Copyright

(1) Copyright

Copyright © 2006, Accelrys Software Inc. All rights reserved. The Accelrys® name and logo are registered trademarks of Accelrys Software Inc.

This product (software and/or documentation) is furnished under a License/Purchase Agreement and may be used only in accordance with the terms of such agreement.

(2) Trademark

The registered trademarks or trademarks of Accelrys Software Inc. include but are not limited to: ACCELERYS® & ACCELERYS Logo, ACCORD, BIOSYM®, CATALYST®, CERIUS®, CERIUS2®, CHARMm®, CHEMEXPLORER®, DIAMOND DISCOVERY®, DISCOVER®, DISCOVERY STUDIO®, DIVA®, FLEXSERVICES®, GCG®, GENEATLAS®, INSIGHT®, INSIGHT II®, MACVECTOR®, MATERIALS STUDIO®, OMIGA®, QUANTA®, SEQARRAY, SEQFOLD®, SEQLAB®, SEQMERGE®, SEQSTORE®, SEQWEB®, TOPKAT®, TSAR®, UNICHEM®, WEBKIT, WEBLAB®, WISCONSIN PACKAGE®. All other trademarks are the property of their respective holders.

(3) Restrictions on Government Use

This is a “commercial” product. Use, release, duplication, or disclosure by the United States Government agencies is subject to restrictions set forth in DFARS 252.227-7013 or FAR 52.227-19, as applicable, and any successor rules and regulations.

(4) Acknowledgments and References

To print photographs or files of computational results (figures and/or data) obtained using Accelrys software; acknowledge the source in an appropriate format. For example:

“Computational results obtained using software programs from Accelrys Software Inc. Dynamics calculations performed with the Discover program using the CFF force field, ab initio calculations performed with the DMol3 program, and graphical displays generated with the Cerius2 molecular modelling system.”

To reference an Accelrys publication in another publication, Accelrys Software, Inc., is the author and the publisher. For example:


(5) Request for Permission to Reprint and Acknowledgment

Accelrys may grant permission to republish or reprint its copyrighted materials. Requests should be submitted to Accelrys Scientific Support, either through electronic mail to support@accelrys.com, or in writing to:

Accelrys Scientific Support
10188 Telesis Court, Suite 100
San Diego, CA 92121

Please include an acknowledgement “Reprinted with permission from Accelrys Software Inc., Document name, Month Year, Accelrys Software Inc., San Diego.” For example:

# TABLE OF CONTENTS

**Programs organized by function**

- Comparison ........................................................................................................ 1
- Database searching .......................................................................................... 3
- DNA/RNA secondary structure ........................................................................... 7
- Editing and publication ...................................................................................... 8
- Evolution ............................................................................................................ 8
- Fragment assembly ............................................................................................. 9
- Gene finding and pattern recognition ................................................................ 9
- Importing and exporting ................................................................................... 11
- Mapping .......................................................................................................... 12
- Primer selection ............................................................................................... 13
- Protein analysis ................................................................................................. 13
- Translation ....................................................................................................... 15
- Utilities ......................................................................................................... 15

**Introduction: Description of a Typical Program**

- 

**R**

- REFORMAT ................................................................................................. 34
- REPEAT ................................................................................................. 46
- REVERSE ............................................................................................... 51

**S**

- SEQCONV+ ........................................................................................... 55
- SEQSTAT+ ............................................................................................ 62
- SEQMANIP+ ......................................................................................... 67
- SAMPLE ................................................................................................. 76
- SEG ......................................................................................................... 81
- SEGMENTS ............................................................................................ 86
- SEQLAB ................................................................................................. 94
- SEQMERGE ........................................................................................... 95
- SHUFFLE ................................................................................................. 96
- SIMPLIFY .............................................................................................. 98
- SPSCAN ................................................................................................. 102
- SPSCAN+ ............................................................................................. 111
- SSEARCH ............................................................................................. 121
- SSEARCH+ .......................................................................................... 136
<table>
<thead>
<tr>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATPLOT^2</td>
<td>153</td>
</tr>
<tr>
<td>STEMLOOP</td>
<td>160</td>
</tr>
<tr>
<td>STRINGSEARCH</td>
<td>167</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>174</td>
</tr>
<tr>
<td>TERNATOR</td>
<td>178</td>
</tr>
<tr>
<td>TESTCODE^2</td>
<td>182</td>
</tr>
<tr>
<td>TFASTA</td>
<td>189</td>
</tr>
<tr>
<td>TFASTX</td>
<td>229</td>
</tr>
<tr>
<td>TFASTX+</td>
<td>244</td>
</tr>
<tr>
<td>TOIG</td>
<td>262</td>
</tr>
<tr>
<td>TOPIR</td>
<td>265</td>
</tr>
<tr>
<td>TOSTADEN</td>
<td>268</td>
</tr>
<tr>
<td>TRANSLATE</td>
<td>271</td>
</tr>
<tr>
<td>TransMem</td>
<td>278</td>
</tr>
<tr>
<td>TransMem+</td>
<td>286</td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
<tr>
<td>WINDOW</td>
<td>301</td>
</tr>
<tr>
<td>WORDSEARCH^2</td>
<td>306</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XNU</td>
<td>321</td>
</tr>
</tbody>
</table>

**Programs not supported by GCG 11.1**

<table>
<thead>
<tr>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHOPUP</td>
<td>326</td>
</tr>
<tr>
<td>COMPRESSTEXT</td>
<td>329</td>
</tr>
<tr>
<td>DETAB</td>
<td>331</td>
</tr>
<tr>
<td>EXTRACTPEPTIDE</td>
<td>334</td>
</tr>
<tr>
<td>GETSEQ</td>
<td>336</td>
</tr>
<tr>
<td>LINEUP</td>
<td>339</td>
</tr>
<tr>
<td>LISTFILE</td>
<td>359</td>
</tr>
<tr>
<td>LPRINT</td>
<td>362</td>
</tr>
<tr>
<td>ONECASE</td>
<td>365</td>
</tr>
<tr>
<td>PEPDATA</td>
<td>368</td>
</tr>
</tbody>
</table>
REPLACE ........................................................................................................... 372
RED .................................................................................................................. 376
SPEW ................................................................................................................... 402
SHIFTOVER ................................................................................................. 405
SETKEYS ............................................................................................................ 407
SEQED .................................................................................................................. 410

Deprecated Programs: Use seqconv+ to obtain the same functionality

TOFASTA .............................................................................................................. 430
FROMFASTA ....................................................................................................... 439
FROMGENBANK ................................................................................................. 443

APPENDICES ....................................................................................................... 449

I Character Fonts .............................................................................................. 449
II Sequence Symbols .......................................................................................... 451
III Form for Reporting Errors ........................................................................... 454
IV References ....................................................................................................... 455
V Sequence Typing ............................................................................................. 464
VI Data Files ......................................................................................................... 465

Note:
2 Program provides a graphical output
+ Program version with enhanced capabilities.
Programs organized by function

The following table groups programs according to function, with some programs grouped under multiple functional categories. Categories appear alphabetically. You can find each GCG program in the following functional categories:

- Comparison
- DNA/RNA Secondary Structure
- Evolution
- Gene Finding and Pattern Recognition
- Mapping
- Protein Analysis
- Utilities
- Database Searching and Retrieval
- Editing and Publication
- Fragment Assembly
- Importing and Exporting
- Primer Selection
- Translation

Comparison

Sequence comparison techniques fall into one of two categories: pairwise or multiple. Pairwise comparisons can be either alignments or dot-plots. Multiple sequence comparisons are displayed as multiple sequence alignments.

Pairwise Comparison

**Gap**

Gap uses the algorithm of Needleman and Wunsch to find the alignment of two complete sequences that maximizes the number of matches and minimizes the number of gaps.

**BestFit**

BestFit makes an optimal alignment of the best segment of similarity between two sequences. Optimal alignments are found by inserting gaps to maximize the number of matches using the local homology algorithm of Smith and Waterman.

**FrameAlign**

FrameAlign creates an optimal alignment of the best segment of similarity (local alignment) between a protein sequence and the codons in all possible reading frames on a single strand of a nucleotide sequence. Optimal alignments may include reading frame shifts.

**Compare**

Compare compares two protein or nucleic acid sequences and creates a file of the points of similarity between them for plotting with DotPlot. Compare finds the points using either a window/stringency or a word match criterion. The word
comparison is 1,000 times faster than the window/stringency comparison, but somewhat less sensitive.

**DotPlot**
DotPlot makes a dot-plot with the output file from Compare or StemLoop.

**GapShow**
GapShow displays an alignment by making a graph that shows the distribution of similarities and gaps. The two input sequences should be aligned with either Gap or BestFit before they are given to GapShow for display.

**ProfileGap**
ProfileGap makes an optimal alignment between a profile and one or more sequences.

**Multiple Sequence Comparison**

**PileUp**
PileUp creates a multiple sequence alignment from a group of related sequences using progressive, pairwise alignments. It can also plot a tree showing the clustering relationships used to create the alignment.

**ClustalW+**
ClustalW+ creates a multiple sequence alignment from a group of related sequences using progressive, pairwise alignments. It can also create a dendrogram (.dnd) showing the clustering relationships used to create the alignment.

**HmmerAlign**
HmmerAlign uses a profile hidden Markov model (HMM) as a template to create an optimal multiple alignment of a group of sequences.

**SeqLab**
SeqLab is the graphical user interface to GCG. It is an interactive multiple sequence analysis tool that lets you view and refine alignments both by hand and by calling GCG programs like PileUp. You can do any analysis available in GCG from SeqLab. See the SeqLab Guide for more information.

**PlotSimilarity**
PlotSimilarity plots the running average of the similarity among the sequences in a multiple sequence alignment.

**Pretty**
Pretty displays multiple sequence alignments and calculates a consensus sequence. It does not create the alignment; it simply displays it.

**Prettybox**
PrettyBox displays multiple sequence alignments as shaded boxes in Postscript format for printing or displaying with a Postscript compatible device. PrettyBox optionally calculates a consensus sequence. The program does not create the alignment; it simply displays it.

**MEME**
MEME finds conserved motifs in a group of unaligned sequences. MEME saves these motifs as a set of profiles. You can search a database of sequences with these profiles using the MotifSearch program.
MEME+
MEME+ finds conserved motifs in a group of unaligned sequences. MEME+ saves these motifs as a set of profiles. You can search a database of sequences with these profiles using the MotifSearch program.

HmmerBuild
HmmerBuild creates a position-specific scoring table, called a profile hidden Markov model (HMM), that is a statistical model of the consensus of a multiple sequence alignment. The profile HMM can be used for database searching (HmmerSearch), sequence alignment (HmmerAlign) or generating random sequences that match the model (HmmerEmit).

HmmerCalibrate
HmmerCalibrate “calibrates” a profile hidden Markov model in order to increase the sensitivity of database searches performed using that profile HMM as a query. The program compares the original profile HMM with a large number of randomly generated sequences and computes the extreme value distribution (EVD) parameters for this simulated search. The original profile HMM is replaced with a new one that contains these EVD parameters.

ProfileMake
ProfileMake creates a position-specific scoring table, called a profile, that quantitatively represents the information from a group of aligned sequences. The profile can then be used for database searching (ProfileSearch) or sequence alignment (ProfileGap).

ProfileGap
ProfileGap makes an optimal alignment between a profile and one or more sequences.

Overlap
Overlap compares two sets of DNA sequences to each other in both orientations using a WordSearch style comparison.

NoOverlap
NoOverlap identifies the places where a group of nucleotide sequences do not share any common subsequences.

OldDistances
OldDistances makes a table of the pairwise similarities within a group of aligned sequences.

Database searching
You can search sequence databases using either sequence or text queries. Any database sequence, together with its annotations, can be retrieved as a file.

Reference Searching
LookUp
LookUp identifies sequence database entries by name, accession number, author, organism, keyword, title, reference, feature, definition, length, or date. The output is a list of sequences.
StringSearch
StringSearch identifies sequences by searching for character patterns such as “globin” or “human” in the sequence documentation.

Names
Names identifies GCG data files and sequence entries by name. It can show you what set of sequences is implied by any sequence specification.

Sequence Searching

BLAST
BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds.

BLAST+
BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds.

NetBLAST
NetBLAST searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

NetBLAST+
NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

PSIBLAST
PSIBLAST iteratively searches one or more protein databases for sequences similar to one or more protein query sequences. PSIBLAST is similar to BLAST except that it uses position-specific scoring matrices derived during the search.

FastA
FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

FastA+
FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

SSearch
SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

SSearch+
SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein).
This may be the most sensitive method available for similarity searches. Compared to BLAST+ and FastA+, it can be very slow.

**TFastA**

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, “What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?”

**TFastA+**

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

**TFastX**

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, “What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?”

**TFastX+**

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like FastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

**FastX**

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, “What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?”

**FastX+**

FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX+ translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

**FrameSearch**

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.
MotifSearch
MotifSearch uses a set of profiles (representing similarities within a family of sequences) as a query to either a) search a database for new sequences similar to the original family, or b) annotate the members of the original family with details of the matches between the profiles and each of the members. Normally, the profiles are created with the program MEME.

HmmerSearch
HmmerSearch uses a profile hidden Markov model as a query to search a sequence database to find sequences similar to the family from which the profile HMM was built. Profile HMMs can be created using HmmerBuild.

ProfileSearch
ProfileSearch uses a profile (representing a group of aligned sequences) as a query to search the database for new sequences with similarity to the group. The profile is created with the program ProfileMake.

ProfileSegments
ProfileSegments makes optimal alignments showing the segments of similarity found by ProfileSearch.

FindPatterns
FindPatterns identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal.

FindPatterns+
FindPatterns+ identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal.

Motifs
Motifs looks for sequence motifs by searching through proteins for the patterns defined in the PROSITE Dictionary of Protein Sites and Patterns. Motifs can display an abstract of the current literature on each of the motifs it finds.

HmmerPfam
HmmerPfam compares one or more sequences to a database of profile hidden Markov models, such as the Pfam library, in order to identify known domains within the sequences.

HmmerBuild
HmmerBuild creates a position-specific scoring table, called a profile hidden Markov model (HMM), that is a statistical model of the consensus of a multiple sequence alignment. The profile HMM can be used for database searching (HmmerSearch), sequence alignment (HmmerAlign) or generating random sequences that match the model (HmmerEmit).

HmmerCalibrate
HmmerCalibrate “calibrates” a profile hidden Markov model in order to increase the sensitivity of database searches performed using that profile HMM as a query. The program compares the original profile HMM with a large number of randomly generated sequences and computes the extreme value distribution (EVD) parameters for this simulated search. The original profile HMM is replaced with a new one that contains these EVD parameters.
WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

**Segments**
Segments aligns and displays the segments of similarity found by WordSearch.

**Sequence Retrieval**

**Fetch**
Fetch copies GCG sequences or data files from the GCG database into your directory or displays them on your terminal screen.

**Fetch+**
Fetch+ copies GCG sequences or data files from the GCG database into your directory or displays them on your terminal screen.

**NetFetch**
NetFetch retrieves sequences from NCBI listed in a NetBLAST output file. You can also use it to retrieve sequences individually by sequence name or accession number. The output of NetFetch is an RSF file.

**NetFetch+**
NetFetch+ retrieves sequences from NCBI listed in a NetBLAST+ output file. You can also use it to retrieve sequences individually by sequence name or accession number. The output of NetFetch+ is an RSF file.

**DNA/RNA secondary structure**

Michael Zuker’s MFold program predicts optimal and suboptimal RNA and DNA secondary structures, which you then can display graphically in six ways with the PlotFold program. StemLoop finds inverted repeats.

**MFold**
MFold predicts optimal and suboptimal secondary structures for an RNA or DNA molecule using the most recent energy minimization method of Zuker.

**PlotFold**
PlotFold displays the optimal and suboptimal secondary structures for an RNA or DNA molecule predicted by MFold.

**StemLoop**
StemLoop finds stems (inverted repeats) within a sequence. You specify the minimum stem length, minimum and maximum loop sizes, and the minimum number of bonds per stem. All stems or only the best stems can be displayed on your screen or written into a file.

**DotPlot**
DotPlot makes a dot-plot with the output file from Compare or StemLoop.
**Editing and publication**

These programs help you enter and modify protein or nucleic acid sequences as well as visualize and edit multiple sequence alignments. You can also make publication-quality figures of plasmid constructs, sequences, and sequence alignments.

**SeqLab**
SeqLab is the graphical user interface to GCG. It is an interactive multiple sequence analysis tool that lets you view and refine alignments both by hand and by calling GCG programs like PileUp. You can do any analysis available in GCG from SeqLab. See the SeqLab Guide for more information.

**Assemble**
Assemble constructs new sequences from pieces of existing sequences. It concatenates the fragments you specify and writes them out as a new sequence file. SeqEd is a better tool for assembling sequences interactively, but Assemble is best for assembling sequences from fragments defined in a list file.

**Pretty**
Pretty displays multiple sequence alignments and calculates a consensus sequence. It does not create the alignment; it simply displays it.

**PrettyBox**
PrettyBox displays multiple sequence alignments as shaded boxes in Postscript format for printing or displaying with a Postscript compatible device. PrettyBox optionally calculates a consensus sequence. The program does not create the alignment; it simply displays it.

**Publish**
Publish arranges sequences for publication. It creates a text file that you can modify to your own needs with a text editor.

**PlasmidMap**
PlasmidMap draws a circular plot of a plasmid construct. It can display restriction patterns, inserts, and known genetic elements. The plot is suitable for publication, record keeping, or analysis. It is drawn from one or more labeling files such as those written by MapSort.

**Figure**
Figure makes figures and posters by drawing graphics and text together. You can include output from other GCG graphics programs as part of a figure.

**Evolution**

These programs allow you to investigate the relationships within a group of pre-aligned sequences. You can compute the pairwise distances between sequences in an alignment, reconstruct phylogenetic trees using distance methods, and calculate the degree of divergence of two protein coding regions.

**PAUPSearch**
PAUPSearch provides a GCG interface to the tree-searching options in PAUP (Phylogenetic Analysis Using Parsimony). Starting with a set of aligned sequences, you can search for phylogenetic trees that are optimal according to
parsimony, distance, or maximum likelihood criteria; reconstruct a neighbor-joining tree; or perform a bootstrap analysis. The program PAUPDisplay can produce a graphical version of a PAUPSearch trees file. PAUP is the copyrighted property of the Smithsonian Institution. Use the program Fetch to obtain a copy of paup-license.txt to read about rights and limitations for using PAUP.

**PAUPDisplay**

PAUPDisplay provides a GCG interface to tree manipulation, diagnosis, and display options in PAUP (Phylogenetic Analysis Using Parsimony). Starting with a trees file that contains a sequence alignment and one or more trees reconstructed from this alignment (such as the output from PAUPSearch), you can plot the tree(s); compute the score of the tree(s) according to the criteria of parsimony, distance, or maximum likelihood; or calculate a consensus tree (two or more input trees). PAUPDisplay can also plot the trees from a GrowTree trees file.

PAUP is the copyrighted property of the Smithsonian Institution. Use the program Fetch to obtain a copy of paup-license.txt to read about rights and limitations for using PAUP.

**Distances**

Distances creates a table of the pairwise distances within a group of aligned sequences.

**GrowTree**

GrowTree creates a phylogenetic tree from a distance matrix created by Distances using either the UPGMA or neighbor-joining method. You can create a text or graphics output file.

**Diverge**

Diverge estimates the pairwise number of synonymous and nonsynonymous substitutions per site between two or more aligned nucleic acid sequences that code for proteins. It uses a variant of the method published by Li et al.

**Fragment assembly**

The Fragment Assembly System is composed of a series of programs that let you enter and assemble overlapping nucleotide sequence fragments into one continuous consensus sequence.

**SeqMerge**

SeqMerge is GCG’s powerful new fragment assembly application with an X Windows graphical user interface. SeqMerge allows you to intuitively assemble fragments in a sequencing project into contigs, or alignments of overlapping fragments. From the contig, SeqMerge creates a consensus sequence representing the underlying sequence from which your fragments were derived.

**Gene finding and pattern recognition**

These programs help you recognize coding regions, terminators, repeats, and consensus patterns. Several of the programs help you analyze sequence composition.
TestCode helps you identify potential protein coding regions in nucleic acid sequences by plotting a measure of the non-randomness of the composition at every third base. The statistic does not require a codon frequency table.

CodonPreference

CodonPreference is a frame-specific gene finder that tries to recognize protein coding sequences by virtue of the similarity of their codon usage to a codon frequency table or by the bias of their composition (usually GC) in the third position of each codon.

Frames

Frames shows open reading frames for the six translation frames of a DNA sequence. Frames can superimpose the pattern of rare codon choices if you provide it with a codon frequency table.

Terminator

Terminator searches for prokaryotic factor-independent RNA polymerase terminators according to the method of Brendel and Trifonov.

Motifs

Motifs looks for sequence motifs by searching through proteins for the patterns defined in the PROSITE Dictionary of Protein Sites and Patterns. Motifs can display an abstract of the current literature on each of the motifs it finds.

MEME

MEME finds conserved motifs in a group of unaligned sequences. MEME saves these motifs as a set of profiles. You can search a database of sequences with these profiles using the MotifSearch program.

MEME+

MEME+ finds conserved motifs in a group of unaligned sequences. MEME+ saves these motifs as a set of profiles. You can search a database of sequences with these profiles using the MotifSearch program.

Repeat

Repeat finds direct repeats in sequences. You must set the size, stringency, and range within which the repeat must occur; all the repeats of that size or greater are displayed as short alignments.

FindPatterns

FindPatterns identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal.

FromTrace

FromTrace converts one or more ABI or SCF trace files into GCG single sequence files.

FindPatterns+

FindPatterns+ identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal.

Composition

Composition determines the composition of sequence(s). For nucleotide sequence(s), Composition also determines dinucleotide and trinucleotide content.
CodonFrequency
CodonFrequency tabulates codon usage from sequences and/or existing codon usage tables. The output file is correctly formatted for input to the CodonPreference, Correspond, and Frames programs.

Correspond
Correspond looks for similar patterns of codon usage by comparing codon frequency tables.

Window
Window makes a table of the frequencies of different sequence patterns within a window as it is moved along a sequence. A pattern is any short sequence like GC or R or ATG. You can plot the output with the program StatPlot.

StatPlot
StatPlot plots a set of parallel curves from a table of numbers like the table written by the Window program. The statistics in each column of the table are associated with a position in the analyzed sequence.

FitConsensus
FitConsensus uses a consensus table written by Consensus as a probe to find the best examples of the consensus in a DNA sequence. You can specify the number of fits you want to see, and FitConsensus tabulates them with their position, frame, and a statistical measure of their quality.

Consensus
Consensus calculates a consensus sequence for a set of pre-aligned short nucleic acid sequences by tabulating the percent of G, A, T, and C for each position in the set. FitConsensus uses the Consensus output table as a probe to search for the best examples of the derived consensus in other nucleotide sequences.

Xnu
Xnu replaces statistically significant tandem repeats in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

Seg
Seg replaces low complexity regions in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

Importing and exporting

These utilities convert sequences from other formats into GCG format and convert GCG sequences to formats compatible with other popular sequence analysis software tools.

Reformat
Reformat rewrites sequence file(s), scoring matrix file(s), or enzyme data file(s) so that they can be read by GCG programs.

BreakUp
BreakUp reads a GCG-format sequence file containing more than 350,000 sequence characters and writes it as a set of separate, shorter, overlapping sequence files that can be analyzed by GCG programs.
**HmmerConvert**

HmmerConvert converts profile hidden Markov model files into different profile formats.

**SeqConv+**

SeqConv+ is a utility program that provides batch conversions between different sequence formats. The motivation for the program is to allow an end user to easily convert between file formats to easily import data into Accelrys’ bioinformatics applications. In addition, the converter allows the user to convert our internally used formats (e.g. BSML, RSF) into formats more commonly accepted by third-party tools. The supported file formats will include BSML, GenBank, FastA, EMBL, and RSF.

**Mapping**

These programs calculate and display restriction maps and peptide cleavage patterns. There is also a program that simulates RNA fingerprints.

**Map**

Map maps a DNA sequence and displays both strands of the mapped sequence with restriction enzyme cut points above the sequence and protein translations below. Map can also create a peptide map of an amino acid sequence.

**Map+**

Map+ maps a DNA sequence and displays both strands of the mapped sequence with restriction enzyme cut points above the sequence and protein translations below. Map+ can also create a peptide map of an amino acid sequence.

**MapPlot**

MapPlot displays restriction sites graphically. If you do not have a plotter, MapPlot can write a text file that approximates the graph.

**MapSort**

MapSort finds the coordinates of the restriction enzyme cuts in a DNA sequence and sorts the fragments of the resulting digest by size. MapSort can sort the fragments from single or multiple enzyme digests.

**Fingerprint**

Fingerprint identifies the products of T1 ribonuclease digestion.

**PeptideMap**

PeptideMap creates a peptide map of an amino acid sequence.

**PlasmidMap**

PlasmidMap draws a circular plot of a plasmid construct. It can display restriction patterns, inserts, and known genetic elements. The plot is suitable for publication, record keeping, or analysis. It is drawn from one or more labeling files such as those written by MapSort.

**PeptideSort**

PeptideSort shows the peptide fragments from a digest of an amino acid sequence. It sorts the peptides by position, putative molecular weight, and relative HPLC retention at pH 2.1, and shows the composition of each peptide. It also prints a summary of the composition of the whole protein.
**Primer selection**

These programs help you evaluate oligonucleotide primers.

**Prime**
Prime selects oligonucleotide primers for a template DNA sequence. The primers may be useful for the polymerase chain reaction (PCR) or for DNA sequencing. You can allow Prime to choose primers from the whole template or limit the choices to a particular set of primers listed in a file.

**Prime+**
Prime+ selects oligonucleotide primers for a template DNA sequence. The primers may be useful for the polymerase chain reaction (PCR) or for DNA sequencing. You can allow Prime to choose primers from the whole template or limit the choices to a particular set of primers listed in a file.

**PrimePair**
PrimePair evaluates individual primers to determine their compatibility for use as PCR primer pairs. You can provide the primers in files (one for forward, one for reverse primers) or on the command line, or you can enter them interactively from the keyboard.

**MeltTemp**
MeltTemp computes the melting temperature of oligonucleotides. You can provide the oligonucleotide sequences in a file or simply type them in at the keyboard.

**Protein analysis**

These programs do analyses specific to protein sequences. The first two programs identify sequence motifs in protein sequences. The next three programs predict peptide properties. The next five programs look at secondary structure, hydrophobicity, and antigenicity. The last two programs identify repeats and regions of low complexity.

**Motifs**
Motifs looks for sequence motifs by searching through proteins for the patterns defined in the PROSITE Dictionary of Protein Sites and Patterns. Motifs can display an abstract of the current literature on each of the motifs it finds.

**HmmerPfam**
HmmerPfam compares one or more sequences to a database of profile hidden Markov models, such as the Pfam library, in order to identify known domains within the sequences.

**ProfileScan**
ProfileScan uses a database of profiles to find structural and sequence motifs in protein sequences.

**CoilScan**
CoilScan locates coiled-coil segments in protein sequences.

**CoilScan+**
CoilScan+ locates coiled-coil segments in protein sequences.
HTHScan
HTHScan scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

SPScan
SPScan scans protein sequences for the presence of secretory signal peptides (SPs).

HTHScan+
HTHScan+ scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

SPScan+
SPScan+ scans protein sequences for the presence of secretory signal peptides (SPs).

PeptideSort
PeptideSort shows the peptide fragments from a digest of an amino acid sequence. It sorts the peptides by position, putative molecular weight, and relative HPLC retention at pH 2.1, and shows the composition of each peptide. It also prints a summary of the composition of the whole protein.

Isoelectric
Isoelectric plots the charge as a function of pH for any peptide sequence.

PeptideMap
PeptideMap creates a peptide map of an amino acid sequence.

PepPlot
PepPlot plots measures of protein secondary structure and hydrophobicity in parallel panels of the same plot.

PeptideStructure
PeptideStructure makes secondary structure predictions for a peptide sequence. The predictions include (in addition to alpha, beta, coil, and turn) measures for antigenicity, flexibility, hydrophobicity, and surface probability. PlotStructure displays the predictions graphically.

PlotStructure
PlotStructure plots the measures of protein secondary structure in the output file from PeptideStructure. The measures can be shown on parallel panels of a graph or with a two-dimensional “squiggly” representation.

Moment
Moment makes a contour plot of the helical hydrophobic moment of a peptide sequence.

HelicalWheel
HelicalWheel plots a peptide sequence as a helical wheel to help you recognize amphiphilic regions.

TransMem
TransMem scans for likely transmembrane helices in one or more input protein sequences.
**TransMem+**
TransMem+ scans for likely transmembrane helices in one or more input protein sequences.

**Xnu**
Xnu replaces statistically significant tandem repeats in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

**Seg**
Seg replaces low complexity regions in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

**Translation**
These programs translate nucleic acids into proteins and proteins back into nucleic acids.

**Translate**
Translate translates nucleotide sequences into peptide sequences.

**BackTranslate**
BackTranslate backtranslates an amino acid sequence into a nucleotide sequence. The output helps you identify areas with fewer ambiguities that might be candidates for synthetic probes.

**Map**
Map maps a DNA sequence and displays both strands of the mapped sequence with restriction enzyme cut points above the sequence and protein translations below. Map can also create a peptide map of an amino acid sequence.

**Map+**
Map maps a DNA sequence and displays both strands of the mapped sequence with restriction enzyme cut points above the sequence and protein translations below. Map can also create a peptide map of an amino acid sequence.

**Reverse**
Reverse reverses and/or complements a sequence.

**DataSet**
DataSet creates a GCG data library from any set of sequences in GCG format.

**DataSet+**
DataSet+ creates a GCG data library from any set of sequences.

**Utilities**
A number of utilities are available that make your work with sequences, databases, printing/plotting, and files easier. Sequence utilities transform sequences in various ways. Database utilities assemble sequences into a database format that allows them to be searched efficiently. Printing/plotting utilities help you print your output, configure plotting utilities, test
your plotting environment, and customize and reproduce GCG graphics. File utilities transform files in ways that many people find useful. Miscellaneous utilities are difficult to categorize.

### Sequence Utilities

**Reverse**
Reverse reverses and/or complements a sequence.

**Shuffle**
Shuffle randomizes the order of the symbols in a sequence without changing the composition.

**Simplify**
Simplify lets you reduce the number of symbols in a sequence. Such a simplification would allow you, for instance, to treat all hydrophobic amino acids as equivalent.

**CompTable**
CompTable creates a scoring matrix using equivalences defined in a simplification scheme such as the one used for Simplify.

**HmmerEmit**
HmmerEmit generates sequences that match a profile hidden Markov model.

**Corrupt**
Corrupt randomly introduces small numbers of substitutions, insertions, and deletions into nucleotide or protein sequence(s).

**Xnu**
Xnu replaces statistically significant tandem repeats in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

**Seg**
Seg replaces low complexity regions in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

**Sample**
Sample extracts sequence fragments randomly from sequence(s). You can set a sampling rate to determine how many fragments Sample extracts.

**SeqManip+**
SeqManip+ is a utility program that allows the user to perform some manipulations of sequences, including translation, back translation of protein sequences, splitting sequences. While individual programs to perform these tasks already exist in Wisconsin Package 10.3, SeqManip+ provides a single platform to execute all the relevant sequence operations. This saves the users from having to find and run several different applications in order to execute some basic sequence manipulations.

**SeqConv+**
SeqConv+ is a utility program that provides batch conversions between different sequence formats. The motivation for the program is to allow an end user to easily convert between file formats to easily import data into Accelrys’
bioinformatics applications. In addition, the converter allows the user to convert our internally used formats (e.g. BSML, RSF) into formats more commonly accepted by third-party tools. The supported file formats will include BSML, GenBank, FastA, EMBL, and RSF.

SeqStat+
SeqStat is a utility program that reads through any number of input sequences and provides some basic statistics about the files, including total length, number of sequences, and average length. Additionally it provides some extended information about the sequences depending on their type (protein or nucleotide), such as G+C% content.

Database Utilities
DataSet
DataSet creates a GCG data library from any set of sequences in GCG format.

DataSet+
DataSet creates a GCG data library from any set of sequences.

Sample
Sample extracts sequence fragments randomly from sequence(s). You can set a sampling rate to determine how many fragments Sample extracts.

Printing / Plotting Utilities
StatPlot
StatPlot lets you choose a graphics configuration from a menu of available graphics devices at your site. For more information, see Chapter 5, Using Graphics in the User’s Guide.

Figure
Figure makes figures and posters by drawing graphics and text together. You can include output from other GCG graphics programs as part of a figure.

PlotTest
PlotTest plots an example graphic to test your graphics configuration. The graphic created by PlotTest uses every GCG graphics feature. It should resemble the example graphic in the Program Manual.

PostScript
Sets your graphics configuration to PostScript to write to a PostScript-compatible device or file.

HPGL
Sets your graphics configuration to HPGL (Hewlett Packard Graphics Language) to write to a HPGL-compatible device.

X Windows
Sets your graphics configuration to X Windows to display graphics on an X Windows-compatible device.
ReGIS
Sets your graphics configuration to ReGIS to write to a ReGIS-compatible device.

Tektronix
Sets your graphics configuration to Tektronix to write to a Tektronix-compatible device.

GIF
Sets your graphics configuration to GIF (Graphics Interchange Format) to write to a GIF file, which can then be displayed via a web browser.

PNG
Sets your graphics configuration to PNG (Portable Network Graphics) to write to a PNG file, which can then be displayed via a web browser.

Miscellaneous Utilities

Reformat
Reformat rewrites sequence file(s), scoring matrix file(s), or enzyme data file(s) so that they can be read by GCG programs.

HmmerBuild
HmmerBuild creates a position-specific scoring table, called a profile hidden Markov model (HMM), that is a statistical model of the consensus of a multiple sequence alignment. The profile HMM can be used for database searching (HmmerSearch), sequence alignment (HmmerAlign) or generating random sequences that match the model (HmmerEmit).

HmmerCalibrate
HmmerCalibrate “calibrates” a profile hidden Markov model in order to increase the sensitivity of database searches performed using that profile HMM as a query. The program compares the original profile HMM with a large number of randomly generated sequences and computes the extreme value distribution (EVD) parameters for this simulated search. The original profile HMM is replaced with a new one that contains these EVD parameters.

HmmerConvert
HmmerConvert converts profile hidden Markov model files into different profile formats.

HmmerIndex
HmmerIndex creates an index for a profile hidden Markov model database so that profile HMMs can be retrieved from the database with HmmerFetch.

HmmerFetch
HmmerFetch retrieves a profile hidden Markov model (HMM) from a database of profile HMMs that has been indexed by HmmerIndex.

Name
Name creates, changes, deletes, or displays GCG logical name(s) from the GCG logical names table.

Symbol
Symbol creates, changes, deletes, or displays GCG symbol(s) from the GCG symbol table.
Programs not supported by GCG 11.0

ExtractPeptide
ExtractPeptide writes a peptide sequence from one or more of the translation frames displayed in the output from Map. Translate supersedes ExtractPeptide for most applications.

PepData
PepData translates DNA sequence(s) in all six frames, concatenates the translations, and creates a single, protein output file.

Red
Red is a text formatter that creates publication-quality documents on a PostScript printer such as the Apple LaserWriter. You can use 13 different fonts, scaling each font to any size. You can also include figures and graphics from any GCG graphics program within the text of the document.

SetKeys
SetKeys writes a file in your current directory that redefines your keyboard’s keys for easier sequence entry with the SeqEd, LineUp, GelEnter and GelAssemble programs and the SeqLab sequence editor. The output file, called set.keys, can be edited if you want to redefine keys that were not considered by the SetKeys program.

GetSeq
GetSeq reads a sequence from a computer that is acting as a terminal and writes it into a new sequence file in GCG format on the computer running GCG.

SeqEd
SeqEd is an interactive editor for entering and modifying sequences and for assembling parts of existing sequences into new genetic constructs. You can enter sequences from the keyboard or from a digitizer.

Spew
Spew sends a GCG sequence from the computer that runs GCG to a personal computer acting as a terminal.

ChopUp
ChopUp converts a non-GCG sequence file containing lines longer than 511 character and as long as 32,000 characters into a new file containing lines no longer than 50 characters. The new file can be read by Reformat to create a GCG-format sequence file.

Replace
Replace makes character string replacements in text file(s). You provide a table of replacements in a file showing each existing string and its replacement.

CompressText
CompressText removes any or all of the following from files: A) trailing space; B) blank lines; C) extra space between words; D) all space; or E) leading space.
OneCase
OneCase puts all of the alphabetic characters in a file into lower or UPPER case. It can also capitalize every word.

ShiftOver
ShiftOver moves a file to the right or to the left as many columns as you specify.

Detab
Detab replaces the tab characters in one or more files with spaces. The files can be written out in card-image format with records of fixed length.

LPrint
LPrint prints text file(s) on a PostScript printer connected to LPrintPort.

ListFile
Note: This program is not supported by GCG 11.0.
ListFile prints a file on a printer attached to your terminal’s pass-through printer port.

ChopUp
ChopUp converts a non-GCG sequence file containing lines longer than 511 character and as long as 32,000 characters into a new file containing lines no longer than 50 characters. The new file can be read by Reformat to create a GCG-format sequence file.

GCGToBLAST
This program is no longer available. Please use FormatDB+. GCGToBLAST combines any set of GCG sequences into a database that you can search with BLAST.

GelAssemble
GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge.

GelDisassemble
GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

GelEnter
GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files.

GelMerge
GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble.

GelStart
GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project.

GelView
GelView displays the structure of the contigs in a fragment assembly project.
 Deprecated Programs: Use seqconv+ to obtain the same functionality.

**FromEMBL**
FromEMBL reformats sequences from the distribution (flat file) format of the EMBL database into individual sequence files in GCG format.

**FromGenBank**
FromGenBank reformats one or more sequences in the flat file format of the GenBank database into individual sequence files in GCG format.

**FromFastA**
FromFastA reformats one or more sequences from FastA format into single sequence files in GCG format.

**ToFastA**
ToFastA converts GCG sequence(s) into FastA format.
Introduction: Description of a Typical Program

The Program Manual contains an entry for each GCG program and describes that program in detail. The entries are arranged alphabetically. If you do not know what program to use, see the Functional Table of Contents.

This chapter describes the standard format for each program entry.

**FUNCTION**

Contains a short description of the program’s function. This description also appears on your screen when you run the program interactively.

**DESCRIPTION**

Describes the major features of the program.

Most GCG programs want one or two sequence specifications for input and write their output into a text file. You can send the output to your terminal screen, instead of a file, if you answer the output file prompt with Term. If you want to run the program as shown in the example session, first copy the input file for each sample session into your own directory with the Fetch program.

The Program Manual entry for Reformat describes sequence files. You can also turn to in Chapter 2, Using Sequence Files and Databases in the User’s Guide for more information about sequence file types and specifying sequences.

**EXAMPLE**

Displays a screen trace for an interactive session with each program. Each program runs when you type its name at the operating system prompt. The bold parts of the screen trace are what you type. If only part of a word is bold, this bold part represents the fewest characters you can type. All the other non-bold letters in the screen trace are prompts from the program. Most GCG programs ask the same questions in the same order.

Below is a typical screen trace where the first 1,000 bases of each strand of the sequence in the file gamma.seq are compared to each other. Default answers, if available, are displayed between asterisks within parentheses, for example (*gamma.txt*). For all programs, you need to type responses only when you do not want to use the default answer; you accept the default answer by simply pressing <Return>. 


Non-Plus Program Example:

```bash
% codonpreference -NOBIAS
CODONPREFERENCE for what sequence? Bacterial: EcoOmpa

  Begin (* 1 *)?
  End (* 2270 *)?
  Reverse (* No *)?

What codon frequency file (* GenRunData: ecohigh.cod *)?
What codon preference window size (in codons) (* 25 *)?
The minimum density for a one-page plot is 74.48 bases/cm.
What density would you like (* 74.48 *)?
When your LaserWriter attached to tty07 is ready, press <Return>.

Average codon preference for frame 1 = 0.8853
Average codon preference for frame 2 = 0.5070
Average codon preference for frame 3 = 0.4742

Average codon preference for a random sequence = 0.4742
```

Plus Program Example:

```bash
> seqmanip+ -translate -open=50
SeqManip is a utility that accepts DNA or protein sequences to
perform single/multiple operations on the selected sequences like
extract, sample, translate, backtranslate, reverse and reverse
complement.
Manipulate what sequence(s)? ggamma.seq
Begin (* 1 *)?
End (-1 for entire sequence) (* -1 *)?
What should I call the output file (* <sequence_name>.seqmanip+ *)?
Extracting the region 1 to 1698 from ggamma.seq ...
Writing 1 sequence(s) to output file ggamma.seqmanip+
```

Note: The examples shown in the Program Manual may vary slightly from what
you see on your terminal screen.

**OUTPUT**

Displays the contents of the output file from the example session. If output file is
too long, portions may be omitted, denoted by
Note: The output shown in the Program Manual may vary slightly from what you see on your terminal screen.

INPUT FILE

Describes the type of file the program accepts as input, for example nucleotide or protein, or single or multiple sequence specification. Multiple sequence specifications include list, RSF, or MSF files as well as specifications using a wildcard such as (*).

The New “+” programs can convert one or more sequence files into a specified format [BSML, GB (GenBank), FASTA, EMBL, SPT (SPTrEMBL), SW (SwissProt), RSF, SSF (GCG), and MSF]. With multiple files, these programs can either convert each file into a separate file or concatenate them all into one file.

RELATED PROGRAMS

Lists other programs in GCG of similar or related function. Sometimes the related programs are useful for preparing your data beforehand or displaying it afterward.

RESTRICTIONS

Lists program restrictions. For example, sequences cannot be longer than 350,000 symbols, but many searches and comparisons cannot handle segments longer than 30,000. Also, often an analysis is specific to either proteins or nucleotide sequences and has no meaning if the other kind of sequence is used.

Note: These restrictions applies only for old programs and all the new “+” programs can process symbols greater than 350,000.

ALGORITHMS

Describes the algorithm, statistic, or figure of merit.

CONSIDERATIONS

Lists the program’s strengths and weaknesses you may want to factor in when running the program.
SUGGESTIONS

Provides parameter values that give reasonable results for the most obvious uses of the program to help you get started.

GRAPHICS

Reminds you that the GCG graphics configuration must always be set up before you run a program with graphic output. See Chapter 5, Using Graphics in the User’s Guide for more information about configuring your process for graphics.

COMMAND-LINE SUMMARY

While all GCG programs will run interactively when you simply type the program’s name, most programs also allow complete command-line control. This control is for advanced users who want to use all of the features of a program or who use a particular program frequently. Command-line control also lets you write scripts that run several programs in succession.

This topic shows a command-line summary for the program. In the summary, the capitalized letters in the parameter names are the letters that you must type to use the parameter. Square brackets ([ and ]) enclose parameters or values that are optional. For more information, see “Using Program Parameters” in Chapter 3, Using Programs in the User’s Guide.

Non-Plus Program Command-Line Summary:

Minimal Syntax: % reverse [-INfile=]ggamma.seq -Default

Prompted Parameters:

-BEGin=1 -END=1700 sets the range of interest
-NOREVerse doesn't reverse the strand
-NOCOMplement doesn't complement the strand
[-OUTfile=]ggamma.rev names the output file

Local Data Files: None

Optional Parameters: None

Plus Program Command-Line Summary:

Minimal Syntax: % seqconv+ [-infile=]value -Default

Minimal Parameters (case-insensitive):

-infile [Type: List / Default: EMPTY / Aliases: infile1
in]Input file specification
Prompted Parameters (case-insensitive):

- **format**  
  [Type: String / Default: 'BSML' / Aliases: fmt] The desired output format for the files. Should be one of BSML, GB (GenBank), FASTA, EMBL, SPT (SPTreMBL), SW (SwissProt), RSF, SSF (GCG) and MSF.

Optional Parameters (case-insensitive):

- **check**  
  [Type: Boolean / Default: 'false' / Aliases: che help] Prints out this usage message

- **default**  
  [Type: Boolean / Default: 'false' / Aliases: def] Specifies that sensible default values be used for all parameters where possible.

- **documentation**  
  [Type: Boolean / Default: 'true' / Aliases: doc] Prints banner at program startup

- **quiet**  
  [Type: Boolean / Default: 'false' / Aliases: qui] Tells application to print only a minimal amount of information

- **outfile**  
  [Type: OutFile / Default: EMPTY / Aliases: out] File to which all input files are concatenated. A value of '-' means STDOUT. Specifying this option also turns on the 'concat' option. Default value is 'SegConvOut.EXT'

- **concat**  
  [Type: Boolean / Default: 'false'] Flag which governs whether all input files should be concatenated into a single output file.

- **informat**  
  [Type: String / Default: 'BSML' / Aliases: infmt] The specified input format for files whose format can not be detected automatically. Note that this is not a way to filter out only files of the desired format. Also, if the format can be determined automatically, it will not be overridden by the given informat value unless you also specify the 'force' flag. Should be one of BSML, GB (GenBank), FASTA, EMBL, SPT (SPTreMBL), SW (SwissProt), RSF, SSF (GCG), and PHY (Phylip).

- **force**  
  [Type: Boolean / Default: 'false'] Forces all input files to read according to the format specified by the 'informat' parameter. If a file doesn't conform to the given format, a warning will be written.

- **preserveannot**  
  [Type: Boolean / Default: 'true' / Aliases: annot] Attempt to preserve all of the data (seq + annotations) rather than just preserve file name and sequence data

- **summary**  
  [Type: Boolean / Default: 'true'] Print a summary of all conversions

- **breakup**  
  [Type: String / Default: EMPTY / Aliases: extract split] Each sequence converted will be saved to its own output file. This option is incompatible with concatenation option
LOCAL DATA FILES

Some programs require nonsequence data. These data are normally read into the program automatically from data files. For instance, the restriction mapping programs all require a file called enzyme.dat, which contains the names of restriction enzymes and their recognition sites.

Local versions of data files are always optional; you are never required to have one unless the default version is, for some reason, not suitable for your specific needs.

For more information about data files, see Appendix VII in this manual.

PARAMETER REFERENCE

Describes parameters that you can set from the command line. In addition to the optional parameters described for each program, most programs respond to the optional global parameters described below.

Non-Plus Programs Parameter Reference:

-Default

is used to tell the program to suppress all interaction and to use the program’s default values for every parameter that is not explicitly defined on the command line.

-INITialize=programname.init

names a file the program should use as a command-line initializing file.

If a command-line initializing file is not specified on the command line, then the file whose name is the same as the program and whose file extension is .init is assumed to be a command-line initializing file by the corresponding GCG program.

The next six parameters can be permanently set with global switches, which are described in “Using Global Switches” in Chapter 3, Using Programs in the User’s Guide. If a command-line parameter conflicts with a global switch, the command-line parameter overrides the global switch.
-DOCLines=10

sets a program to copy any number of file documentation lines you choose. Usually, GCG programs copy the first six, non-blank lines of input file documentation into output files. The global switch % doclines sets your process to act as if this optional parameter were always on the command line.

-FAITHful

sets a program to copy all of the input file’s documentary heading into output files—including blank lines.

-QUIet

sets a GCG program not to ring the terminal bell, even if an error occurs. The global switch % quiet sets your process to act as if this switch were always on the command line.

-STAden

sets a program to accept sequences in Staden format. The global switch % seqformat Staden sets your process to act as if this switch were always on the command line.

-CHEck

prints a summary of the available command-line parameters and prompts you for any additions you might wish to make to the command line. The global switch % comcheck sets your process to always show this summary and display this prompt.

-NODOCumentation

suppresses the short banner that introduces each program. The global switch % nodocumentation lets your process to act as if this switch were always on the command line.

If you have not used GCG graphics programs before, you should look at Chapter 5, Using Graphics in the User’s Guide before continuing with this section.

The options described below apply to all GCG graphics programs. Most plotting programs stop plotting if you use <Ctrl>C at the terminal keyboard. Some of the switches described below do not work with every graphics device. A platen unit (pu) in the descriptions below is one percent of the length of the vertical axis. The GCG platen always has at least 150 horizontal (X) platen units and 100 vertical (Y) platen units.
-FIGure=programname.figure

Writes the plot as a text file of plotting instructions suitable for input to the Figure program instead of drawing the plot on your plotter. The plotting instructions in the text output file can be customized to suit your needs and be plotted out at any time on any graphics device. (See the Figure program in the Program Manual for a description of the plotting instructions in the output file.)

The name of the file can be set by you on the command line or the program makes up a name for it using the name of the program for the file name and .figure for the file name extension.

-NO\text{TEx} or -FAS\text{T}

suppresses all of the text on the plot. This can sometimes make plotting faster on devices where character plotting is slow.

-\text{FONT}=1

draws all text characters on the plot using font 1 (see Appendix I of the Program Manual).

-\text{COLOR}=1

draws the entire plot with the black pen (the pen in stall 1).

-\text{LINEWidth}=0.5

sets the line thickness for all of the lines on the plot to 0.5 platen unit. A platen unit is one percent of the vertical height of the platen. Many devices do not support this option.

-\text{COPlies}=3

makes three copies of each page on some laser printers.

-\text{GRID}=5,2

draws a grid showing the platen units behind the graph. The first optional parameter value sets the grid interval in platen units. If the first value is negative, the numbering along the bottom axis is suppressed. The second optional parameter value sets the grid color.

-\text{BOX}=0,150,0,100,1,2.0,0.2

draws a box or frame on the plot. The first four optional parameter values set the position of the box. The fifth optional value sets the color. The sixth optional value sets the distance between the inner and outer frames. The seventh optional value sets the line thickness of the outer frame (on some devices).
-NOCLIPPING

If the data points on a line fall outside of the window in which the data are supposed to be represented, most programs will clip the graph at the edge of the window. This switch disables that clipping.

-PSINCLUDE=programname.ps

makes a file of PostScript instructions that can be included within a formatted Red document. The Wisconsin Package must be set to use the PostScript graphics device driver. The name of the file can be set by you on the command line or the program makes up a name for it, using the name of the program for the file name and .ps for the file name extension.

-PLT=/dev/tty15

GCG graphics programs direct their output to a port or queue to which the logical name "PlotPort" has been assigned. This parameter lets you direct graphics output to a different port or to a disk file.

-SPEED=8.0

lets you choose a pen speed between 1.0 and 10.0 to achieve higher quality plots, for those x-y plotters that allow pen speed selection, so that you can trade speed for quality. Note that 1.0 is the slowest pen speed available and 10.0 is the fastest; the default is usually 10.0.

-AUTOFEED

advances to the second and all subsequent pages of the plot automatically, on plotters equipped with automatic paper feed. The first page must be loaded in the usual manner. Plotters equipped with automatic page feeding must usually be set up locally to enable this feature. For example, the HP7550 must have the auto-feed button pushed and must have paper in the feed tray.

If your plotter is queued or if you are writing the plotting instructions into a file, then the AUTOFEED parameter is automatically in effect—you do not need to specify it on the command line.

-NOUNLOAD

A few x-y plotters (like the HP7550) let you draw a second picture over the top of an existing plot even if the plotter would normally unload the paper automatically after plotting each page. This option directs such plotters to keep the existing page on the platen so that you can draw the
output from a second session on top of the plot from this session. Queued devices and laser printers do not support this option. In addition, this option doesn’t work with the X Windows driver.

**-PASS through**

sends the plot to a graphics output device attached to the terminal’s printer port. Many devices (for instance, printers without keyboards) are always configured this way if you have them attached to the terminal, and this switch is unnecessary. There are a few plotters, however, that can be attached to your terminal either between the computer and terminal or behind the terminal on the terminal’s pass-through printer port. For such plotters attached behind the terminal, this switch insists that your printer port should be turned on before any instructions are sent to the plotter.

With the parameters described below you can expand or reduce the plot (zoom), move it in either direction (pan), or rotate it 90 degrees (rotate).

**-SCA le=0.5**

allows you to make the plot larger or smaller by setting a scaling factor that is normally 1.0 to some number larger or smaller than 1.0. A scaling factor of 0.5 for each axis causes the outside dimensions of the plot to be half as long, thus making the plot one-fourth as large.

**-XSCAle=0.7 -YSCAle=0.8**

lets you set the scaling factor independently for each axis if you want to change the aspect ratio of the plot. The values used above compress the vertical dimension by 30 percent and the horizontal dimension by 20 percent.

**-PORtrait**

rotates the plot 90 degrees on the page. (GCG plots are reduced or enlarged automatically to fit on the page.)

**-XPAN=30.0**

moves the plot to the right or left. The value 30.0 would move the plot to the right by 30 platen units.

**-YPAN=30.0**

moves the plot up or down. The value 30.0 would move the plot up by 30 platen units.
Plus Programs Parameter Reference:

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-\texttt{-default, -d}

is used to tell the program to suppress all interaction and to use the program’s default values for every parameter that is not explicitly defined on the command line.

-\texttt{-initialize=programname.init, -init}

names a file the program should use as a command-line initializing file. If a command-line initializing file is not specified on the command line, then the file whose name is the same as the program and whose file extension is .init is assumed to be a command-line initializing file by the corresponding GCG program.

The next six parameters can be permanently set with global switches, which are described in “Using Global Switches” in Chapter 3, Using Programs in the User’s Guide. If a command-line parameter conflicts with a global switch, the command-line parameter overrides the global switch.

-\texttt{-doclines=10, -docl}

sets a program to copy any number of file documentation lines you choose. Usually, GCG programs copy the first six, non-blank lines of input file documentation into output files. The global switch % doclines sets your process to act as if this optional parameter were always on the command line.

-\texttt{-quiet, -qui}

sets a GCG program not to ring the terminal bell, even if an error occurs. The global switch % quiet sets your process to act as if this switch were always on the command line.

-\texttt{-check, -che}

prints a summary of the available command-line parameters and prompts you for any additions you might wish to make to the command line. The global switch % comcheck sets your process to always show this summary and display this prompt.
-nodocumentation , -nodoc

suppresses the short banner that introduces each program. The global switch %
nodocumentation sets your process to act as if this switch were always on the
command line.

If you have not used GCG graphics programs before, you should look at Chapter
5, Using Graphics in the User’s Guide before continuing with this section.

The options described below apply to all GCG graphics programs. Most plotting
programs stop plotting if you use <Ctrl>C at the terminal keyboard. Some of the
switches described below do not work with every graphics device. A platen unit
(pu) in the descriptions below is one percent of the length of the vertical axis. The
GCG platen always has at least 150 horizontal (X) platen units and 100 vertical
(Y) platen units.

-plot=/dev/tty15 , -plo

GCG graphics programs direct their output to a port or queue to which the logical
name “PlotPort” has been assigned. This parameter lets you direct graphics
output to a different port or to a disk file.
REFORMAT

FUNCTION

Reformat rewrites sequence file(s), scoring matrix file(s), or enzyme data file(s) so that they can be read by GCG programs.

DESCRIPTION

Reformat rewrites sequence or data files to make them usable by Accelrys GCG (GCG). It can also be used to alter the appearance of single sequence files. The following are some of the manipulations that Reformat can perform:

- Converting single sequence files that were prepared or edited with a text editor into GCG format.
- Converting between multiple sequence (MSF), rich sequence (RSF) and single sequence GCG formats.
- Correcting the sequence type (protein or nucleic acid) of single sequence files that have no type or that were incorrectly typed when they were created.
- Converting nucleic acid sequences between DNA (T, t) and RNA (U, u) representations.
- Converting protein sequences between one-letter and three-letter amino acid representations.
- Converting sequences to all uppercase or all lowercase characters.
- Removing gap characters from sequence files.

In order to use Reformat on single sequence files, the files must contain a heading, a dividing line, and a sequence, as described below. You can use a text editor to make your "foreign" sequence files conform to this arrangement.

HEADING

The heading of a sequence file may contain any number of lines of text at the top of the file to describe the sequence. The heading must not contain two adjacent periods (..) anywhere within it. This area is optional.

DIVIDING LINE

The heading is followed by a dividing line: a line containing two adjacent periods (...). Any information on the line other than the two periods is lost during reformatting. The dividing line may be omitted if there is absolutely no heading. All GCG data files contain a dividing line to separate the data from a documentary heading.

SEQUENCE

After the dividing line comes the sequence in any format you wish. It is conventional to use uppercase letters for known parts of the sequence and lowercase letters for uncertain parts. As in the example below, the sequence may have documentary comments.
embedded within it. You may either use two adjacent slash characters (//) to mark the end of the sequence data or just allow the sequence to go on until the end of the file.

SEQUENCE CHARACTERS

The alphabet of legitimate sequence characters and their meanings are defined in Appendix III. GCG programs support the IUB-IUPAC standard ambiguity codes for the representation of nucleic acid ambiguities and the standard one-letter amino acid codes. Reformat, like all other GCG programs, will ignore all characters that are not in the alphabet of legitimate sequence characters.

EXAMPLE

Here is a session using Reformat to rewrite a sequence file prepared with a text editor (see the INPUT FILE topic below) to GCG format:

```
% reformat

REFORMAT what sequence file(s) ? reformat.txt

reformat.txt  length: 1636 bp

%
```

OUTPUT FILE

Here is part of the output file from the example above:

```
!!NA_SEQUENCE 1.0

Human fetal Beta globin G gamma
from Shen, Slightom and Smithies, Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.

The region below is used to demonstrate REFORMAT. It
starts at base 2101 of the sequence reported in Cell (gamma.seq).

reformat.txt  Length: 1636  September 29, 1998 17:28  Type: N
Check: 398  ..

1  AGGAAGCACC CTTCAGCAGT TCCACA
   <Cap (G gamma)>
   CACT CGCTTCTGGA ACGTCTGAGG

51  TTATCAATAA GCTCCTAGTC CAGACGCC
   >coding (G gamma)>
   AT GGGTCATTTC ACAGAGGAGG

//---------------------------------//

1551  CTTTCAAGGA TAGGCTTTAT TCTGCAAGCA ATACAAATAA TAAATCTATT

1601  CTGCTAAGAG ATCAC
   <POLYA (G gamma)>
```
INPUT FILES

Here is part of the input file used for the example above:

Human fetal Beta globin G gamma
from Shen, Slightom and Smithies, Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.

The region below is used to demonstrate REFORMAT. It
starts at base 2051 of the sequence reported in Cell.

..>

AGGAAGCACC CTTCAGCAGT TCCACA>Cap (G gamma)>CACT CGTT
CTGGA ACGTCTGAGG
TTATCAATAA GCTCCTAGTC CAGACGCC>coding (G gamma)>AT

GCTCAGTGCC CATGATGCAG
AGCTTTCAAG GATAGGCTTT ATTCTGCAAG CAATACAAAT AATAAATCTA
TTCTGCTAAG AGATCAC>POLYA (G gamma)<ACATGGTGTCTTCTGACCTT

RELATED PROGRAMS

SeqEd is a general purpose sequence editor.

All GCG programs that write single sequence files, such as Assemble, BackTranslate,
ExtractPeptide, FromStaden, GetSeq, PepData, PileUp, Reverse, SeqEd, Shuffle, and
Translate, write these files in GCG format.

The programs FromEMBL, FromFastA, FromGenBank, FromIG, FromPIR, and
FromStaden are designed to bring files from six popular formats into GCG format.
These specialized reformattting programs, in addition to reformattting the sequences, also
convert the sequence characters into the nearest IUB-IUPAC equivalent character (see
Appendix III).

ChopUp converts a non-GCG sequence file containing lines longer than 511 characters
and as long as 32,000 characters into a new file containing lines no longer than 50
characters. The new file can be read by Reformat to create a GCG-format sequence file.

BreakUp reads a GCG-format sequence file containing more than 350,000 sequence
characters and writes it as a set of separate, shorter, overlapping sequence files that can
be analyzed by GCG programs.
DataSet creates a GCG data library from any set of sequences in GCG format. GCGToBLAST combines any set of GCG sequences into a database that you can search with BLAST.

SeqConv+ is a utility program that provides batch conversions between different sequence formats. The motivation for the program is to allow an end user to easily convert between file formats to easily import data into Accelrys’s bioinformatics applications. In addition, the converter allows the user to convert our internally used formats (e.g. BSML, RSF) into formats more commonly accepted by third-party tools. The supported file formats will include BSML, GenBank, FastA, EMBL, and RSF.

RESTRICTIONS

A sequence may not contain more than 350,000 sequence characters. BreakUp can convert a GCG-format sequence file containing more than 350,000 sequence characters into a set of separate, shorter overlapping sequence files. Embedded comments more than 125 characters long are truncated to 125 characters. Input lines may not be more than 511 characters. ChopUp can convert a file with lines exceeding 511 characters to a file suitable for input to Reformat.

CONSIDERATIONS

Filename Extensions

Nucleic acid and protein sequences are generally named with the filename extensions .seq and .pep, respectively.

Use Staden Format Directly

The command % seqformat Staden sets your process so that most programs accept input sequences in Staden format without the need for reformatting. The command % seqformat GCG restores the system to expect sequences in GCG format.

You can use Reformat on Staden files (or any files that contain only sequence characters) without modification as long as all the sequence characters in the file belong to the IUB-IUPAC code representation. If your Staden file contains any of Staden’s ambiguity codes, use the FromStaden program instead.

Use FastA Format Directly

The command % seqformat FastA sets your process so that most programs accept input sequences in FastA format without the need for reformatting. The command % seqformat GCG restores the system to expect sequences in GCG format.
Input from stdin

Reformat accepts input from stdin with `-INfile=-`. If the stdin input does not contain a heading that is separated from the sequence by a line containing two dots (..), then also use `-NOHEADING`.

Multiple Sequence Format (MSF) and Rich Sequence Format (RSF) Files

Reformat can be used to convert between MSF, RSF, single sequence format and list files. When single sequence files are specified using a list file, any sequence attributes specified in the list file (e.g. begin and end ranges) are ignored during the conversion to the new file type. When converting from an RSF file any sequence features are lost. Access to sequence features is currently available only from within SeqLab. (Refer to Chapter 2 of the Users' Guide, Using Sequence Files and Databases, for details. See "Using Multiple Sequence Format (MSF) Files" for help in specifying sequences in MSF files, "Using Rich Sequence Format Files" for help with RSF files, and "Using List Files" for information about list files.)

Following are several examples of the commands you might type to convert between MSF or RSF and single sequence format files. These examples use the files hsp70.msf, hsp70.rsf and pretty.list, which can be copied to your local directory with the `% fetch` command.

To copy all of the sequences in hsp70.msf into separate sequence files, use

```bash
% reformat hsp70.msf[*]
```

To copy all of the sequences in hsp70.rsf into separate sequence files, use

```bash
% reformat hsp70.rsf[*]
```

To copy the sequence Hs70_Plafa from hsp70.msf into a single sequence file, use

```bash
% reformat hsp70.msf{hs70_plafa}
```

To convert pretty.list into an RSF file, use

```bash
% reformat -RSF @pretty.list
```

If you edit hsp70.msf with a text editor to manually adjust the alignment, you must use Reformat to rewrite the MSF file so that it can be used with GCG programs by using

```bash
% reformat -MSF hsp70.msf[*]
```

FORMAT CONTROL

For single sequence files and MSF files, you can control the number of sequence characters per line and the number of characters in each block by setting parameters on
the command line. Additionally for single sequence files, you can control how many blank lines appear between sequence lines. Reformat defaults to groups of 10 characters in lines of 50, with one blank line between each sequence line.

CHECKSUM

For each sequence in an MSF, RSF or single sequence file, Reformat calculates a checksum based on the exact sequence. Reformat always adds the checksum to the file containing the sequence. All GCG programs that read sequences recalculate the checksum and compare it to the value written by Reformat to ensure the integrity of the data. If there is disagreement between the newly calculated and previously written checksum values, the program stops and displays an error message. There is one chance in ten thousand that two different sequences would have the same checksum.

EMBEDDED COMMENTS

You may embed comments of up to 125 characters within a sequence in an single sequence file by enclosing them in special comment-delimiting characters. Comments are very helpful for documenting sequences, especially sequences assembled from several sources or sequences containing many genes.

Comment Delimiting Characters

Embedded comments can begin with one of the characters <, >, or $. Each comment must begin and end with the same character.

Suggestions

The embedded comments below seem useful for the sequences we have annotated.

>coding> beginning of coding sequence
<coding< termination of coding sequence
>Cap> cap site
>IVS> intervening sequence donor
<IVS< intervening sequence acceptor
<PolyA< poly-A addition site
>Transcript> beginning of transcript
<Transcript< end of transcript
>Promoter> promoter
>Ribosome> ribosome binding site

Comment Limitations

Comments must start and end with the same delimiting character and may not be more than 125 characters long. Comments that are too long are truncated to 125 characters. Reformat searches through the whole file, if need be, for the second delimiting character that closes the field of a comment. Reformat prints a warning for unclosed comments, but not for comments that are too long.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % reformat [-INfile=]reformat.txt -Default

Prompted Parameters: None

Local Data Files:

-DATa=translate.txt names file of three-letter to one-letter codes

Optional Parameters:

[-OUTfile=]newseqname names the output file
-EXTension=.seq specifies a file name extension for the output
-LISTfile=[reformat.list] writes a list file of output sequence names
-MSF reformats sequences into an MSF output file
-RSF reformats sequences into an RSF output file
-PROtein or -NUCleotide insists that the sequences are reformatted as protein or nucleotide sequences
-DEGap removes gap characters (. and ~) from the sequence
-LINesize=50 sets number of characters per line
-BLOCKsize=10 sets number of characters per block
-BLANKlines=1 puts blank lines between the sequence lines
-NONUMbering suppresses numbering
-NOCOMments suppresses comments
-DNA changes U into T
-RNA changes T into U
-UPPer makes all sequence characters uppercase
-LOWer makes all sequence characters lowercase
-ONEIntothree translates one-letter peptides into three-letter
-THREEintoone translates three-letter peptides into one-letter
-NOHEAding doesn't include header information for input sequence from stdin
-COMparison reformats a scoring matrix instead of a sequence (used with -PROtein or -NUCleotide, insists
that the matrix is reformatted as a protein or nucleotide scoring matrix)
-\texttt{-GAPweight=8} specifies the gap creation penalty associated with the scoring matrix
-\texttt{-LENgthweight=2} specifies the gap extension penalty associated with the scoring matrix
-\texttt{-SCAle=10} multiplies each value in the scoring matrix by 10 (use any number from .01 to 100.0)
-\texttt{-PROtein} or \texttt{-NUCleotide} insists that the sequences are reformatted as
-\texttt{-EQUALSformat} writes the scoring matrix in a form that may be more easily read
-\texttt{-OLDCMPformat} converts a pre-Version 9 scoring matrix into a Version 9 scoring matrix (all options used with \texttt{-COMparison} can also be used with \texttt{-OLDCMPformat}. \texttt{-PROtein} or \texttt{-NUCleotide} must be specified with \texttt{-OLDCMPformat}
-\texttt{-TRANSLate=filename.txt} names the translation table
-\texttt{-NOMONitor} suppresses the screen trace showing each output file

**SCORING MATRICES**

After modifying a scoring matrix, you may want to reformat it to give it a nicer appearance. To use Reformat for this purpose, run the program with $\% \texttt{reformat -COMparison}$. (See Appendix VII for more information about scoring matrices.)

**LOCAL DATA FILES**

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like $\% \texttt{DATa1=myfile.dat}$. For more information see Chapter 4, Using Data Files in the User's Guide.

In the rare event that you are using Reformat to convert a three-letter amino acid sequence into a one-letter sequence, Reformat looks for translate.txt as a local data file.

The translation of codons to amino acids, the identification of potential start codons and stop codons, and the mappings of one-letter to three-letter amino acid codes are all defined in a translation table in the file translate.txt. If the standard genetic code does not apply to your sequence, you can provide a modified version of this file in your working directory or name an alternative file on the command line with an expression like $\% \texttt{TRANSLate=mycode.txt}$. Translation tables are discussed in more detail in Appendix VII.
PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-OUTfile=newseqname

Selects an output filename other than the name of the input file. This option is most useful for single sequence conversions.

-EXTension=.seq

Selects a filename extension other than the input filename extension. This option if most useful for multiple sequence conversions to a list file

-LISTfile=reformat.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then Reformat makes one up using reformat for the file name and .list for the file name extension. If -MSF is on the command line, this parameter is ignored and a list file will not be written.

-MSF

Reformats all input sequences into a multiple sequence format (MSF) output file.

-RSF

Reformats all input sequences into a rich sequence format (RSF) output file.

-PROtein or -NUCleotide

Explicitly reformats the sequences as proteins or nucleic acids.

-DEGap

Removes all gap characters (. and ~) from sequences.

-LINesize=50

Lets you set the number of sequence characters per line to any number between 1 and 120 in MSF and single sequence files.

-BLOCKsize=10

Lets you set the number of sequence characters in each block to any number between 1 and the line size in MSF and single sequence files.
-BLAnklines=1

Leaves zero or more blank lines between the sequence lines in single sequence files.

-NONUMbering

Suppresses the numbering next to each sequence line in single sequence files.

-NOCOMments

Removes any comments from single sequence files.

-DNA

Substitutes T for U and t for u in sequences.

-RNA

Substitutes U for T and u for t in sequences.

-UPPer

Puts all sequence characters into uppercase.

-LOWer

Puts all sequence characters into lowercase.

-ONEIntothree

Converts a protein sequence in one-letter code to three-letter code (see Appendix III). GCG programs use protein sequences in one-letter code only.

-THReeintoone

Converts a protein sequence from three-letter code to one-letter code (see Appendix III). GCG programs use protein sequences in one-letter codes only.

-COMparison

Reformats a scoring matrix.

-GAPweight

Specifies a default gap creation penalty associated with a scoring matrix. This penalty is written in the auxiliary data block of scoring matrix files. If you do not specify a default gap creation penalty with -GAPweight, the program calculates
a reasonable default and writes it in the *auxiliary data block*. (See Appendix VII for information about the *auxiliary data block* in scoring matrix files.)

**-LEN**gtweight

Specifies the default gap extension penalty associated with a scoring matrix. This penalty is written in the *auxiliary data block* of scoring matrix files. If you do not specify a default gap extension penalty with **-LEN**gtweight, the program calculates a reasonable default and writes it in the *auxiliary data block*. (See Appendix VII for information about the *auxiliary data block* in scoring matrix files.)

**-SCA**le=10

Multiplies each value in the scoring matrix and the gap penalties in the *auxiliary data block* by 10. (See Appendix VII for information about the *auxiliary data block* in scoring matrix files.) You can specify any value from 0.01 to 100.0 and each value in the matrix and the gap penalties are multiplied by this number and rounded to the nearest integer.

**-PRO**tein or **-NUC**leotide

Reformats the matrix as either a protein or nucleotide scoring matrix. (See Appendix VII for information about scoring matrix types.)

**-EQUALS** format

Writes the scoring matrix in a format which is less compact but may be more easily read. Files converted with this option are readable by all GCG programs.

**-OLDCMP** format

Converts a pre-version 9 scoring matrix to the version 9 scoring matrix format. By default, each floating point value in the pre-version 9 matrix is first multiplied by 10 and then rounded to the nearest integer. You must add either **-PRO**tein or **-NUC**leotide to specify the type of the converted scoring matrix. (See Appendix VII for information about scoring matrix types.) All of the optional parameters that may be used with **-COM**parison may also be used with **-OLDCMP** format.

**-NOHEA**ding

Expects input sequences from stdin to contain no header information.

**-TRANS**late=filename.txt

Usually, translation is based on the translation table in a default or local data file called translate.txt. This parameter allows you to use a translation table in a different file. (See Appendix VII for information about translation tables.)
-MONitor

This program normally monitors its progress on your screen. However, when you use `default` to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

Printed: February 10, 2006 19:41
**REPEAT**

**FUNCTION**

Repeat finds direct repeats in sequences. You must set the size, stringency, and range within which the repeat must occur; all the repeats of that size or greater are displayed as short alignments.

**DESCRIPTION**

Repeat lets you choose a minimum repeat length (window), a stringency within the window, and a search range and then finds all the repeats of at least that size and stringency within the search range chosen. The repeats are sorted by position and displayed in an output file as alignments of those parts of the sequence that make up the repeats. Repeat tells you the number of repeats found for your settings of window and stringency before filing the results. If you feel there are too many repeats, you may reset the parameters before writing the repeats out to a file. You can limit the number of repeats shown, or sort the repeats by quality so that the longest repeats come at the top of the list. See the ALGORITHM topic below to understand precisely what Repeat does.

**EXAMPLE**

Here is a session using Repeat to find all the direct repeats in the first 1,000 bases of gamma.seq that are 10 bases or longer and that occur within 100 bases of each other and that have at least 9 out of 10 matched bases:

```
% repeat
REPEATs from what sequence ? gamma.seq
Begin (* 1 *) ?
End (* 11375 *) ? 1000
What minimum repeat window (* 7 *) ? 10
What minimum stringency (* 10 *) ? 9
Find repeats through what range (* 50 *) ? 100
There are 11 repeats, would you like to
  1) File the repeats
  4) Set new parameters
Please choose one (* 1 *):
What should I call the output file (* gamma.rpt *)
```

%
OUTPUT

Each repeat is shown as an alignment of the repeated regions along with the beginning and ending coordinates of each region. The size and stringency of each repeat is shown to the right of the alignment. The stringency is the sum of the repeat's pairwise residue values which are found in the scoring matrix. Here is some of the output file for the example above:

REPEAT of: gamma.seq  check: 6474  from: 1  to: 1000

Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies,  Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.

Window: 10  Stringency: 9  Range: 100  Repeats: 11  October 7, 1998 10:29...

79  TGTAATCCCA 88
    || | | | | | | |  
109 10 9

158  TGAAATCCCCA 167
    || || | | | | | |  
170 13 11

213  TGAAATCCCCATCT 225
    || || | | | | | |  
228 13 11

395  ACCAGTCTCT 404
    || | | | | | | |  
404 10 9

444  ACCAGACTCT 453

 /////:////////://:////////

937  AAAAAACAAAA 947
    || | | | | | | | | | | | | 
947 11 10

965  AAAAAATAAAA 975

965  AAAAAATAAAAA 976
    || | | | | | | | | | | | | 
981 12 11

985  AAAAAATAAAGA 996

981  AAAGAAAAA 989
    || | | | | | | | | | | | 
990 9 9

992  AAAGAAAAA 1000

INPUT FILES

Repeat accepts a single sequence file as input. The function of Repeat depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.
RELATED PROGRAMS

Xnu replaces statistically significant tandem repeats in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

Using Compare/DotPlot to create a dot-plot comparison of a sequence to itself is functionally equivalent to running Repeat. The dot-plot is a much more graphic way to show where the repeats occur and what the background of random repeats looks like.

RESTRICTIONS

Repeat cannot find more than 1,000 repeats.

ALGORITHM

For window/stringency comparisons, Repeat reads a scoring matrix that defines a match value for every possible GCG symbol comparison. (See Chapter 4, Using Data Files in the User's Guide for more information.) Repeat then slides the sequence along itself in order to generate every register of comparison (diagonal) for the search range you have set. For each diagonal, Repeat slides a window along the pair of sequences. The match values for each pair of symbols within the window are summed to determine a score at each position. When the score under the window is greater than or equal to the set stringency, then the match criterion has been met and the repeat is recorded.

Repeat Nibbling

Before the repeats are presented, they are nibbled from both ends so that the symbol pair on each end has a scoring matrix value at least as great as the average positive non-identical comparison value in the matrix. You can reset this minimum match threshold with the -PAIR command-line parameter. Thus, repeats less than the minimum repeat length may be shown.

CONSIDERATIONS

Repeat can show several repeats that are part of the same structure if there is a simple sequence with a repeat period shorter than the minimum repeat length.

Repeat chooses a default minimum stringency that is appropriate for the scoring matrix it reads. If you select a different scoring matrix with the -MATRX command-line parameter, the program will adjust the default minimum stringency accordingly.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
REPEAT does not support complete command-line control.

Local Data Files:
- `MATTRix=repeatdna.cmp` assigns the scoring matrix for nucleic acids
- `MATTRix=blosum62.cmp` assigns the scoring matrix for proteins

Optional Parameters:
- `-LIMit` limits the number of repeats written into the output file
- `-SORT` sorts the repeats on quality
- `-PAIr=5` sets match threshold for displaying "|

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like `-DATa1=myfile.dat`. For more information see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name MyData; or 3) name a file on the command line with an expression like `-MATTRix=mymatrix.cmp`. If you do not include a directory specification when you name a file with `-MATTRix`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

By default, Repeat uses the scoring matrix found in either repeatdna.cmp (for nucleotide sequences) or blosum62.cmp (for protein sequences) to find the pairwise match values when determining the stringency of the repeat. You should recognize that stringency is really the sum of the match values (defined in this file) for the symbols compared under the window. The public version of repeatdna.cmp scores a 1 for all IUPAC-IUB nucleic acid ambiguity symbol comparisons where there is ANY overlap between the sets defined by the symbols (see Appendix III). No symbols match the symbols X or N, however. In the public version of blosum62.cmp, the scores for pairwise values for amino acids range from -4 to +11. You can use the Fetch program to create copies of these scoring matrix files in your working directory, where you may modify them to suit your own needs.
You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-MATRix=mymatrix.cmp

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with -MATRix, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

-SORT

Sorts the repeats by quality score instead of position so that the longest repeats (those with the highest quality scores) are at the top of the output.

-LIMIT

Limits the output report to the largest repeats. This parameter automatically causes the repeats to be sorted by quality score instead of position. If you use this parameter, the program asks you to specify how many repeats you want to see.

-PAIR=1

The output from this program has a '|' (vertical bar) between sequence symbols that match. This match display character is added to the output whenever the symbol comparison value for the two symbols in your scoring matrix is greater than or equal to the average positive non-identical comparison value in the matrix. The -PAIR parameter lets you specify a match display threshold appropriate for the scoring matrix you are using.

The repeat nibbling, referred to in the ALGORITHM topic above, uses the threshold value set by this command-line parameter to decide what repeats should be nibbled away from the structure. If you set the pairing threshold too high, all repeats will be nibbled away!

Printed: February 10, 2006 19:41
REVERSE

FUNCTION

Reverse reverses and/or complements a sequence.

DESCRIPTION

Reverse reverses and/or complements the symbols in a sequence. The complements of all of the supported IUPAC-IUB nucleic acid symbols are listed in Appendix III. The output is written into a new sequence file.

EXAMPLE

The file test.seq contains all of the legitimate GCG sequence characters. Here is a session using Reverse to show how they would look on the opposite strand:

```bash
% reverse
REVERSE of what sequence? test.seq

Begin (* 1 *) ?
End (* 389 *) ?

Do you want to:
1) reverse only
2) complement only
3) reverse and complement

Please choose one (* 3 *):

What should I call the output file (* test.rev *)

%

OUTPUT

Here is the output file test.rev:

```
!!NA_SEQUENCE 1.0

REVERSE-COMPLEMENT of: test.seq check: 3365 from: 1 to: 389

This sequence contains every symbol in the alphabet of legitimate GCG sequence characters (Appendix III).

test.rev Length: 389 October 21, 1996 10:15 Type: N Check: 8096 ..
```
Reverse accepts a single sequence as input. The function of Reverse depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

Restrictions

A reversed sequence is renumbered so that the first base corresponds to the last base of the range you chose. It only makes sense to complement nucleotide sequences! If you do not reverse and complement a sequence, you are in danger of having the sequence in 3' to 5' orientation. All GCG programs and all databases assume that nucleotide sequences are in 5' to 3' order, so be careful. Peptide sequences are generally kept in amino-to-carboxyl orientation. Many legitimate sequence symbols are not IUPAC-IUB-supported nucleic acid symbols, so they have no sensible complement (see Appendix III).

Embedded comments are lost.

Input file

Here is the input file test.seq used in the example above:

!!NA_SEQUENCE 1.0
This sequence contains every symbol in the alphabet of legitimate GCG sequence characters (Appendix III).

1  >starts with the codons from appendix iii>
   GCTGCCGAG GGGCXGATGA CAATAACRAY TGTTGCTGYG ATGACGAYGA
51  AGAGGARTTT TTCTTYGGTG GCGGAGGGGG XCATCACCAY ATTATCATAA
101  THAAAAAGAA RTTGTTACTT CTCTACTGT TRCTXYTAYT GYTRYTXATG
151  AATAACAAYC CTCCCCCACCC GXXXCAACAG CARCGTGCAC GCCGCGGAG
201  AAGCGGXAGR MGAMGGMGRM GTXCTTCTTC ATCGAGTAGC TCXAGYWSXA
251  CTACCACAAC GACXGTGTGC GTAGTGTXT GGXXXTATT A CTAYGAAGAG
301  CAACAGSART AATAGTGA RTRATRR  >continues with all
    uppercase sequence characters>
   ABC DEFGHIJKLM NOPQRSTUVWXYZ
351  XYZ.-@&*ab cdefghijkl mnopqrstuvwxyz wxyz@&~.
   <ends with

   all lowercase sequence characters<

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use  \texttt{-CHEc\k} to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you \textit{must} type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: \texttt{% reverse [-INfile=}ggamma.seq -Default

Prompted Parameters:

- \texttt{-BEGIN=}1 -END=1700 sets the range of interest
- \texttt{-NOREVerse} doesn't reverse the strand
- \texttt{-NOCOMPlement} doesn't complement the strand
- \texttt{[-OUTfile=}ggamma.rev names the output file

Local Data Files: None

Optional Parameters: None

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
-NOCOMPlement

Doesn’t complement the strand.

Printed: February 10, 2006 19:41
SEQCONV+

FUNCTION

SeqConv+ is a utility program that provides batch conversions between different sequence formats.

DESCRIPTION

Advantages of Plus “+” Programs:

✓ Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

✓ Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

SeqConv+ rewrites the sequence files into any of the standard sequence formats. The following are some of the operations that SeqConv+ can perform:

- Converting single sequence files that were prepared or edited with a text editor into GCG format.

- Conversion between multiple sequence (MSF), rich sequence (RSF) and single sequence (SSF) GCG formats.

- Inter-conversion between standard sequence file formats- GenBank, Genpept, Swissprot, SP-TrEMBL, and FastA.

- Support to BSML format (Bio-Sequence Markup Language).

- The functions of individual GCG programs such as FromEMBL, FromGenBank, FromFastA, and ToFastA have been incorporated in SeqConv+.

EXAMPLE

Here is an example of using SeqConv+ for converting the Hemolysin Precursor sequence in RSF format into a FastA format sequence.

```bash
% SeqConv+
```

SeqConv+ is a new (batch) sequence conversion utility. The program can convert one or more sequence files into a specified format [BSML, GB (GenBank), FASTA, EMBL, SPT (SPTrEMBL), SW (SwissProt), RSF, SSF (GCG) and MSF]. With multiple files, SeqConv can either convert each file into a separate file or concatenate them all into one file.
SeqConv+ of what sequence(s) ? P15320.RSF

Desired output format (* BSML *) ? FASTA

Written 1 sequence from P15320.RSF to P15320.fa

OUTPUT

>P15320 Hemolysin precursor.
MKNNNFRLSAAGKLAAALAILAASAGAYAAEIVAAANGANGPGVSTAATGACQVVDIVAPN
GNGLSHNQYQDFNVPNQPGAVLNNSEAGLSQLAGQLGANPNLGGREASVILNEV
IGRNPSSLHGQQEIIFMGMAADYVLANPNQSCGQFSINTSSHSSLVGVNWENGPVQLQGYS
TFGNRNLTSNGLTNAGVGLDIAPKIDSREGEIVQDFKQNSGKVTSAAINASGLNRVA
RDGVQNASQMPDLTDSYLGSMQAGRININTAQGSGVNLACSLNAGDELKVKAYDIRS
ESRVDASSNKNGGDNYQNYRGGIYVNDRESSQTLTRTELKGNISLVDNHALHTATDIRGEDIT
LQGKILTLDQQLQKQTQGHTDDWFYWSQYDVTREEREQOLQQAGSTVAASGSAKL
ISTQEDVKKLGNVSDARALSVAARDVHLAGVLVEKDASSERQYRNHTSSKRTGWSNS
DESELKSASRLSEEGLTLLKAGRNVSTQGAKVHAQSLTIDADNQIQVGVQKTANAKVR
DDKTSWGGIGGGDNKNNNSRREISHASELTSGTLRLNGQQQGVTITGSKARGQKGEVTA
THGLRIDNALSSTVVDKIDARTGTAFNITSSSHKHADNYSQSTASLEKSDLNLTSVHSDK
ADVIGSVQVASGGELSVEKTGNINVKAEQRQNIDEQKTVTNYAKEAGDKQYRAGLRIEHTRD
SEKTTRTENSASSLSGVSGLCAEKDSVFSGSKLVADKGDASVSNKVSFAADDKATSN
TEQTASTGIGHYTGGIDKLGSGVEAYEYNNKRTQAQSSKAITSGDVGNLTINA
RDKLTQQGAQHSVGGAYQENAAGVDHLAAADTASTTTTSDKVGNIGANVDYSAVTRPVE
RAVQAKLQDATGVIDRGIGAPNVGLDIAQGQSSEKRSSSSQAVVSqvQAGSIDINA
KGQVRDQGTQYQASKGAVNLTADSHRSEAANRQDEEQSRDTRGSAGVRVYTFTGTDSLTVDA
AKGEGGQDRSNSSASSAQVTTGSIDAAANGINVNVKKDAICYQGTALNGGKTANAGGDIRLDQASDK
QSESRSGFNVKASAKGGFTADSKNFGAGFGGTHNGESSSSSTAVSNISQQQV
ELKAGR
DLTLQGTDVKSQGDVSLSAGNKVALQAAESTQTRKESKLGNIDLGAGSSESKTGNGNL
SAGGAFAKAVNESATERQGATIASDGKVTLANGKDDALHLQGAKVSGGSAA
LEAKNG
GILLESAKNEQHKDNWLSLGIAKANAKGGQTGFNKDAGGKVDPNTGKDTHTLGAQLK
VGVEQQ
DKTTHANTGIGTAGDVTLSNGKDTLRAGVRDADSQGKVGDDLHVEESRKDVENGVKVDV
AGLHSNDFGSISITKLSKVGTQPRAGKVEKLEAGVNKDADATTDDKYSNVAR
LDPQGD
TTGAVSFSAEKKGVTLPATPGKPCQGPLWDRGARTVGGAVKDSTGPAQRQGH
LKVNAD
VVNNNAVQGEQSAIAAGKNGVALQVGGQTQLTGGEIRSGQGKVLELGSQVSQDVN
GQRYQG
GGRVDAAATVGGLLGAAKQSVAGNVFPASHASTQQADAKAGVFSQG

INPUT FILES

The input to SeqConv+ is one or more nucleotide or protein sequences. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*) ; or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.

RELATED PROGRAMS

SeqManip+: SeqManip+ is a utility program that allows the user to perform some manipulations of sequences, including translation, back translation of protein sequences, splitting sequences.

Reformat: Reformat rewrites sequence file(s), scoring matrix file(s), or enzyme data file(s) so that they can be read by GCG programs.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -check to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
Minimal Syntax: % seqconv+ [-infile=]value -Default

Minimal Parameters (case-insensitive):

-infile [Type: List / Default: EMPTY / Aliases: infile1 in]
  Input file specification.

Prompted Parameters (case-insensitive):

-format [Type: String / Default: 'BSML' / Aliases: fmt]
  The desired output format for the files.
  Should be one of BSML, GB (GenBank), FASTA, EMBL, SPT (SPTrEMBL), SW (SwissProt),
  RSF, SSF(GCG) and MSF.

Optional Parameters (case-insensitive):

-check [Type: Boolean / Default: 'false' / Aliases: che help]
  Prints out this usage message.

-default [Type: Boolean / Default: 'false' / Aliases: d def]
  Specifies that sensible default values be used for all parameters where possible.

-documentation [Type: Boolean / Default: 'true' / Aliases: doc]
  Prints banner at program startup.

-quiet [Type: Boolean / Default: 'false' / Aliases: qui]
  Tells application to print only a minimal amount of information.

-outfile [Type: OutFile / Default: EMPTY / Aliases: out]
  File to which all input files are concatenated. A value of '-' means STDOUT.
  Specifying this option also turns on the 'concat' option. Default value is 'SeqConvOut.EXT'.

-concat [Type: Boolean / Default: 'false']
Flag which governs whether all input files should be concatenated into a single output file.

-`informat` [Type: String / Default: 'BSML' / Aliases: infmt]
The specified input format for files whose format can not be detected automatically. Note that this is not a way to filter out only files of the desired format. Also, if the format can be determined automatically, it will not be overridden by the given `informat` value unless you also specify the 'force' flag. Should be one of BSML, GB (GenBank), FASTA, EMBL, SPT (SPTrEMBL), SW (SwissProt), RSF, SSF(GCG), and PHY (Phylip).

-`force` [Type: Boolean / Default: 'false']
Forces all input files to read according to the format specified by the 'informat' parameter. If a file doesn't conform to the given format, a warning will be written.

-`preserveannot` [Type: Boolean / Default: 'true' / Aliases: annot]
Attempt to preserve all of the data (seq + annotations) rather than just preserve file name and sequence data.

-`summary` [Type: Boolean / Default: 'true']
Print a summary of all conversions.

-`breakup` [Type: String / Default: EMPTY / Aliases: extract split] each sequence converted will be saved to its own output file. This option is incompatible with concatenation option.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
-format, -fmt

The desired output format for the files. This should be one of BSML, GB (GenBank), FASTA, EMBL, SPT (SPTrEMBL), SW (SwissProt), RSF, SSF(GCG), and MSF.

-infile, -in, -infile1

Input file specification.

-check, -che, -help

Prints out this usage message.

-default, -def, -d

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.

-quiet, -qui

This parameter is not supported.

 outfile, -out

File to which all input files are concatenated. Specifying this option also turns on the 'concat' option. Default value is 'SeqConvOut.EXT'

-concat

Flag which governs whether all input files should be concatenated into a single output file.

-informat, -infmt

The specified input format for files, whose format can not be detected automatically. Note that this is not a way to filter out only files of the desired format. Also, if the format can be determined automatically, it will not be overridden by the given information value unless you also specify the 'force' flag. This value should be one of BSML, GB (GenBank), FASTA, EMBL, SPT (SPTrEMBL), SW (SwissProt), RSF, SSF(GCG), or PHY (Phylip)
-force

Forces all input files to read according to the format specified by the 'informat' parameter. If a file doesn't conform to the given format, a warning will be written.

-preserveannot, -annot

Attempt to preserve all of the data (seq + annotations) rather than just preserve file name and sequence data

-summary

Writes a summary of the program's completion to the screen. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -summary=false.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.

-breakup, -extract, -split

Each sequence converted will be saved to its own output file. This option shall not be used along with -concat option
SEQSTAT+

FUNCTION

SeqStat+ is a utility program that reads through any number of input sequences and provides general statistics about the files, such as total length, number of sequences, and average length. Additionally it provides some extended information about the sequences depending on their type (protein or nucleotide), such as G+C% content. SeqStat+ provides an option which allows the user to configure what results are produced.

DESCRIPTION

Advantages of Plus “+” Programs:

✓ Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

✓ Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

SeqStat+ supports all input file formats such as BSML, FASTA, GenBank, Swissprot, and EMBL formats. SeqStat+ shall also read from STDIN and GCG-format databases, as well as regular files. When a directory of files is specified as input, SeqStat+ will recursively process all files within that directory as input.

EXAMPLE

%seqstat+

SeqStat+ reads in any number of sequences and prints general statistics about them, such as number of bases, average length, and percent GC.

seqstat+ of what sequence(s) ? gb_na:a16stm2*

<table>
<thead>
<tr>
<th>File</th>
<th>#Bases</th>
<th>#Seq</th>
<th>AvgLen</th>
<th>MinLen</th>
<th>MaxLen</th>
<th>%GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>gb_na:A16STM208</td>
<td>1349</td>
<td>1</td>
<td>1349.0</td>
<td>1349</td>
<td>1349</td>
<td>57.52%</td>
</tr>
<tr>
<td>gb_na:A16STM210</td>
<td>1348</td>
<td>1</td>
<td>1348.0</td>
<td>1348</td>
<td>1348</td>
<td>57.86%</td>
</tr>
<tr>
<td>gb_na:A16STM213</td>
<td>1349</td>
<td>1</td>
<td>1349.0</td>
<td>1349</td>
<td>1349</td>
<td>58.56%</td>
</tr>
<tr>
<td>gb_na:A16STM214</td>
<td>1349</td>
<td>1</td>
<td>1349.0</td>
<td>1349</td>
<td>1349</td>
<td>56.56%</td>
</tr>
<tr>
<td>gb_na:A16STM220</td>
<td>1372</td>
<td>1</td>
<td>1372.0</td>
<td>1372</td>
<td>1372</td>
<td>57.65%</td>
</tr>
<tr>
<td>gb_na:A16STM226</td>
<td>1352</td>
<td>1</td>
<td>1352.0</td>
<td>1352</td>
<td>1352</td>
<td>58.57%</td>
</tr>
<tr>
<td>gb_na:A16STM232</td>
<td>1324</td>
<td>1</td>
<td>1324.0</td>
<td>1324</td>
<td>1324</td>
<td>55.51%</td>
</tr>
<tr>
<td>gb_na:A16STM262</td>
<td>1324</td>
<td>1</td>
<td>1324.0</td>
<td>1324</td>
<td>1324</td>
<td>55.51%</td>
</tr>
</tbody>
</table>

Total          10808  8  1351.0  1342  1372  57.40%
OUTPUT

Output shown above is written to the Console by default. If you include –outfile in your commandline, the output is redirected to the file specified.

As SeqStat+ processes the input, it will keep track of a number of statistics. The statistics for each input file or sequence will be kept and output separately. In addition, the statistics for all sequences processed will be kept and output at the end of processing.

The statistics displayed to the user will be controlled by a format string. This string defines which fields are printed. The available fields are:

File name
Sequence Type (N or P)
Number of residues
Number of sequences
Minimum sequence length
Maximum sequence length
Average sequence length
Median sequence length
G+C% content (nucleotide only)
N% content (nucleotide only)

INPUT FILES

SeqStat+ accepts multiple (one or more) nucleotide or protein sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf{*}; or by using a sequence specification with an asterisk (*) wildcard, for example Genbank:*.*. If SeqStat+ rejects your nucleotide sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RESTRICTIONS

To prevent deeply nested processing, SeqStat+ will report on sequences in a directory but not on sequences in any subdirectory of that directory.
SeqStat+ requires that all sequence from input files will be of the same molecule type but no error message is provided if a mix of type is encountered.

SeqStat+ cannot process multiple sequences in a single file. For example, if an RSF file contains more than one sequence, SeqStat+ fails to read all the sequences in a given file and gives wrong results.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use `-check` to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% seqstat+ [-infile=]value -Default`

Minimal Parameters (case-insensitive):

- `-infile` [Type: List / Default: EMPTY / Aliases: infile1 in]
  Input file specification

Optional Parameters (case-insensitive):

- `-check` [Type: Boolean / Default: 'false' / Aliases: che help]
  Prints out this usage message.

- `-default` [Type: Boolean / Default: 'false' / Aliases: d def] Specifies that sensible default values be used for all parameters where possible.

- `-documentation` [Type: Boolean / Default: 'true' / Aliases: doc] Prints banner at program startup.

- `-quiet` [Type: Boolean / Default: 'false' / Aliases: qui] Tells application to print only a minimal amount of information.

- `-outfile` [Type: OutFile / Default: '-' / Aliases: out] File to which statistics are written. A value of '-' means STDOUT.
-fmtstr [Type: String / Default: 'FBSAIXG' / Aliases: fmt] Format string for statistics. Consists of one or more of the following characters:

- F: File name
- T: File type
- B: Number of bases
- S: Number of sequences
- A: Average sequence length
- I: Minimum sequence length
- X: Maximum sequence length
- G: G+C% content
- N: N% content

PARAMETER REFERENCE

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-check, -che, -help

Prints out the usage summary.

-default, -d, -def

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup (default). Skip banner with: -doc=false

-quiet, -qui

This parameter is not supported.

-in[filename] = filename..., -infile1, -in

This is a space-separated list of input files. If this is the first thing on the command line then the parameter tag can be omitted. If the input file name is "-", then SeqStat+ will read from STDIN. This is a required parameter.

-out[filename] = filename, -out

Specifies that statistics should be written to filename. If filename is omitted or is "-", then statistics are written to STDOUT. The format of the statistics is controlled by the –-fmtstr option. The default value for –outfile is "-".
-fmt[str] = optionletters, -fmt

This option controls what appears in the output and the order of appearance. 
optionletters lists the fields that are to be written. It consists any of the following letters in any order:

- f: File name
- t: File format type
- b: Number of bases
- s: Number of sequences
- i: Minimum sequence length
- x: Maximum sequence length
- a: Average sequence length
- m: Median sequence length
- g: G+C% content
- n: N% content

The default value for fmtstr is “ftbsaix”.

Printed: February 10, 2006 19:41
SEQMANIP+

FUNCTION

SeqManip+ is a utility program that allows users to manipulate sequences in ways that include splitting sequences into a set of overlapping segments, extracting a segment from a sequence, translating or complementing nucleotide sequences and back translating protein sequences. SeqManip+ provides a single program for performing operations similar to those previously done in GCG with Translate, Backtranslate, Reverse, and Breakup.

DESCRIPTION

Advantages of Plus “+” Programs:

✓ Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

✓ Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

Operations that can be invoked from SeqManip+ are:

1. Translate
2. Backtranslate
3. Reverse complement
4. Sample
5. Reverse
6. Complement
7. Extract

EXAMPLE

Here is a session using SeqManip+ to translate the G-gamma gene in gamma.seq into the protein sequence for the human fetal beta globin G gamma.

% seqmanip+ -translate -open=50

SeqManip+ is a utility that accepts DNA or protein sequences to perform single/multiple operations on the selected sequences like extract, sample, translate, backtranslate, reverse and reverse complement.

Manipulate what sequence(s) ? ggamma.seq
Begin (* 1 *) ?

End (-1 for entire sequence) (* -1 *) ?

What should I call the output file (* <sequence_name>.seqmanip+ *) ?

Extracting the region 1 to 1700 from ggamma.seq ...

Writing 1 sequence(s) to output file ggamma.seq.seqmanip+

OUTPUT

Here is the output file ggamma.seq.seqmanip+:

!!RICH_SEQUENCE 1.0

..

{

name  ggamma.seq_c366

descrip    ORF translation of ggamma.seq from : 366 to : 650. [ 95 residues ]

type    PROTEIN

checksum  9400

creation-date 02/17/2005 13:45:23

strand  1

sequence

MGNPKVKAHGKKVLTSGLDAIKHLDDLGTKFAQLSELHCDKLHVDPENFKVSPGDVSALL

PLVSRQLRQLSLIDLSAGCELFEETGVEGETAED

}

INPUT FILES

SeqManip+ accepts multiple (one or more) sequences as input. Input sequences may be either nucleotides or proteins, but only one type of sequence can be analyzed at a time. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf[*]; or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.
Single Sequence Input

If you specify a single sequence on the command line or in response to the first program prompt, and -default is not on the command line, SeqManip+ prompts you for the sequence range. After reading that range, SeqManip+ performs the operation specified in the commandline. If no operation is specified on commandline then SeqManip+ converts the given sequence(s) into an RSF format by default.

Multiple Sequence Input

When you specify multiple sequences, SeqManip+, by default, converts all the sequences in the MSF file into a single RSF format. You can also use a list file to specify multiple sequences.

For more information about list files, see "Using List Files" in Chapter 2, Using Sequence Files and Databases in the User's Guide.

RELATED PROGRAMS

SeqConv+: SeqConv+ is a utility program used for conversion of sequences in to standard sequence formats. (BSML, FastA, SwissProt, and GenBank)

Translate: Translate translates nucleotide sequences into peptide sequences

Sample: Sample extracts sequence fragments randomly from sequence(s). You can set a sampling rate to determine how many fragments Sample extracts

Backtranslate: Backtranslate backtranslates an amino acid sequence into a nucleotide sequence. The output helps you identify areas with fewer ambiguities that might be candidates for synthetic probes.

Reverse: Reverse reverses and/or complements the symbols in a sequence. The output is written into a new sequence file.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -check to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
Minimal Syntax: % seqmanip+ [-infile=value] -Default

Minimal Parameters (case-insensitive):

-infile [Type: InFile / Default: EMPTY / Aliases: infile, infile1, in] The name of the input sequence(s).

Prompted Parameters (case-insensitive):

-begin [Type: Integer / Default: '1' / Aliases: beg] First base to read for each query sequence.

-end [Type: Integer / Default: '-1'] Last base to read for each query sequence.


Optional Parameters (case-insensitive):

-check [Type: Boolean / Default: 'false' / Aliases: che, help] Prints out this usage message.

-default [Type: Boolean / Default: 'false' / Aliases: d, def] Specifies that sensible default values be used for all parameters where possible.

-documentation [Type: Boolean / Default: 'true' / Aliases: doc] Prints banner at program startup.

-quiet [Type: Boolean / Default: 'false' / Aliases: qui] Tells application to print only a minimal amount of information.

-bsml [Type: Boolean / Default: 'false'] Output file will be in bsml format.

-complement [Type: Boolean / Default: 'false' / Aliases: comp compl] Complement all input sequences (NA only)

-revcomp [Type: Boolean / Default: 'false' / Aliases: revcompl] Reverse and complement all input sequences (NA only)

-sample [Type: Double / Default: '100.0'] Percent chance that an input sequence is included in output sequences.

-translate [Type: Boolean / Default: 'false' / Aliases: trans] Translate all input sequences (NA only).

-open [Type: Integer / Default: '20'] Translates open reading frames only if they exceed the specified minimum peptide length. This option works only if '-Translate' is set to True.

-frame [Type: Integer / Default: '1'] Translate in specified reading frame (1, 2, 3, -1, -2, -3).


-backtranslate [Type: Boolean / Default: 'false' / Aliases: back backtrans revtrans] Back translate all input sequences (AA only).


-winlen  [Type: Integer / Default: '0'] Split sequence into windows of no more than winlen bases (0 means no splitting).


-extract  [Type: Boolean / Default: 'true'] Extracts a range of the input sequence. Range is specified using -begin and -end option.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-infile, -infile1, -in

The name of the input sequence(s).

-begin, -beg

First base to read for each query sequence.

-end

Last base to read for each query sequence.
-outfile, -out, -outfile1

Names the output file.

-check, -che, -help

Prints out this usage message.

-default, -def

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.

-quiet, -qui

This parameter is not supported.

-bsml

Output file will be in bsml format.

-reverse, -rev

Reverse all input sequences.

-complement, -comp

Complement all input sequences (NA only).

-revcomp

Reverse and complement all input sequences (NA only).

-sample

Percent chance that an input sequence is included in output sequences.

-translate, -trans

Translate all input sequences (NA only).
-open

Translates open reading frames only if they exceed the specified minimum peptide length. This option works only if `-Translate' is set to True.

-frame

Translate in specified reading frame (1, 2, 3, -1, -2, -3).

-allframes

Translate in all 6 reading frames.

-backtranslate, -backtrans

Back translate all input sequences (AA only).

-listtranstables

Lists available genetic codes.

-transtable

Genetic code to use for (back) translations ("-listtranstables" for listing).

-codonfreq

File specifying codon frequencies for best-guess back translation (AA only).

-ambiguities, -ambig

Use ambiguities in back translated sequences instead of best-guess (AA only).

-winlen

Split sequence into windows of no more than winlen bases (0 means no splitting).

-winoverlap, -winover

Overlap of windows when splitting sequences.

-listfile, -lis

Writes a listfile of the output sequence names.
-extract

Extracts a range of the input sequence. Range is specified using -begin and -end option.

Printed: February 10, 2006 19:41
SAMPLE

FUNCTION

Sample extracts sequence fragments randomly from sequence(s). You can set a sampling rate to determine how many fragments Sample extracts.

DESCRIPTION

Sample is a validation tool we use to extract small random samples of sequence data. It uses a random number generator to extract fragments of constant length randomly from somewhere within a sequence. You can set the length of the fragments extracted. You can set the sampling rate to 1 in 10, for example, to make Sample extract its fragment from every 10th sequence in the set of sequences you have specified.

The output is a set of sequence files, each containing a single fragment. Each file documents where its fragment came from. The current time is used to seed the random number generator, so each run with Sample should yield different results.

If you give Sample a single input sequence, you can choose the range, strand, and output file name. Otherwise, Sample uses the top strand of the whole sequence and names the output file with the sequence name followed by the file name extension .sample. For a single input sequence, you can choose to extract more than one sequence fragment. In this case, the output files are named with the sequence name and the number of the extracted fragment, followed by the file name extension .sample (e.g., ecoompa1.sample, ecoompa2.sample, ...).

EXAMPLE

Here is a session using Sample to extract a sample of 300 base pair fragments from every 100th bacterial sequence in GenBank:

% sample

Sample from what sequence(s) ? Bacterial:*

Extract fragments of what length (* 300 *) ?

Sample one in every how many sequences (* 1 *) ? 100

Gb_BA:AB000222 Len: 2,558
Gb_BA:AB001637 Len: 1,677

///////////////////////////////

GB_BA:YK16SRRN Len: 1,495
GB_BA:ZMOFRK Len: 1,080

SAMPLE complete with:
INPUT sequences: 45,946
Output sequences: 459
Fragment length: 300
   Reversed: 0
   Not Reversed: 459
Sampling Rate: 1 in 100

Output files called: "*.sample"
%

OUTPUT

Each .sample output file would contain a 300 base pair fragment from a bacterial sequence. Here is the first one:

!!NA_SEQUENCE 1.0
(300 bp) SAMPLE of: hihi0043 check: 5029 from: 1 to: 1065
starting at position: 122 ending at position: 421

ID   HIHI0043   standard; DNA; PRO; 1065 BP.
AC   L44687; L42023;
NI   g1004185
DT   04-OCT-1995 (Rel. 45, Created)
DT   04-OCT-1995 (Rel. 45, Last updated, Version 1)
DE   Haemophilus influenzae Rd predicted coding region HI0043
gene, . . .

hihi0043.sample Length: 300 October 5, 1998 13:12 Type: N
Check: 5029 ..

    1 CTCAACTTGA ACAAGCATTG AAACCAAAAT CCAGTTTTAG AAAAACTTTA
    51 TTAAAATTTA CTGCACTTTT ATTTGGCTTG GCGACGGTTG CGCAATCCGT
   101 GCAGTGGATT TGGGATAGCT ATCAAAAACA TCAATGGATT TATCTTGCTT
   151 TTGCTTTAGT CAGTTTGATT ATCATTTTAT TGGGTATTAA AGAGATTATT
   201 TGTGAGTGGC GACGTTTAGT TCGTTTAAAA AAACGTGAGC AATGGCAACA
   251 ACAAAAGTCAG CAGATTGGTT TAGAAAGTGC GGTAAAAAAT GGTGATGTTT

INPUT FILES

Sample accepts a single sequence or multiple sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf{*}; or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.
The function of Sample depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your
sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

Corrupt randomly introduces small numbers of substitutions, insertions, and deletions into nucleotide or protein sequence(s). Shuffle randomizes the order of the symbols in a sequence without changing the composition. SeqEd is an interactive editor for entering and modifying sequences and for assembling parts of existing sequences into new genetic constructs. You can enter sequences from the keyboard or from a digitizer.

RESTRICTIONS

If you give Sample more than one sequence as input, Sample only extracts one fragment from any particular sequence. The sequences chosen are not random. For a sampling rate of 1 in 100, the first sequence after every 100 sequences that is longer than the set fragment length is used to extract a fragment. Contact us if you would like to have Sample sample in some other way.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHeck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % sample [-INfile=]ba:* -Default

Prompted Parameters:

-LENgth=300 extracts fragments of length 300

Prompted Parameters: (for single sequences)

-BEGin=1 -END=11375 sets the range of interest
-REVerse uses the reverse strand (for nucleotides)
-SAMplingrate=100 extracts 100 fragments from the input sequence
[-OUTfile=]gamma.sample names the output file

Prompted Parameters: (multiple sequences only)

-SAMplingrate=100 extracts fragments from 1 in every 100 sequences

Local Data Files: None
Optional Parameters:

-BOTHstrands          selects from both strands (nucleotide sequences only)
-EXTension=.sample    sets the default output file name extension
-LISTfile[=sample.list] writes a list file of output sequence names
-VALIDate             displays details for each sampling action
-NOMONitor             suppresses screen monitor of input sequence names
-NOSUMmary              suppresses the screen summary

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-LENgth=300

Specifies the length of extracted fragments.

-SAMPlingrate=100

Sets the sampling rate for the specified set of sequences to 1 in 100.

-BOTHstrands

Sample normally extracts fragments from nucleotide sequences only from the top strand. With this parameter it will select the fragments randomly from both strands.

-EXTension=.sample

This program normally creates output file names by using the original input file name for the base name and the program name for the name extension. Use this parameter to specify some other file name extension.

-LISTfile=sample.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then Sample makes one up using sample for the file name and .list for the file name extension.
-VALIDate

Displays the location of each sample on your screen (name of sequence sampled, beginning and end coordinates and strand of sample taken, etc.).

-MONitor

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

-SUMmary

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.
SEG

FUNCTION

Seg replaces low complexity regions in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

DESCRIPTION

The Karlin-Altschul statistics that underlie BLAST assume that the probability of finding a residue at any particular position in a sequence is simply proportional to its composition. Low-complexity regions and polymers violate this assumption. Such regions occur frequently in proteins. Query sequences containing low-complexity sequences may give highly significant similarity scores when compared to unrelated low-complexity sequences of similar composition.

Seg uses the method of Wootton & Federhen (Computers and Chemistry 17; 149-163, (1993)) to divide a sequence into regions of high and low complexity. The output is a sequence just like the input sequence except that if low-complexity regions are found, the amino acid characters in these regions are replaced by X's. A BLAST search ignores these X regions.

EXAMPLE

Seg is used to find the regions of low complexity in a human major protein prion precursor.

% seg

SEG of what input sequence(s) ? Pir:ujhu

   Begin (* 1 *) ?
   End (* 253 *) ?

What should I call the output file (* ujhu.seg *) ?

PIR1:UJHU   Len: 253

%

OUTPUT

Each output file contains the input sequence with the amino acid characters in low-complexity regions changed into X's. Here is the output file from the session above.

!!AA_SEQUENCE 1.0
   SEG of: a05017   check: 8781   from: 1 to: 253

P1;UJHU - major prion protein precursor - human
N; Alternate names: 11K amyloid protein; 27-30K sialoglycoprotein; PrP 27-30; PrP 33-35C; scrapie prion protein
C; Species: Homo sapiens (man)
C; Date: 25-Oct-1987 #sequence_revision 12-Apr-1996 #text_change 05-Sep-1997
C; Accession: A24173; A40372; A05017; S14078; I54322; I68597; I58135; I59184; I79633; I79634
R; Kretzschmar, H.A.; Stowring, L.E.; Westaway, D.; Stubblebine, W.H.; Prusiner, S.B.; Dearmond, S.J

a05015.seq Length: 253 October 13, 1998 16:29 Type: P Check: 4122 ..
  1  MANLGCWMLV LFVATWSDLG LCKKRKPQGG WNTGGSRYPG QGSPGGNRYX
  51  XXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXTS\S\XQW
 101  KPSKPKTNMK HMXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXRPIIH FGSDYEDRYY
 151  RENMHRYPQ QVYRPMDEYS NQNNFVHD\XCV NITIKQHXX XXXXXXXXXX
 201  XDVKM\MERVV EQMCITQYER E\S\QA\YQRGS SMVLFSXXX XXXXXXXXXX
 251  XXG

INPUT FILES

You can specify either a single protein sequence or multiple protein sequences as input to Seg. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf{*}; or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.

If Seg rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

Xnu replaces statistically significant tandem repeats in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

RESTRICTIONS

Seg only accepts protein sequences. If you give Seg more than one sequence as input, Seg will not prompt you for begin and end positions or for the output file name.

CONSIDERATIONS

If 20 different characters were distributed randomly, but with equal probability along a sequence, then each character would add 4.322 bits of information to the sequence.
If a sequence contained only one character, then each character would add 0.0 bits of information to the sequence (log(base 2) 1).

The parameters **-LOW**cut and **-HIGH**cut are cutoffs in units of bits/residue that set the "lowness" of complexity of the regions you want to find.

To find all homopolymers of length six or greater, set the window to six and both the cutoffs to zero.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use **-CHECK** to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

**Minimal Syntax:** % seg [-INfile=]pir1:ujhu -Default

**Prompted Parameters:** (for single sequences)

- `-BEGin=1` `-END=253` sets the range of interest
- `-OUTfile=ujhu.seg` names the output file

**Local Data Files:** None

**Optional Parameters:**

- `-BEGin=1` `-END=100` sets the range of interest (for multiple sequences)
- `-WINDOW=12` sets the minimum size of first stage segment
- `-LOWcut=2.2` sets the maximum complexity of a first stage segment
- `-HIGHcut=2.5` sets the maximum complexity of a second stage segment
- `-MINhighlen=0` sets the minimum length of a high-complexity segment
- `-EXTension=.seg` sets the default output file name extension
- `-LISTfile[=seg.list]` writes a list file of output sequence names
- `-NOMONitor` suppresses screen monitor of input sequence names
- `-NOSUMmary` suppresses the screen summary

**LOCAL DATA FILES**

None.
PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-BEGIN=1

Sets the beginning position for all input sequences. When the beginning position is set from the command line, Seg ignores beginning positions specified for individual sequences in a list file.

-END=100

Sets the ending position for all input sequences. When the ending position is set from the command line, Seg ignores ending positions specified for sequences in a list file.

-WINDOW=12

To limit the computing required, Seg starts off by looking for low-complexity regions of size 12 or greater. You can set this window size lower if you want to find shorter low-complexity regions.

-LOWCUT=2.2

In the first stage of the algorithm, Seg identifies segments having a complexity equal to or less than the cutoff in bits/residue set by this parameter (Equation (3) from Wootton and Federhen). The range of acceptable values for this parameter is from 0.0 to 4.322.

This parameter is K(1) in Wootton and Federhen and is referred to as the trigger segment complexity. See the CONSIDERATIONS topic above.

-HIGHCUT=2.5

In the second stage of the algorithm, Seg extends the low-complexity segments found in the first stage into overlapping low-complexity segments that have a complexity equal to or less than the cutoff in bits/residue set by this parameter (Equation (3) from Wootton and Federhen). The complexity of these extension segments can be higher than the complexity for the first stage segments. The range of acceptable values for this parameter is from the lowcut setting to 4.322.

This parameter is K(2) in Wootton and Federhen and is referred to as the extension segment complexity. When this second stage of the algorithm is finished, the resulting extended segment-contigs are referred to as "raw segments". See the CONSIDERATIONS topic above.
The maximal BLAST bit score of a segment pair is 4.322 times the length of the pair. If you are searching with a very short region, even though it may be locally complex, it may not contain enough total information to reach the cutoff score and it will therefore seem to find nothing at all even when there are related sequence segments in the database. This parameter lets you set a minimum acceptable length for a high complexity segment. If Seg finds one shorter than this minimum length, it extends it into adjacent low-complexity segments.

This program normally creates output file names by using the original input file name for the base name and the program name for the name extension. Use this parameter to specify some other file name extension.

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then Seg makes one up using seg for the file name and .list for the file name extension.

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.
SEGMENTS

FUNCTION

Segments aligns and displays the segments of similarity found by WordSearch.

DESCRIPTION

WordSearch uses word comparison, which is very fast, to identify regions of possible similarity between a query sequence and some set of sequences. Segments uses optimal alignment, which is slow but precise, to display the best segment of similarity in the regions identified by WordSearch. WordSearch uses a method similar to the method of Wilbur and Lipman (Proc. Natl. Acad. Sci.(USA) 80; 726-730 (1983)) to find the regions of possible similarity. Segments uses the alignment procedure of Smith and Waterman (Advances in Applied Mathematics 2; 482-489 (1981)) to search for the segments.

Segments uses a scoring matrix, a gap creation penalty, and a gap extension penalty to find the best region of similarity between two sequences. The best region has the highest quality, where quality is the sum of the matches minus the sum of the mismatches minus the sum of the gap creation and extension penalties for the gaps added. The best region must fall within some "width" around the peak diagonal.

EXAMPLE

Here is a session using Segments to align the regions of similarity between a human globin coding sequence and sequences in the GenBank nucleotide sequence database found in the example session for WordSearch:

```
% segments

(BestFit) SEGMENTS from what WORDSEARCH file ? ggammacod.word

What should I call the output file (* ggammacod.pairs *) ?

Aligning ......................-...
GB_PR2:HUMHBGG    545 bp  Gaps:  0  Quality:   4440 / Length: 444
Aligning ......................-...
GB_PAT:I42109    443 bp  Gaps:  0  Quality:   4440 / Length: 444
Aligning .....................-..
GB_PR1:HSGGGPHG    521 bp  Gaps:  0  Quality:   4440 / Length: 444
Aligning .....................-..
GB_PR1:HSGGGPHG    521 bp  Gaps:  0  Quality:   4440 / Length: 444
```

%
Here is part of the output file:

(BestFit) SEGMENTS from: ggammacod.word  October 19, 1998 15:06

(Masked) (Nucleotide) WORDSEARCH of: GenDocData:ggammacod.seq
check: 2906
from: 1 to: 444

ASSEMBLE    July 27, 1994 11:40
Symbols:     1 to: 92    from: gamma.seq  ck: 6474,  2179 to: 2270
Symbols:    93 to: 315   from: gamma.seq  ck: 6474,  2393 to: 2615
Symbols:   316 to: 444   from: gamma.seq  ck: 6474,  3502 to: 3630

Human fetal beta globins G and A gamma . . .

AvMatch: 3.84  AvMisMatch: -6.00  GapWeight: 50  LengthWeight: 3

Match display thresholds for the alignment(s):

| = IDENTITY
.: = 3
.: = 1

ggammacod.seq check: 2906 from: 1 to: 444
GB_PR2:HUMHBGG check: 7917 from: 17 to: 545

M15386 Human hemoglobin gamma-G (HBG2) mRNA, partial cds.
3/97
Gaps: 0  Quality: 4440  Ratio: 10.000  Score: 442  Width: 3
Limits: +/-4

1 ATGGGTCATTTCACAGAGGAGGACAAGGCTACTATCACAAGCCTGTGGGG 50
18 ATGGGTCATTTCACAGAGGAGGACAAGGCTACTATCACAAGCCTGTGGGG 67
51 CAAGGTGAATGTGGAAGATGCTGGAGGAGAAACCCTGGGAAGGCTCCTGG 100
68 CAAGGTGAATGTGGAAGATGCTGGAGGAGAAACCCTGGGAAGGCTCCTGG 117

Limits: +/-4

91 AGGCTCCTGGTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGG 140
2409 AGGCTCCTGGTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGG 2458
141 CAACCTGTCCTCCTGCTGCCATCATGGGCAACCCCCAAGTC2AGGCAAC 190

M. mulatta gamma-globin-1(G), gamma-globin-2(A) genes and ...

Limits: +/-4

91 AGGCTCCTGGTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGG 140
2409 AGGCTCCTGGTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGG 2458
141 CAACCTGTCCTCCTGCTGCCATCATGGGCAACCCCCAAGTC2AGGCAAC 190
INPUT FILES

Segments accepts the output file of WordSearch as input. If any of the search set sequences listed in this file have been changed or deleted, Segments acts as if they do not exist. If the WordSearch query sequence listed in this file no longer exists, Segments complains and stops. Segments also reads the beginning and ending positions of the query sequence in the output file from WordSearch. If Segments cannot read this range, the entry query sequence is used.

RELATED PROGRAMS

Segments is an automated version of the BestFit program run with `-LIMit`, with the limits set to plus and minus `width+1`. The output file of WordSearch is the input file for Segments. Compare/DotPlot and BestFit are more flexible tools for examining the relationship between two sequences when automation is not desired.

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds.

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST. TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?" TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"
FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

**RESTRICTIONS**

The diagonal of comparison cannot be longer than 30,000 and the surface of comparison may not be larger than one million. The surface of comparison can be estimated by multiplying the average length of the two sequences being compared by the sum of the two gap shift limits. (See the ALGORITHM topic below for more information about gap shift limits.) Segments truncates sequences that exceed 30,000 symbols and squeezes the gap shift limits to keep the surface within the one-million limit.

**ALGORITHM**

Segments reads the query sequence and the set of sequences and diagonals in the output list from WordSearch and then executes a limited BestFit on each pair of sequences to make an alignment near that diagonal. For a detailed description, see BestFit (-LIMIT), and imagine that the gap shift limits are both set to width + 1. Width is defined as the width of a structure in the histogram from a word comparison (see the WordSearch program). Width is the fifth column of data in the WordSearch output file.

**CONSIDERATIONS**

There is strong reason to believe that the BestFit algorithm used by Segments is the best way to search for segments of similarity (Lipman and Pearson, "Rapid and Sensitive Protein Similarity Searches," Science 227; 1435-1441 (1985)), but the best parameters to use for Segments are not clear. Like any alignment program, Segments produces alignments that are very different depending on the values assigned for match, mismatch, gap creation penalty, and gap extension penalty. Segments chooses default gap creation and extension penalties that are appropriate for the scoring matrix it reads. If you select a different scoring matrix with -MATRIX, the program will adjust the default gap penalties accordingly. (See Appendix VII for information about how to set the default gap penalties for any scoring matrix.) Similarly, if you have done a simplified word search and adjust the match and mismatch comparison values with -MATCH and -MISMATCH, the program will adjust the default gap penalties accordingly. You can use -GAPweight and -LENgthweight to specify alternative gap penalties if you do not want to accept the default values.

**The Public Scoring Matrix is Quite Stringent**

The public scoring matrix file segdna.cmp scores matches as +10 and mismatches as -6, which means that the segment shown is cut off if there is any significant region where mismatches outnumber matches by about a 2:1 ratio. If the words scored by WordSearch were dispersed along the diagonal, then some of them may not appear in the alignment for that diagonal.
The Alignments Miss Some Words

Segments often fail to display every word scored for the peak diagonal if the words were not tightly grouped along the diagonal. You can use `-WHOle` to get Needleman-Wunsch alignments that traverse the entire length of the diagonal. If you run Compare with `-WORd` and plot the output with DotPlot, you see the exact pattern of word identities between two sequences.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use `-CHEck` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax:  % segments [-INfile=]ggammacod.word -Default

Prompted Parameters:

[-OUTfile=]ggammacod.pairs  names the output file

Local Data Files:

-MATRix=segdna.cmp    assigns the scoring matrix for nucleic acids
-MATRix=blosum62.cmp assigns the scoring matrix for proteins

Optional Parameters:

-GAPweight=50         sets the gap creation penalty
-LENgthweight=3       sets the gap extension penalty
-WHOle                aligns the whole diagonal, not just the best segment
-MATch=10             sets symbol match value for simplified word searches
-MISmatch=-5          sets symbol mismatch value for simplified word searches
-PAIr=x,5,1           thresholds for displaying '|', ':', and '.'
-WIDTH=50             the number of sequence symbols per line
-PAGe=60              adds a line with a form feed every 60 lines
-NOBIGGaps            suppresses abbreviation of large gaps with '.'s
-NOMONitor            suppresses the screen monitor

**LOCAL DATA FILES**

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on
the command line with an expression like `\texttt{-DAT\_a=myfile.dat}`. For more information see Chapter 4, Using Data Files in the User's Guide.

**Local Scoring Matrices**

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program’s default scoring matrix in a public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name MyData; or 3) name a file on the command line with an expression like `\texttt{-MATR\_x=mymatrix.cmp}`. If you do not include a directory specification when you name a file with `\texttt{-MATR\_x}`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

Segments reads comparison values from the scoring matrix file segdna.cmp (nucleic acids) or blosum62.cmp (peptides). If the WordSearch sequences were simplified, Segments would use the same simplification table used by WordSearch to construct a scoring matrix.

Segments run with `\texttt{-WHO\_e}` uses the scoring matrix files seggapdna.cmp for nucleotide sequence comparison instead of segdna.cmp. The scoring matrix for protein sequence comparisons, blosum62.cmp, is unchanged.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

\textbf{-MATR\_x=mymatrix.cmp}

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with `\texttt{-MATR\_x}`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

\textbf{-GAP\_weight=50}

Lets you designate a gap creation penalty if you do not want to use the default penalty. (See the ALGORITHM topic in BestFit for a description of gap creation penalties.)
-**LEN**gthweight=3

Lets you select a gap extension penalty if you do not want to use the default penalty. (See the **ALGORITHM** topic in BestFit for a description of gap extension penalties.)

-**WHO**le

Causes this program to make alignments using the method of Needleman and Wunsch instead of the default method of Smith and Waterman. The difference between these two methods is the same as the difference between the programs Gap and BestFit. The Needleman and Wunsch method displays the whole length of both sequences after alignment, while the Smith and Waterman method shows only the best segment of similarity from each sequence.

-**WHO**le causes Segments to read the local data file seggapdna.cmp for nucleotide sequence comparisons.

-**MAT**ch=10

If you have done a simplified word search, Segments must make up a scoring matrix that looks like your simplification scheme. The matrix normally assigns 10 for all the symbol comparisons you treated as equivalent and \(-20/Alphabet\ size\) for all other symbol comparisons. **-MAT**ch and **-MIS**match allow you to set values other than 10 for matches and \(-20/Alphabet\ size\) for mismatch.

-**MIS**match=-5

See **-MAT**ch for a description of **-MIS**match.

-**PAI**r=4,2,1

The paired output file from this program displays sequence similarity by printing one of three characters between similar sequence symbols: a pipe character(|), a colon (:), or a period (.). Normally a pipe character is put between symbols that are the same, a colon is put between symbols whose comparison value is greater than or equal to the average positive non-identical comparison value in the scoring matrix, and a period is put between symbols whose comparison value is greater than or equal to 1. You can change these match display thresholds from the command line. The three values associated with **-PAI**r are the display thresholds for the pipe character, colon, and period. The match display criterion for a pipe character changes from symbolic identity (the default) to the quantitative threshold you have set in the first parameter. A pipe character will no longer be inserted between identical symbols unless their comparison values are greater than or equal to this threshold. If you still want a pipe character to connect identical symbols, use \(\times\) instead of a number as the first value. (See Appendix VII for more information about scoring matrices.)
-WIDth=50

Puts 50 sequence symbols on each line of the output file. You can set the width to anything from 10 to 150 symbols.

-PAGe=60

Printed output from this program may cross from one page to another in an annoying way. Use this parameter to add form feeds to the output file in order to try to keep clusters of related information together. You can set the number of lines per page by supplying a number after -PAGe.

-NOBIGGaps

Suppresses large gap abbreviations, showing all the sequence characters across from large gaps. Usually, gaps that extend one sequence by more than one complete line of output are abbreviated with three dots arranged in a vertical line.

-MONitor

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

Printed: February 10, 2006 19:41
SeqLab is the graphical user interface for GCG. To run the program, you must have an X Windows compatible display, such as a workstation console or a personal computer running X Windows emulation software.

SeqLab allows you to run most programs in GCG from its menus as well as manage, edit, and annotate your sequences. Extensive online help is available within SeqLab.

For more information on SeqLab refer to the SeqLab Guide.

To run SeqLab, type the following from the command line:

```
% seqlab
```

Note: You may need to first tell the computer running the GCG which X display it should connect to. For example:

```
% setenv DISPLAY mypc.mynetwork.com:0.0
```

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % seqlab

Optional Parameters:

-`-MODE=list` Start in the Main List mode of operation
-`-MODE=editor` Start in the Editor mode of operation

Printed: February 10, 2006 19:41
SEQMERGE

FUNCTION

SeqMerge is GCG’s powerful new fragment assembly application with an X Windows graphical user interface.

DESCRIPTION

SeqMerge allows you to intuitively assemble fragments in a sequencing project into contigs, or alignments of overlapping fragments.

From the contig, SeqMerge creates a consensus sequence representing the underlying sequence from which your fragments were derived.

For more information on SeqMerge refer to the SeqMerge Tutorial.

EXAMPLE

To run SeqMerge, type the following from the command line:

Note: You may need to first tell the computer running the GCG which X display it should connect to. For example:

```
% setenv DISPLAY mypc.mynetwork.com:0.0
% seqmerge
```

Or

```
% setenv DISPLAY mypc.mynetwork.com:0.0
% Seqlab
% Navigate to Functions ➔ Fragment Assembly ➔ SeqMerge and click Run Button to open the SeqMerge application window.
```

COMMAND-LINE SUMMARY

Minimal Syntax: % seqlab

Optional Parameters:

- MODE=list Start in the Main List mode of operation
- MODE=editor Start in the Editor mode of operation

Printed: February 10, 2006 19:41
SHUFFLE

FUNCTION

Shuffle randomizes the order of the symbols in a sequence without changing the composition.

DESCRIPTION

Shuffle uses a random number generator to scramble the positions of the symbols in a sequence. The generator is initialized with the current time, so repeated shuffles should yield different results.

EXAMPLE

Here is a session using Shuffle to randomize the bases of gamma.seq:

```
% shuffle

SHUFFLE of what sequence ? gamma.seq

Begin (* 1 *) ?
End (* 11375 *) ?
Reverse (* No *) ?

What should I call the output file (* gamma.shuffle *) ?

%
```

OUTPUT

The file gamma.shuffle now contains the shuffled contents of gamma.seq.

INPUT FILES

Shuffle accepts a single sequence as input. The function of Shuffle depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

Sample extracts sequence fragments randomly from sequence(s). You can set a sampling rate to determine how many fragments Sample extracts. Corrupt randomly introduces small numbers of substitutions, insertions, and deletions into nucleotide or protein sequence(s).
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use \texttt{--Check} to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you \textit{must} type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: \texttt{\% shuffle [-INfile=]gamma.seq -Default}

Prompted Parameters:

- \texttt{-BEGIN=1 -END=11375} sets the range of interest
- \texttt{-REVerse} uses the reverse strand
- \texttt{[-OUTfile=}gamma.shuffle\texttt{]} names the output file

Local Data Files: None

Optional Parameters:

- \texttt{-PREServe=2} preserves dinucleotide or dipeptide composition in shuffled sequence
- \texttt{-PREServe=3} preserves trinucleotide or tripeptide composition in shuffled sequence
- \texttt{-NONUMbering} suppresses the numbering in the output file

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

\texttt{-PREServe=2}

Preserves the input sequence's dinucleotide or dipeptide composition in the output shuffled sequence. Use \texttt{-PREServe=3} to preserve the trinucleotide or tripeptide composition.

\texttt{-NONUMbering}

Suppresses the numbering in the output file.

Printed: February 10, 2006 19:41
SIMPLIFY

FUNCTION

Simplify lets you reduce the number of symbols in a sequence. Such a simplification would allow you, for instance, to treat all hydrophobic amino acids as equivalent.

DESCRIPTION

Scientists searching for the basic design features in protein sequences believe that there may be functionally similar amino acids that can be substituted without causing radical changes in the function of the protein. Therefore, it may be useful to treat some amino acids as equivalent in peptide sequence comparisons. The simplifications below are from Dr. Miguel A. Jimenez-Montano, who worked with Dr. Hugo Martinez at the University of California in San Francisco, and is now at Univ. de las Americas-Puebla (Mexico). You can determine your own simplification by changing the local data file simplify.txt. Here are the default simplifications in the public data file.

A = P,A,G,S,T (neutral, weakly hydrophobic)
D = Q,N,E,D,B,Z (hydrophilic, acid amine)
H = H,K,R (hydrophilic, basic)
I = L,I,V,M (hydrophobic)
F = F,Y,W (hydrophobic, aromatic)
C = C (cross-link forming)
All other characters are unchanged.

The simplify.txt file in the public data directory is only appropriate for simplifying peptide sequences. You must create your own simplify.txt file to define equivalences for nucleic acid simplifications.

EXAMPLE

Here is a session using Simplify to make a simplification of gzeinaa.pep:

% simplify
SIMPLIFY what sequence(s) ? gzeinaa.pep
  Begin (* 1 *) ? 18
  End (* 285 *) ? 243

What should I call the output file (* gzeinaa.sim *) ?

%

INPUT FILES

Simplify accepts a single sequence or multiple sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by
using a sequence specification with an asterisk (*) wildcard, for example GenBank:*. The function of Simplify depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

CompTable writes a scoring matrix based on the simplifications from a simplification file like simplify.txt. You can assign match and mismatch values.

SIMPLIFICATION FILE

You can use Fetch to make a copy of simplify.txt in your own directory, and then modify it with an editor to suit your own needs. Here is the default version:

```
!!SIMPLIFY 1.0
A standard simplification used by SIMPLIFY and WORDSEARCH to simplify peptide sequences. The first line below means "for all of the P, A, G, S, or T characters in the sequence, substitute A." The program COMPTABLE can construct a symbol comparison table with the equivalences from this file.
10/7/84 ..
A PAGST
D QNEDBZ
H HKR
I LIVM
F FYW
C C
```

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % simplify [-INfile=]ggamma.pep -Default

Prompted Parameters: (for a single sequence)
-BEGIN=1 -END=444 sets the range of interest
[-OUTfile=]ggamma.sim names the output file

Local Data Files:

-DATa=simplify.txt specifies a file of equivalences

Optional Parameters:

-EXTension=.sim sets the default output file name extension

-LISTfile[=simplify.list] writes a list file of output sequence names

-NOMONitor suppresses the screen trace

The default simplification is as follows:

A  =  P,A,G,S,T  (neutral, weakly hydrophobic)
D  =  Q,N,E,D,B,Z  (hydrophilic, acid amine)
H  =  H,K,R  (hydrophilic, basic)
I  =  I,L,V,M  (hydrophobic)
F  =  F,Y,W  (hydrophobic, aromatic)
C  =  C  (cross-link forming)
All other characters are unchanged.

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

Simplify reads the file simplify.txt to find the equivalences you desire. The first letter in each equivalence row is the letter that is substituted for all of the rest of the letters in the row.

The simplify.txt file in the public data directory is only appropriate for simplifying peptide sequences. You must create your own simplify.txt file to define equivalences for nucleic acid simplifications.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-EXTension=.sim

Sets the default file output file name extension.
-**LIST**file=simplify.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then Simplify makes one up using simplify for the file name and .list for the file name extension.

---

-**MONitor**

This program normally monitors its progress on your screen. However, when you use -**Default** to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

Printed: February 10, 2006 19:41
SPSCAN

FUNCTION

SPScan scans protein sequences for the presence of secretory signal peptides (SPs).

DESCRIPTION

SPScan predicts secretory signal peptides (SPs) in protein sequences. For each sequence, SPScan prints a list of possible secretory signal peptides sorted in descending order according to score. Associated with each score is the probability of achieving that score in the target sequence by chance using the given weight matrix. SPScan has weight matrices for eukaryotes, Gram-positive prokaryotes, and Gram-negative prokaryotes.

EXAMPLE

Here is a session with SPScan that was used to find SPs in the apolipoprotein A-I precursor protein sequence from Salmo salar:

```
% spscan

SPScan of what sequence(s)? PIR:Jh0472

Begin (* 1 *) ?
End (* 258 *) ?

Search using weight matrix for which organism type:
   A. Eukaryote
   B. Gram-Positive Prokaryote
   C. Gram-Negative Prokaryote

Please choose one: (* A *):

Only display SPs whose score exceeds (* 7.0 *) ?

What should I call the output file (* jh0472.spscan *) ?

Number of input sequences processed: 1
Number of sequences with predicted SPs: 1
   Output file: jh0472.spscan
   CPU time (sec): 1.01
```

OUTPUT

Here is the output file:
SPScan of PIR:Jh0472  September 29, 1998 15:02

Weight matrix: GenRunData:speuk.dat
Minimum score for SPs (threshold): 7.0

Predicted cleavage sites indicated by '^'.

> sequence: pir2:jh0472
name: jh0472  check: 8711  from: 1  to: 258

1. 1 MKFLVLALTILLAAGTQA^FP 20
Score: 12.2
Probability: 1.455E-03
SP length: 18
McGeoch scan succeeded:
Charged-region statistics:
  Length: 2   Charge: 1
Hydrophobic-region statistics:
  Length: 9   Offset: 3   Total hydropathy: 67.8
  Maximum 8-residue hydropathy: 60.6, starting at 5

Databases searched:
Input sequences searched: 1
Number of sequences with predicted SPs: 1
CPU time (sec): 0.42

SP Representation

The N-terminus->C-terminus direction of the predicted SP is from left to right. The position of the first residue in the SP is shown to the left, and the position of the second residue after the cleavage site is shown to the right. The predicted position of the cleavage site itself is indicated with a caret (^).

SP Data

Each predicted SP displayed in the output is followed by a summary of the information used to make the prediction:

Score gives the score computed using the weight matrix for the predicted SP. This is the maximum score generated from the weight matrix as it is moved over a region no longer than 70 residues downstream from a putative SP start site (an SP start site is either an initiator methionine or the first amino acid residue of the sequence, if the sequence didn't start with a methionine). The region immediately downstream of the putative start site is evaluated for certain characteristics indicative of a SP before the weight matrix was applied to the sequence. If you use -ADJustscores, the score is lessened by an amount proportional to that by which the length of the predicted SP exceeds the suggested maximum for the organism type. All the SPs predicted for a particular sequence are sorted according to this value, with highest scores appearing first.
Unadjusted score, when present, gives the score computed by applying the weight matrix to the predicted SP. This information will appear only when you use -ADJJustscores.

Probability, when present, is the probability of the random occurrence of a score at least as high as the one reported in a sequence with the same amino acid composition as that portion of the target sequence scanned (see ALGORITHM topic below) whose positions are all independent of each other. -EVEn causes SPScan to compute score probabilities based on a sequence with even amino acid residue distribution whose positions are all independent of each other. -NOPROBabilities causes SPScan to forgo the calculation of probability. If you specify -ADJJustscores, the probability always applies to the unadjusted score.

SP length is the length of the predicted SP from the putative SP start site to the residue immediately preceding the site of enzymatic cleavage. Note that the SP sequence display shows an indication of the cleavage site followed by the first two residues after the SP; the final two residues are not included in the SP length because they are not part of the SP.

McGeoch scan reports either "succeeded" or "failed," based on the result of the scan for McGeoch's criteria for a minimum SP. (See ALGORITHM topic below.)

Charged-region statistics are present only if the McGeoch scan succeeds.

Length gives the length of the charged region, or n-region (see ALGORITHM topic below), as measured from the putative SP start site to the distal charged residue. In a typical SP, the charged region is 1 to 5 amino acids in length and carries a positive charge.

Charge gives the total charge of the n-region. The total charge is the sum of the charges of the charged amino acids in the n-region.

Hydrophobic-region statistics are present only if the McGeoch scan succeeds.

Length gives the length of the hydrophobic region, or h-region (see ALGORITHM topic below), as measured from the residue immediately following the distal charged residue of the charged region to the last amino acid in the maximally hydrophobic 8-residue window beginning 8 to 15 residues downstream from the putative SP start site.

Offset gives the position of the first residue in the hydrophobic region of the potential SP relative to the beginning of the predicted SP.

Total hydropathy gives the total Kyte-Doolittle hydropathy of the hydrophobic region.

Maximum 8-residue hydropathy gives the Kyte-Doolittle hydropathy of the maximally hydrophobic 8-residue window in the hydrophobic region (see ALGORITHM topic below). The position of the first residue in this window is indicated. The final residue of this window is the last amino acid of the hydrophobic region.
INPUT FILES

The input to SPScan is one or more protein sequences. If SPScan rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.

RELATED PROGRAMS

Motifs looks for sequence motifs by searching through proteins for the patterns defined in the PROSITE Dictionary of Protein Sites and Patterns. Motifs can display an abstract of the current literature on each of the motifs it finds. FindPatterns identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal. HTHScan scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation. CoilScan locates coiled-coil segments in protein sequences. TransMem scans for likely transmembrane helices in one or more input protein sequences. HTHScan+ scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation. CoilScan+ locates coiled-coil segments in protein sequences. TransMem scans for likely transmembrane helices in one or more input protein sequences.

SPScan+ scans protein sequences for the presence of secretory signal peptides (SPs).

CONSIDERATIONS

Under normal circumstances it is likely that SPScan will predict more than one SP in your sequence. Often one of these will have a score significantly greater than the others. If not, keep the following points in mind when evaluating the results of SPScan (from Nielsen, H. et al. Protein Engineering 10(1); 1-6 (1997)):

- SPs in eukaryotes are very rarely longer than 35 residues in length (40 residues for Gram-negative bacteria, 45 for Gram-positive bacteria). `ADJUSTscores` causes the scores of long predictions to linearly diminish as the predicted SP lengthens beyond those empirical limits.

- SPs shorter than 15 residues are extremely rare in both eukaryotes and prokaryotes. SPScan won't find any SPs shorter than 15 residues in length.

The probability value attached to each score, being a measure of the probability of achieving that score or higher by chance with the given weight matrix and target sequence, is extremely useful to use when evaluating SP predictions. A probability close to 0.0 indicates that achieving the score purely by chance is very unlikely, and that you can have more confidence in the SP prediction. Probabilities closer to 1.0 indicate that it's likely that you have gotten the score by chance alone, making the SP prediction more dubious.
Ambiguity codes (such as B or Z) in protein sequences contribute exactly 0 to the score of the sequence window within which they are found. Therefore, the scores and probabilities associated with any predicted motifs from such a sequence window are likely to differ to varying extents from what they would be otherwise. You shouldn't routinely encounter this problem because ambiguity codes are extremely rare in protein sequences.

The "McGeoch scan" information is included in the results to help you decide whether predicted SPs are real when their scores are only marginal or when the probability of achieving those scores seems rather high. The McGeoch scan looks at the upstream part of the predicted SP, beginning with the putative initiator methionine, to determine whether the sequence meets McGeoch's criteria for a minimum acceptable SP (see the ALGORITHM topic below). If a low-scoring SP fails the McGeoch scan, it may be a false positive prediction; if the McGeoch scan succeeds, that SP might merit a closer look.

Because of the way SPScan sorts and stores predicted SPs during scanning, no particular ordering is guaranteed among SPs that have exactly the same score (see the ALGORITHM topic below).

ALGORITHM

SPScan uses the weight matrix method of von Heijne (von Heijne, G. Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit. (1987)), in concert with McGeoch's description of a minimum acceptable SP (McGeoch, D. Virus Research 3; 271-286 (1985)) to predict secretory signal peptides within a protein sequence.

von Heijne's weight matrix method is widely used for detecting SPs in protein sequences. However, this method can misclassify non-functional SPs resulting from events like point mutations. To help reduce false positive predictions, SPScan also determines whether potential SPs meet McGeoch's criteria. SP predictions which fail to meet these criteria are more dubious in general than those that do.

The following is a brief description of how these methods are used to make SP predictions.

Each input protein sequence is scanned from beginning to end. The first residue in each sequence is always examined as a potential SP starting point; subsequently, only methionine residues are considered as potential SP starting points.

At each potential SP starting point, SPScan first checks to see whether McGeoch's criteria for a minimum SP are met. SPScan looks for what von Heijne refers to as an n-region and what McGeoch calls the charged region or CR. This is a window of 11 or fewer residues (including the potential starting residue) containing at least one charged amino acid residue (the charged amino acids are arginine, lysine, asparagine, and glutamic acid). In a real SP, the charged region usually has a charge in the range -1 to +2. If a charged residue is not found, the potential SP has failed to meet McGeoch's criteria.
If a charged region is found, the distal charged amino acid residue is taken as the end of the charged region. The scan continues downstream for an 8-residue window within 15 residues of the end of the charged region. This is referred to by von Heijne as the h-region, and by McGeoch as the uncharged region or UR. To qualify as an uncharged region, the maximally hydrophobic 8-residue window within this 15-residue range should have a hydrophobicity on the Kyte-Doolittle scale of at least 15. If a good uncharged region is found, we take the end of that maximally hydrophobic 8-residue window to be the end of the uncharged region and the potential SP is deemed to have met the McGeoch criteria. The potential SP will be evaluated using von Heijne's weight matrix method in the next stage of the scan. If a good h-region is not found, the potential SP has failed to meet McGeoch's criteria.

The potential SP is then subjected to scanning using von Heijne's weight matrix method. The weight matrix is applied beginning with the potential starting residue for the SP, and scanning continues residue by residue until a region 70 residues long has been examined (very few SPs will be longer than 70 residues in eukaryotes or prokaryotes). The cleavage site predicted by the weight matrix application yielding the highest score is reported. The score reported for a predicted SP is just the von Heijne weight matrix score; the result of the scan for the McGeoch criteria is not reflected in that score, but is simply reported as success or failure.

The statistical significance of each score is computed as the probability of random occurrence of that score in a sequence with the same amino acid residue distribution as that portion of the target sequence scanned and whose positions are all independent of each other (Claverie, J.-M. and Audic, S. CABIOS 12(5); 431-439 (1996)).

The weight matrices used to compute scores for potential SPs are from data given in Nielsen, H. et al. Protein Engineering 10(1); 1-6 (1997). There are matrices for eukaryotes, Gram-positive prokaryotes, and Gram-negative prokaryotes.

There is no guarantee of the relative ordering between predicted SPs having exactly the same score. For example, as we scan from the beginning of the sequence to the end, if the first two SPs encountered each have the score 3.7, SPScan may list the second SP before the first in the final report.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % spscan [-INfile=]pir:jh0472 -Default

Prompted Parameters:

-BEGin=1 -END=258 sets the range of interest
-THRESHold=7.0
specifies minimum score for SP detection

[-OUTfile=]jh0472.spscan
specifies name of results file

Local Data Files:

-DA1a=speuk.dat
assigns the weight matrix for eukaryotic SPs
-DA1a=spgpos.dat
assigns the weight matrix for Gram-positive
prokaryotic SPs
-DA1a=spgneg.dat
assigns the weight matrix for Gram-negative
prokaryotic SPs

Optional Parameters:

-NUMTOPscores=3
specifies maximum number of SPs to report
-GRAMPositive
uses Gram-positive prokaryote weight matrix
-GRAMNegative
uses Gram-negative prokaryote weight matrix
-ADJjustscores
reduces scores of very long SPs
-EVEN
assumes even target residue distribution
-NOPROBabilities
doesn't compute score probabilities
-VERbose
uses verbose output
-RSF[=spscan.rsf]
saves predicted SPs as features in the RSF
file
-MONitor
displays screen trace of progress
-NOSUMmary
suppresses screen summary at the end of the
program

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program
automatically reads them from a public data directory unless you either 1) have a data
file with exactly the same name in your current working directory; or 2) name a file on
the command line with an expression like -DATa1=myfile.dat. For more
information see Chapter 4, Using Data Files in the User's Guide.

If you use -GRAMPositive, SPScan will use the weight matrix file for Gram-positive
prokaryotes called spgpos.dat. If you use -GRAMNegative, SPScan will use the
weight matrix file for Gram-negative prokaryotes called spgneg.dat. The default
behavior is to use the weight matrix file for detecting eukaryotic SPs, speuk.dat.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information,
see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-THRESHold=7.0
Sets the minimum score for secretory signal peptide detection.

-NUMTOPscores=3
Allows you to specify the maximum number of predicted SPs to report for each
sequence scanned. For example, if you specify -NUMTOPscores=3, SPScan
will display no more than three of the highest scoring SPs predicted for each sequence. Use \texttt{-NUMTOP} \texttt{scores}=1 if you want to see only the highest-scoring SP in each sequence. By default, SPScan will display all SPs whose scores meet or exceed the threshold.

\texttt{-GRAMPositive}

Tells SPScan to use the Gram-positive prokaryote weight matrix described in Nielsen, H. et al. Protein Engineering \textbf{10}(1); 1-6 (1997). The default weight matrix is the one for eukaryotes described in the same paper.

\texttt{-GRAMNegative}

Tells SPScan to use the Gram-negative prokaryote weight matrix described in Nielsen, H. et al. Protein Engineering \textbf{10}(1); 1-6 (1997). The default weight matrix is the one for eukaryotes described in the same paper.

\texttt{-ADJJustscores}

Tells SPScan to reduce each computed score by an amount proportional to the difference between the length of the predicted SP and the empirical "maximum" length of SPs for the appropriate organism type (eukaryote, Gram-positive prokaryote, or Gram-negative prokaryote). These maxima are not absolute limits, but are described in Nielsen, H. et al. (Protein Engineering \textbf{10}(1); 1-6 (1997)) as being the length beyond which genuine SPs appear only very rarely. Use this option to cause predicted SPs that are probably too long to be real to be printed later in the sorted list of predictions.

\texttt{-EVEN}

Tells SPScan to assume that amino acid residues are distributed evenly throughout the length of the target sequence for the purpose of calculating score probabilities. This makes SPScan run a little faster, because it does not have to compute the actual distribution of residues in each input sequence, but reliability of the score probability calculations may be adversely effected.

\texttt{-NPROBabilities}

Tells SPScan to forgo the calculation of the probability of random occurrence of the score in a sequence with even amino acid residue distribution whose positions are all independent of each other. This makes SPScan run much faster.

\texttt{-VERbose}

Tells SPScan to print more documentation about each sequence to the output file. The number of lines of documentation printed depends upon the value of the \% \texttt{DocLines} global switch described in "Using Global Switches" in Chapter 3, Using Programs in the User's Guide.
-RSF=spscan.rsf

Writes an RSF (rich sequence format) file containing the input sequences annotated with features generated from the results of SPScan. This RSF file is suitable for input to other GCG programs that support RSF files. In particular, you can use SeqLab to view this features annotation graphically. If you do not specify a file name with this parameter, then the program creates one using spscan for the file basename and .rsf for the extension. For more information on RSF files, see "Using Rich Sequence Format (RSF) Files" in Chapter 2 of the User's Guide. Or, see "Rich Sequence Format (RSF) Files" in Appendix C of the SeqLab Guide.

-MONitor=10

Monitors this program's progress on your screen. Use this parameter to see this same monitor in the log file for a batch process. If the monitor is slowing down the program because your terminal is connected to a slow modem, suppress it with -NOMONitor.

The monitor is updated every time the program processes 10 sequences or files. You can use a value after the parameter to set this monitoring interval to some other number.

-SUMmary

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.
SPSCAN+

FUNCTION

SPScan+ scans protein sequences for the presence of secretory signal peptides (SPs).

DESCRIPTION

Advantages of Plus “+” Programs:

- Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

- Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

SPScan+ predicts secretory signal peptides (SPs) in protein sequences. For each sequence, SPScan+ prints a list of possible secretor signal peptides sorted in descending order according to score. Associated with each score is the probability of achieving that score in the target sequence by chance using the given weight matrix. SPScan+ has weight matrices for eukaryotes, Gram-positive prokaryotes, and Gram-negative prokaryotes.

EXAMPLE

Here is a session with SPScan+ that was used to find SPs in the apolipiprotein A-I precursor protein sequence from *Salmo salar*:

```
%spscan+
SPScan+ scans protein sequences for the presence of secretory signal peptides
SPScan with what sequence(s) ? pir:jh0472
Begin (* 1 *) ? 1
End (-1 for entire sequence) (* -1 *) ?
Only display SPs whose score exceeds (* 7.0 *) ?
What should I call the output file (* <sequence_name>.spscan+ *) ?
jh0472.spscan+

SPScan of pir:jh0472  December 03, 2004 14:56

Weight matrix: SHARE_MATRIX:speuk.dat
Minimum score for SPs (threshold): 7.0

Predicted cleavage sites indicated by '^'.
```
Analyzing sequence 'JH0472' from 'pir2:JH0472'
Processing results...

Input sequences processed              : 1
Number of sequences with predicted SPs : 1
Output File                            : jh0472.spscan+

Results written to jh0472.spscan+

OUTPUT

Here is the output file:

> sequence: pir2:JH0472
  name: JH0472  check: 8711  from: 1  to: 258

  1. 1 MKFLVLALTILLAAGTQA^FP 20
  Score: 12.2
  Probability: 1.455E-03
  SP length: 18
  McGeoch scan succeeded:
    Charged-region statistics:
      Length: 2  Charge: 1
    Hydrophobic-region statistics:
      Length: 9  Offset: 3  Total hydropathy: 67.8
      Maximum 8-residue hydropathy: 60.6, starting at 5

*** SUMMARY ***

Input sequences processed              : 1
Number of sequences with predicted SPs : 1

INPUT FILES

The input to SPScan+ is one or more protein sequences. If SPScan+ rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf[*]; or by using a sequence specification with an asterisk (*) wildcard, for example Genbank:*.

RELATED PROGRAMS

Motifs looks for sequence motifs by searching through proteins for the patterns defined in the PROSITE Dictionary of Protein Sites and Patterns. Motifs can display an abstract of the current literature on each of the motifs it finds.
FindPatterns+ identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal.

HTHScan+ scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

CoilScan+ locates coiled-coil segments in protein sequences. TransMem scans for likely transmembrane helices in one or more input protein sequences.

SPScan scans protein sequences for the presence of secretory signal peptides (SPs).

CONSIDERATIONS

Under normal circumstances it is likely that SPScan+ will predict more than one SP in your sequence. Often one of these will have a score significantly greater than the others. If not, keep the following points in mind when evaluating the results of SPScan+ (from Nielsen, H. et al. Protein Engineering 10(1); 1-6 (1997)):

- SPs in eukaryotes are very rarely longer than 35 residues in length (40 residues for Gram-negative bacteria, 45 for Gram-positive bacteria). -adjustscores causes the scores of long predictions to linearly diminish as the predicted SP lengthens beyond those empirical limits.

- SPs shorter than 15 residues are extremely rare in both eukaryotes and prokaryotes. SPScan+ won't find any SPs shorter than 15 residues in length.

The probability value attached to each score, being a measure of the probability of achieving that score or higher by chance with the given weight matrix and target sequence, is extremely useful to use when evaluating SP predictions. A probability close to 0.0 indicates that achieving the score purely by chance is very unlikely, and that you can have more confidence in the SP prediction. Probabilities closer to 1.0 indicate that it's likely that you have gotten the score by chance alone, making the SP prediction more dubious.

Ambiguity codes (such as B or Z) in protein sequences contribute exactly 0 to the score of the sequence window within which they are found. Therefore, the scores and probabilities associated with any predicted motifs from such a sequence window are
likely to differ to varying extents from what they would be otherwise. You shouldn't routinely encounter this problem because ambiguity codes are extremely rare in protein sequences.

The "McGeoch scan" information is included in the results to help you decide whether predicted SPs are real when their scores are only marginal or when the probability of achieving those scores seems rather high. The McGeoch scan looks at the upstream part of the predicted SP, beginning with the putative initiator methionine, to determine whether the sequence meets McGeoch's criteria for a minimum acceptable SP (see the ALGORITHM topic below). If a low-scoring SP fails the McGeoch scan, it may be a false positive prediction; if the McGeoch scan succeeds, that SP might merit a closer look.

Because of the way SPScan+ sorts and stores predicted SPs during scanning, no particular ordering is guaranteed among SPs that have exactly the same score (see the ALGORITHM topic below).

ALGORITHM

SPScan+ uses the weight matrix method of von Heijne (von Heijne, G. Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit. (1987)), in concert with McGeoch's description of a minimum acceptable SP (McGeoch, D. Virus Research 3; 271-286 (1985)) to predict secretor signal peptides within a protein sequence.

Von’s weight matrix method is widely used for detecting SPs in protein sequences. However, this method can misclassify non-functional SPs resulting from events like point mutations. To help reduce false positive predictions, SPScan+ also determines whether potential SPs meet McGeoch's criteria. SP predictions which fail to meet these criteria are more dubious in general than those that do.

The following is a brief description of how these methods are used to make SP predictions.

Each input protein sequence is scanned from beginning to end. The first residue in each sequence is always examined as a potential SP starting point; subsequently, only methionine residues are considered as potential SP starting points.

At each potential SP starting point, SPScan+ first checks to see whether McGeoch's criteria for a minimum SP are met. SPScan+ looks for what von Heijne refers to as an n-region and what McGeoch calls the charged region or CR. This is a window of 11 or fewer residues (including the potential starting residue) containing at least one charged amino acid residue (the charged amino acids are arginine, lysine, asparagine, and glutamic acid). In a real SP, the charged region usually has a charge in the range -1 to +2. If a charged residue is not found, the potential SP has failed to meet McGeoch's criteria.

If a charged region is found, the distal charged amino acid residue is taken as the end of the charged region. The scan continues downstream for an 8-residue window within 15 residues of the end of the charged region. This is referred to by von Heijne as the h-region, and by McGeoch as the uncharged region or UR. To qualify as an uncharged
region, the maximally hydrophobic 8-residue window within this 15-residue range should have hydrophobicity on the Kyte-Doolittle scale of at least 15. If a good uncharged region is found, we take the end of that maximally hydrophobic 8-residue window to be the end of the uncharged region and the potential SP is deemed to have met the McGeoch criteria. The potential SP will be evaluated using von Heijne's weight matrix method in the next stage of the scan. If a good h-region is not found, the potential SP has failed to meet McGeoch's criteria.

The potential SP is then subjected to scanning using von Heijne's weight matrix method. The weight matrix is applied beginning with the potential starting residue for the SP, and scanning continues residue by residue until a region 70 residues long has been examined (very few SPs will be longer than 70 residues in eukaryotes or prokaryotes). The cleavage site predicted by the weight matrix application yielding the highest score is reported. The score reported for a predicted SP is just the von Heijne weight matrix score; the result of the scan for the McGeoch criteria is not reflected in that score, but is simply reported as success or failure.

The statistical significance of each score is computed as the probability of random occurrence of that score in a sequence with the same amino acid residue distribution as that portion of the target sequence scanned and whose positions are all independent of each other (Claverie, J.-M. and Audic, S. CABIOS 12(5); 431-439 (1996)).

The weight matrices used to compute scores for potential SPs are from data given in Nielsen, H. et al. Protein Engineering 10(1); 1-6 (1997). There are matrices for eukaryotes, Gram-positive prokaryotes, and Gram-negative prokaryotes.

There is no guarantee of the relative ordering between predicted SPs having exactly the same score. For example, as we scan from the beginning of the sequence to the end, if the first two SPs encountered each have the score 3.7, SPScan+ may list the second SP before the first in the final report.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use -check to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

SPScan+ scans protein sequences for the presence of secretory signal peptides.

Minimal Syntax: % spscan+ [-infile=]value -Default
Minimal Parameters (case-insensitive):

-infile [Type: InFile / Default: EMPTY / Aliases: infile in]  
The name of the input file.

Prompted Parameters (case-insensitive):

-begin [Type: Integer / Default: '1' / Aliases: beg]  
First base of interest for each query sequence.

-end [Type: Integer / Default: '-1']  
Last base of interest for each query sequence.

-threshold [Type: Double / Default: '7.0' / Aliases: thresh]  
Sets minimum score for SP detection.

-outfile [Type: OutFile / Default: '<sequence_name>.spscan+' / Aliases: out outfile1]  
Names the output file.

Optional Parameters (case-insensitive):

-check [Type: Boolean / Default: 'false' / Aliases: che help]  
Prints out this usage message.

-default [Type: Boolean / Default: 'false' / Aliases: d def]  
Specifies that sensible default values be used for all parameters where possible.

-documentation [Type: Boolean / Default: 'true' / Aliases: doc]  
Prints banner at program startup.

-quiet [Type: Boolean / Default: 'false' / Aliases: qui]  
Tells application to print only a minimal amount of information.

-grampositive [Type: Boolean / Default: 'false' / Aliases: gramp]  
Uses Gram-positive prokaryote weight matrix.

-gramnegative [Type: Boolean / Default: 'false' / Aliases: gramn]  
Uses Gram-negative prokaryote weight matrix.

-adjustscores [Type: Boolean / Default: 'false' / Aliases: adj]  
Reduces scores of very long SPs.

-data [Type: String / Default: EMPTY / Aliases: dat]  
Assigns weight matrix.

-seqout [Type: OutFile / Default: EMPTY / Aliases: rsf]  
Annotated sequence output.

-numtopscores [Type: Integer / Default: '-1' / Aliases: numtop maxhits]  
Specifies maximum number of SPs to report.

-even [Type: Boolean / Default: 'false' / Aliases:]  
Assumes even target residue distribution.
-probabilities  [Type: Boolean / Default: 'true' / Aliases: prob]  
Compute score probabilities.

-verbose  [Type: Boolean / Default: 'false' / Aliases: ver]  
Print more documentation about each sequence to the output file.

-monitor  [Type: Boolean / Default: 'false' / Aliases: mon]  
Displays screen trace of progress.

-summary  [Type: Boolean / Default: 'true' / Aliases: sum]  
Displays screen summary at end of the program.

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -data1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

If you use -grampositive, SPScan+ will use the weight matrix file for Gram-positive prokaryotes called spgpos.dat ($GCGROOT/share/matrix). If you use -gramnegative, SPScan+ will use the weight matrix file for Gram-negative prokaryotes called $GCGROOT/share/matrix/spneg.dat. The default behavior is to use the weight matrix file for detecting eukaryotic SPs, speuk.dat.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-infile, -infile1, -in

The name of the input file.

-begin, -beg

First base of interest for each query sequence.

-end

Last base of interest for each query sequence.

-outfile, -out, -outfile1

Names the output file.
-check, -che, -help

Prints out this usage message.

-default, -d, -def

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.

-quiet, -qui

This parameter is not supported.

-threshold=7.0, -thresh

Sets the minimum score for secretor signal peptide detection.

-data, -dat

Assigns weight matrix.

-seqout, -rsf

Annotated sequence output.

-numtopscores=3, -numtop

Allows you to specify the maximum number of predicted SPs to report for each sequence scanned. For example, if you specify -numtopscores=3, SPScan+ will display no more than three of the highest scoring SPs predicted for each sequence. Use -numtopscores=1 if you want to see only the highest-scoring SP in each sequence. By default, SPScan+ will display all SPs whose scores meet or exceed the threshold.

-grampositive, -gramp

Tells SPScan+ to use the Gram-positive prokaryote weight matrix described in Nielsen, H. et al. Protein Engineering 10(1); 1-6 (1997). The default weight matrix is the one for eukaryotes described in the same paper.

-gramnegative, -gramn

Tells SPScan+ to use the Gram-negative prokaryote weight matrix described in Nielsen, H. et al. Protein Engineering 10(1); 1-6 (1997). The default weight matrix is the one for eukaryotes described in the same paper.
-adjustscores, -adj

Tells SPScan+ to reduce each computed score by an amount proportional to the
difference between the length of the predicted SP and the empirical "maximum"
length of SPs for the appropriate organism type (eukaryote, Gram-positive
prokaryote, or Gram-negative prokaryote). These maxima are not absolute limits,
but are described in Nielsen, H. et al. (Protein Engineering 10(1); 1-6 (1997)) as
being the length beyond which genuine SPs appear only very rarely. Use this
option to cause predicted SPs that are probably too long to be real to be printed
later in the sorted list of predictions.

-even, -eve

Tells SPScan+ to assume that amino acid residues are distributed evenly
throughout the length of the target sequence for the purpose of calculating score
probabilities. This makes SPScan+ run a little faster, because it does not have to
compute the actual distribution of residues in each input sequence, but reliability
of the score probability calculations may be adversely effected.

-probabilities, -prob

Compute score probabilities.

-verbose, -ver

Tells SPScan+ to print more documentation about each sequence to the output
file. The number of lines of documentation printed depends upon the value of the
% DocLines global switch described in "Using Global Switches" in Chapter 3,
Using Programs in the User's Guide.

-monitor=10, -mon

Program monitors its progress on your screen by displaying a screen trace of
progress. However, when you use -default to suppress all program interaction,
you also suppress the monitor. You can turn it back on with this parameter. If you
are running the program in batch, the monitor will appear in the log file.

The monitor is updated every time the program processes 10 sequences or files.
You can use a value after the parameter to set this monitoring interval to some
other number.

-summary, -sum

Writes a summary of the program's completion to the screen. A summary
typically displays at the end of a program run interactively. You can suppress the
summary for a program run interactively with -summary=false.

You can also use this parameter to cause a summary of the program's work to be
written in the log file of a program run in batch.
**SSEARCH**

**FUNCTION**

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

**DESCRIPTION**

SSearch uses William Pearson's implementation of the method of Smith and Waterman (Advances in Applied Mathematics 2: 482-489 (1981)) to search for similarities between one sequence (the *query*) and any group of sequences of the same type (nucleic acid or protein) as the query sequence.

**EXAMPLE**

Here is a session using SSearch to identify sequences in the PIR protein sequence database that are similar to a human globin protein sequence:

```bash
% ssearch
SSEARCH with what query sequence ? ggamma.pep
Removing terminal * from query sequence...
    Begin (* 1 *) ?
    End (* 147 *) ?
Search for query in what sequence(s) (* PIR:* *) ?
Do not show scores whose E() value exceeds: (* 10.0 *):
What should I call the output file (* ggamma.ssearch *) ?

1 Sequences        105 aa searched    PIRl:CCHU
501 Sequences      93,217 aa searched    PIRl:IHQFT

OUTPUT

The output from SSearch is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

Here is some of the output file:

!!SEQUENCE_LIST 1.0

(Peptide) SSEARCH of: ggamma.pep from: 1 to: 147 October 16, 1998 12:08

TRANSLATE of: gamma.seq check: 6474 from: 2179 to: 2270
  and of: gamma.seq check: 6474 from: 2393 to: 2615
  and of: gamma.seq check: 6474 from: 3502 to: 3630
generated symbols 1 to: 148.
Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies, Cell 26; 191-203. . . .

TO: PIR:* Sequences: 109,075 Symbols: 34,814,664

Databases searched:

Scoring matrix: GenRunData:Blosum50.Cmp
Variable pamfactor used
Gap creation penalty: 12 Gap extension penalty: 2

Histogram Key:
Each histogram symbol represents 179 search set sequences
Each inset symbol represents 17 search set sequences
z-scores computed from opt scores

<table>
<thead>
<tr>
<th>z-score obs</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>(=)</td>
<td>(*)</td>
</tr>
<tr>
<td>&lt; 20</td>
<td>879</td>
</tr>
<tr>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>15</td>
</tr>
<tr>
<td>26</td>
<td>21</td>
</tr>
<tr>
<td>28</td>
<td>63</td>
</tr>
<tr>
<td>30</td>
<td>153</td>
</tr>
<tr>
<td>32</td>
<td>433</td>
</tr>
<tr>
<td>34</td>
<td>1130</td>
</tr>
<tr>
<td>36</td>
<td>2412</td>
</tr>
<tr>
<td>38</td>
<td>4595</td>
</tr>
<tr>
<td>40</td>
<td>6860</td>
</tr>
<tr>
<td>42</td>
<td>8728</td>
</tr>
<tr>
<td>44</td>
<td>10204</td>
</tr>
<tr>
<td>9988</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>10709</td>
</tr>
<tr>
<td>10173</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>10286</td>
</tr>
<tr>
<td>9740</td>
<td></td>
</tr>
</tbody>
</table>

122 Program Manual
Smith-Waterman (PGopt): reg.-scaled

The best scores are:                  s-w    z-sc
E(108303)..

PIR1:HGCZG
! hemoglobin gamma-G chain - chimpanzee  971  1317.6  1.5e-66

PIR1:I37025
! hemoglobin gamma-G chain - gorilla    971  1317.6  1.5e-66

PIR1:HGHUG
! hemoglobin gamma-G chain - human      971  1317.6  1.5e-66

\\End of List
PIR1:HGCZG – hemoglobin gamma-G chain – chimpanzee

N: Alternate names: hemoglobin gamma-1 chain

C: Species: Pan troglodytes (chimpanzee)

C: Date: 31-May-1996 #sequence_revision 21-Jan-1997 #text_change 14-Nov-1997

C: Accession: I36939; I61853


SCORES

z-score: 1317.6

E(): 1.5e-66

Smith-Waterman score: 971;

100.0% identity in 147 aa overlap

SCORES

z-score: 1317.6

E(): 1.5e-66

Smith-Waterman score: 971;

100.0% identity in 147 aa overlap

10        20        30        40        50

ggamma.pep

MGHFTEEDKATITSLWGKVNVEDAGGETLGRLLVVYPWTQRFFDSFGNLSSASAIMGNPK

HGCZG

MGHFTEEDKATITSLWGKVNVEDAGGETLGRLLVVYPWTQRFFDSFGNLSSASAIMGNPK

60

ggamma.pep

VKAHGKKVLTLGDAIKHLDLKGTFAQLSELHCDEKLLHDVFENFKLGLNLVTVLAIFG

HGCZG

VKAHGKKVLTLGDAIKHLDLKGTFAQLSELHCDEKLLHDVFENFKLGLNLVTVLAIFG

120

ggamma.pep

KEFTPEVQASWQKMTGVASALSSRYH

HGCZG

KEFTPEVQASWQKMTGVASALSSRYH

130

140

///////////////////////////////////////////////////////////////////////
//

! Distributed over 1 thread.
! Completion time: Fri Oct 16 12:08:38 1998

! CPU time used:
!     Database scan:  0:10:19.8
!     Post-scan processing:  0:00:04.6
!     Total CPU time:  0:10:24.6
! Output File: ggamma.ssearch
What is the Output?

The first part of the output file contains a histogram showing the distribution of the z-scores between the query and search set sequences. (See the ALGORITHM topic for an explanation of z-score.) The histogram is composed of bins of size 2 that are labeled according to the higher score for that bin (the leftmost column of the histogram). For example, the bin labeled 24 stores the number of sequence pairs that had scores of 23 or 24.

The next two columns of the histogram list the number of z-scores that fell within each bin. The second column lists the number of z-scores observed in the search and the third column lists the number of z-scores that were expected.

The body of the histogram displays a graphical representation of the score distributions. Equal signs (=) indicate the number of scores of that magnitude that were observed during the search, while asterisks (*) plot the number of scores of that magnitude that were expected.

At the bottom of the histogram is a list of some of the parameters pertaining to the search.

Below the histogram, SSearch displays a listing of the best scores. Strand: after the sequence name in this list indicates that the match was found between search set sequence and the reverse complement of the query sequence.

Following the list of best scores, SSearch displays the alignments of the regions of best overlap between the query and search sequences. /rev following the query sequence name indicates that the search sequence is aligned with the reverse complement of the query sequence.

This program displays only the region of overlap between the two aligned sequences (plus some residues on either side of the region to provide context for the alignment) unless you use -SHOWall. The display of identities and conservative replacements between the aligned sequences depends on the value of -MARKx. By default ( -MARKx=3), the pipe character (|) is used to denote identities and the colon (:) to denote conservative replacements.

INPUT FILES

SSearch accepts a single protein sequence or a single nucleic acid sequence as the query sequence. The search set is either a single sequence or multiple sequences of the same type as the query. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf{*}; or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.

The function of SSearch depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above
the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

**RELATED PROGRAMS**

SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST+ and FastA+, it can be very slow.

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.
ProfileSearch and MotifSearch use a profile (derived from a set of aligned sequences) instead of a query sequence to search a collection of sequences. FindPatterns uses a pattern described by a regular expression to search a collection of sequences. HmmerSearch uses a profile hidden Markov model as a query to search a sequence database to find sequences similar to the family from which the profile HMM was built. Profile HMMs can be created using HmmerBuild.

StringSearch, LookUp, and Names identify sequences by searching the annotation (non-sequence) portions of sequence files or sequence databases.

FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like TFastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX+ translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

**REstrictions**

The query sequence cannot be longer than 32,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The sequence type (nucleic acid or protein) of the query sequence and the search set sequences must match.

For the estimates of statistical significance to be valid, the search set must contain a large sample of unrelated sequences. The statistical estimates will not be calculated at all if there are fewer than 10 sequences in the search set (20 sequences if only one strand is searched).
ALGORITHM

SSearch uses William Pearson's implementation of the method of Smith and Waterman (Advances in Applied Mathematics 2; 482-489 (1981)) to search for similarities between one sequence (the query) and any group of sequences of the same type (nucleic acid or protein) as the query sequence. This method uses a scoring matrix (containing match/mismatch scores), a gap creation penalty, and a gap extension penalty as scoring criteria to determine the best region of local similarity between a pair of sequences. This score is reported as the Smith-Waterman score.

After the Smith-Waterman score for a pairwise alignment is determined, SSearch uses a simple linear regression against the natural log of the search set sequence length to calculate a normalized z-score for the sequence pair. (See William R. Pearson, Protein Science 4; 1145-1160 (1995) for an explanation of how this z-score is calculated.)

The distribution of the z-scores tends to closely approximate an extreme-value distribution; using this distribution, the program can estimate the number of sequences that would be expected to produce, purely by chance, a z-score greater than or equal to the z-score obtained in the search. This is reported as the E() score.

When all of the search set sequences have been compared to the query, the list of best scores is printed. If alignments were requested, the alignments are also printed.

In evaluating the E() scores, the following rules of thumb can be used: for searches of a protein database of 10,000 sequences, sequences with E() less than 0.01 are almost always found to be homologous. Sequences with E() between 1 and 10 frequently turn out to be related as well.

CONSIDERATIONS

GCG version of SSearch searches using both strands of nucleic acid queries unless you use -ONE strand. The SSEARCH program distributed with Dr. Pearson's FASTA package searches with one strand only.

The E() scores are affected by similarities in sequence composition between the query sequence and the search set sequence. Unrelated sequences may have "significant" scores because of composition bias.

If there is a database entry that overlaps your query in several places, but there are large gaps between the matching regions, only the best overlap appears in the alignment display.

There are two ways to control the size of the list of best scores. By default, scores are listed until a specific E() value is reached. You may set the value in response to the program prompt or by using -EXPect; otherwise the program uses 10.0 for protein searches, 2.0 for nucleic acid searches. (If you are running the program interactively, it will show no more than 40 scores initially, and ask if you want to see more scores if there are any more that are less than the E() value.)
If you use `-LISTsize`, the E() value is ignored, and the program will list the number of scores you requested.

You can control the number of alignments using `-NOALIGN` and `-ALIGN`. The program behaves differently depending on whether it is being run noninteractively (in batch or with `-default` on the command line) or interactively. In the noninteractive case, the program displays the number of alignments set by `-ALIGN`. (If this is not present, it shows 40 alignments or the number of scores that were listed, whichever is smaller.) If you run the program interactively, it displays the list of best scores, and then asks you how many alignments you want to see. (This prompt does not appear if you use `-NOALIGN` or `-ALIGN`.)

**Adjusting Gap Creation and Extension Penalties**

Unlike other GCG programs, SSearch does not read the default gap creation and gap extension penalties from the scoring matrix file. It uses default gap creation and extension penalties that were empirically determined to be appropriate for the default scoring matrices. If you select a different scoring matrix with `-MATRIX`, you may need to change the gap penalties. The histogram display gives a qualitative view of the quality of fit between the actual distribution of scores and the expected distribution of scores. This information may indicate whether or not suitable gap creation and extension penalties were used for the search. When the histogram shows poor agreement between the actual distribution and the theoretical distribution, you might consider using `-GAPweight` and/or `-LENGTHweight` to specify higher gap creation and extension penalties, respectively. For example, you might increase the gap creation penalty from 12 to 16 and the gap extension penalty from 2 to 4.

**Differences in Applying Gap Extension Penalties**

There are two different philosophies on how to penalize gaps in an alignment. One way is to penalize a gap by the gap creation penalty plus the extension penalty times the length of the gap (`gapweight + (lengthweight x gap length)`). The other way is to use the gap creation penalty plus the extension penalty times the gap length excluding the first residue in the gap (`gapweight + (lengthweight x (gap length - 1))`).

"Native" GCG programs, such as Framesearch and Bestfit, handle gap extension penalties the first way, while the FastA-family programs use the second way. Therefore a value for `-LENGTHweight` that gives good results with one of the FastA-family programs may not give equivalent results with a native GCG program, and vice versa.

**Increasing Program Speed Using Multithreading**

This program is multithreaded. It has the potential to run faster on a machine equipped with multiple processors because different parts of the analysis can be run in parallel on different processors. By default, the program assumes you have one processor, so the analysis is performed using one thread. You can use `-PROCESSORS` to increase the number of threads up to the number of physical processors on the computer.
Under ideal conditions, the increase in speed is roughly linear with the number of processors used. But conditions are rarely ideal. If your computer is heavily used, competition for the processors can reduce the program's performance. In such an environment, try to run multithreaded programs during times when the load on the system is light.

As the number of threads increases, the amount of memory required increases substantially. You may need to ask your system administrator to increase the memory quota for your account if you want to use more than two threads.

Never use \texttt{-PROCessors} to set the number of threads higher than the number of physical processors that the machine has -- it does not increase program performance, but instead uses up a lot of memory needlessly and makes it harder for other users on the system to get processor time. Ask your system administrator how many processors your computer has if you aren't sure.

\textbf{SUGGESTIONS}

\textbf{Identifying the Search Set}

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" topic of Chapter 2, Using Sequence Files and Databases of the User's Guide for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. For information about naming groups of sequences for the search set, see the topics "Specifying Files" and "Using Wildcards" in Chapter 1, Getting Started, and "Using Database Sequences," "Using Multiple Sequence Format (MSF) Files", "Using Rich Sequence Format (RSF) Files", and "Using List Files" in Chapter 2, Using Sequence Files and Databases of the User's Guide.

\textbf{Batch Queue}

SSearch is one of the few programs in GCG that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using \texttt{-BATch}. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

\textbf{Interrupting a Search: \texttt{<Ctrl>C}}

You can type \texttt{<Ctrl>C} to interrupt a search and see the results from the part of the search that has already been completed. Because the program is multithreaded, the search may not be interrupted immediately, but will continue until one of the threads finishes processing its data and returns for more data.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % ssearch [-INfile1=]ggamma.pep -Default

Prompted Parameters:

[-INfile2=]pir:* specifies the search set
[-OUTfile=]ggamma.ssearch names the output file
-BEGIN=1 -END=148 sets the range of interest
-EXPect=2.0 lists scores until E() value reaches 2.0

Local Data Files:

-MATRIX=fastadna.cmp assigns the scoring matrix for nucleic acids
-MATRIX=blosum50.cmp assigns the scoring matrix for proteins

Optional Parameters:

-PROCessors=2 sets the number of threads devoted to the analysis on a multiprocessor computer
-MINLength=1000 searches only sequences of 1000 or more residues
-MAXLength=5000 searches only sequences of 5000 or fewer residues
-SINce=6.90 limits search to sequences dated on or after June 1990
-ONEstrand searches using only the top strand of nucleotide queries
-GAPweight=16 sets the gap creation penalty (12 is protein default)
-LENgthweight=4 sets the gap extension penalty (2 is protein default)
-LISTsize=40 shows the best 40 scores (overrides EXPect)
-ALIgn=20 shows the best 20 alignments
-NOALIgn suppresses sequence alignments
-SHOWall shows complete sequences in alignment, not just overlaps
-MARKx=3 sets the alignment display mode
-NOHIStogram suppresses printing the histogram
-LINesize=60 sets number of sequence symbols per line of the alignment
-NODOCLines suppresses sequence documentation in the alignment
-BATch submits the program to run in the batch queue
LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like `-DAT[a1]=myfile.dat`. For more information see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name MyData; or 3) name a file on the command line with an expression like `-MATR[aix]=mymatrix.cmp`. If you do not include a directory specification when you name a file with `-MATR[aix]`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

SSearch reads a scoring matrix containing the values for every possible match from your working directory or the public database. The files fastadna.cmp (for nucleic acid sequences) and blosum50.cmp (for protein sequences) contain the default values for matches. blosum50.cmp is a BLOSUM50 matrix. You can use the Fetch program to obtain a copy of these files in order to modify them to suit your own needs.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

```
-MATR[aix]=mymatrix.cmp
```

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with `-MATR[aix]`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.
-**EXPect**=2.0

Shows all scores whose E() value is less than 2.0. Ignored if -**LISTsize** is used.

-**PROCessors**=2

Tells the program to use 2 threads for the database search on a multiprocessor computer.

-**MINLength**=1000

Restricts the search to search set sequences that are equal to or longer than 1000 residues.

-**MAXLength**=5000

Restricts the search to search set sequences that are equal to or shorter than 5000 residues.

-**SINce**=6.1990

Limits the search to sequences that have been entered into the database or modified since June 1990. As this is being written, only the EMBL, GenBank, and SWISS-PROT databases support this parameter.

-**ONEstrand**

Searches using only the top strand of a nucleotide query sequence.

-**GAPweight**=12

Specifies the gap creation penalty that is subtracted from the alignment score whenever a gap is created.

-**LEN**g**th**weight=2

Specifies the gap extension penalty that is subtracted from the alignment score for each residue added to an existing gap.

-**LISTsize**=40

Shows the best 40 scores. Overrides -**EXPect**.

-**ALIgn**=10

Limits the number of alignments to display in the output file to the 10 best matches in the list. Use the -**NOALIgn** to suppress the sequence alignments in the output file.
-**SHOWall**

Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-**MARKx=3**

Determines the alignment display mode -- especially the symbols that identify matches and mismatches. The default value, 3, uses a pipe character (|) to show identities and a colon (:) to show conservative replacements. -**MARKx=0** uses a colon to show identities and a period (.) to show conservative replacements. -**MARKx=1** will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X. If -**MARKx=2**, the residues in the second sequence are shown only if they differ from the first sequence.

Use -**MARKx=10** to get aligned sequences in the FastA "parsable" output format. A document describing this format appears after FastA in the Program Manual.

-**NOHISogram**

Suppresses printing the histogram.

-**LINesize=60**

Lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-**NODOCLines**

Suppresses the documentation from the search set sequence accompanying the alignment in the output file. Use -**DOCLines=5** to copy only five non-blank lines of documentation.

-**BATch**

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

-**MONitor=500**

Monitors this program's progress on your screen. Use this parameter to see this same monitor in the log file for a batch process. If the monitor is slowing down
the program because your terminal is connected to a slow modem, suppress it
with \texttt{-NOMONitor}.

The monitor is updated every time the program processes 500 sequences or files.
You can use a value after the parameter to set this monitoring interval to some
other number.
SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST+ and FastA+, it can be very slow.

Advantages of Plus “+” Programs:

- Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

- Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

SSearch+ uses William Pearson's implementation of the method of Smith and Waterman (Advances in Applied Mathematics 2; 482-489 (1981)) to search for similarities between one sequence (the query) and any group of sequences of the same type (nucleic acid or protein) as the query sequence.

Here is a session using SSearch+ to identify sequences in the GenBank rodent sequence database that are similar to an actinomycetes protein sequence:

```
% ssearch+
 ssearch+ with what query sequence(s) ? a16stm210.gb_ba
 Begin (* 1 *) ?
 End (-1 for entire sequence) (* -1 *) ?
 Enter value for search set (*Default DB*) ? Genbank:ro*
 What should I call the output file (*
 <sequence_name>,<program_name> *) ?
 %
```

The output from SSearch+ is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

Here is some of the output file:

```
# $GCGROOT/bin/SSearch+34_native -O /var/tmp/bslskBAA16aGBY.tmp -E 2.0
- b 10 -T 1 /var/tmp/bslskBAA36aGBY.fa "Genbank:ro* 17"
```
SSEARCH+ searches a sequence database using the Smith-Waterman algorithm version 3.4t21 May 14, 2003
Please cite:

Query library /var/tmp/bslskDAA36aGBY.fa vs Genbank:ro* library searching Genbank:ro* 17 library

1>>>A16STM210 Actinomyces species 16S ribosomal RNA (isolate TM210).
- 1348 nt
vs Genbank:ro* library

<table>
<thead>
<tr>
<th>opt</th>
<th>E()</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 20</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>2: *</td>
</tr>
<tr>
<td>34</td>
<td>6</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>9</td>
</tr>
<tr>
<td>42</td>
<td>23</td>
</tr>
<tr>
<td>44</td>
<td>22</td>
</tr>
<tr>
<td>46</td>
<td>41</td>
</tr>
<tr>
<td>48</td>
<td>27</td>
</tr>
<tr>
<td>50</td>
<td>47</td>
</tr>
<tr>
<td>52</td>
<td>28</td>
</tr>
<tr>
<td>54</td>
<td>41</td>
</tr>
<tr>
<td>56</td>
<td>27</td>
</tr>
<tr>
<td>58</td>
<td>18</td>
</tr>
<tr>
<td>60</td>
<td>11</td>
</tr>
<tr>
<td>62</td>
<td>8</td>
</tr>
<tr>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>66</td>
<td>10</td>
</tr>
<tr>
<td>68</td>
<td>8</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
</tr>
<tr>
<td>72</td>
<td>2</td>
</tr>
<tr>
<td>74</td>
<td>4</td>
</tr>
<tr>
<td>76</td>
<td>2</td>
</tr>
<tr>
<td>78</td>
<td>2</td>
</tr>
<tr>
<td>80</td>
<td>5</td>
</tr>
<tr>
<td>82</td>
<td>1</td>
</tr>
<tr>
<td>84</td>
<td>1</td>
</tr>
<tr>
<td>86</td>
<td>1</td>
</tr>
<tr>
<td>88</td>
<td>2</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>94</td>
<td>1</td>
</tr>
<tr>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>98</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>102</td>
<td>0</td>
</tr>
</tbody>
</table>
The first part of the output file contains a histogram showing the distribution of the \( z \)-scores between the query and search set sequences. (See the ALGORITHM topic for an explanation of \( z \)-score.) The histogram is composed of bins of size 2 that are labeled according to the higher score for that bin (the leftmost column of the histogram). For
example, the bin labeled 24 stores the number of sequence pairs that had scores of 23 or 24.

The next two columns of the histogram list the number of z-scores that fell within each bin. The second column lists the number of z-scores observed in the search and the third column lists the number of z-scores that were expected.

The body of the histogram displays a graphical representation of the score distributions. Equal signs (=) indicate the number of scores of that magnitude that were observed during the search, while asterisks (*) plot the number of scores of that magnitude that were expected.

At the bottom of the histogram is a list of some of the parameters pertaining to the search.

Below the histogram, SSearch+ displays a listing of the best scores. Strand: - after the sequence name in this list indicates that the match was found between search set sequence and the reverse complement of the query sequence.

Following the list of best scores, SSearch+ displays the alignments of the regions of best overlap between the query and search sequences. /rev following the query sequence name indicates that the search sequence is aligned with the reverse complement of the query sequence.

This program displays only the region of overlap between the two aligned sequences (plus some residues on either side of the region to provide context for the alignment) unless you use -showall. The display of identities and conservative replacements between the aligned sequences depends on the value of -markx. By default (-markx=3), the pipe character (|) is used to denote identities and the colon (:) to denote conservative replacements.

**INPUT FILES**

SSearch+ accepts a single protein sequence or a single nucleic acid sequence as the query sequence. The search set is either a single sequence or multiple sequences of the same type as the query. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example Genbank:*. The function of SSearch+ depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

**RELATED PROGRAMS**

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may
be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like TFastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX+ translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

ProfileSearch and MotifSearch use a profile (derived from a set of aligned sequences) instead of a query sequence to search a collection of sequences. FindPatterns+ uses a pattern described by a regular expression to search a collection of sequences. HmmerSearch uses a profile hidden Markov model as a query to search a sequence database to find sequences similar to the family from which the profile HMM was built. Profile HMMs can be created using HmmerBuild.
StringSearch, LookUp, and Names identify sequences by searching the annotation (non-sequence) portions of sequence files or sequence databases.

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

**RESTRICTIONS**

The query sequence cannot be longer than 20,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The sequence type (nucleic acid or protein) of the query sequence and the search set sequences must match.

For the estimates of statistical significance to be valid, the search set must contain a large sample of unrelated sequences. The statistical estimates will not be calculated at all if there are fewer than 10 sequences in the search set (20 sequences if only one strand is searched).

Fast suite of programs work with the flat file databases only. Users can't specify Blast databases as a database specification for Ssearch+.

For Tru64 (OSF) Ssearch+ fails with an error message:

"While running the child process: Child was terminated by signal 6 (SIGABRT)" Error in cleaning up after application:
Exception: Error reading fast program output: Unable to open Ssearch output file: "/tmp/bslskAAAMGXMCf.tmp" (at /tmp/bslskAAAMGXMCf.tmp:0).

Workaround

There is an upper limit on the amount of memory that is allocated per process. For tru64 machine the limit for datasize is set to 128M. To increase this limit, execute

> unlimit datasize (csh) or
> ulimit datasize (ksh)

This will increase the limit on the datasize to 1024M. This is the maximum amount of memory that an individual process can take on Tru64 machine. So, default settings for the search set parameter (-infile2) for the fasta suite of programs may cause a crash. Please execute the programs with a smaller subset. The programs have been tested successfully using a search set of 400 thousand sequences.

ALGORITHM

SSearch+ uses William Pearson's implementation of the method of Smith and Waterman (Advances in Applied Mathematics 2; 482-489 (1981)) to search for similarities between one sequence (the query) and any group of sequences of the same type (nucleic acid or protein) as the query sequence. This method uses a scoring matrix (containing match/mismatch scores), a gap creation penalty, and a gap extension penalty as scoring criteria to determine the best region of local similarity between a pair of sequences. This score is reported as the Smith-Waterman score.

After the Smith-Waterman score for a pairwise alignment is determined, SSearch+ uses a simple linear regression against the natural log of the search set sequence length to calculate a normalized z-score for the sequence pair. (See William R. Pearson, Protein Science 4; 1145-1160 (1995) for an explanation of how this z-score is calculated.)

The distribution of the z-scores tends to closely approximate an extreme-value distribution; using this distribution, the program can estimate the number of sequences that would be expected to produce, purely by chance, a z-score greater than or equal to the z-score obtained in the search. This is reported as the E() score.

When all of the searches set sequences have been compared to the query, the list of best scores is printed. If alignments were requested, the alignments are also printed.

In evaluating the E() scores, the following rules of thumb can be used: for searches of a protein database of 10,000 sequences, sequences with E() less than 0.01 are almost always found to be homologous. Sequences with E() between 1 and 10 frequently turn out to be related as well.
CONSIDERATIONS

GCG version of SSearch+ searches using both strands of nucleic acid queries unless you use -onestrand. The SSEARCH+ program distributed with Dr. Pearson's FASTA+ package searches with one strand only.

The E() scores are affected by similarities in sequence composition between the query sequence and the search set sequence. Unrelated sequences may have "significant" scores because of composition bias.

If there is a database entry that overlaps your query in several places, but there are large gaps between the matching regions, only the best overlap appears in the alignment display.

There are two ways to control the size of the list of best scores. By default, scores are listed until a specific E() value is reached. You may set the value in response to the program prompt or by using -expect; otherwise the program uses 10.0 for protein searches, 2.0 for nucleic acid searches. (If you are running the program interactively, it will show no more than 40 scores initially, and ask if you want to see more scores if there are any more that are less than the E() value.)

If you use -listsize, the E() value is ignored, and the program will list the number of scores you requested.

You can control the number of alignments using -noalign and -align. The program behaves differently depending on whether it is being run noninteractively (in batch or with -Default on the command line) or interactively. In the noninteractive case, the program displays the number of alignments set by -align. (If this is not present, it shows 40 alignments or the number of scores that were listed, whichever is smaller.) If you run the program interactively, it displays the list of best scores, and then asks you how many alignments you want to see. (This prompt does not appear if you use -noalign or -align.)

Adjusting Gap Creation and Extension Penalties

Unlike other GCG programs, SSearch+ does not read the default gap creation and gap extension penalties from the scoring matrix file. It uses default gap creation and extension penalties that were empirically determined to be appropriate for the default scoring matrices. If you select a different scoring matrix with -matrix, you may need to change the gap penalties. The histogram display gives a qualitative view of the quality of fit between the actual distribution of scores and the expected distribution of scores. This information may indicate whether or not suitable gap creation and extension penalties were used for the search. When the histogram shows poor agreement between the actual distribution and the theoretical distribution, you might consider using -gapweight and/or -lenghtweight to specify higher gap creation and extension penalties, respectively. For example, you might increase the gap creation penalty from 12 to 16 and the gap extension penalty from 2 to 4.
Differences in Applying Gap Extension Penalties

There are two different philosophies on how to penalize gaps in an alignment. One way is to penalize a gap by the gap creation penalty plus the extension penalty times the length of the gap \((\text{gapweight} + (\text{lengthweight} \times \text{gap length}))\). The other way is to use the gap creation penalty plus the extension penalty times the gap length excluding the first residue in the gap \((\text{gapweight} + (\text{lengthweight} \times (\text{gap length} - 1)))\).

"Native" GCG programs, such as Framesearch and Bestfit, handle gap extension penalties the first way, while the FastA+-family programs use the second way. Therefore a value for \(-\text{lengthweight}\) that gives good results with one of the FastA+-family programs may not give equivalent results with a native GCG program, and vice versa.

Increasing Program Speed Using Multithreading

This program is multithreaded. It has the potential to run faster on a machine equipped with multiple processors because different parts of the analysis can be run in parallel on different processors. By default, the program assumes you have one processor, so the analysis is performed using one thread. You can use \(-\text{processors}\) to increase the number of threads up to the number of physical processors on the computer.

Under ideal conditions, the increase in speed is roughly linear with the number of processors used. But conditions are rarely ideal. If your computer is heavily used, competition for the processors can reduce the program's performance. In such an environment, try to run multithreaded programs during times when the load on the system is light.

As the number of threads increases, the amount of memory required increases substantially. You may need to ask your system administrator to increase the memory quota for your account if you want to use more than two threads.

Never use \(-\text{processors}\) to set the number of threads higher than the number of physical processors that the machine has -- it does not increase program performance, but instead uses up a lot of memory needlessly and makes it harder for other users on the system to get processor time. Ask your system administrator how many processors your computer has if you aren't sure.

SUGGESTIONS

Identifying the Search Set

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" topic of Chapter 2, Using Sequence Files and Databases of the User's Guide for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. For information about naming groups of sequences for the search set, see the topics "Specifying Files" and "Using Wildcards" in Chapter 1, Getting Started, and "Using Database Sequences,"
Batch Queue

SSearch+ is one of the few programs in GCG that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using \texttt{-batch}. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

Interrupting a Search: <Ctrl>C

You can type <Ctrl>C to interrupt a search and see the results from the part of the search that has already been completed. Because the program is multithreaded, the search may not be interrupted immediately, but will continue until one of the threads finishes processing its data and returns for more data.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use \texttt{-check} to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % ssearch+ [-infile1=value] -Default

Minimal Parameters (case-insensitive):

-\texttt{-infile1}  [Type: List / Default: EMPTY / Aliases: infile in1 in]  
  Input file specification.

Prompted Parameters (case-insensitive):

-\texttt{-begin}  [Type: Integer / Default: '1' / Aliases: beg]  
  Starting point of the range of interest in the input sequence.

-\texttt{-end}  [Type: Integer / Default: '-1']  
  End point of the range of interest in the input sequence. A value of '-1' indicates that the range extends till the end of input sequence.
-infile2 [Type: List / Default: EMPTY / Aliases: in2 db]
Search set specification.

-outfile [Type: OutFile / Default:
'<sequence_name>.<program_name>' / Aliases: out] File to
which output is written. A value of '-' means STDOUT.
Specifying this option also turns on the 'concat' option. Default value is '-'.

Optional Parameters (case-insensitive):

-check [Type: Boolean / Default: 'false' / Aliases: che help]
Prints out this usage message.

-default [Type: Boolean / Default: 'false' / Aliases: d def]
Specifies that sensible default values be used for all
parameters where possible.

documentation [Type: Boolean / Default: 'true' / Aliases: doc]
Prints banner at program startup.

-quiet [Type: Boolean / Default: 'false' / Aliases: qui]
Tells application to print only a minimal amount of
information.

-expect [Type: Double / Default: '2.0' / Aliases: exp]
Shows all scores whose E() value is less than the
specified value of expect.

-matrix [Type: String / Default: EMPTY / Aliases: mat]
Assigns the scoring matrix for the comparison.

-processors [Type: Integer / Default: '1' / Aliases: proc]
On a multiprocessor computer, this parameter controls
the number of threads to use for database search.

-minlength [Type: Integer / Default: EMPTY / Aliases: minl]
The search set is restricted to sequences whose length
is more than the value specified by this parameter.

-maxlength [Type: Integer / Default: EMPTY / Aliases: maxl]
The search set is restricted to sequences whose length
is less than the value specified by this parameter.

-pamfactor [Type: Boolean / Default: 'DEFAULT_PARAM_VALUE' /
Aliases: pam] This parameter governs whether a scoring
matrix should be used for calculating initial diagonal
scores, instead of using the identical match scores
from the scoring matrix. Default is to use FASTA+
internal behavior, which differs for protein and
nucleotide searches.

-gapweight [Type: Integer / Default: EMPTY / Aliases: gap]
This parameter specifies the gap creation penalty that
is substracted from an alignment every time a gap is
created.
-lengthweight  
[Type: Integer / Default: EMPTY / Aliases: len]
This parameter specifies the gap extension penalty that is substracted from an alignment every time a gap is extended by one residue.

-listsize    
[Type: Integer / Default: '10' / Aliases: lis]
This parameter controls the number of top scores show. Overrides the -EXPect parameter.

-alignments  
[Type: Integer / Default: '20' / Aliases: align ali]
This parameter limits the number of alignments to display in the output file to the 10 best matches in the list. Use -NOALIgn to suppress the sequence alignments in the output file.

-showall    

-native     
[Type: Boolean / Default: 'false']
Output native FastA+ formatted output.

-markx     
[Type: Integer / Default: EMPTY / Aliases: mark]
This parameter determines the alignment display mode—especially the symbols that identify matches and mismatches. The default value, 3, uses a pipe character (|) to show identities and a colon (:)) to show conservative replacements.

-MARKx=0 uses a colon to show identities and a period (.) to show conservative replacements. -MARKx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X. If -MARKx=2, the residues in the second sequence are shown only if they differ from the first sequence. Use -MARKx=10 to get aligned sequences in the FastA+ "parsable" output format.

-histogram  
[Type: Boolean / Default: 'true' / Aliases: his]
Start/Suppress printing the histogram.

-linesize   
[Type: Integer / Default: EMPTY / Aliases: lin]
This parameter lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-batch     
[Type: Boolean / Default: 'false']
Allows submitting a job to a batch queue.

-onestrand  
[Type: Boolean / Default: 'false' / Aliases: one]
Translates only the 3 forward frames.
LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like `-data=myfile.dat`. For more information see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either

1) Have a data file with exactly the same name as the program default scoring matrix in your current working directory; or

2) Have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name Share_Matrix; or

3) Name a file on the command line with an expression like `-matrix=mymatrix.cmp`. If you do not include a directory specification when you name a file with `-matrix`, the program searches for the file first in your local directory, then in the directory with the logical name Share_Matrix. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

SSearch+ reads a scoring matrix containing the values for every possible match from your working directory or the public database. The files fastadna.cmp (for nucleic acid sequences) and blosum50.cmp (for protein sequences) contain the default values for matches. blosum50.cmp is a BLOSUM50 matrix. You can use the Fetch+ program to obtain a copy of these files in order to modify them to suit your own needs.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

```
-infile1, -infile, -in1, -in
```

Inputs file specification.

```
-begin, -beg
```

Starting point of the range of interest in the input sequence.
-end

End point of the range of interest in the input sequence. A value of '-1' indicates that the range extends till the end of input sequence.

-infile2, -in2, -db

Search set specification.

-outfile, -out

File to which output is written. A value of '-' means STDOUT. Specifying this option also turns on the 'concat' option. Default value is '-'

-wordsize=2, -wor

Sets the size of the word (k-tuple) to use for the hashing step.

-optall=20, -opt

Immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to 20. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search. -nooptall doesn't compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.

-matrix=mymatrix.cmp, -matr

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with -matrix, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

-expct=2.0, -exp

Shows all scores whose E() value is less than 2.0. Ignored if -listsize is used.

-processors=2, -proc

Tells the program to use 2 threads for the database search on a multiprocessor computer.
-check, -che, -help

Prints out this usage message.

-default, -def, -d

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.

-quiet, -qui

This parameter is not supported.

-pamfactor, -pam

This parameter governs whether a scoring matrix should be used for calculating initial diagonal scores, instead of using the identical match scores from the scoring matrix.

Default is to use FASTA+ internal behavior, which differs for protein and nucleotide searches.

-histogram

Start/suppress printing the histogram.

-minlength=1000, -minl

Restricts the search to search set sequences that are equal to or longer than 1000 residues.

-maxlength=5000, -maxl

Restricts the search to search set sequences that are equal to or shorter than 5000 residues.

-onestrand, -one

Searches using only the top strand of a nucleotide query sequence.

-gapweight=12, -gap

Specifies the gap creation penalty that is subtracted from the alignment score whenever a gap is created.
-lengthweight=2, -len

Specifies the gap extension penalty that is subtracted from the alignment score for each residue added to an existing gap.

-listsize=40, -lis

Shows the best 40 scores. Overrides -expect.

-alignments, -align, -ali

Limits the number of alignments to display in the output file to the 10 best matches in the list. Use the -noalign to suppress the sequence alignments in the output file.

-showall, -show

Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-markx, -mark

This parameter determines the alignment display mode - especially the symbols that identify matches and mismatches. The default value, -markx=0 uses a colon to show identities and a period (.) to show conservative replacements.

-markx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X.

If -markx=2, the residues in the second sequence are shown only if they differ from the first sequence.

-markx=3 displays the aligned library sequences without the query sequences; these can be used to build a primitive multiple alignment.

-markx=4 provides a graphical display of the boundaries of the alignments.

-markx=5 provides a combination of -markx=4 and -markx=0.

-markx=6 provides -markx=5 plus HTML formatting.

-markx=9 provides percent identity and coordinates with the initial list of high scores as well as the conventional

-markx=0 alignments.
Use -markx=10 to get aligned sequences in the FastA "parsable" output format.

-native

Output native FastA+ formatted output.

-linesize=60, -lin

Lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-batch, -bat

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.
STATPLOT

FUNCTION

StatPlot plots a set of parallel curves from a table of numbers like the table written by the Window program. The statistics in each column of the table are associated with a position in the analyzed sequence.

DESCRIPTION

StatPlot is a display program for programs like Window that make sliding window measurements on a sequence. The statistics in each column of the table are associated with some position in a sequence. StatPlot figures out a scale for each column and then plots all of the statistics in parallel. You can choose the density in bases per cm along the horizontal axis so that different runs of StatPlot may be compared.

EXAMPLE

Here is a session using StatPlot to plot the functions from the example session with Window:

% statplot

STATPLOT what stat file ? gamma.wdw

gamma.wdw contains 6 columns of 134 statistics for:

<table>
<thead>
<tr>
<th>Name</th>
<th>Check</th>
<th>Begin</th>
<th>End</th>
<th>Dir</th>
</tr>
</thead>
<tbody>
<tr>
<td>gamma.seq</td>
<td>6474</td>
<td>1</td>
<td>500</td>
<td>forward</td>
</tr>
</tbody>
</table>

The minimum density for a one-page plot is 23.15 bases/cm.

What density would you like (* 23.15 *) ?

STATPLOT will take 1 pages. Would you like to:

P)lot the statistics
D)ifferent density
G)et another stat file to plot
Q)uit

Please select one (* P *):

When your LaserWriter attached to tty07 is ready, press <Return>.

P)lot the statistics
D)ifferent density
G)et another stat file to plot
Q)uit
Please select one (* P *): Q

% OUTPUT

The plot from this session is shown at the end of this program entry.

INPUT FILES

The output files from some GCG programs, such as Window, can be read as input by StatPlot. You could also create an input file using a text editor. If you do so, here are the format requirements for the input file to StatPlot.

The first line of the file must identify the sequence, checksum, and range after the words of:, check:, from:, and to:. The word reverse identifies reversed sequence ranges. Reversed ranges are numbered backwards on GCG plots.

The second non-blank line is printed on the plot without interpretation.

The dividing line (the line containing the "..") is read and the words from the second column onwards are taken to be the column headings for labeling each part of the plot. The number of words in this line between the first word (in this example, "position") and the ".." is taken to be the number of columns of statistics to be plotted. There must be a space between the last column heading and the two periods.

The data start two lines below the dividing line. The numbers are in the format I8, 6F12.3. This means that the position numbers are integers right justified in the first eight character columns. Each statistic has three figures to the right of the decimal and is right justified in a field 12 character-columns wide.

Here is some of the input file gamma.wdw, which you can Fetch for further inspection:

WINDOW of: gamma.seq  check: 6474  from: 1 to: 500
Window: 100  Shift: 3  MatchType: Subset MisMatch: 0

Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies,  Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.

October 13, 1998 13:06

Position C(obsrv) G(obsrv) CG(obsrv) CG_ob-ex(l) GC(obsrv) GC_ob-ex(l) ..
50   17.000   30.000    1.000    -4.049    4.000    -1.049
53   19.000   29.000    1.000    -4.455    5.000    -0.455
56   17.000   30.000    1.000    -4.049    5.000    -0.049
RELATED PROGRAMS

Window makes a table of the frequencies of different sequence patterns within a window as it is moved along a sequence. A pattern is any short sequence like GC or R or ATG. You can plot the output with the program StatPlot.

RESTRICTIONS

No more than six columns of measurements are allowed. No more than 300,000 measurements may appear in each column. There are a number of input file format restrictions discussed above under the INPUT FILES topic.

On Hewlett Packard plotters, density in bases per centimeter is only defined for paper that is 11 x 17 inches.

GRAPHICS

GCG must be configured for graphics before you run any program with graphics output! If the % setplot command is available in your installation, this is the easiest way to establish your graphics configuration, but you can also use commands like % postscript that correspond to the graphics languages GCG supports. See Chapter 5, Using Graphics in the User's Guide for more information about configuring your process for graphics.

<CTRL>C

If you need to stop this program, use <Ctrl>C to reset your terminal and session as gracefully as possible. Searches and comparisons write out the results from the part of the search that is complete when you use <Ctrl>C. The graphics device should stop plotting the current page and start plotting the next page. If the current page is the last page, plotters should put the pen away and graphic terminals should return to interactive mode.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use ~CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
STATPLOT does not support complete command-line control.

Local Data Files:

- MARk=gamma.mrk marks the plot with known features

Optional Parameters:

- LABel makes vertical axis labels on every page
- POInt makes points instead of a continuous curve
- CONsistent scales every field the same
- SCAling lets you set each fields scale limits interactively

All GCG graphics programs accept these and other switches. See the Using Graphics chapter of the USERS GUIDE for descriptions.

- FIGure[=filename] stores plot in a file for later input to FIGURE
- FONT=3 draws all text on the plot using font 3
- COlor=1 draws entire plot with pen in stall 1
- SCAle=1.2 enlarges the plot by 20 percent (zoom in)
- XPAN=10.0 moves plot to the right 10 platen units (pan right)
- YPAN=10.0 moves plot up 10 platen units (pan up)
- PORtrait rotates plot 90 degrees

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like - DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

If you are studying a sequence with known features, this program can mark the plot with small boxes showing the positions of these features. The presence of a file in your directory with the same name as your sequence and the filename extension .mrk causes the program to mark each range specified in the file. You can provide a marking file on the command line with an expression like - MARk=gamma.mrk. The file gamma.mrk contains information about the format of marking files. The figure for the example session shows marked regions.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- LABel

Makes vertical axis labels on both vertical axes of every page of a multi-page plot.
-POInt

Places a point at each measurement instead of drawing a continuous curve.

-CONSistent

Because StatPlot scales each field to use the whole physical vertical axis dimension, it may cause vertical exaggeration when you want to compare similar measurements. You can use the -CONSistent parameter to cause StatPlot to plot all of the measures with the same scaling. This scaling may cause weird-looking results if the measures are of different kinds as in the plot in the example. The -SCAling parameter allows you to choose the absolute scaling for each field.

-SCAling

Allows you to set the scaling on the vertical axis. If you use this parameter you are asked for the bottom and top of each panel in the plot. The query shows the defaults calculated for each panel.

-MARk=gamma.mrk

If you are studying a sequence with known features, this program can mark the plot with small boxes showing the positions of these features. The presence of a file in your directory with the same name as your sequence and the file name extension .mrk causes the program to mark each range specified in the file. The file gamma.mrk contains information about the format of marking files.

The parameters below apply to all GCG graphics programs. These and many others are described in detail in Chapter 5, Using Graphics of the User's Guide.

-FIGure=programname.figure

Writes the plot as a text file of plotting instructions suitable for input to the Figure program instead of sending it to the device specified in your graphics configuration.

-FONT=3

Draws all text characters on the plot using Font 3 (see Appendix I).

-COLor=1

Draws the entire plot with the pen in stall 1.

The parameters below let you expand or reduce the plot (zoom), move it in either direction (pan), or rotate it 90 degrees (rotate).
-SCAle=1.2

Expands the plot by 20 percent by resetting the scaling factor (normally 1.0) to 1.2 (zoom in). You can expand the axes independently with -XSCAle and -YSCAle. Numbers less than 1.0 contract the plot (zoom out).

-XPAN=30.0

Moves the plot to the right by 30 platen units (pan right).

-YPAN=30.0

Moves the plot up by 30 platen units (pan up).

-PORtrait

Rotates the plot 90 degrees. Usually, plots are displayed with the horizontal axis longer than the vertical (landscape). Note that plots are reduced or enlarged, depending on the platen size, to fill the page.

Printed: February 10, 2006 19:41
STEMLOOP

FUNCTION

StemLoop finds stems (inverted repeats) within a sequence. You specify the minimum stem length, minimum and maximum loop sizes, and the minimum number of bonds per stem. All stems or only the best stems can be displayed on your screen or written into a file.

DESCRIPTION

StemLoop searches for inverted repeats in your sequence after you choose a minimum stem length and minimum and maximum loop sizes. You must also specify a minimum number of bonds per stem with G-T, A-T/U, and G-C scored as 1, 2, and 3 bonds, respectively. The stems found can be sorted by position, size (stem length), or quality (number of bonds) and can be either filed or displayed on the screen. StemLoop tells you the number of stems found for your settings of minimum stem size, maximum loop size, minimum loop size, and minimum bonds per stem. If you feel there are too many stems, you may reset the parameters without reviewing the stems found or view only the best stems found. To view only the best stems, there must be more than 25 stems found and you must sort them by quality or size. (See the ALGORITHM topic below to understand precisely what StemLoop does.)

EXAMPLE

Here is a session using StemLoop to see the 8 inverted repeats that have at least 18 bonds within stems at least 8 base pairs in length in an Alu consensus sequence:

```
% stemloop
STEMLOOP of what sequence ? alucons.seq

Begin (* 1 *) ?
End (* 290 *) ?

What minimum stem length (* 6 *) ? 8
What minimum number of bonds/stem (* 16 *) ? 18
What maximum loop size (* 20 *) ?
What minimum loop size (* 3 *) ?

just a second ... 

There are 8 stems. Would you like to:
1) See the stems
2) See the stem coordinates
3) File the stems
4) File the stems as points for DOTPLOT
```
5) Choose new parameters,
6) Get a different sequence

Q)uit ?

Please choose one (* 1 *): 3

Sort Stems by:

1) Position
2) Quality
3) Size

Q)uit

Please choose one (* 1 *): 2

What should I call the output file (* alucons.stem *) ?

There are 8 stems. Would you like to:

1) See the stems,
2) See the stem coordinates,
3) File the stems,
4) File the stems as points for DOTPLOT
5) Choose new parameters,
6) Get a different sequence

Q)uit ?

Please choose one (* Q *):

% OUTPUT

StemLoop creates an output file if you choose to file the stems from any search; otherwise, you may view the stems on your screen. In either case, the stem is shown, as below, with vertical bars ('|') indicating the base pairs. The associated loop is shown to the right of the stem. If either the stem or loop is too long to be displayed in its entirety on the line, then only that part that fits on the line is shown. The first and last coordinates of the stem are displayed on the left, and the length of the stem (size), the number of bonds in the stem (quality), and the loop size are shown on the right. Here is part of the file alucons.stem created by the example session above:

STEMLOOP of: alucons.seq  check: 1861  from: 1  to: 290

Alu consensus sequence
Labuda, D. and Striker, G. (1989) Sequence conservation in Alu
evolution. Nucleic Acids Research 17, 2477-2491.

Minimum Stem: 8  Minimum bonds/stem: 18  Maximum loop size: 20
Stems found: 8  Stems shown: 8
Average Match: 1.80  Average Mismatch: 0.00  Nibbling Threshold: 1

October 6, 1998 14:16  ..

217 AGGCTGCAGTAGGCGTGAT  11, 25
         |||||||   C
257 TCCGGCCTCAC  GTCACCGCG  19

135 TAGCGGAGCGGTG  GG  11, 22
         ||||| |
160 GTCCGCGCGCGG  GT  4

//////////////////////////////////////////////////

221 TGCAGTG  AGCCGTG  7, 18
         |||||
248 ACGTCAC  CGCGCTA  14

35 CACTTCGG  GA  8, 18
         |||||
54 GCGGAGCC  GG  4

You may choose to see only the numbers defining each stem on your screen by choosing option '2' in the first menu. This is what that screen output would look like if you choose option '2' in the first menu and then choose to sort by quality in the second menu:

<table>
<thead>
<tr>
<th>Loop</th>
<th>Start</th>
<th>End</th>
<th>Size</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>217</td>
<td>257</td>
<td>11</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>135</td>
<td>160</td>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>139</td>
<td>160</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>69</td>
<td>95</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>25</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>213</td>
<td>247</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>221</td>
<td>248</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>54</td>
<td>8</td>
<td>18</td>
</tr>
</tbody>
</table>

StemLoop can also make an output file with points for plotting with DotPlot.

INPUT FILES

StemLoop accepts a single nucleotide sequence as input. If StemLoop rejects your nucleotide sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

MFold predicts optimal and suboptimal secondary structures for an RNA or DNA molecule using the most recent energy minimization method of Zuker. PlotFold displays the optimal and suboptimal secondary structures for an RNA or DNA molecule predicted by MFold.
Using Compare-DotPlot to create a dot-plot of the similarities between a nucleotide sequence and its reverse-complement strand is functionally equivalent to running StemLoop. Repeat uses the same algorithm as StemLoop to find repeats that are not inverted. DotPlot shows you the output from Compare or StemLoop on a surface of comparison.

RESTRICTIONS

StemLoop only searches for loops through a range that is equal to twice the minimum stem length, plus the maximum loop size. You may extend the search range by increasing the maximum loop size; however, the maximum range for the search may not exceed 2,000 bases. StemLoop cannot find more than 1,000 loops.

ALGORITHM

StemLoop uses a window and stringency match criterion in exactly the same manner as Compare. For every position in each register shift, a window set by you as the minimum stem size is moved along the sequence, and if the minimum number of bonds per stem or more are found, then a stem is recorded covering all of the bases under the window. The number of the bonds under the window at each window position is the sum of the scoring matrix values for each base pair found in the file stemloop.cmp (see the LOCAL DATA FILES topic below). Mismatches can be scored negatively, although the public data file simply scores matches with G-T, A-T/U, and G-C worth 1, 2, and 3, respectively. Several adjacent mismatches may be found within a long stem if there are strong matches on either side. The criterion for a stem is that the minimum number of bonds occur within a length set by you as the minimum stem length.

Stem Extension and Nibbling

Before the stems are presented, they are extended (or nibbled) from both ends so that the first base on each end participates in a bond. The criterion for a bond between pairing bases is that the value in the scoring matrix file (stemloop.cmp) for the pair is greater than or equal to the average positive non-identical comparison value in the scoring matrix. You can reset the threshold for nibbling with -PAI\text{x}. You could set a pairing threshold high enough so that all stems are nibbled away!

Since stem nibbling occurs, stems shorter than the minimum stem length are commonly reported. If, on the other hand, extra pairing bases are found adjacent to the stem, the stem is extended until a pair of bases do not have a bond between them. If the nibbling function finds, after nibbling a stem, that the now enlarged loop is longer than the specified maximum loop size, the stem is not reported.

CONSIDERATIONS

StemLoop chooses a default minimum number of bonds per stem that is appropriate for the scoring matrix it reads. If you select a different scoring matrix with -MATR\text{x}, the program will adjust the default minimum number of bonds per stem accordingly.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % stemloop [-INfile=]alucons.seq -Default

Promted Parameters:
- BEGin=1 -END=290 sets the range of interest
- STEMlength=6 sets the minimum stem length
- BONds=12 sets the minimum bonds per stem
- MINLoopsize=3 sets the minimum loop size
- MAXLoopsize=20 sets the maximum loop size (distance to furthest inverted repeat)
- MENU1=1 specifies output type: 1=see stems, 2=see coordinates, 3=file, 4=dotplot file
- MENU2=1 sorts by: 1=position, 2=quality, 3=size
- MAXSTems=25 sets the maximum number of stems to show (quality or size sorts only)
[-OUTfile=]alucons.stem names the output file

Local Data Files:
- MATRix=stemloop.cmp assigns the scoring matrix for finding bonds/stem

Optional Parameters:
- PAIr=1 sets threshold for nibbling, match (|), and point display

Note: StemLoop does not cycle through the menus repeatedly if you specify either -MENU1 or -MENU2 on the command line.

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a
public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name MyData; or 3) name a file on the command line with an expression like `-MATRIX=mymatrix.cmp`. If you do not include a directory specification when you name a file with `-MATRIX`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

StemLoop uses a scoring matrix of the kind described in Appendix VII to find the number of bonds between any possible pair of bases. Every non-zero value is defined in the scoring matrix. StemLoop reads the scoring matrix file stemloop.cmp in your local directory, or if it fails to find such a file there, it uses the public file of the same name. The file can be customized so that any score, positive or negative, can be assigned to any possible pair of bases (GCG symbols). You can get the public file with `% fetch stemloop.cmp`. The values in the file assign G-T, A-T/U, and G-C to 1, 2, and 3 respectively, with all other pairs valued at zero. A more realistic set of values might assign some negative score to the mismatches, especially purine-purine pairs. This would make the output sorted by quality more significant.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- `stemlength=6`
  Sets the minimum stem length.

- `bonds=12`
  Sets the minimum bonds per stem.

- `minloopsize=3`
  Sets the minimum loop size.

- `maxloopsize=20`
  Sets the maximum loop size (distance to furthest inverted repeat).

- `menvu1=1`
  Indicates the type of output. `menvu1=1` means display the stems; other values for `menvu1` are: 2) display the coordinates, 3) save the stems to a file, and 4) save the stem coordinates to a DotPlot file.
-MEN:\textit{2}=1

Indicates how to sort the stems in the output. \textit{-MEN:\textit{2}=1} means sort by position; other values for \textit{-MEN:\textit{2}} are: 2) sort by quality and 3) sort by size.

-MAXST_{ems}=25

Sets the maximum number of stems to show (only applies when stems are sorted by quality or size).

-MATR{ix}=mymatrix.cmp

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with \textit{-MATR{ix}}, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

-PAI:\textit{r}=1

The output from this program has a '|' (vertical bar) between sequence symbols that match. This \textit{match display character} is added to the output whenever the symbol comparison value for the two symbols in your scoring matrix is greater than or equal to the average positive non-identical comparison value in the matrix. The \textit{-PAI:\textit{r}} parameter lets you specify a match display threshold appropriate for the scoring matrix you are using.

Stem structure nibbling also uses the threshold value set by this parameter to decide what pairs should be nibbled away from the structure. \textit{You can set a pairing threshold high enough so that all stems are nibbled away!}

Printed: February 10, 2006 19:41
**STRINGSEARCH**

**FUNCTION**

StringSearch identifies sequences by searching for character patterns such as "globin" or "human" in the sequence documentation.

**DESCRIPTION**

Annotations and Definitions

In addition to the actual sequence data, GCG databases contain two additional types of data: sequence annotations, and definitions.

The annotations contain the complete documentation for each entry in the sequence database, including journal and author names, sequence features, comments, etc. The annotations appear at the top of sequences copied from a GCG database with the Fetch program.

The definitions contain a minimal amount of the annotations documentation for each entry: the name of the organism, the name of the gene, the sequence length, and usually the date. Definitions for the GenBank, EMBL, and SWISS-PROT databases also contain the primary accession number for the sequences.

The StringSearch program searches through either the definitions alone or the complete sequence annotations for text patterns that you specify. Annotations take much longer to search than definitions.

Searching Sequence Definitions

The expression `%% stringsearch GenBank:* human` finds every entry in the GenBank sequence database whose definition contains the text pattern `human`. The databases available in addition to GenBank are EMBL, SWISS-PROT, and PIR-Protein. GenBank specifies the sequences in both GenBank and EMBL. Additionally, definitions searches can be done on any of the individual divisions in GenBank and/or EMBL. If you believe that a published human sequence in the database is 1,531-bases long, you can search for entries that contain both `human` and `1531`.

When searching definitions, you can specify the set of sequences you want to search in the same way as for all other GCG programs with the following exception. The specified sequences must be contained in a database; you cannot search the definitions of user sequences. For instance, the specification `Primate:hum*` would search through the definitions for all of the sequences in the Primate division of GenBank that begin with the pattern `hum`. You may also specify the database sequences to search by means of a list file. Each sequence in a list file must be preceded by a logical name for one of the databases or database divisions. Sequence specification is described in detail in Chapter 2, Using Sequence Files and Databases of the User's Guide.
Searching Complete Sequence Annotations

When you are searching complete sequence annotations, you can specify the set of sequences you want to search in the same way as for all other GCG programs. Sequence specification is described in detail in Chapter 2, Using Sequence Files and Databases of the User's Guide.

If your sequence specification is not preceded by a logical name, StringSearch looks in all of the databases and in all of the GCG data files to find all possible entry names. The specification GenBank:hum* will search only GenBank for sequences whose names begin with hum, while hum* will search GenBank and also databases other than GenBank and all GCG data files. A search of all the entries in all the databases takes a very long time.

Special Considerations for Searching

Keep in mind that filenames are case sensitive and database entry names are case insensitive. Because this program searches for both filenames and database entry names, you must take care when you enter the character pattern that makes up your specification.

For example, if you entered Gamma* as a file specification, this program would find all entries in the databases whose names begin with Gamma but no GCG-supplied files would be found. This is because all the files in GCG are named using lowercase letters. Conversely, if you entered gamma*, this program would find all of the entries in the databases and all GCG supplied files whose names begin with gamma.

Searching for More Than One Pattern

You can search for more than one text pattern in response to the program prompt with Human, Globin. StringSearch then finds all the entries that contain both human and globin. You can set StringSearch to show all the entries that contain either human or globin with -MATCH=OR.

Specifying Patterns

Blank spaces are removed from the beginning of each pattern unless that pattern is enclosed in double quotes. For instance, specifying the pattern Globin shows all entries that contain globin, while specifying "Globin" excludes entries containing terms like myoglobin in which globin is not preceded by a space.

To specify a double quote (") as part of a pattern, use two double quote marks (""). To specify a comma as part of a pattern, enclose the whole pattern in quotes.
EXAMPLE

Here is a session using StringSearch to search for nucleotide sequences with pseudogenes in GenBank:

% stringsearch

STRINGSEARCH through what sequence(s) (* GenBank:* *) ?

Do you want to search through:

A) definitions
   B) complete sequence annotation

Please choose one (* A *):

Search for what text patterns ? Pseudo

What should I call the output file (* GenBank.strings *) ?

*** Gbba:Ab000361 ***

AB000361 Pseudomonas cichorii gene for D-Tagatose 3-epimerase, ...

//////////////////////

*** Gbsts:Ppu85464 ***

U85464 Pseudomyrmex pallidus clone Psd2523CAC trinucleotide ...

   Sequences searched:  552323
   Sequences with matches:     6237
   Patterns sought:   Pseudo

   Output file: GenBank.strings

%

OUTPUT

The output file from StringSearch is a list file. (See Chapter 2, Using Sequence Files and Databases of the User's Guide for more information.) Here is what the output from the example session looks like:

!!SEQUENCE_LIST 1.0
! STRINGSEARCH from: GenBank:*  October 22, 1998 11:38

! searching for: "pseudo" ..
INPUT FILES

StringSearch takes as input any valid GCG sequence database specification. This may represent a single sequence, for example GenBank:humcyc. But usually you specify multiple sequences by using a database specification with an asterisk (*) wildcard, for example GenBank:*

When searching complete sequence annotations, you may also search one or more user sequences, using a wildcard asterisk or a list file to specify multiple sequences. To search user sequences in your own directories instead of in the GCG data files directories, you must preface the specification with the path to your sequences, for example: /usr/user/burgess/seqs/*.seq

RELATED PROGRAMS

LookUp identifies sequence database entries by name, accession number, author, organism, keyword, title, reference, feature, definition, length, or date. The output is a list of sequences.

RESTRICTIONS

The search is case insensitive.

LIST REFINEMENT

The database programs LookUp, Names, StringSearch, FindPatterns, FastA, TFastA, FastX, TFastX, SSearch, and WordSearch can be used for list refinement if you are looking for sequences with something in common. For instance, you could identify human globin nucleotide sequences with LookUp. The output list from LookUp could then be refined further with FindPatterns to show only those human globin sequences containing EcoRI sites. If you run FindPatterns with -NAMes, you could then do a FastA sequence search on the FindPatterns list file output to see if a sequence you have is similar to any of these EcoRI-containing human globin sequences.
Adding Lists Together

You can add two lists together by simply appending one of the files to the other. It is better if you use a text editor to modify the heading of the combined list so that the annotation in the list correctly reflects what you have done. Remember to delete the text heading from the second file so that it does not occur in the middle of the list.

Suppressing Items

Suppress any item in a list by typing an exclamation point (!) in front of the item. You can also put comments into a list anywhere on a line by placing an exclamation point before the comment.

CONSIDERATIONS

You cannot assume that a text pattern search is exhaustive. The text you choose may not have been used by the data collectors. Worse yet, all databases contain errors -- the misspelling \textit{psuedo} appears 14 times in the definitions for GenBank Release 108.0!

Hyphenation is particularly prone to inconsistent usage. A search for \textit{pseudogene} would only be complete if \textit{pseudo-gene} (or \textit{pseudo gene}) were never used to refer to pseudogenes.

Using a nonspecific pattern such as \textit{pseudo} to find sequences of pseudogenes will result in many false matches. (In the example session, there were 1570 instances of \textit{Pseudomonas}, 321 instances of \textit{pseudoobscura}, and 42 instances of \textit{pseudoautosomal} out of 6237 total matches.) But restricting the search by setting \texttt{-MAtch=AND} and searching for \textit{pseudo, gene} will miss pseudogene sequences whose definitions use terms like \textit{pseudoexon} or the prefix \textit{pseudo}- used with the name of the gene. It's usually better to use a less-specific search pattern and then edit the resulting list file to remove entries that you aren't interested in.

The conclusion is that a search with StringSearch can only tell you what is available.

The complete annotation search takes a lot of computing, but the search includes a lot of information, such as author and journal names, that is not found in the sequence definitions. You can speed up the search considerably by using a sequence specification like \texttt{Primate:hum*} to look only at the group of sequences in which you really expect the text pattern to be found.

Use the expression \% \texttt{typedata primate:hum*} to see some examples of sequence annotations.

SUGGESTIONS

Batch Queue

StringSearch is one of the few programs in GCG that can take more than a few minutes to run. Searches should probably be run in the batch queue if an entire database is being searched, especially if the complete annotations search (\texttt{-MEnu=B}) is chosen. You can
specify that this program run at a later time in the batch queue by using \(-\text{BAT}ch\). Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide.

**STOPPING SCREEN OUTPUT**

If you want this (or any) program to stop so you can read the screen, use \(<\text{Ctrl}>S\). Restart the program by using \(<\text{Ctrl}>Q\).

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use \(-\text{CHEck}\) to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % stringsearch [-INfile=]GenBank:* [- STRings=]pseudo -Default

Prompted Parameters:

\(-\text{MENu=}a\) selects the sequence documentation to search:
\(A=\)definitions, \(B=\)complete records

[-OUTfile=]GenBank.strings names the output list file

Optional Parameters:

\(-\text{MATch=}or\) finds entries with any of the patterns specified
\(-\text{WIDTH=}100\) limits length of documentation in the output file
\(-\text{NOHEAding}\) suppresses the heading in the output file
\(-\text{BATch}\) submits the program to run in the batch queue
\(-\text{NOSCReen}\) suppresses the screen output
\(-\text{NOMONitor}\) suppresses the \('.\)'s in the screen trace

**LOCAL DATA FILES**

None.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

\(-\text{STRings=}\text{Pseudo}\)

String pattern or patterns to search for.
-MENu=A

Searches complete entry records (-MENu=B) or just the definition lines of the entries (-MENu=A, the default).

-MATCH=OR

When you are looking for more than one text pattern, this parameter sets StringSearch to find sequence entries that contain any one, but not necessarily all, of the text patterns you have specified. -MATCH=AND requires that the sequences found contain all of the patterns sought. -MATCH=2 requires that each of the sequences found have two of the patterns sought.

-WIDTH=100

StringSearch normally appends a line of documentation after each sequence name in the output list file, starting at the 20th column. Use this parameter to set the length of the documentation. A value of 100 gives lines that are a maximum of 120 characters long. -WIDTH=0 suppresses the documentation next to each sequence name completely.

-NOHEADING

Suppresses the heading at the top of the list file that shows the input specification and the time.

-BATCH

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

-NOSCREEN

Suppresses the output on the screen that shows each sequence as it is found. You must direct output to a file for this parameter to work.

-NOMONITOR

When searching complete sequence annotations, a dot normally appears on your screen for every 50 complete sequence annotations that are searched without a find. This parameter suppresses the display of the dots.

Printed: February 10, 2006 19:41
SYMBOL

FUNCTION

Symbol creates, changes, deletes, or displays GCG symbol(s) from the GCG symbol table.

DESCRIPTION

Programs of GCG use symbols to refer to various optional values that control operation of GCG. Examples of these symbols include the plotting protocol and device to be used, file protections, and the level of command-prompt verbosity. Each symbol has an identifier and a single associated value. For example, the identifier PlotDriver has as its value the name of the driver used to perform graphics output for GCG. The symbol identifiers and their values are maintained in a table by a server process. The Symbol program provides a command-line interface to the symbol table.

Symbol identifiers are case insensitive, meaning you can enter the letters in uppercase, lowercase, or mixed case. For example, the symbol you specify as PlotDriver is the same as specifying plotdriver. An identifier can have up to 31 characters and can contain any combination of alphanumeric characters, plus the dollar sign ($), underscore (_), and hyphen (-) characters. The values associated with a symbol are case sensitive and can contain up to 1,024 characters, but must not contain any null characters.

When GCG is initialized at the start of a session, the Symbol program is used to establish a table of symbols with values that are appropriate for your site. During the course of your session, some programs may establish new symbols and may modify the values of some existing symbols as a side effect of their execution.

To display the current contents of the table, type `% symbol`. To find the symbols that start with a specific character pattern, you can use the asterisk (*) and question mark (?) wildcards. For example, to find all symbols that start with di you would type `% symbol di*`. In this case the di* acts as an identifier specification because it may refer to more than one symbol.

EXAMPLE

Here is an example that shows you how to create a symbol:

```
% symbol -set
Set what symbol ? documentation
Set to what value ? FALSE
```

For the rest of your GCG session, the package programs that examine the value of this symbol identifier, which determines whether or not to display program documentation, will read the value FALSE and as a result, not display documentation text.

Here is a session that shows you how to delete the symbol you just created:

```% symbol -unset
Unset what name? documentation
%
```

With the `-Unset` parameter, you can also use an identifier specification to refer to a related group of symbols. For example, you could have entered `% symbol -unset plot*` to unset any or all symbols with identifiers that begin with `plot`.

**OUTPUT**

There is no output for this program.

**RELATED PROGRAMS**

Name creates, changes, deletes, or displays GCG logical name(s) from the GCG logical names table. (Also referred to as the name(1) command.)

**RESTRICTIONS**

The TableMap program must establish a memory table before Symbol can operate.

The values associated with a symbol are case sensitive and can contain up to 1,024 characters, but must not contain any null characters. An identifier can have up to 31 characters and can contain any combination of alphanumeric characters, plus the dollar sign ($), underscore (_), and hyphen (-) characters. The values associated with a symbol are case sensitive and can contain up to 1,024 characters, but must not contain any null characters.

**CONSIDERATIONS**

Symbols differ from environment variables in the manner in which they are inherited by programs. When you use Symbol to change a current symbol or create a new one, the change to the symbol table becomes immediately visible to all GCG processes running in the context of the current session. In contrast, processes always see the contents of the environment table as it existed at the time that they were started.

The order of the parameters for `-Set` is `identifier` followed by `value`. If you like, you can bypass this implied order by using the `-Name=identifier` and `-Value=value` parameters in any order on the command line. (See the PARAMETER REFERENCE topic for more information.)
(If you use Symbol in a shell script, note that it sets its exit status to zero when operation is successful, or to -1 if unsuccessful.)

SUGGESTIONS

Any change you make to the symbols table for a session is in effect during that session only; once you log off, the change is lost. If you want to assign values to symbols at login, you should put the information in your .ggrc file. (See "Customizing Your Login" in Chapter 1, Getting Started of the User's Guide for information about the .ggrc file.)

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % symbol

Promoted Parameters: None.

Optional Parameters:
- -Set identifier value changes the value of the symbol or adds a new symbol
- -Unset identifier-spec deletes all symbols matching identifier-spec
- -Name=identifier lets you enter a symbol identifier anywhere on the command line
- -Value=value lets you enter a symbol value anywhere on the command line
- -Quiet suppress all messages
- -KILL stops the symbols service for the session
- -List identifier-spec lists all symbols matching the identifier-spec

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Note that although there are no required parameters for the Symbol command, you would generally use it with the -Set or -Unset parameters.
-Set identifier value

Creates a symbol or assigns a new value to an existing one. For example, `% symbol -s documentation TRUE` changes the symbol Documentation to TRUE. If you leave out the identifier value, Symbol prompts you for the information.

-Unset identifier-spec

Deletes an existing symbol. This parameter is the opposite of the -Set parameter.

-Quiet

Suppresses any screen display (exit status is set as usual).

-KILL

Stops the symbol service from running. (If you use this parameter, and we can think of no reason why you would, you must restart GCG by entering `% gcg`.)

-Name=identifier

Lets you enter a symbol identifier anywhere on the command line.

-Value=value

Lets you enter a value for the symbol anywhere on the command line.

-List identifier-spec

Displays the symbols that match an identifier specification, along with the values associated with those symbols.

Printed: February 10, 2006 19:41
TERMINATOR

FUNCTION

Terminator searches for prokaryotic factor-independent RNA polymerase terminators according to the method of Brendel and Trifonov.

DESCRIPTION

Terminator uses a table of the dinucleotide frequencies for each position from a set of known terminators to find places in a new sequence where terminator-like sequences occur. Terminator finds all discrete examples in the searched sequence where a measurement falls above some user-defined threshold value. The measurement for each alignment of the table over the sequence is the sum of the values in the table for each dinucleotide from the sequence. The method can also restrict the set of terminator-like sequences shown to those that fall above some threshold for the presence of a GC-rich dyad symmetry near the poly-U region.


EXAMPLE

Here is a session using Terminator to search for terminator-like sequences in synpbr322:

% terminator

TERMINATOR search of what sequence ? GenBank:SynpBR322

Begin (* 1 *) ?
End (* 4361 *) ?
Reverse (* No *) ?

Primary structure threshold value (* 3.50 *) ?
Secondary structure threshold value (* 0 *) ?

What should I call the output file (* synpbr322.trm *) ?
Searching . . .

%

OUTPUT

Here is the output file:
TERMINATOR search on: synpbr322 check: 5483 from: 1 to: 4361

J01749 Cloning vector pBR322, complete genome. 6/96
LOCUS SYNPBR322 4361 bp DNA circular SYN 07-JUN-1996
DEFINITION Cloning vector pBR322, complete genome.
ACCESSION J01749 K00005 L08654 M10284 M10285 M10356 M10784 M10785 M10786 M33694 V01119
NID g208958 . . .

Primary structure threshold: 3.50 Secondary structure threshold: 0

October 6, 1998 15:02 . .

-40 -35 -30 -25 -20 -15 -10 -5 -1+ +5 p s
921=> CCATTATGCCGCCATGGGCGCCAGCGCTGGGCTACGTCTTGCTGGCGT 3.80 0
1398=> CACCTCCAGGCACCGCCCGCCGACCGCTGGGCTACGTCTTGCTGGCGT 3.62 0
1573=> TCTGCTCGCTCGCTCTCGCTGGGCTACGTCTTGCTGGCGT 3.62 0
1583=> GAGCAACAATGGGTCTCTGGGCTACGTCTTGCTGGGCTACGTCTTGCTGGGCTACGTCTTGCTGGCGT 3.62 0
1881=> CATGAACAGAAATCCCCCTTACACCGAGAGCCATCAGTGACCAAAACAGGAAA 3.57 16

--- --    -- ---
914=> AGTGACCAAACAGGAAAAAACGCCCTTAAACATGCGCCCGCTTTATCGAAG 4.47 0
2320=> GATGCGTAGAGGAAAAATACCGCATCCGACCGCTTTCCGCTTCTCGCTCA 3.73 48

2492=> GCAAAGGCGCAAGCGGTTAAAGGAGCCGCGGGCTGCTGGCCTTTCCCATAG 4.35 0
2497=> AGGCCAGAAACCCTAAAGGAGCCGCGGCTGCTGGCCTTTCCCATAGGCTCCC 3.95 0
3039=> TGATCCGGAAACCCCTTACACCGGCTGCTGGGCTTTTTTGTTCGAAAAG 6.92 95

-------------
3101=> GCCGCAAGAAAAAGGATCTCAAGAAGATCTTTTCTGGGCTACCATGGGCT 4.18 68

-------------
3199=> GATTATCAAAAGGATCTTCACCTAGAATCTCTTAAATTAAAAATAGGGTT 4.62 19

-------------
3502=> GTGGTCTCTGACTTATCTGGGCTCTCCATCCAGTCATTTATATGGGCTGCG 3.59 0
4226=> TATTGAAAAATAAAAATATAGGGTTCCGCCACATTTCCCCAGAAAGT 4.49 0
4311=> ACATTAACCTATAAAAAATAGGCGTATCCAGGAGCCCTTTGCCCTTTCAAGAA 3.69 0

INPUT FILES

Terminator takes a single nucleic acid sequence as input. If Terminator rejects your nucleotide sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

None
RESTRICTIONS

The pattern recognition method used by Terminator is only applicable to the search for prokaryotic factor-independent terminators. As mentioned above, Terminator is not really a GCG program, but was adapted to run with the package by Greg Hamm. Its behavior is not completely known, and it may not adhere to all GCG conventions. Accelrys (GCG) is very grateful to Drs. Brendel and Trifonov for generously allowing them to distribute their program.

ALGORITHM

The algorithm is described clearly in the CODATA paper.

CONSIDERATIONS

The default primary structure threshold is such that about 95 percent of known factor-independent, prokaryotic terminators should be found by Terminator in a set of terminator-like sequences, based on primary structure alone.

The program predicts terminators in those parts of the sequence composed entirely of lower- and uppercase G, A, T, and C. Parts of the sequence containing other sequence symbols are given a primary structure value of 0.0 and a secondary structure value of 0.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use –CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % terminator [-INfile=]genbank:synpbr322 -Default

Prompted Parameters:

-BEGin=1 -END=4363 sets the range of interest
-REVerse uses the reverse strand
-PTHRESHold=3.50 sets the primary structure threshold value
-SPTHRESHold=0 sets the secondary structure threshold value
[-OUTfile=]synpbr322.trm names the output file

Local Data Files:

-DATa1=pmatrix.dat contains the normalized dinucleotide fractions
-DATa2=smatrix.dat contains the significant GC-rich dyad diagonals
Optional Parameters: None

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

The file pmatrix.dat is taken from Figure 3 of the CODATA paper. It is similar to Figure 3 of the NAR paper. It contains the normalized fractions of each dinucleotide observed in the set thought to be determining terminator structure. The file smatrix.dat is from Figure 2 of the CODATA paper. It contains the significant diagonals for the GC-rich dyad symmetry. Both pmatrix.dat and smatrix.dat must be provided to Terminator as local data files.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-PTHRESHold=3.50

Sets the primary threshold value for display of sequence ranges. The default value is set to find 95 percent of known, factor-independent, prokaryotic terminators.

-STHRESHold=0

Sets the secondary structure threshold value. This secondary structure is GC-rich dyad symmetry near poly-U regions of a sequence.
TESTCODE

FUNCTION

TestCode helps you identify potential protein coding regions in nucleic acid sequences by plotting a measure of the non-randomness of the composition at every third base. The statistic does not require a codon frequency table.

DESCRIPTION

TestCode helps identify genes when you do not have specific knowledge of codon preferences for the DNA being examined. TestCode plots a measure of the period three constraint of each region of a DNA sequence using a statistic developed by Dr. James Fickett at Los Alamos (Nucl. Acids Res. 10(17); 5303-5318 (1982)).

The statistic is independent of the reading frame and is based on measurements of the period three compositional constraints found in regions known to be coding and non-coding. The output file plot is divided into three regions for which the statistic makes predictions. For windows larger than 200 nucleotides, the top region is supposed to predict coding regions to a 95 percent level of confidence. The bottom region is supposed to predict non-coding regions to the same confidence level. The middle region is the window of vulnerability for the method where the statistic can make no significant prediction.

In the plot, there are markings above the curve that identify the potential start codons (ATG) and stop codons for the forward reading frame of the sequence. Starts are indicated by short vertical lines and stops by small diamonds.

EXAMPLE

Here is a session using TestCode to plot the TestCode statistic for the E. coli outer membrane proteins in the sequence Bacterial:EcoOmpa:

```bash
% testcode
Plot TESTCODE for what sequence ? Bacterial:EcoOmpa
   Begin (* 1 *) ?
   End (* 2270 *) ?
   Reverse (* No *) ?

What window size in bp (* 200 *) ?

The minimum density for a one page plot is: 2270.0 bases/page
A typical density is about 3000.0 bases/page

What density would you like (* 2270.0 *) ?

When your LaserWriter attached to tty07 is ready, press <Return>.

%```
OUTPUT

The plot from this session is shown in the figure at the end of this program entry.

INPUT FILES

TestCode accepts a single nucleotide sequence as input. If TestCode rejects your nucleotide sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

The method of Gribskov et al. (Nucl. Acids Res. 12(1); 539-549 (1984)) is available in the CodonPreference program if you have an appropriate codon frequency table in GCG format. CodonPreference displays a separate plot for each of the forward reading frames.

RESTRICTIONS

Unknown.

FICKETT'S TESTCODE STATISTIC

Fickett's TestCode statistic was described by James Fickett in Nucleic Acids Research 10(17); 5303-5318 (1982). We believe that TestCode is a formal implementation of Fickett's method.

The statistic is high when measures of compositional bias with a periodicity of three are high. The key measures of bias are simply the three measures:

\[
\frac{\text{Maximum}(n_1, n_2, n_3)}{\text{Minimum}(n_1, n_2, n_3)}
\]

where \(n_1\), \(n_2\), and \(n_3\) are the composition of each nucleotide at positions (1,4,7,...), (2,5,8,...) and (3,6,9,...). The composition is simply the number of observations of \(n\) in the window.

The path to the final TestCode statistic is quite tortuous, but there is good reason. Fickett measured the biases for the coding and noncoding sequences that were then in the database and derived an empirical statistic that would separate coding sequences from non-coding sequences. He did not take a sliding-window approach to that measurement but instead used whole coding sequences. Unfortunately, the exons of many eukaryotic coding sequences are considerably shorter than the resolution of the method. The TestCode statistic does not claim to make a significant prediction for windows of less than 200 bases.

Fickett also found that compositional constraint is characteristic of coding sequences, and his TestCode statistic takes composition into account. However, we have received two personal communications suggesting that the TestCode statistic is actually more sensitive when composition is ignored. We have done no experiments to confirm this.
CONSIDERATIONS

The method was designed to detect coding regions that are more than 200 bases long. Therefore, the method misses many eukaryotic coding sequences that are considerably shorter than this. The statistic is very sensitive when coding regions have strong codon preferences.

Frameshift errors in the data reduce the TestCode statistic as the window passes over them.

Plotting at a density of more than 5,000 bases per page may make a pattern difficult to read.

GRAPHICS

GCG must be configured for graphics before you run any program with graphics output! If the % setplot command is available in your installation, this is the easiest way to establish your graphics configuration, but you can also use commands like % postscript that correspond to the graphics languages GCG supports. See Chapter 5, Using Graphics in the User's Guide for more information about configuring your process for graphics.

<CTRL>C

If you need to stop this program, use <Ctrl>C to reset your terminal and session as gracefully as possible. Searches and comparisons write out the results from the part of the search that is complete when you use <Ctrl>C. The graphics device should stop plotting the current page and start plotting the next page. If the current page is the last page, plotters should put the pen away and graphic terminals should return to interactive mode.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % testcode [-INfile=]bacterial:ecoompa -Default

Prompted Parameters:

-BEGIN=1 -END=2270    sets the range of interest
-REVerse             uses the reverse strand
-WINDow=200           sets the window size
-DENSity=2270         sets the density in bp per 100 platen units
Local Data Files:

-\texttt{MARk=ecoompa.mrk} marks the plot with regions of known interest

Optional Parameters:

-\texttt{INCrement=3} lets you set the window slide increment
-\texttt{POInts} makes points instead of a curve

All GCG graphics programs accept these and other switches. See the Using Graphics chapter of the USERS GUIDE for descriptions.

-\texttt{FIGure[=filename]} stores plot in a file for later input to FIGURE
-\texttt{FONT=3} draws all text on the plot using font 3
-\texttt{COLor=1} draws entire plot with pen in stall 1
-\texttt{SCALE=1.2} enlarges the plot by 20 percent (zoom in)
-\texttt{XPAN=10.0} moves plot to the right 10 platen units (pan right)
-\texttt{YPAN=10.0} moves plot up 10 platen units (pan up)
-\texttt{PORtrait} rotates plot 90 degrees

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like \texttt{-DATa1=myfile.dat}. For more information see Chapter 4, Using Data Files in the User's Guide.

If you are studying a sequence with known features, this program can mark the plot with small boxes showing the positions of these features. The presence of a file in your directory with the same name as your sequence and the filename extension .mrk causes the program to mark each range specified in the file. You can provide a marking file on the command line with an expression like \texttt{-MARk=gamma.mrk}. The file gamma.mrk contains information about the format of marking files. The figure for the example session shows marked regions.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

\texttt{-WINdow=200}

Sets the width of the window, in bases, used for each TestCode measurement. The default is 200.

\texttt{-DENsity=1000}

Sets the number of bases or amino acids per 100 platen units (PU). This is usually equivalent to the number of bases or amino acids per page. Output from different
GCG graphics programs that are run at the same density can be compared by lining up the plots on a light box.

-MARK=ecoompa.mrk

If you are studying a sequence with known features, this program can mark the plot with small boxes showing the positions of these features. The presence of a file in your directory with the same name as your sequence and the file name extension .mrk causes the program to mark each range specified in the file. The file gamma.mrk contains information about the format of marking files.

-INCREMENT=3

Allows you to set the distance that the window is moved after each TestCode measurement. The default is three.

-POINTS

Causes TestCode to plot unconnected points instead of a continuous line.

The parameters below apply to all GCG graphics programs. These and many others are described in detail in Chapter 5, Using Graphics of the User's Guide.

-Figure=programname.figure

Writes the plot as a text file of plotting instructions suitable for input to the Figure program instead of sending it to the device specified in your graphics configuration.

-FONT=3

Draws all text characters on the plot using Font 3 (see Appendix I).

-COLOR=1

Draws the entire plot with the pen in stall 1.

The parameters below let you expand or reduce the plot (zoom), move it in either direction (pan), or rotate it 90 degrees (rotate).

-SCALE=1.2

Expands the plot by 20 percent by resetting the scaling factor (normally 1.0) to 1.2 (zoom in). You can expand the axes independently with -XSCALE and -YSCALE. Numbers less than 1.0 contract the plot (zoom out).

-XPAN=30.0

Moves the plot to the right by 30 platen units (pan right).
-YPAN=30.0

Moves the plot up by 30 platen units (pan up).

- POR trait

Rotates the plot 90 degrees. Usually, plots are displayed with the horizontal axis longer than the vertical (landscape). Note that plots are reduced or enlarged, depending on the platen size, to fill the page.

Printed: February 10, 2006 19:41
TFASTA

FUNCTION

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

DESCRIPTION

TFastA uses the method of Pearson and Lipman (Proc. Natl. Acad. Sci. USA 85: 2444-2448 (1988)) to search for similarities between a query protein sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six frames before performing the comparison. Each translated reading frame is treated as a separate sequence to be searched.

In the first step of this search, the comparison can be viewed as a set of dot plots, with the query as the vertical sequence and the group of sequences to which the query is being compared as the different horizontal sequences. This first step finds the registers of comparison (diagonals) having the largest number of short perfect matches (words) for each comparison. In the second step, these "best" regions are rescored using a scoring matrix that allows conservative replacements, ambiguity symbols, and runs of identities shorter than the size of a word. In the third step, the program checks to see if some of these initial highest-scoring diagonals can be joined together. Finally, the search set sequences with the highest scores are aligned to the query sequence for display.

What is a Word?

A word is any short sequence (n-mer or k-tuple) where you have set n to some small integer less than or equal to six. The word GGATGG is one of the 4,096 possible words of length six that can be created from an alphabet consisting of the four letters G, A, T, and C. The word QL is one of the 400 possible words of length two that you can make with the 20 letters of the amino acid alphabet.

EXAMPLE

Here is a session using TFastA to identify sequences in the GenBank nucleotide sequence database that contain translated regions similar to a human globin protein:

% tfasta

TFASTA with what query sequence ? ggamma.pep

Removing terminal * from query sequence...

Begin (* 1 *) ?
End (* 147 *) ?
Search for query in what sequence(s) (* GenBank:* *) ?

What word size (* 2 *) ?

Do not show scores whose E() value exceeds: (* 10.0 *):

What should I call the output file (* ggamma.tfasta *) ?

1 Sequences       1,497 aa searched    GB_BA:A16SRRNA
501 Sequences     908,944 aa searched    GB_BA:AB004059

----------------------------------------------------------------------

CPU time used:
Database scan:  0:30:17.2
Post-scan processing:  0:00: 7.0
Total CPU time:  0:30:24.2

Output file: ggamma.tfasta

% 

OUTPUT

The output from TFastA is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

Here is some of the output file:

!!SEQUENCE_LIST 1.0

(Peptide) TFASTA of: ggamma.pep  from: 1 to: 147  September 23, 1998 15:27

TRANSLATE of: gamma.seq check: 6474 from: 2179 to: 2270
    and of: gamma.seq check: 6474 from: 2393 to: 2615
    and of: gamma.seq check: 6474 from: 3502 to: 3630
generated symbols 1 to: 148.

Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies, Cell 26; 191-203. . . .

TO: GENBANK:*  Sequences:    552,297  Symbols: 1,036,534,764
Word Size: 2

Sequences too short to analyze: 26 (118 symbols)
Databases searched:

Searching all six frames.
Scoring matrix: GenRunData:Blosum50.Cmp
Variable pamfactor used
Gap creation penalty: 16  Gap extension penalty: 2
Histogram Key:
Each histogram symbol represents 1010 search set sequences
Each inset symbol represents 15 search set sequences
z-scores computed from opt scores

<table>
<thead>
<tr>
<th>z-score</th>
<th>obs</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; 20)</td>
<td>762</td>
<td>0:=</td>
</tr>
<tr>
<td>22</td>
<td>287</td>
<td>0:=</td>
</tr>
<tr>
<td>24</td>
<td>354</td>
<td>1:*</td>
</tr>
<tr>
<td>26</td>
<td>580</td>
<td>12:*</td>
</tr>
<tr>
<td>28</td>
<td>196</td>
<td>125:*</td>
</tr>
<tr>
<td>30</td>
<td>307</td>
<td>760:*</td>
</tr>
<tr>
<td>32</td>
<td>983</td>
<td>2937:=*</td>
</tr>
<tr>
<td>34</td>
<td>2854</td>
<td>7965:==*</td>
</tr>
<tr>
<td>36</td>
<td>8036</td>
<td>16358:==*</td>
</tr>
<tr>
<td>38</td>
<td>16819</td>
<td>27033:*</td>
</tr>
<tr>
<td>40</td>
<td>30999</td>
<td>37709:*</td>
</tr>
<tr>
<td>42</td>
<td>45853</td>
<td>46095:*</td>
</tr>
<tr>
<td>44</td>
<td>55899</td>
<td>50847:*</td>
</tr>
<tr>
<td>46</td>
<td>60552</td>
<td>51789:==*</td>
</tr>
<tr>
<td>48</td>
<td>59619</td>
<td>49582:==*</td>
</tr>
<tr>
<td>50</td>
<td>52834</td>
<td>45243:*</td>
</tr>
<tr>
<td>52</td>
<td>44792</td>
<td>39776:*</td>
</tr>
<tr>
<td>54</td>
<td>37576</td>
<td>33976:*</td>
</tr>
<tr>
<td>56</td>
<td>29109</td>
<td>28380:*</td>
</tr>
<tr>
<td>58</td>
<td>22227</td>
<td>23300:*</td>
</tr>
<tr>
<td>60</td>
<td>16395</td>
<td>18874:*</td>
</tr>
<tr>
<td>62</td>
<td>12883</td>
<td>15132:*</td>
</tr>
<tr>
<td>64</td>
<td>9798</td>
<td>12034:*</td>
</tr>
<tr>
<td>66</td>
<td>8069</td>
<td>9511:*</td>
</tr>
<tr>
<td>68</td>
<td>7098</td>
<td>7481:*</td>
</tr>
<tr>
<td>70</td>
<td>5521</td>
<td>5863:*</td>
</tr>
<tr>
<td>72</td>
<td>4551</td>
<td>4581:*</td>
</tr>
<tr>
<td>74</td>
<td>3626</td>
<td>3572:*</td>
</tr>
<tr>
<td>76</td>
<td>2725</td>
<td>2780:*</td>
</tr>
<tr>
<td>78</td>
<td>2267</td>
<td>2161:*</td>
</tr>
<tr>
<td>80</td>
<td>1763</td>
<td>1678:*</td>
</tr>
<tr>
<td>82</td>
<td>1463</td>
<td>1284:*</td>
</tr>
<tr>
<td>84</td>
<td>1119</td>
<td>1017:*</td>
</tr>
<tr>
<td>86</td>
<td>801</td>
<td>787:*</td>
</tr>
<tr>
<td>88</td>
<td>630</td>
<td>609:*</td>
</tr>
<tr>
<td>90</td>
<td>487</td>
<td>471:*</td>
</tr>
<tr>
<td>92</td>
<td>423</td>
<td>364:*</td>
</tr>
<tr>
<td>94</td>
<td>330</td>
<td>282:*</td>
</tr>
<tr>
<td>96</td>
<td>228</td>
<td>218:*</td>
</tr>
<tr>
<td>98</td>
<td>141</td>
<td>169:*</td>
</tr>
<tr>
<td>100</td>
<td>154</td>
<td>131:*</td>
</tr>
<tr>
<td>102</td>
<td>122</td>
<td>101:*</td>
</tr>
<tr>
<td>104</td>
<td>80</td>
<td>78:*</td>
</tr>
<tr>
<td>106</td>
<td>64</td>
<td>61:*</td>
</tr>
<tr>
<td>108</td>
<td>61</td>
<td>47:*</td>
</tr>
</tbody>
</table>
Joining threshold: 36, opt. threshold: 36, opt. width: 16, reg.-scaled

The best scores are:

<table>
<thead>
<tr>
<th>Frame</th>
<th>Init1</th>
<th>Initn</th>
<th>Opt</th>
<th>Z-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>E(...)..</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GB_PR2:HUMHBGG

! M15386 Human hemoglobin gamma-G (HB...)(3)  971 971 971 1547.8 1.2e-78

GB_PR1:HSGGGPHG

! X55656 H.sapiens mRNA for gamma-G g... (2)  843 843 843 1343.8 2.7e-67

GB_PAT:I42109

! I42109 Sequence 4 from patent US 56... (3)  765 765 776 1238.0 2.1e-61

\\End of List

ggamma.pep

GB_PR2:HUMHBGG

LOCUS HUMHBGG 545 bp mRNA PRI 24-MAR-1997
DEFINITION Human hemoglobin gamma-G (HBG2) mRNA, partial cds.
ACCESSION M15386
NID g183884
KEYWORDS .
SOURCE human. . .

SCORES Frame: (3) Init1: 971 Initn: 971 Opt: 971 z-score: 1547.8 E(): 1.2e-78
100.0% identity in 147 aa overlap

<table>
<thead>
<tr>
<th>GGFTEEDKATITSLWGVHNLSSASAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUMHBGG</td>
</tr>
<tr>
<td>PFPDAMGFTEEDKATITSLWGVHNLSSASAI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GGFTEEDKATITSLWGVHNLSSASAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUMHBGG</td>
</tr>
<tr>
<td>PFPDAMGFTEEDKATITSLWGVHNLSSASAI</td>
</tr>
</tbody>
</table>

192 Program Manual
What is the Output?

The first part of the output file contains a histogram showing the distribution of the z-scores between the query and search set sequences. (See the ALGORITHM topic for an explanation of z-score.) The histogram is composed of bins of size 2 that are labeled according to the higher score for that bin (the leftmost column of the histogram). For example, the bin labeled 24 stores the number of sequence pairs that had scores of 23 or 24.

The next two columns of the histogram list the number of z-scores that fell within each bin. The second column lists the number of z-scores observed in the search and the third column lists the number of z-scores that were expected.

The body of the histogram displays a graphical representation of the score distributions. Equal signs (=) indicate the number of scores of that magnitude that were observed during the search, while asterisks (*) plot the number of scores of that magnitude that were expected.

At the bottom of the histogram is a list of some of the parameters pertaining to the search.

Below the histogram, TFastA displays a listing of the best scores. This listing includes the reading frame in the original nucleotide sequence from which the reported translated sequence is derived.
Following the list of best scores, TFastA displays the alignments of the regions of best overlap between the query and search sequences. In these alignments, stop codons are represented by the letter X.

This program displays only the region of overlap between the two aligned sequences (plus some residues on either side of the region to provide context for the alignment) unless you use \texttt{-SHOW\text{**}}. The display of identities and conservative replacements between the aligned sequences depends on the value of \texttt{-MARK\text{**}}. By default \texttt{(-MARK\text{**}=3)}, the pipe character (|) is used to denote identities and the colon (: ) to denote conservative replacements.

**INPUT FILES**

TFastA accepts a single protein sequence as the query sequence. The search set is either a single nucleic acid sequence or multiple nucleic acid sequences. You can specify multiple sequences in a number of ways: by using a list file, for example \texttt{@project.list}; by using an MSF or RSF file, for example \texttt{project.msf{(*)}}; or by using a sequence specification with an asterisk (*) wildcard, for example \texttt{GenBank:*}. If TFastA rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence.

**RELATED PROGRAMS**

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"
FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

ProfileSearch and MotifSearch use a profile (derived from a set of aligned sequences) instead of a query sequence to search a collection of sequences. FindPatterns uses a pattern described by a regular expression to search a collection of sequences. HmmerSearch uses a profile hidden Markov model as a query to search a sequence database to find sequences similar to the family from which the profile HMM was built. Profile HMMs can be created using HmmerBuild.

StringSearch, LookUp, and Names identify sequences by searching the annotation (non-sequence) portions of sequence files or sequence databases.

FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST+ and FastA+, it can be very slow.

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"
FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX+ translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

**RESTRICTIONS**

The query sequence may not be longer than 32,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The word size must be either 1 or 2.

For the estimates of statistical significance to be valid, the search set must contain a large sample of unrelated sequences. The statistical estimates will not be calculated at all if there are fewer than 60 frames searched (equivalent to 10 sequences when all six frames are searched, or 20 sequences when only the top three frames are searched).

With `-NOOPT all`, the estimates of statistical significance will not be accurate.

For Tru64 (OSF) TFastA fails with an error message:

"While running the child process: Child was terminated by signal 6 (SIGABRT)" Error in cleaning up after application: Exception: Error reading fast program output: Unable to open tfasta output file: "/tmp/bslskAAAMGXMCF.tmp" (at /tmp/bslskAAAMGXMCF.tmp:0)."

**Workaround**

There is an upper limit on the amount of memory that is allocated per process. For tru64 machine the limit for datasize is set to 128M. To increase this limit, execute

```
> unlimit datasize (csh) or
> ulimit datasize (ksh)
```

This will increase the limit on the datasize to 1024M. This is the maximum amount of memory that an individual process can take on Tru64 machine. So, default settings for the search set parameter (-infile2) for the fasta suite of programs may cause a crash. Please execute the programs with a smaller subset. The programs have been tested successfully using a search set of 400 thousand sequences.

**ALGORITHM**

For a description of the algorithm, see the FastA program documentation.

**CONSIDERATIONS**

TFastA treats each reading frame as a different sequence. If a nucleotide sequence contains a gene coding for a protein similar to your query, but with an intervening
sequence that changes the reading frame, the program will find and display two matches, one for each reading frame. If the individual matches each have fairly low scores, they may not make the list of best scores. If you suspect that the gene for your query sequence contains intervening sequences, or if you are searching a nucleotide database known to contain sequencing errors that may cause a frameshift (such as the EST division of GenBank), use **TFastX** instead of **TFastA**.

**TFastA** translates stop codons in search set sequences to the sequence symbol X.

The E() scores are affected by similarities in sequence composition between the query sequence and the search set sequence. Unrelated sequences may have "significant" scores because of composition bias.

If there is a database entry that overlaps your query in several places, but there are large gaps between the matching regions, only the best overlap appears in the alignment display.

There are two ways to control the size of the list of best scores. By default, scores are listed until a specific E() value is reached. You may set the value in response to the program prompt or by using **-EXP ect**; otherwise the program uses 10.0 for protein searches, 2.0 for nucleic acid searches. (If you are running the program interactively, it will show no more than 40 scores initially, and ask if you want to see more scores if there are any more that are less than the E() value.)

If you use **-LISTsize**, the E() value is ignored, and the program will list the number of scores you requested.

You can control the number of alignments using **-NOALIgn** and **-ALIgn**. The program behaves differently depending on whether it is being run noninteractively (in batch or with **-Default** on the command line) or interactively. In the noninteractive case, the program displays the number of alignments set by **-ALI gn**. (If this is not present, it shows 40 alignments or the number of scores that were listed, whichever is smaller.) If you run the program interactively, it displays the list of best scores, then asks you how many alignments you want to see. (This prompt does not appear if you use **-NOALI gn** or **-ALI gn**.)

**Increasing Sensitivity By Adjusting Word Size**

By default, **TFastA** uses a word size of 2. If it finds few or no matches, especially if your query sequence is short, rerun the search using **-WORDsize=1** to increase the sensitivity. Note that this will dramatically increase the amount of CPU time required to run the program.

**Adjusting Gap Creation and Extension Penalties**

Unlike other GCG programs, **TFastA** does not read the default gap creation and gap extension penalties from the scoring matrix file. It uses default gap creation and extension penalties that were empirically determined to be appropriate for the BLOSUM50 scoring matrix. If you select a different scoring matrix with **-MATR ix**, 

---

*Accelrys GCG 197*
you may need to change the gap penalties. The histogram display gives a qualitative view of the quality of fit between the actual distribution of scores and the expected distribution of scores. This information may indicate whether or not suitable gap creation and extension penalties were used for the search. When the histogram shows poor agreement between the actual distribution and the theoretical distribution, you might consider using -GAPweight and/or -LENgthweight to specify higher gap creation and extension penalties, respectively. For example, you might increase the gap creation penalty from 16 to 20 and the gap extension penalty from 4 to 6.

**Differences in Applying Gap Extension Penalties**

There are two different philosophies on how to penalize gaps in an alignment. One way is to penalize a gap by the gap creation penalty plus the extension penalty times the length of the gap \((\text{gapweight} + (\text{lengthweight} \times \text{gap length}))\). The other way is to use the gap creation penalty plus the extension penalty times the gap length excluding the first residue in the gap \((\text{gapweight} + (\text{lengthweight} \times (\text{gap length} - 1)))\).

"Native" GCG programs, such as Framesearch and Bestfit, handle gap extension penalties the first way, while the FastA-family programs use the second way. Therefore a value for -LENgthweight that gives good results with one of the FastA-family programs may not give equivalent results with a native GCG program, and vice versa.

**Increasing Program Speed Using Multithreading**

This program is multithreaded. It has the potential to run faster on a machine equipped with multiple processors because different parts of the analysis can be run in parallel on different processors. By default, the program assumes you have one processor, so the analysis is performed using one thread. You can use -PROCessors to increase the number of threads up to the number of physical processors on the computer.

Under ideal conditions, the increase in speed is roughly linear with the number of processors used. But conditions are rarely ideal. If your computer is heavily used, competition for the processors can reduce the program's performance. In such an environment, try to run multithreaded programs during times when the load on the system is light.

As the number of threads increases, the amount of memory required increases substantially. You may need to ask your system administrator to increase the memory quota for your account if you want to use more than two threads.

Never use -PROCessors to set the number of threads higher than the number of physical processors that the machine has -- it does not increase program performance, but instead uses up a lot of memory needlessly and makes it harder for other users on the system to get processor time. Ask your system administrator how many processors your computer has if you aren't sure.
SUGGESTIONS

Identifying the Search Set

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" topic of Chapter 2, Using Sequence Files and Databases of the User's Guide for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. For information about naming groups of sequences for the search set, see the topics "Specifying Files" and "Using Wildcards" in Chapter 1, Getting Started, and "Using Database Sequences," "Using Multiple Sequence Format (MSF) Files", "Using Rich Sequence Format (RSF) Files", and "Using List Files" in Chapter 2, Using Sequence Files and Databases of the User's Guide.

Batch Queue

TFastA is one of the few programs in the GCG that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using -BATch. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

Interrupting a Search: <Ctrl>C

You can type <Ctrl>C to interrupt a search and see the results from the part of the search that has already been completed. Because the program is multithreaded, the search may not be interrupted immediately, but will continue until one of the threads finishes processing its data and returns for more data.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % tfasta [-INfile1=]ggamma.pep -Default

Promoted Parameters:

[-INfile2=]GenBank:* specifies search set
[-OUTfile=]ggamma.tfasta names the output file
-BEGIN=1 -END=148 sets the range of interest
-WORDsize=2 sets the word size
-EXPect=2.0 lists scores until E() value reaches 2.0
Local Data Files:

-MATRix=blosum50.cmp assigns the scoring matrix for proteins

Optional Parameters:

-PROCessors=2 sets the number of threads devoted to the analysis on a multiprocessor computer
-MINLength=1000 searches only sequences of 1000 or more residues
-MAXLength=5000 searches only sequences of 5000 or fewer residues
-SINce=6.90 limits search to sequences dated on or after June 1990
-DBTOPstrand translates and searches only the three forward reading frames of the search set sequences
-DBBOTtomstrand translates and searches only the three reverse reading frames of the search set sequences
-NOPAMfactor uses a constant factor to calculate initial diagonal scores
-GAPweight=16 sets the gap creation penalty
-LENgthweight=2 sets the gap extension penalty
-OPTall=20 computes opt score when the inith score is 20 or higher; sorts on opt score
-NOOPTall doesn't compute opt score during search; sorts on inith
-SWalign creates final alignment as unlimited Smith-Waterman
-LISTsize=40 shows the best 40 scores (overrides EXPect)
-ALIgn=20 shows the best 20 alignments
-NOALIgn suppresses sequence alignments
-SHOWall shows complete sequences in alignment, not just overlaps
-MARKx=3 sets the alignment display mode
-NOHISTogram suppresses printing the histogram
-LINEsize=60 sets number of sequence symbols per line of the alignment
-NODOCLines suppresses sequence documentation in the alignment
-BATch submits the program to run in the batch queue
-NOMONitor suppresses the screen trace for each search set sequence

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DAT1=myfile.dat. For more information see Chapter 4, Using Data Files in the User’s Guide.
Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name MyData; or 3) name a file on the command line with an expression like `-MATRix=mymatrix.cmp`. If you do not include a directory specification when you name a file with `-MATRix`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

TFastA reads a scoring matrix containing the values for every possible match from your working directory or the public database. The default matrix is blosum50.cmp, which is a BLOSUM50 matrix. You can use the Fetch program to obtain a copy of this file if you need to modify it for your own needs.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- **WORdsize=2**

  Sets the size of the word (k-tuple) to use for the hashing step.

- **MATRix=mymatrix.cmp**

  Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with `-MATRix`, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

  For more information see the Local Scoring Matrices section.

- **CHEck**

  Prints out this usage message.

- **DEFault**

  Specifies that sensible default values be used for all parameters where possible.
-DOCDumentation

Prints banner at program startup.

-QUIet

Tells application to print only a minimal amount of information.

-ALIGNments

This parameter limits the number of alignments to display in the output file to the 10 best matches in the list. Use -NOALIgn to suppress the sequence alignments in the output file.

-HISTogram

Start/suppress printing the histogram.

-EXPect=2.0

Shows all scores whose E() value is less than 2.0. Ignored if -LISTsize is used.

-PROCESSors=2

Tells the program to use 2 threads for the database search on a multiprocessor computer.

-PAMfactor

This parameter governs whether a scoring matrix should be used for calculating initial diagonal scores, instead of using the identical match scores from the scoring matrix.

Default is to use FASTA internal behavior, which differs for protein and nucleotide searches

-MINLength=1000

Restricts the search to search set sequences that are equal to or longer than 1000 residues.

-MAXLength=5000

Restricts the search to search set sequences that are equal to or shorter than 5000 residues.
-SIN<sub>ce</sub>=6.1990

Limits the search to sequences that have been entered into the database or modified since June 1990. As this is being written, only the EMBL, GenBank, and SWISS-PROT databases support this parameter.

-DBTOP<sub>strand</sub>

Translates and searches only the three forward reading frames.

-DBBOT<sub>tomstrand</sub>

Translates and searches only the three reverse complement reading frames.

-NOPAM<sub>factor</sub>

Uses a constant factor for the calculation of initial diagonal scores, instead of using the identical match scores from the scoring matrix.

-GAPweight=12

Specifies the gap creation penalty that is subtracted from the alignment score whenever a gap is created.

-LENgLenthweight=2

Specifies the gap extension penalty that is subtracted from the alignment score for each residue added to an existing gap.

-OPTall=20

Immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to 20. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search. -NOOPTall doesn't compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.

-SWalign

Does an unlimited Smith-Waterman alignment as the final alignment for TFastA+ searches, instead of the "alignment in a band" version of Smith-Waterman. (Note: this can be very slow.)

-LISLStsize=40

Shows the best 40 scores. Overrides -EXPect.
-ALIGN=10

Limits the number of alignments to display in the output file to the 10 best matches in the list. Use the -NOALIGN to suppress the sequence alignments in the output file.

-SHOWALL

Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-MARKx=3

Determines the alignment display mode -- especially the symbols that identify matches and mismatches. The default value, 3, uses a pipe character (|) to show identities and a colon (:) to show conservative replacements. -MARKx=0 uses a colon to show identities and a period (.) to show conservative replacements. -MARKx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X. If -MARKx=2, the residues in the second sequence are shown only if they differ from the first sequence.

Use -MARKx=10 to get aligned sequences in the FastA "parsable" output format. A document describing this format appears after FastA in the Program Manual.

-NOHISTOGRAM

Suppresses printing the histogram.

-LINESIZE=60

Lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-NODOCLINES

Suppresses the documentation from the search set sequence accompanying the alignment in the output file. Use -DOCLINES=5 to copy only five non-blank lines of documentation.

-BATCH

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.
-MONitor=500

Monitors this program's progress on your screen. Use this parameter to see this same monitor in the log file for a batch process. If the monitor is slowing down the program because your terminal is connected to a slow modem, suppress it with -NOMONitor.

The monitor is updated every time the program processes 500 sequences or files. You can use a value after the parameter to set this monitoring interval to some other number.
TFASTA+

FUNCTION

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

DESCRIPTION

Advantages of Plus “+” Programs:

✓ Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

✓ Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

TFastA+ uses the method of Pearson and Lipman (Proc. Natl. Acad. Sci. USA 85; 2444-2448 (1988)) to search for similarities between a query protein sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six frames before performing the comparison. Each translated reading frame is treated as a separate sequence to be searched.

In the first step of this search, the comparison can be viewed as a set of dot plots, with the query as the vertical sequence and the group of sequences to which the query is being compared as the different horizontal sequences. This first step finds the registers of comparison (diagonals) having the largest number of short perfect matches (words) for each comparison. In the second step, these "best" regions are rescoring using a scoring matrix that allows conservative replacements, ambiguity symbols, and runs of identities shorter than the size of a word. In the third step, the program checks to see if some of these initial highest-scoring diagonals can be joined together. Finally, the search set sequences with the highest scores are aligned to the query sequence for display.

What is a Word?

A word is any short sequence (n-mer or k-tuple) where you have set n to some small integer less than or equal to six. The word GGATGG is one of the 4,096 possible words of length six that can be created from an alphabet consisting of the four letters G, A, T, and C. The word QL is one of the 400 possible words of length two that you can make with the 20 letters of the amino acid alphabet.
EXAMPLE

Here is a session using TFastA+ to identify sequences in the GenBank nucleotide sequence database that contain translated regions similar to a human globin protein:

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

tfasta+ with what query sequence(s) ? ggamma.pep
Begin (* 1 *) ?

End (-1 for entire sequence) (* -1 *) ?
Enter value for search set (*Default DB*) ?
What should I call the output file (* <sequence_name>.<program_name> *) ?

# $GCGROOT/bin/tfasta34_native -O /var/tmp/bslskBAAO2a4Ro.tmp -E 2.0 -b 10 -T 1 /var/tmp/bslskBAAAQ2a4Ro.fa "Genbank:* 17"
TFASTA translates and searches a DNA sequence data bank
version 3.4t21 May 14, 2003
Please cite:

Query library /var/tmp/bslskBAAAQ2a4Ro.fa vs Genbank:* library
searching Genbank:* 17 library

1>>>GGAMMA.PEP TRANSLATE of: ggamma.seq check: 7694 from: 1 to: 1700
- 566 aa
vs Genbank:* library

103380 residues in 14 sequences
MLE_cen statistics: Lambda= 0.1223; K=0.0006321 (cen=0)

TFASTA (3.45 Mar 2002) function [optimized, BL50 matrix (15:-5)] ktup:
2
join: 37, opt: 37, open/ext: -14/-2, width: 16
Scan time: 0.490
The best scores are: opt bits
E(14)
AB107101 AB107101 Homo sapiens chromosome 3 cl (91914) [4] 129 33.4
0.065
AB107101 AB107101 Homo sapiens chromosome 3 cl (91914) [5] 128 33.2
0.073
AB107101 AB107101 Homo sapiens chromosome 3 cl (91914) [6] 108 29.7
0.82
A00001 A00001 Cauliflower mosaic virus satelli (335) [2] 56 20.5
1.7
A00001 A00001 Cauliflower mosaic virus satelli (335) [6] 49 19.3
3.6
AAB2MCG1 AF032092 Aotus azarai beta-2-microglo (289) [3] 46 18.7
4.4
AB000556 AB000556 Synthetic unidentified bacte (924) [2] 54 20.2
5.1
A165RRNA X87617 Actinomycete (genus unknown) 1 (1497) [3] 55 20.3
6.6
AA12SRRNA X67626 A.australis mitochondrial gen (386) [6] 43 18.2
7.2
AAG311130 AJ311130 Apodemus agrarius mitochond (955) [3] 48 19.1
8.7

566 residues in 1 query sequences
103380 residues in 14 library sequences
Scomplib [34t21]
  Total Scan time:  0.490 Total Display time:  2.310

Function used was TFASTA [version 3.4t21 May 14, 2003]

OUTPUT

!!SEQUENCE_LIST 1.0
# $GCGROOT/bin/tfasta34_native -O
/var/tmp/bs1skBAAO2a4Ro.tmp -E 2.0 -b 10 -T 1
/var/tmp/bs1skDAAQ2a4Ro.fa "Genbank:* 17"
TFASTA translates and searches a DNA sequence data bank
version 3.4t21 May 14, 2003
Please cite:
GGAMMA.PEP, 566 aa
vs Genbank:* library

103380 residues in 14 sequences
MLE_cen statistics: Lambda= 0.1223;  K=0.0006321 (cen=0)

TFASTA (3.45 Mar 2002) function [optimized, BL50 matrix
(15:-5)] ktup: 2
  join: 37, opt: 37, open/ext: -14/-2, width: 16
The best scores are:
opt bits E(14)

..
Genbank:AB107101 Begin: 9933 End: 9871
! AB107101 Homo sapiens chromosome 3 clone 15N2...(4) 125
161 129 87.4 0.065 Genbank:AB107101 Begin: 9932 End: 9879
! AB107101 Homo sapiens chromosome 3 clone 15N2...(5) 128
248 128 86.5 0.073 Genbank:AB107101 Begin: 9955 End: 9887
! AB107101 Homo sapiens chromosome 3 clone 15N2...(6) 107
241 108 67.4 0.82 Genbank:A00001 Begin: 80 End: 214
! A00001 Cauliflower mosaic virus satellite cDNA....(2) 43
43 56 61.6 1.7 Genbank:A00001 Begin: 282 End: 217
! A00001 Cauliflower mosaic virus satellite cDNA....(6) 49
49 49 54.9 3.6 Genbank:A00001 Begin: 129 End: 188
! AF032092 Aotus azarai beta-2-microglobulin pr...(3) 40 40
46 53.2 4.4
Genbank:A0000556 Begin: 485 End: 637
! AB000556 Synthetic unidentified bacterium/pla...(2) 49 97
54 51.8 5.1
Genbank:A16SRRNA Begin: 663 End: 725
! X87617 Actinomycete (genus unknown) 16S ribos...(3) 51 71
55 49 6.6
Genbank:AA12SRRNA Begin: 366 End: 304
! X67626 A.australis mitochondrial gene for 12...(6) 43 43
43 48.1 7.2
Genbank:AAG311130 Begin: 576 End: 638
! AJ311130 Apodemus agrarius mitochondrial 12S...(3) 48 48
48 45.8 8.7
\End of List

>>Genbank:AB107101 AB107101 Homo sapiens chromosome 3 clone
15N2 (91914 aa)
Frame: 4 initn: 161 init1: 125 opt: 129  Z-score: 87.4
bits: 33.4 E(): 0.065
banded Smith-Waterman score: 129;  60.870% identity
(66.667% ungapped) in 23 aa overlap (364-386:9933-9871)

390       350       360       370       380
GGAMMA
PLIPDGGKVCPCGVRNN*NWAGVDVESQLCVCVCVCRVCLVCVCSVCF*RFQPTAYR
V

AB1071  LYWEKTLGKIRSSQCFEDEKLARGN*FVWVCVCVC--
CVCVCLSVCLCYSFIRCAG*E
10000   9970   9940   9910   9880

400      410      420      430      440
GGAMMA
HGGKKITRFKLWPVTSAAARRTTTCI*WESKISGFEGS*HRLDSDKLGV*LSGGQAGAL
S

Accelrys GCG 209
>>Genbank:AB107101 AB107101 Homo sapiens chromosome 3 clone 15N2 (91914 aa)
bits: 33.2 E(): 0.073
banded Smith-Waterman score: 128; 77.778% identity
(77.778% ungapped) in 18 aa overlap (365-382:9932-9879)

340 350 360 370 380
GGAMMA
LIPDGGKVCPCGVRNN*NIWAGVDFESQLCVCVCVCARVCLVCESVCFF*RFQPTAYRV
H

400 410 420 430 440
GGAMMA
GGKKITRFKLWPVTSAARRTTTCI*WESKISGFEGS*HRLDSgwklGV*LSGQqAgALS
S

450

>>Genbank:AB107101 AB107101 Homo sapiens chromosome 3 clone 15N2 (91914 aa)
bits: 29.7 E(): 0.82
banded Smith-Waterman score: 108; 56.522% identity
(56.522% ungapped) in 23 aa overlap (360-382:9955-9887)

330 340 350 360 370 380
GGAMMA
TNLYPLIPDGGKVCPCGVRNN*NIWAGVDFESQLCVCVCVCARVCLVCESVCFF*RFQPT
T

... ::::: ::::: ::::: :::::
AB1071
LCFHGLHTILGEDTR*RNKISVF*RR*TDGKLVCMVCVCVCVCVCVCVCSCSLV*LL
*  10040  10010  9980  9950  9920  9890  9860  9830  9800  9770  9740  9710

390  400  410  420  430  440

GGAMMA
AYRVHGKKITRFKLPVTSAARRTTTCT*WESKISGFEGS*HRLDSGWKLGV*LSGGQ
A

AB1071
TSWLRKTSLRGDFRQ**LEQISHAKKWTTRVFQEEQIATTKAPKPE*A*QFTEVERRHVC
F

110
60  70  80  90  100

GGAMMA R*ALVTRTREGRKDPVPGKSPGRFSGFVAPSDCQTVLVLGSGW-

LSTHGPRGSLATALAT

.  .:.  ::.  :...  .:  .:.

A00001
FCLMENCAEGLYLREDLSLGGVGYLPAKAG*VMFPRGDRWLASYVRYSYQYTLI*

120  130  140  150  160

170
GGAMMA
CPLPPLPSwatPKSRHMARRC*LPWEMP*STWMSRAPLPS*VNCTVTSCMWILRTSR*V
Q

.  .:.  .:.:.:.:.:.:

A00001 APAQFASR----
TRHMVRRYHGSKETLCC*VV*VMTHAGRGGLCYADLRECLSLHLRT

170  200  230  260  290

320

>>Genbank:A00001 A00001 Cauliflower mosaic virus satellite cDNA. (335 aa)
Frame: 2 initn:  43 init1:  43 opt:  56 Z-score:  61.6
bits: 20.5 E():  1.7
banded Smith-Waterman score: 56; 24.490% identity (27.273% ungapped) in 49 aa overlap (87-134:80-214)

0  20  50  80  110  140

200  230  260  290

>>Genbank:A00001 A00001 Cauliflower mosaic virus satellite cDNA. (335 aa)
Frame: 6 initn:  49 init1:  49 opt:  49 Z-score:  54.9
bits: 19.3 E():  3.6
banded Smith-Waterman score: 49; 22.727% identity (22.727% ungapped) in 22 aa overlap (118-139:282-217)

90 100 110 120 130
GGAMMA
DCQTVLVTGSWLSTHGPRGLTALTCLPLPSWATPKSRHMMARCC*LPMEMP*STWM
:::...::...::...
A00001
VLCRNDRHSRRSA*HKP*PLPACVMTHTT*QSVSFEIPWYRRARMRV

150 160 170 180 190
GGAMMA
ISRAPLPS*VNCTVTSWMLRTSR*VQEMFHCCCL*SRGSLDSV*LLI*AQQGVSCLKL

A00001
LLAN*AGAQMRV*Y*ET*LASQRPVSGNTQPALAGRYPTPSSRRRRNPSAQFS

190 160 130 100 70 40
>>Genbank:AAB2MC G1 AF032092 Aotus azarai beta-2-microglobulin pr (289 aa)
Frame: 3 initn: 40 init1: 40 opt: 46 Z-score: 53.2
bits: 18.7 E(): 4.4
banded Smith-Waterman score: 46; 30.769% identity (42.105% ungapped) in 26 aa overlap (295-319:129-188)

270 280 290 300 310
GGAMMA
FTFPLLDVVLKHLGSGRTSMVVKMVAEGIYWLSQSGELWPNHIC*GYSYI-SWHIK
:::...:::...:::
AAB2MC ALS*LAVPDSV*HK*RRVARALLQRTTLGSRWAS---WWPC---

180
SCYSCLAWRLSS

330 340 350 360 370
380
GGAMMA
CC*CFITNLPLDPGGKVCPGVRNN*NIWAGVDFESQLCVCVCAVCLCVCESVCF
F
AAB2MC VSLSSRPALVPLPLPPSVAVSVLGSFVT
  210  240  270

>>Genbank:AB000556 AB000556 Synthetic unidentified
bacterium/pla (924 aa)
Frame: 2 initn:  97 init1:  49 opt:  54  Z-score: 51.8
bits: 20.2 E():  5.1
banded Smith-Waterman score: 54;  21.569% identity (22.449%
ingapped) in 51 aa overlap (368-416:485-637)
  340  350  360  370  380
390
GGAMMA
DGGKVCPGVRNN*NIWAGVDFESQLCVCVCAVCLCVCESVCF*RFQPTAYRVHGG
K
  :    .  : . :. :.  ..
AB0005
RRSTLKPGHTV*RVCARCASTTV*CTATNCRPTNCRLCSVSTWPSR*SKTTARASPS
S
  410  440  470  500  530  560
  400  410  420  430  440
450
GGAMMA KITRFKL--
WPVTSAARRTTTCI*WESKISGFEGS*HRLDSGWKLGV*LSGQAGALSSL
  .  : .  ::::::::
AB0005
SACRPRRTTWLPSSTRRRASSITSITSHSSWKPGKTFCAPPT*SP*PTPRSIARPVTVP*PT
A
  590  620  650  680  710  740

>>Genbank:A16SRRNA X87617 Actinomycete (genus unknown) 16S
ribos (1497 aa)
Frame: 3 initn:  71 init1:  51 opt:  55  Z-score: 49.0
bits: 20.3 E():  6.6
banded Smith-Waterman score: 55;  33.333% identity (33.333%
ingapped) in 21 aa overlap (393-413:663-725)
  370  380  390  400  410
420
GGAMMA
CVCVCAVCLCVCESVCF*RFQPTAYRVHGGKKITRFKLPVTSAAARRTTTTCI*WES
A16SRR
RA*LWACSGYGQARVW*GRLEFLV*R*NAQISGGTPVAKAGLWATDDEERKHGERTGLD

600       630       660       690       720
750
480
GGAMMA
KISGFEGS*HRDLGDKWKLGV*LSGGQAGALSSLWVHLYCLLSSQQLGNNVLLVTVLAIHFG

A16SRR
TLVVHAVNVGR*VWGTFHVFCAAANALSAPPGEYGRKAKTQRN*RGPAQAAEAD*FDA

780       810       840       870       900
930

>>Genbank:AA12SRRNA X67626 A.australis mitochondrial gene
for 12 (386 aa)
bits: 18.2 E(): 7.2
banded Smith-Waterman score: 43; 28.571% identity (28.571%
ungapped) in 21 aa overlap (253-273:366-304)

230       240       250       260       270
280
GGAMMA
QWQCFRA*GVPLKI*MDNFDPEKREKVEEMKMTFLY*ISVERTFTFPLLLFVLKHLSSGG
R

AA12SR
PRAGLK*TFLFCFTAKSSF*GGFHTLFRSILF*KM*

370       340       310
280

590       600       610       620       630
640
340
GGAMMA
TSMVVKKMQAEGIYWLSQSGELWPNIHC*GYSYISWTHIKCC*CFITNLYPLIPDGK
V

AA12SR
PISPISWAIP*PVLLAGVAVVLAALSF*AGWRRRYVGCVCGRMLGVSVIDYRTGSSRW
V
## INPUT FILES

TFasta+ accepts a single protein sequence as the query sequence. The search set is either a single nucleic acid sequence or multiple nucleic acid sequences. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example Genbank:.*.
If TFastA+ rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST+ and FastA+, it can be very slow.

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like TFastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX+ translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.
WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

ProfileSearch and MotifSearch use a profile (derived from a set of aligned sequences) instead of a query sequence to search a collection of sequences. FindPatterns+ uses a pattern described by a regular expression to search a collection of sequences. HmmerSearch uses a profile hidden Markov model as a query to search a sequence database to find sequences similar to the family from which the profile HMM was built. Profile HMMs can be created using HmmerBuild.

StringSearch, LookUp, and Names identify sequences by searching the annotation (non-sequence) portions of sequence files or sequence databases.

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like TFastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

**RESTRICTIONS**

The query sequence may not be longer than 20,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The word size must be either 1 or 2.

For the estimates of statistical significance to be valid, the search set must contain a large sample of unrelated sequences. The statistical estimates will not be calculated at
all if there are fewer than 60 frames searched (equivalent to 10 sequences when all six 
frames are searched, or 20 sequences when only the top three frames are searched).

With `-nooptall`, the estimates of statistical significance will not be accurate.

**ALGORITHM**

For a description of the algorithm, see the FastA+ program documentation.

**CONSIDERATIONS**

TFastA+ treats each reading frame as a different sequence. If a nucleotide sequence 
contains a gene coding for a protein similar to your query, but with an intervening 
sequence that changes the reading frame, the program will find and display two 
matches, one for each reading frame. If the individual matches each have fairly low 
scores, they may not make the list of best scores. If you suspect that the gene for your 
query sequence contains intervening sequences, or if you are searching a nucleotide 
database known to contain sequencing errors that may cause a frameshift (such as the 
EST division of GenBank), use TFastX+ instead of TFastA+.

TFastA+ translates stop codons in search set sequences to the sequence symbol X.

The E() scores are affected by similarities in sequence composition between the query 
sequence and the search set sequence. Unrelated sequences may have "significant" 
scores because of composition bias.

If there is a database entry that overlaps your query in several places, but there are large 
gaps between the matching regions, only the best overlap appears in the alignment 
display.

There are two ways to control the size of the list of best scores. By default, scores are 
listed until a specific E() value is reached. You may set the value in response to the 
program prompt or by using `-expect`; otherwise the program uses 10.0 for protein 
searches, 2.0 for nucleic acid searches. (If you are running the program interactively, it 
will show no more than 40 scores initially, and ask if you want to see more scores if 
there are any more that are less than the E() value.)

If you use `-listsized`, the E() value is ignored, and the program will list the number 
of scores you requested.

You can control the number of alignments using `-noalign` and `-align`. The 
program behaves differently depending on whether it is being run noninteractively (in 
batch or with `-Default` on the command line) or interactively. In the noninteractive 
case, the program displays the number of alignments set by `-align`. (If this is not 
present, it shows 40 alignments or the number of scores that were listed, whichever is 
smaller.) If you run the program interactively, it displays the list of best scores, and then 
asks you how many alignments you want to see. (This prompt does not appear if you 
use `-noalign` or `-align`.)
Increasing Sensitivity By Adjusting Word Size

By default, TFastA+ uses a word size of 2. If it finds few or no matches, especially if your query sequence is short, rerun the search using \texttt{-wordsize=1} to increase the sensitivity. Note that this will dramatically increase the amount of CPU time required to run the program.

Adjusting Gap Creation and Extension Penalties

Unlike other GCG programs, TFastA+ does not read the default gap creation and gap extension penalties from the scoring matrix file. It uses default gap creation and extension penalties that were empirically determined to be appropriate for the BLOSUM50 scoring matrix. If you select a different scoring matrix with \texttt{-matrix}, you may need to change the gap penalties. The histogram display gives a qualitative view of the quality of fit between the actual distribution of scores and the expected distribution of scores. This information may indicate whether or not suitable gap creation and extension penalties were used for the search. When the histogram shows poor agreement between the actual distribution and the theoretical distribution, you might consider using \texttt{-gapweight} and/or \texttt{-lengthweight} to specify higher gap creation and extension penalties, respectively. For example, you might increase the gap creation penalty from 16 to 20 and the gap extension penalty from 4 to 6.

Differences in Applying Gap Extension Penalties

There are two different philosophies on how to penalize gaps in an alignment. One way is to penalize a gap by the gap creation penalty plus the extension penalty times the length of the gap ($\text{gapweight} + (\text{lengthweight} \times \text{gap length})$). The other way is to use the gap creation penalty plus the extension penalty times the gap length excluding the first residue in the gap ($\text{gapweight} + (\text{lengthweight} \times (\text{gap length} - 1))$).

"Native" GCG programs, such as Framesearch and Bestfit, handle gap extension penalties the first way, while the FastA+-family programs use the second way. Therefore a value for \texttt{-lengthweight} that gives good results with one of the FastA+-family programs may not give equivalent results with a native GCG program, and vice versa.

Increasing Program Speed Using Multithreading

This program is multithreaded. It has the potential to run faster on a machine equipped with multiple processors because different parts of the analysis can be run in parallel on different processors. By default, the program assumes you have one processor, so the analysis is performed using one thread. You can use \texttt{-processors} to increase the number of threads up to the number of physical processors on the computer.

Under ideal conditions, the increase in speed is roughly linear with the number of processors used. But conditions are rarely ideal. If your computer is heavily used, competition for the processors can reduce the program's performance. In such an environment, try to run multithreaded programs during times when the load on the system is light.
As the number of threads increases, the amount of memory required increases substantially. You may need to ask your system administrator to increase the memory quota for your account if you want to use more than two threads.

Never use `-processors` to set the number of threads higher than the number of physical processors that the machine has -- it does not increase program performance, but instead uses up a lot of memory needlessly and makes it harder for other users on the system to get processor time. Ask your system administrator how many processors your computer has if you aren't sure.

**SUGGESTIONS**

**Identifying the Search Set**

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" topic of Chapter 2, Using Sequence Files and Databases of the User's Guide for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. For information about naming groups of sequences for the search set, see the topics "Specifying Files" and "Using Wildcards" in Chapter 1, Getting Started, and "Using Database Sequences," "Using Multiple Sequence Format (MSF) Files", "Using Rich Sequence Format (RSF) Files", and "Using List Files" in Chapter 2, Using Sequence Files and Databases of the User's Guide.

**Batch Queue**

TFastA+ is one of the few programs in the GCG that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using `-batch`. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

**Interrupting a Search: <Ctrl>C**

You can type <Ctrl>C to interrupt a search and see the results from the part of the search that has already been completed. Because the program is multithreaded, the search may not be interrupted immediately, but will continue until one of the threads finishes processing its data and returns for more data.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use `-check` to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a
specified alias. If “Type” is “Boolean”, then the presence of the parameter on the
command line indicates a true condition. A false condition needs to be stated as,
parameter=false. For more information, see "Using Program Parameters" in Chapter 3,
Using Programs in the User's Guide.

TFastA+ does a Pearson and Lipman search for similarity between a protein query
sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide
sequences in all six reading frames before performing the comparison. It is designed to
answer the question, "What implied protein sequences in a nucleotide sequence
database are similar to my protein sequence?"

Minimal Syntax: % tfasta+ [-infile1=]value -Default.

Minimal Parameters (case-insensitive):

(infile1) [Type: List / Default: EMPTY / Aliases: infile in1 in]
Input files specification.

Prompted Parameters (case-insensitive):

-begin
[Type: Integer / Default: '1' / Aliases: beg]
Starting point of the range of interest in the input
sequence.

-end
[Type: Integer / Default: '-1']
End point of the range of interest in the input
sequence. A value of '-1' indicates that the range
extends till the end of input sequence.

-infile2 [Type: List / Default: EMPTY / Aliases: in2 db]
Search set specification.

-outfile [Type: OutFile / Default: 
'<sequence_name>.<program_name>' /
Aliases: out] File to which output is written. A value
of '-' means STDOUT.
Specifying this option also turns on the 'concat'
option. Default value is '-'.

Optional Parameters (case-insensitive):

-check [Type: Boolean / Default: 'false' / Aliases: che help]
Prints out this usage message.

-default [Type: Boolean / Default: 'false' / Aliases: d def]
Specifies that sensible default values be used for all
parameters where possible.

-documentation [Type: Boolean / Default: 'true' / Aliases: doc]
Prints banner at program startup.

-quiet [Type: Boolean / Default: 'false' / Aliases: qui]
Tells application to print only a minimal amount of
information.
-wordsize [Type: Integer / Default: EMPTY / Aliases: wor]
Size of word (k-tuple) used in the hashing step.

-expect [Type: Double / Default: '2.0' / Aliases: exp]
Shows all scores whose E() value is less than the specified value of expect.

-matrix [Type: String / Default: EMPTY / Aliases: mat]
Assigns the scoring matrix for the comparison.

-processors [Type: Integer / Default: '1' / Aliases: proc]
On a multiprocessor computer, this parameter controls the number of threads to use for database search.

-minlength [Type: Integer / Default: EMPTY / Aliases: minl]
The search set is restricted to sequences whose length is more than the value specified by this parameter.

-maxlength [Type: Integer / Default: EMPTY / Aliases: maxl]
The search set is restricted to sequences whose length is less than the value specified by this parameter.

-pamfactor [Type: Boolean / Default: 'DEFAULT_PARAM_VALUE' / Aliases:pam]
This parameter governs whether a scoring matrix should be used for calculating initial diagonal scores, instead of using the identical match scores from the scoring matrix. Default is to use FASTA+ internal behavior, which differs for protein and nucleotide searches.

-gapweight [Type: Integer / Default: EMPTY / Aliases: gap]
This parameter specifies the gap creation penalty that is subtracted from an alignment every time a gap is created.

-lengthweight [Type: Integer / Default: EMPTY / Aliases: len]
This parameter specifies the gap extension penalty that is subtracted from an alignment every time a gap is extended by one residue.

-optall [Type: Boolean / Default: 'DEFAULT_PARAM_VALUE' / Aliases:opt]
With this parameter, the program immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to the value specified by this parameter. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search.

-NOOPTall doesn't compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.

-listsize [Type: Integer / Default: '10' / Aliases: lis]
This parameter controls the number of top scores show. Overrides the -expect parameter.
-alignments [Type: Integer / Default: '20' / Aliases: align ali]
This parameter limits the number of alignments to display in the output file to the 10 best matches in the list. Use -noalign to suppress the sequence alignments in the output file.


-native [Type: Boolean / Default: 'false']
Output native fasta formatted output.

-markx [Type: Integer / Default: EMPTY / Aliases: mark]
This parameter determines the alignment display mode - especially the symbols that identify matches and mismatches. The default value, -MARKx=0 uses a colon to show identities and a period (.) to show conservative replacements.

-MARKx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X.
If -MARKx=2, the residues in the second sequence are shown only if they differ from the first sequence.
-MARKx=3 displays the aligned library sequences without the query sequences; these can be used to build a primitive multiple alignment.
-MARKx=4 provides a graphical display of the boundaries of the alignments.
-MARKx=5 provides a combination of -MARKx=4 and -MARKx=0.
-MARKx=6 provides -MARKx=5 plus HTML formatting.
-MARKx=9 provides percent identity and coordinates with the initial list of high scores as well as the conventional -MARKx=0 alignments.
Use -MARKx=10 to get aligned sequences in the FastA "parsable" output format.

-histogram [Type: Boolean / Default: 'true' / Aliases: his]
Start/Suppress printing the histogram.

-linesize [Type: Integer / Default: EMPTY / Aliases: lin]
This parameter lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-batch [Type: Boolean / Default: 'false']
Allows submitting a job to a batch queue.

-salign [Type: Boolean / Default: 'false' / Aliases: sw]
Does an unlimited Smith-Waterman alignment as the final alignment for the nucleotide searches, instead of 'alignment in a band'.

-dbtopstrand [Type: Boolean / Default: 'false' / Aliases: dbtop]
Translate and search only the top strand of search set sequences.

-dbbottomstrand [Type: Boolean / Default: 'false' / Aliases: dbbot]
Translate and search only the bottom strand of search set sequences.

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -data1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either

1) Have a data file with exactly the same name as the program default scoring matrix in your current working directory; or

2) Have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name Share_Matrix; or

3) Name a file on the command line with an expression like -matrix=mymatrix.cmp. If you do not include a directory specification when you name a file with -matrix, the program searches for the file first in your local directory, then in the directory with the logical name Share_Matrix. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

TFastA+ reads a scoring matrix containing the values for every possible match from your working directory or the public database. The default matrix is blosum50.cmp, which is a BLOSUM50 matrix. You can use the Fetch+ program to obtain a copy of this file if you need to modify it for your own needs.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
-infile1, -infile, -in1, -in

Inputs file specification.

-begin, -beg

Starting point of the range of interest in the input sequence.

-end

End point of the range of interest in the input sequence. A value of '-1' indicates that the range extends till the end of input sequence.

-infile2, -in2, -db

Search set specification.

-outfile, -out

File to which output is written. A value of '-' means STDOUT. Specifying this option also turns on the 'concat' option. Default value is '-'

-wordsze=2, -wor

Sets the size of the word (k-tuple) to use for the hashing step.

-matrix=mymatrix.cmp, -matr

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with -matrix, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

-check, -che, -help

Prints out this usage message.

-default, -default

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.
-quiet, -qui

This parameter is not supported.

-alignment, -align -ali

This parameter limits the number of alignments to display in the output file to the 10 best matches in the list. Use -noalign to suppress the sequence alignments in the output file.

-histogram, -his

Start/suppress printing the histogram.

-expect=2.0, -exp

Shows all scores whose E() value is less than 2.0. Ignored if -listsize is used.

-processors=2, -proc

Tells the program to use 2 threads for the database search on a multiprocessor computer.

-pamfactor, -pam

This parameter governs whether a scoring matrix should be used for calculating initial diagonal scores, instead of using the identical match scores from the scoring matrix.

Default is to use FASTA+ internal behavior, which differs for protein and nucleotide searches

-minlength=1000, -minl

Restricts the search to search set sequences that are equal to or longer than 1000 residues.

-maxlength=5000, -maxl

Restricts the search to search set sequences that are equal to or shorter than 5000 residues.

-dbtopstrand, -dbtopstrand

Translates and searches only the three forward reading frames.

-dbbottomstrand, -dbbot

Translates and searches only the three reverse complement reading frames.
-gapweight=12, -gap

Specifies the gap creation penalty that is subtracted from the alignment score whenever a gap is created.

-lengthweight=2, -len

Specifies the gap extension penalty that is subtracted from the alignment score for each residue added to an existing gap.

-optall=20, -opt

Immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to 20. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search. -nooptall doesn't compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.

-swalign, -sw

Does an unlimited Smith-Waterman alignment as the final alignment for TFasta+ searches, instead of the "alignment in a band" version of Smith-Waterman. (Note: this can be very slow.)

-listsize=40, -lis

Shows the best 40 scores. Overrides -expect.

-showall, -show

Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-markx, -mark

This parameter determines the alignment display mode - especially the symbols that identify matches and mismatches. The default value, -markx=0 uses a colon to show identities and a period (.) to show conservative replacements.

-markx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X.

If -markx=2, the residues in the second sequence are shown only if they differ from the first sequence.

-markx=3 displays the aligned library sequences without the query sequences; these can be used to build a primitive multiple alignment.
-markx=4 provides a graphical display of the boundaries of the alignments.

-markx=5 provides a combination of -markx=4 and -markx=0.

-markx=6 provides -markx=5 plus HTML formatting.

-markx=9 provides percent identity and coordinates with the initial list of high scores as well as the conventional

-markx=0 alignments.

Use -markx=10 to get aligned sequences in the FastA "parsable" output format.

-native

Output native FastA+ formatted output.

-linesize=60, -lin

Lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-batch, -bat

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

Printed: February 10, 2006 19:41
TFASTX

FUNCTION

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

DESCRIPTION


TFastX can be considered to be an enhanced version of TFastA. TFastA treats each of the six reading frames of a nucleotide sequence as a separate sequence, resulting in three separate alignments for each strand. TFastX, on the other hand, compares the protein query sequence to only one translated protein per strand of the nucleotide sequence, resulting in one alignment per strand. It calculates a similarity score for alignments that takes frameshifts into account, allowing it to "join" short regions separated by frameshifts into a single long alignment. TFastX may alert you to more meaningful hits than TFastA does when the nucleotide sequences contain frameshift errors.

TFastX can also be used in situations where FrameSearch is used. TFastX is faster, but FrameSearch is more sensitive.

In the first step of this search, the comparison can be viewed as a set of dot plots, with the query as the vertical sequence and the group of sequences to which the query is being compared as the different horizontal sequences. This first step finds the registers of comparison (diagonals) having the largest number of short perfect matches (words) for each comparison. In the second step, these "best" regions are rescored using a scoring matrix that allows conservative replacements, ambiguity symbols, and runs of identities shorter than the size of a word. In the third step, the program checks to see if some of these initial highest-scoring diagonals can be joined together. Finally, the search set sequences with the highest scores are aligned to the query sequence for display.

What is a Word?

A word is any short sequence (n-mer or k-tuple) where you have set n to some small integer less than or equal to six. The word GGATGG is one of the 4,096 possible words of length six that can be created from an alphabet consisting of the four letters G, A, T, and C. The word QL is one of the 400 possible words of length two that you can make with the 20 letters of the amino acid alphabet.
EXAMPLE

Here is a session using TFastX to identify sequences in a collection of nucleotide sequences that may contain translated regions similar to a human globin protein:

% tfastx

TFASTX with what query sequence? ggamma.pep

Removing terminal * from query sequence...

Begin (* 1 *) ?
End (* 147 *) ?

Search for query in what sequence(s)? (GenBank:* *) ?

fragments.rsf(*)

What word size (* 2 *) ?

Do not show scores whose E() value exceeds: (* 10.0 *):

What should I call the output file? (ggamma.tfastx *) ?

1 Sequences 400 nt searched Fragments.Rsf(Aa004794)

////////////////////////////////////////////////////////////////////////

CPU time used:
Database scan: 0:00: 0.6
Post-scan processing: 0:00: 1.4
Total CPU time: 0:00: 2.1

Output file: ggamma.tfastx

%

OUTPUT

The output from TFastX is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

Here is some of the output file:

!!SEQUENCE_LIST 1.0

(Peptide) TFASTX of: ggamma.pep from: 1 to: 147 October 14, 1998 13:11

TRANSLATE of: gamma.seq check: 6474 from: 2179 to: 2270
and of: gamma.seq check: 6474 from: 2393 to: 2615
and of: gamma.seq check: 6474 from: 3502 to: 3630
generated symbols 1 to: 148.

Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies, Cell 26; 191-203. . . .
TO: FRAGMENTS.RSF(*)
Sequences: 31 Symbols: 12,095 Word Size: 2

Searching both strands.
Scoring matrix: GenRunData:Blosum50.Cmp
Variable pamfactor used
Gap creation penalty: 15 Gap extension penalty: 2 Frameshift penalty: 20

Histogram Key:
Each histogram symbol represents 1 search set sequences
z-scores computed from opt scores

<table>
<thead>
<tr>
<th>z-score</th>
<th>obs</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>1:*</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>2:*</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>2:*</td>
</tr>
</tbody>
</table>
| 42      | 10  | 3:==*=======
| 44      | 6   | 3:==*====
| 46      | 3   | 3:==*    |
| 48      | 1   | 3:==*    |
| 50      | 1   | 3:==*    |
| 52      | 1   | 2:==*    |
| 54      | 2   | 2:==*    |
| 56      | 2   | 2:==*    |
| 58      | 1   | 1:*     |
| 60      | 0   | 1:*     |
| 62      | 0   | 1:*     |
| 64      | 0   | 1:*     |
| 66      | 1   | 1:*     |
| 68      | 0   | 0:      |
| 70      | 1   | 0:      |
| 72      | 1   | 0:      |
| 74      | 0   | 0:      |
| 76      | 0   | 0:      |
| 78      | 1   | 0:      |
| 80      | 0   | 0:      |
| 82      | 0   | 0:      |
| 84      | 0   | 0:      |
| 86      | 0   | 0:      |
| 88      | 0   | 0:      |
| 90      | 0   | 0:      |
| 92      | 0   | 0:      |
| 94      | 0   | 0:      |
| 96      | 0   | 0:      |
| 98      | 0   | 0:      |
| 100     | 0   | 0:      |
Joining threshold: 36, opt. threshold: 24, opt. width: 16, reg.-scaled

The best scores are:

<table>
<thead>
<tr>
<th>FRAGMENTS.RSF{AA096098}</th>
<th>! SAMPLE of: 17628.seq.F Fetal heart, ... (f)</th>
<th>681</th>
<th>761</th>
<th>763</th>
<th>78.7 0.43</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAGMENTS.RSF{AA038757}</td>
<td>! SAMPLE of: mi94d08.r1 Soares mouse ... (f)</td>
<td>574</td>
<td>596</td>
<td>625</td>
<td>71.8 1</td>
</tr>
<tr>
<td>FRAGMENTS.RSF{AA063750}</td>
<td>! SAMPLE of: mj79g09.r1 Soares mouse ... (f)</td>
<td>337</td>
<td>540</td>
<td>580</td>
<td>69.6 1.4</td>
</tr>
</tbody>
</table>

\\End of List

ggamma.pep
FRAGMENTS.RSF{AA096098}
Description: SAMPLE of: 17628.seq.F Fetal heart, Lambda ZAP Express
Accession/ID:

SCORES Strand: (f) Init1: 681 Initn: 761 Opt: 763 z-score: 78.7 E(): 0.43
Smith-Waterman score: 763; 93.0% identity in 129 aa overlap

<table>
<thead>
<tr>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MGHFTEEDKATITSLWGKVNVEDAGGETLGRLLVVYPWTQRRFDSDLSSASAIMGNPK</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAGMENTS.RS</td>
</tr>
<tr>
<td>MGHFTEEDKATITSLWGKVNVEDAGGETPGRLVVYPWTQRRFDSDLSSASAIMGNPK</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>70</td>
</tr>
</tbody>
</table>

| 119 |
| ggamma.pep VKAHGKvltslgdaihlldlkgtfaqloqteslhcldkhlvdpen- |
| FKLLGNVLVTVLAIHF |
| FRAGMENTS.RS |
| VKAHGKvltslgdaihlldlkgtfaqloqteslhcldkhlvdpen/lkllgnvlvpvlaihf |
The first part of the output file contains a histogram showing the distribution of the z-scores between the query and search set sequences. (See the ALGORITHM topic for an explanation of z-score.) The histogram is composed of bins of size 2 that are labeled according to the higher score for that bin (the leftmost column of the histogram). For example, the bin labeled 24 stores the number of sequence pairs that had scores of 23 or 24.

The next two columns of the histogram list the number of z-scores that fell within each bin. The second column lists the number of z-scores observed in the search and the third column lists the number of z-scores that were expected.

The body of the histogram displays a graphical representation of the score distributions. Equal signs (=) indicate the number of scores of that magnitude that were observed during the search, while asterisks (*) plot the number of scores of that magnitude that were expected.

At the bottom of the histogram is a list of some of the parameters pertaining to the search.

Below the histogram, TFastX displays a listing of the best scores. This listing includes the strand (f or r) of the original nucleotide sequence from which the reported translated sequence is derived.

Following the list of best scores, TFastX displays the alignments of the regions of best overlap between the query and search sequences. In these alignments, stop codons are represented by the letter X.

This program displays only the region of overlap between the two aligned sequences (plus some residues on either side of the region to provide context for the alignment) unless you use −SHOWall. The display of identities and conservative replacements
between the aligned sequences depends on the value of \(-\text{MARK}x\). By default (\(-\text{MARK}x=3\)), the pipe character (|) is used to denote identities and the colon (:) to denote conservative replacements.

INPUT FILES

TFastX accepts a single protein sequence as the query sequence. The search set is either a single nucleic acid sequence or multiple nucleic acid sequences. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.

If TFastX rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like TFastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"
FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

ProfileSearch and MotifSearch use a profile (derived from a set of aligned sequences) instead of a query sequence to search a collection of sequences. FindPatterns uses a pattern described by a regular expression to search a collection of sequences.

StringSearch, LookUp, and Names identify sequences by searching the annotation (non-sequence) portions of sequence files or sequence databases.

FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"
RESTRICTIONS

The query sequence may not be longer than 32,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The word size must be either 1 or 2.

For the estimates of statistical significance to be valid, the search set must contain a large sample of unrelated sequences. The statistical estimates will not be calculated at all if there are fewer than 10 sequences in the search set (20 sequences when both strands are searched).

With \texttt{-NOOPTall}, the estimates of statistical significance will not be accurate.

ALGORITHM

For a description of the algorithm, see the FastA program documentation. TFastX always uses an unrestricted Smith-Waterman algorithm for the final alignment, so this step may take a long time.

CONSIDERATIONS

TFastX translates stop codons in search set sequences to the sequence symbol X.

The E() scores are affected by similarities in sequence composition between the query sequence and the search set sequence. Unrelated sequences may have "significant" scores because of composition bias.

If there is a database entry that overlaps your query in several places, but there are large gaps between the matching regions, only the best overlap appears in the alignment display.

There are two ways to control the size of the list of best scores. By default, scores are listed until a specific E() value is reached. You may set the value in response to the program prompt or by using \texttt{-EXPect}; otherwise the program uses 10.0 for protein searches, 2.0 for nucleic acid searches. (If you are running the program interactively, it will show no more than 40 scores initially, and ask if you want to see more scores if there are any more that are less than the E() value.)

If you use \texttt{-LISTsize}, the E() value is ignored, and the program will list the number of scores you requested.

You can control the number of alignments using \texttt{-NOALIGN} and \texttt{ALIGN}. The program behaves differently depending on whether it is being run noninteractively (in batch or with \texttt{-Default} on the command line) or interactively. In the noninteractive case, the program displays the number of alignments set by \texttt{ALIGN}. (If this is not present, it shows 40 alignments or the number of scores that were listed, whichever is smaller.) If you run the program interactively, it displays the list of best scores, then asks you how many alignments you want to see. (This prompt does not appear if you use \texttt{-NOALIGN} or \texttt{ALIGN}.)
Increasing Sensitivity By Adjusting Word Size

By default, TFastX uses a word size of 2. If it finds few or no matches, especially if your query sequence is short, rerun the search using `-WORDsize=1` to increase the sensitivity. Note that this will dramatically increase the amount of CPU time required to run the program.

Adjusting Gap Creation, Gap Extension, and Frameshift Penalties

Unlike other GCG programs, TFastX does not read the default gap creation, gap extension, and frameshift penalties from the scoring matrix file. It uses default penalties that were empirically determined to be appropriate for the BLOSUM50 scoring matrix. If you select a different scoring matrix with `-MATRIX`, you may need to change the gap penalties. The histogram display gives a qualitative view of the quality of fit between the actual distribution of scores and the expected distribution of scores. This information may indicate whether or not suitable gap creation and extension penalties were used for the search. When the histogram shows poor agreement between the actual distribution and the theoretical distribution, you might consider using `-GAPweight` and/or `-LENweight` to specify higher gap creation and extension penalties, respectively. For example, you might increase the gap creation penalty from 15 to 20 and the gap extension penalty from 2 to 6. You may also need to use `-FRAMEweight` to adjust the frameshift penalty.

Differences in Applying Gap Extension Penalties

There are two different philosophies on how to penalize gaps in an alignment. One way is to penalize a gap by the gap creation penalty plus the extension penalty times the length of the gap (\(\text{gapweight} + (\text{lengthweight} \times \text{gap length})\)). The other way is to use the gap creation penalty plus the extension penalty times the gap length excluding the first residue in the gap (\(\text{gapweight} + (\text{lengthweight} \times (\text{gap length} - 1))\)).

"Native" GCG programs, such as Framesearch and Bestfit, handle gap extension penalties the first way, while the FastA-family programs use the second way. Therefore a value for `-LENweight` that gives good results with one of the FastA-family programs may not give equivalent results with a native GCG program, and vice versa.

Increasing Program Speed Using Multithreading

This program is multithreaded. It has the potential to run faster on a machine equipped with multiple processors because different parts of the analysis can be run in parallel on different processors. By default, the program assumes you have one processor, so the analysis is performed using one thread. You can use `-PROCessors` to increase the number of threads up to the number of physical processors on the computer.

Under ideal conditions, the increase in speed is roughly linear with the number of processors used. But conditions are rarely ideal. If your computer is heavily used, competition for the processors can reduce the program's performance. In such an environment, try to run multithreaded programs during times when the load on the system is light.
As the number of threads increases, the amount of memory required increases substantially. You may need to ask your system administrator to increase the memory quota for your account if you want to use more than two threads.

Never use \texttt{--processors} to set the number of threads higher than the number of physical processors that the machine has -- it does not increase program performance, but instead uses up a lot of memory needlessly and makes it harder for other users on the system to get processor time. Ask your system administrator how many processors your computer has if you aren't sure.

\textbf{SUGGESTIONS}

\textbf{Identifying the Search Set}

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" topic of Chapter 2, Using Sequence Files and Databases of the User's Guide for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. For information about naming groups of sequences for the search set, see the topics "Specifying Files" and "Using Wildcards" in Chapter 1, Getting Started, and "Using Database Sequences," "Using Multiple Sequence Format (MSF) Files", "Using Rich Sequence Format (RSF) Files", and "Using List Files" in Chapter 2, Using Sequence Files and Databases of the User's Guide.

\textbf{Batch Queue}

TFastX is one of the few programs in the GCG that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using \texttt{--batch}. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

\textbf{Interrupting a Search: \textless Ctrl\textgreater C}

You can type \textless Ctrl\textgreater C to interrupt a search and see the results from the part of the search that has already been completed. Because the program is multithreaded, the search may not be interrupted immediately, but will continue until one of the threads finishes processing its data and returns for more data.

\textbf{COMMAND-LINE SUMMARY}

All parameters for this program may be added to the command line. Use \texttt{--check} to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
Minimal Syntax: % tfastx [\-INfile1=]ggamma.pep -Default

Prompted Parameters:

[\-INfile2=}GenBank:* specifies search set
[\-OUTfile=}ggamma.tfastx names the output file
-BEGin=1 -END=148 sets the range of interest
-WORDsize=2 sets the word size
-EXPect=2.0 lists scores until E() value reaches 2.0

Local Data Files:

-MATRix=blosum50.cmp assigns the scoring matrix for proteins

Optional Parameters:

-PROCessors=2 sets the number of threads devoted to the analysis
  on a multiprocessor computer
-MINLength=1000 searches only sequences of 1000 or more residues
-MAXLength=5000 searches only sequences of 5000 or fewer residues
-SINce=6.90 limits search to sequences dated on or after June 1990
-DBTOPstrand translates and searches only the top (forward)
  strand of
  the search set sequences
-DBBOTtomstrand translates and searches only the reverse complement
  strand of the search set sequences
-NOPAMfactor uses a constant factor to calculate initial diagonal
  scores
-GAPweight=15 sets the gap creation penalty
-LENgthweight=2 sets the gap extension penalty
-FRAmeweight=20 sets the frame shift penalty
-OPTall=20 computes opt score when the initn score is 20
  or higher; sorts on opt score
-NOOPTall doesn't compute opt score during search; sorts on
  initn
-LISTsize=40 shows the best 40 scores (overrides EXPect)
-ALIgn=20 shows the best 20 alignments
-NOALIgn suppresses sequence alignments
-SHOWall shows complete sequences in alignment, not just
  overlaps
-MARKx=3 sets the alignment display mode
-NOHIStogram suppresses printing the histogram
-LINEsize=60 sets number of sequence symbols per line of the
  alignment
-NODOCLines suppresses sequence documentation in the alignment
-BATch submits the program to run in the batch queue
-NOMONitor suppresses the screen trace for each search set
  sequence

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program
automatically reads them from a public data directory unless you either 1) have a data
file with exactly the same name in your current working directory; or 2) name a file on
the command line with an expression like -DATa1=myfile.dat. For more information
see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence
characters. The program automatically reads the program's default scoring matrix in a
public data directory unless you either 1) have a data file with exactly the same name as
the program default scoring matrix in your current working directory; or 2) have a data
file with exactly the same name as the program default scoring matrix in the directory
with the logical name MyData; or 3) name a file on the command line with an
expression like -MATRi=x=mymatrix.cmp. If you do not include a directory
specification when you name a file with -MATRi=x, the program searches for the file
first in your local directory, then in the directory with the logical name MyData, then in
the public data directory with the logical name GenMoreData, and finally in the public
data directory with the logical name GenRunData. For more information see "Using a
Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the
User's Guide.

TFastX reads a scoring matrix containing the values for every possible match from your
working directory or the public database. The default matrix is blosum50.cmp, which is
a BLOSUM50 matrix. You can use the Fetch program to obtain a copy of this file if you
need to modify it for your own needs.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information,
see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-WORdsize=2

Sets the size of the word (k-tuple) to use for the hashing step.

-MATRi=x=mymatrix.cmp

Allows you to specify a scoring matrix file name other than the program default.
If you do not include a directory specification when you name a file with -MATRi=x, the program searches for the file first in your local directory, then in the
directory with the logical name MyData, then in the public data directory with the
logical name GenMoreData, and finally in the public data directory with the
logical name GenRunData.

For more information see the Local Scoring Matrices section.

-EXpec=2.0

Shows all scores whose E() value is less than 2.0. Ignored if -LISTsize is used.
-PROCessors=2

Tells the program to use 2 threads for the database search on a multiprocessor computer.

-MINLenght=1000

Restricts the search to search set sequences that are equal to or longer than 1000 residues.

-MAXLenght=5000

Restricts the search to search set sequences that are equal to or shorter than 5000 residues.

-SINce=6.1990

Limits the search to sequences that have been entered into the database or modified since June 1990. As this is being written, only the EMBL, GenBank, and SWISS-PROT databases support this parameter.

-DBTOP strand

Translates and searches only the top strand of search set sequences.

-DBBOTtomstrand

Translates and searches only the reverse complement strand of search set sequences.

-NOPAM factor

Uses a constant factor for the calculation of initial diagonal scores, instead of using the identical match scores from the scoring matrix.

-GAPweight=12

Specifies the gap creation penalty that is subtracted from the alignment score whenever a gap is created.

-LENgthweight=2

Specifies the gap extension penalty that is subtracted from the alignment score for each residue added to an existing gap.

-FRAMEweight=20

Specifies the penalty that is subtracted from the alignment score whenever a frameshift is inserted.
-OPTall=20

Immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to 20. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search. -NOOPTall doesn't compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.

-LISTsize=40

Shows the best 40 scores. Overrides -EXPect.

-ALIGN=10

Limits the number of alignments to display in the output file to the 10 best matches in the list. Use the -NOALIGN to suppress the sequence alignments in the output file.

-SHOWall

Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-MARKx=3

Determines the alignment display mode -- especially the symbols that identify matches and mismatches. The default value, 3, uses a pipe character (|) to show identities and a colon (:) to show conservative replacements. -MARKx=0 uses a colon to show identities and a period (.) to show conservative replacements. -MARKx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X. If -MARKx=2, the residues in the second sequence are shown only if they differ from the first sequence.

Use -MARKx=10 to get aligned sequences in the FastA "parsable" output format. A document describing this format appears after FastA in the Program Manual.

-NOHISTogram

Suppresses printing the histogram.

-LINEsizé=60

Lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.
-NODOCLines

Suppresses the documentation from the search set sequence accompanying the alignment in the output file. Use -DOCLines=5 to copy only five non-blank lines of documentation.

-BATCH

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

-MONitor=500

Monitors this program's progress on your screen. Use this parameter to see this same monitor in the log file for a batch process. If the monitor is slowing down the program because your terminal is connected to a slow modem, suppress it with -NOMONitor.

The monitor is updated every time the program processes 500 sequences or files. You can use a value after the parameter to set this monitoring interval to some other number.
TFastX+

FUNCTION

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like FastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

DESCRIPTION

Advantages of Plus “+” Programs:

✓ Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

✓ Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.


TFastX+ can be considered to be an enhanced version of FastA+. FastA+ treats each of the six reading frames of a nucleotide sequence as a separate sequence, resulting in three separate alignments for each strand. TFastX+, on the other hand, compares the protein query sequence to only one translated protein per strand of the nucleotide sequence, resulting in one alignment per strand. It calculates a similarity score for alignments that takes frameshifts into account, allowing it to "join" short regions separated by frameshifts into a single long alignment. TFastX+ may alert you to more meaningful hits than TFastA+ does when the nucleotide sequences contain frameshift errors.

TFastX+ can also be used in situations where FrameSearch is used. TFastX+ is faster, but FrameSearch is more sensitive.

In the first step of this search, the comparison can be viewed as a set of dot plots, with the query as the vertical sequence and the group of sequences to which the query is being compared as the different horizontal sequences. This first step finds the registers of comparison (diagonals) having the largest number of short perfect matches (words) for each comparison. In the second step, these "best" regions are rescored using a scoring matrix that allows conservative replacements, ambiguity symbols, and runs of identities shorter than the size of a word. In the third step, the program checks to see if some of these initial highest-scoring diagonals can be joined together. Finally, the
search set sequences with the highest scores are aligned to the query sequence for display.

What is a Word?

A word is any short sequence (n-mer or k-tuple) where you have set n to some small integer less than or equal to six. The word GGATGG is one of the 4,096 possible words of length six that can be created from an alphabet consisting of the four letters G, A, T, and C. The word QL is one of the 400 possible words of length two that you can make with the 20 letters of the amino acid alphabet.

EXAMPLE

Here is a session using TFastX+ to identify sequences in a collection of nucleotide sequences from Genbank:humbh* that may contain translated regions similar to a human globin protein:

10:34~215> tfastx+

TFastX+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA+, and like TFastA+, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

tfastx+ with what query sequence(s) ? ggamma.pep
Begin (* 1 *) ?

End (-1 for entire sequence) (* -1 *) ?
Enter value for search set (*Default DB*) ? Genbank:humbh*
What should I call the output file (* <sequence_name>.<program_name> *) ?

# $GCGROOT/bin/tfastx34_native -O /var/tmp/bslskBAA23aqOL.tmp -E 2.0 -b 10 -T 1 /var/tmp/bslskDAA43aqOL.fa "Genbank:humbh* 17"

TFASTX compares a protein to a translated DNA data bank 
version 3.4t21 May 14, 2003

Please cite: 

Query library /var/tmp/bslskDAA43aqOL.fa vs Genbank:humbh* 17 library 

% 

OUTPUT

The output from TFastX+ is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)
Here is some of the output file:

# $GCGROOT/bin/tfastx34_native -O /var/tmp/bslskBAA23aqOL.tmp -E 2.0 -b 10 -T 1 /var/tmp/bslskDAA43aqOL.fa "Genbank:humbh* 17"
TFASTX compares a protein to a translated DNA data bank
version 3.4t21 May 14, 2003
Please cite:

Query library /var/tmp/bslskDAA43aqOL.fa vs Genbank:humbh* library
searching Genbank:humbh* 17 library

1>>>GGAMMA.PEP TRANSLATE of: ggamma.seq check: 7694 from: 1 to: 1700
- 566 aa
vs Genbank:humbh* library

<table>
<thead>
<tr>
<th>opt</th>
<th>E()</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 20</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>2</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>3</td>
</tr>
<tr>
<td>54</td>
<td>1</td>
</tr>
<tr>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>66</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>1</td>
</tr>
<tr>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>74</td>
<td>0</td>
</tr>
<tr>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>82</td>
<td>0</td>
</tr>
<tr>
<td>84</td>
<td>0</td>
</tr>
<tr>
<td>86</td>
<td>0</td>
</tr>
<tr>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>94</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>0</td>
</tr>
</tbody>
</table>

one = represents 1 library sequences
98  0  0:
100 0  0:
102 0  0:
104 0  0:
106 0  0:
108 0  0:
110 0  0:
112 0  0:
114 0  0:
116 0  0:
118 0  0:
>120 0  0:

19900 residues in 21 sequences
Expectation_n fit: rho(ln(x)) = 17.3479 +/- 0.267; mu = -51.4258 +/- 12.631
mean_var=58.6557/-28.422, 0's: 0 Z-trim: 5 B-trim: 0 in 0/5
Lambda= 0.167463
Kolmogorov-Smirnov statistic: 0.3635 (N=12) at 50

TFASTX (3.46 May 2003) function [optimized, BL50 matrix (o=15:-5:-1)]
ktup: 2
join: 37, opt: 33, open/ext: -14/-2 shift: -20, width: 16
Scan time: 0.330
The best scores are:

<table>
<thead>
<tr>
<th>E(21)</th>
<th>opt bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUMBHA04 M16414 Human beta-hexosaminidase alph ( 68) [r]</td>
<td>38  20.0 0.76</td>
</tr>
<tr>
<td>HUMBHA06 M16416 Human beta-hexosaminidase alph ( 123) [r]</td>
<td>44  19.8 1.5</td>
</tr>
<tr>
<td>HUMBHA12 M16422 Human beta-hexosaminidase alph ( 112) [r]</td>
<td>42  19.6 1.6</td>
</tr>
<tr>
<td>HUMBHA14 M16424 Human beta-hexosaminidase alph ( 718) [r]</td>
<td>70  21.2 3.2</td>
</tr>
<tr>
<td>HUMBHA12 M16422 Human beta-hexosaminidase alph ( 112) [f]</td>
<td>36  18.1 4.2</td>
</tr>
<tr>
<td>HUMBHA13 M16423 Human beta-hexosaminidase alph ( 126) [f]</td>
<td>37  18.0 4.9</td>
</tr>
<tr>
<td>HUMBHA05 M16415 Human beta-hexosaminidase alph ( 132) [f]</td>
<td>37  17.9 5.5</td>
</tr>
<tr>
<td>HUMBHA02 M16412 Human beta-hexosaminidase alph ( 114) [r]</td>
<td>33  17.4 6.7</td>
</tr>
<tr>
<td>HUMBHA04 M16414 Human beta-hexosaminidase alph ( 68) [f]</td>
<td>24  16.6 6.7</td>
</tr>
<tr>
<td>HUMBHA07 M16417 Human beta-hexosaminidase alph ( 154) [r]</td>
<td>38  17.7 6.9</td>
</tr>
</tbody>
</table>

566 residues in 1 query sequences
19900 residues in 21 library sequences
Scomplib [34t21]
start: Thu Dec  9 10:35:12 2004 done: Thu Dec  9 10:35:22 2004
Total Scan time: 0.330 Total Display time: 3.820

Function used was TFASTX [version 3.4t21 May 14, 2003]
What is the Output?

The first part of the output file contains a histogram showing the distribution of the z-scores between the query and search set sequences. (See the ALGORITHM topic for an explanation of z-score.) The histogram is composed of bins of size 2 that are labeled according to the higher score for that bin (the leftmost column of the histogram). For example, the bin labeled 24 stores the number of sequence pairs that had scores of 23 or 24.

The next two columns of the histogram list the number of z-scores that fell within each bin. The second column lists the number of z-scores observed in the search and the third column lists the number of z-scores that were expected.

The body of the histogram displays a graphical representation of the score distributions. Equal signs (=) indicate the number of scores of that magnitude that were observed during the search, while asterisks (*) plot the number of scores of that magnitude that were expected.

At the bottom of the histogram is a list of some of the parameters pertaining to the search.

Below the histogram, TFastX+ displays a listing of the best scores. This listing includes the strand (f or r) of the original nucleotide sequence from which the reported translated sequence is derived.

Following the list of best scores, TFastX+ displays the alignments of the regions of best overlap between the query and search sequences. In these alignments, stop codons are represented by the letter X.

This program displays only the region of overlap between the two aligned sequences (plus some residues on either side of the region to provide context for the alignment) unless you use -showall. The display of identities and conservative replacements between the aligned sequences depends on the value of -markx. By default (-markx=3), the pipe character (|) is used to denote identities and the colon (:) to denote conservative replacements.

INPUT FILES

TFastX+ accepts a single protein sequence as the query sequence. The search set is either a single nucleic acid sequence or multiple nucleic acid sequences. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example Genbank:*. If TFastX+ rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is
designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastA+ does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA+ may be more sensitive than BLAST+.

BLAST+ searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST+ can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST+ can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch+ does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST+ and FastA+, it can be very slow.

TFastA+ does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA+ translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX+ does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX+ translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

ProfileSearch and MotifSearch use a profile (derived from a set of aligned sequences) instead of a query sequence to search a collection of sequences. FindPatterns+ uses a pattern described by a regular expression to search a collection of sequences.

StringSearch, LookUp, and Names identify sequences by searching the annotation (non-sequence) portions of sequence files or sequence databases.
FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST+ searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA.

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?"

**RESTRICTIONS**

The query sequence may not be longer than 20,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The word size must be either 1 or 2.

For the estimates of statistical significance to be valid, the search set must contain a large sample of unrelated sequences. The statistical estimates will not be calculated at all if there are fewer than 10 sequences in the search set (20 sequences when both strands are searched).

With `-nooptall`, the estimates of statistical significance will not be accurate.

For Tru64 (OSF), TFastX+ fails with an error message:

"While running the child process: Child was terminated by signal 6 (SIGABRT)"  Error in cleaning up after application: Error reading fast program output: Unable to open tfastx output file: " /tmp/bslskAAAMGXM Cf.tmp" (at /tmp/bslskAAAMGXM Cf tmp:0)."
Workaround

There is an upper limit on the amount of memory that is allocated per process. For tru64 machine the limit for datasize is set to 128M. To increase this limit, execute

```csh
> unlimit datasize
```

```ksh
> ulimit datasize
```

This will increase the limit on the datasize to 1024M. This is the maximum amount of memory that an individual process can take on Tru64 machine. So, default settings for the search set parameter (-infile2) for the fasta suite of programs may cause a crash. Please execute the programs with a smaller subset. The programs have been tested successfully using a search set of 400 thousand sequences.

**ALGORITHM**

For a description of the algorithm, see the FastA+ program documentation. TFastX+ always uses an unrestricted Smith-Waterman algorithm for the final alignment, so this step may take a long time.

**CONSIDERATIONS**

TFastX+ translates stop codons in search set sequences to the sequence symbol X.

The E() scores are affected by similarities in sequence composition between the query sequence and the search set sequence. Unrelated sequences may have "significant" scores because of composition bias.

If there is a database entry that overlaps your query in several places, but there are large gaps between the matching regions, only the best overlap appears in the alignment display.

There are two ways to control the size of the list of best scores. By default, scores are listed until a specific E() value is reached. You may set the value in response to the program prompt or by using `-expect`; otherwise the program uses 10.0 for protein searches, 2.0 for nucleic acid searches. (If you are running the program interactively, it will show no more than 40 scores initially, and ask if you want to see more scores if there are any more that are less than the E() value.)

If you use `-listsize`, the E() value is ignored, and the program will list the number of scores you requested.

You can control the number of alignments using `-noalign` and `-align`. The program behaves differently depending on whether it is being run noninteractively (in batch or with `-default` on the command line) or interactively. In the noninteractive case, the program displays the number of alignments set by `-align`. (If this is not present, it shows 40 alignments or the number of scores that were listed, whichever is smaller.) If you run the program interactively, it displays the list of best scores, and then
asks you how many alignments you want to see. (This prompt does not appear if you use \texttt{-noalign} or \texttt{-align}.)

**Increasing Sensitivity By Adjusting Word Size**

By default, TFastX+ uses a word size of 2. If it finds few or no matches, especially if your query sequence is short, rerun the search using \texttt{-wordsiz}=1 to increase the sensitivity. Note that this will dramatically increase the amount of CPU time required to run the program.

**Adjusting Gap Creation, Gap Extension, and Frameshift Penalties**

Unlike other GCG programs, TFastX+ does not read the default gap creation, gap extension, and frameshift penalties from the scoring matrix file. It uses default penalties that were empirically determined to be appropriate for the BLOSUM50 scoring matrix. If you select a different scoring matrix with \texttt{-matrix}, you may need to change the gap penalties. The histogram display gives a qualitative view of the quality of fit between the actual distribution of scores and the expected distribution of scores. This information may indicate whether or not suitable gap creation and extension penalties were used for the search. When the histogram shows poor agreement between the actual distribution and the theoretical distribution, you might consider using \texttt{-gapweight} and/or \texttt{-lengthweight} to specify higher gap creation and extension penalties, respectively. For example, you might increase the gap creation penalty from 15 to 20 and the gap extension penalty from 2 to 6. You may also need to use \texttt{-frameweight} to adjust the frameshift penalty.

**Differences in Applying Gap Extension Penalties**

There are two different philosophies on how to penalize gaps in an alignment. One way is to penalize a gap by the gap creation penalty plus the extension penalty times the length of the gap \((\text{gapweight} + (\text{lengthweight} \times \text{gap length}))\). The other way is to use the gap creation penalty plus the extension penalty times the gap length excluding the first residue in the gap \((\text{gapweight} + (\text{lengthweight} \times (\text{gap length} - 1)))\).

"Native" GCG programs, such as Framesearch and Bestfit, handle gap extension penalties the first way, while the FastA+-family programs use the second way. Therefore a value for \texttt{-lengthweight} that gives good results with one of the FastA+-family programs may not give equivalent results with a native GCG program, and vice versa.

**Increasing Program Speed Using Multithreading**

This program is multithreaded. It has the potential to run faster on a machine equipped with multiple processors because different parts of the analysis can be run in parallel on different processors. By default, the program assumes you have one processor, so the analysis is performed using one thread. You can use \texttt{-processors} to increase the number of threads up to the number of physical processors on the computer.
Under ideal conditions, the increase in speed is roughly linear with the number of processors used. But conditions are rarely ideal. If your computer is heavily used, competition for the processors can reduce the program's performance. In such an environment, try to run multithreaded programs during times when the load on the system is light.

As the number of threads increases, the amount of memory required increases substantially. You may need to ask your system administrator to increase the memory quota for your account if you want to use more than two threads.

Never use `-processors` to set the number of threads higher than the number of physical processors that the machine has -- it does not increase program performance, but instead uses up a lot of memory needlessly and makes it harder for other users on the system to get processor time. Ask your system administrator how many processors your computer has if you aren't sure.

**SUGGESTIONS**

**Identifying the Search Set**

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" topic of Chapter 2, Using Sequence Files and Databases of the User's Guide for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. For information about naming groups of sequences for the search set, see the topics "Specifying Files" and "Using Wildcards" in Chapter 1, Getting Started, and "Using Database Sequences," "Using Multiple Sequence Format (MSF) Files", "Using Rich Sequence Format (RSF) Files", and "Using List Files" in Chapter 2, Using Sequence Files and Databases of the User's Guide.

**Batch Queue**

TFastX+ is one of the few programs in the GCG that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using `-batch`. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

**Interrupting a Search: <Ctrl>C**

You can type <Ctrl>C to interrupt a search and see the results from the part of the search that has already been completed. Because the program is multithreaded, the search may not be interrupted immediately, but will continue until one of the threads finishes processing its data and returns for more data.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -check to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % tfastx+ [-infile1=]value -Default

Minimal Parameters (case-insensitive):

(infile1       [Type: List / Default: EMPTY / Aliases: infile in1 in]  
  Input file specification.

Prompted Parameters (case-insensitive):

-begin          [Type: Integer / Default: '1' / Aliases: beg]  
  Starting point of the range of interest in the input sequence.

-end            [Type: Integer / Default: '-1']  
  End point of the range of interest in the input sequence. A value of '-1' indicates that the range extends till the end of input sequence.

-infile2       [Type: List / Default: EMPTY / Aliases: in2 db]  
  Search set specification.

-outfile       [Type: OutFile / Default:  
    '<sequence_name>.<program_name>' / Aliases: out] File to which output is written. A value of '-' means STDOUT. Specifying this option also turns on the 'concat' option. Default value is '-'

Optional Parameters (case-insensitive):

-check          [Type: Boolean / Default: 'false' / Aliases: che help]  
  Prints out this usage message.

-default        [Type: Boolean / Default: 'false' / Aliases: d def]  
  Specifies that sensible default values be used for all parameters where possible.

-documentation  [Type: Boolean / Default: 'true' / Aliases: doc]  
  Prints banner at program startup.

-quiet          [Type: Boolean / Default: 'false' / Aliases: qui]  
  Tells application to print only minimal amount of information.
<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Default</th>
<th>Alias(es)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-wordsize</td>
<td>Integer</td>
<td>EMPTY</td>
<td>wor</td>
<td>Size of word (k-tuple) used in the hashing step.</td>
</tr>
<tr>
<td>-expect</td>
<td>Double</td>
<td>'2.0'</td>
<td>exp</td>
<td>Shows all scores whose E() value is less than the specified value of EXPect</td>
</tr>
<tr>
<td>-matrix</td>
<td>String</td>
<td>EMPTY</td>
<td>mat</td>
<td>Assigns the scoring matrix for the comparison.</td>
</tr>
<tr>
<td>-processors</td>
<td>Integer</td>
<td>'1'</td>
<td>proc</td>
<td>On a multiprocessor computer, this parameter controls the number of threads to use for database search.</td>
</tr>
<tr>
<td>-minlength</td>
<td>Integer</td>
<td>EMPTY</td>
<td>minl</td>
<td>The search set is restricted to sequences whose length is more than the value specified by this parameter.</td>
</tr>
<tr>
<td>-maxlength</td>
<td>Integer</td>
<td>EMPTY</td>
<td>maxl</td>
<td>The search set is restricted to sequences whose length is less than the value specified by this parameter.</td>
</tr>
<tr>
<td>-pamfactor</td>
<td>Boolean</td>
<td>'DEFAULT_PARAM_VALUE'</td>
<td>pam</td>
<td>This parameter governs whether a scoring matrix should be used for calculating initial diagonal scores, instead of using the identical match scores from the scoring matrix. Default is to use FASTA+ internal behavior, which differs for protein and nucleotide searches.</td>
</tr>
<tr>
<td>-gapweight</td>
<td>Integer</td>
<td>EMPTY</td>
<td>gap</td>
<td>This parameter specifies the gap creation penalty that is subtracted from an alignment every time a gap is created.</td>
</tr>
<tr>
<td>-lengthweight</td>
<td>Integer</td>
<td>EMPTY</td>
<td>len</td>
<td>This parameter specifies the gap extension penalty that is subtracted from an alignment every time a gap is extended by one residue.</td>
</tr>
<tr>
<td>-optall</td>
<td>Boolean</td>
<td>'DEFAULT_PARAM_VALUE'</td>
<td>opt</td>
<td>With this parameter, the program immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to the value specified by this parameter. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search. -nooptall doesn't compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.</td>
</tr>
<tr>
<td>-listszie</td>
<td>Integer</td>
<td>'10'</td>
<td>lis</td>
<td>This parameter controls the number of top scores show. Overrides the -expect parameter.</td>
</tr>
</tbody>
</table>
-alignments
[Type: Integer / Default: '20' / Aliases: align ali]
This parameter limits the number of alignments to display in the output file to the 10 best matches in the list. Use -noalign to suppress the sequence alignments in the output file.

-showall
[Type: Boolean / Default: 'DEFAULT_PARAM_VALUE' / Aliases: show]
Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-native
[Type: Boolean / Default: 'false']
Output native FastA+ formatted output.

-markx
[Type: Integer / Default: EMPTY / Aliases: mark]
This parameter determines the alignment display mode – especially the symbols that identify matches and mismatches. The default value, -MARKx=0 uses a colon to show identities and a period (.) to show conservative replacements.

-MARKx=1
will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X.
If -MARKx=2, the residues in the second sequence are shown only if they differ from the first sequence.

-MARKx=3
displays the aligned library sequences without the query sequences; these can be used to build a primitive multiple alignment.

-MARKx=4
provides a graphical display of the boundaries of the alignments.

-MARKx=5
provides a combination of -MARKx=4 and -MARKx=0.

-MARKx=6
provides -MARKx=5 plus HTML formatting.

-MARKx=9
provides percent identity and coordinates with the initial list of high scores as well as the conventional alignments.

-MARKx=0
Use -MARKx=10 to get aligned sequences in the FastA "parsable" output format.

-histogram
[Type: Boolean / Default: 'true' / Aliases: his]
Start/Suppress printing the histogram.

-linesize
[Type: Integer / Default: EMPTY / Aliases: lin]
This parameter lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-batch
[Type: Boolean / Default: 'false']
Allows submitting a job to a batch queue.

-frameweight
[Type: Integer / Default: EMPTY / Aliases: frame fra]

-dbtopstrand
[Type: Boolean / Default: 'false' / Aliases: dbtop]
Translate and search only the top strand of search set sequences.

-dbbottomstrand [Type: Boolean / Default: 'false' / Aliases: dbbot] Translate and search only the bottom strand of search set sequences.

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATal=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either

1) Have a data file with exactly the same name as the program default scoring matrix in your current working directory; or

2) Have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name Share_Matrix; or

3) Name a file on the command line with an expression like -matrix=mymatrix.cmp. If you do not include a directory specification when you name a file with -matrix, the program searches for the file first in your local directory, then in the directory with the logical name Share_Matrix. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

TFastX+ reads a scoring matrix containing the values for every possible match from your working directory or the public database. The default matrix is blosum50.cmp, which is a BLOSUM50 matrix. You can use the Fetch+ program to obtain a copy of this file if you need to modify it for your own needs.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-infile1, -infile, -in1, -in

Inputs file specification.
-begin, -beg

Starting point of the range of interest in the input sequence.

-end

End point of the range of interest in the input sequence. A value of '-1' indicates that the range extends till the end of input sequence.

-infile2, -in2, -db

Search set specification.

-outfile, -out

File to which output is written. A value of '-' means STDOUT. Specifying this option also turns on the 'concat' option. Default value is '-'

-wordsize=2, -wor

Sets the size of the word (k-tuple) to use for the hashing step.

-matrix=mymatrix.cmp, -matr

Allows you to specify a scoring matrix file name other than the program default. If you do not include a directory specification when you name a file with -matrix, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

-check, -che, -help

Prints out this usage message.

-default, -def

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.

-quiet, -qui

This parameter is not supported.
-alignments, -align, -ali

This parameter limits the number of alignments to display in the output file to the 10 best matches in the list. Use -noalign to suppress the sequence alignments in the output file.

-pamfactor, -pam

This parameter governs whether a scoring matrix should be used for calculating initial diagonal scores, instead of using the identical match scores from the scoring matrix.

Default is to use FASTA+ internal behavior, which differs for protein and nucleotide searches

-expect=2.0, -exp

Shows all scores whose E() value is less than 2.0. Ignored if -listsize is used.

-processors=2, -proc

Tells the program to use 2 threads for the database search on a multiprocessor computer.

-minlength=1000, -minl

Restricts the search to search set sequences that are equal to or longer than 1000 residues.

-maxlength=5000, -maxl

Restricts the search to search set sequences that are equal to or shorter than 5000 residues.

-dbtopstrand, -dbtop

Translates and searches only the top strand of search set sequences.

-dbbottomstrand, -dbbot

Translates and searches only the reverse complement strand of search set sequences.

-gapweight=12, -gap

Specifies the gap creation penalty that is subtracted from the alignment score whenever a gap is created.
-lengthweight=2, -len

   Specifies the gap extension penalty that is subtracted from the alignment score for each residue added to an existing gap.

-frameweight=20, -frame

   Specifies the penalty that is subtracted from the alignment score whenever a frameshift is inserted.

-optall=20, -opt

   Immediately performs an alignment and calculates the opt score when the initn score is greater than or equal to 20. This parameter allows you to override the default threshold calculated by the program. Scores are sorted and saved by opt score during the search. -nooptall doesn’t compute the opt score until the search is complete. In this case scores are sorted and saved by initn score instead of by opt score.

-listsize=40, -lis

   Shows the best 40 scores. Overrides -expect.

-showall, -show

   Shows entire sequences in the alignment display, instead of just the best region of overlap and its surroundings.

-markx, -mark

   This parameter determines the alignment display mode - especially the symbols that identify matches and mismatches. The default value, -markx=0 uses a colon to show identities and a period (.) to show conservative replacements.

   -markx=1 will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X.

   If -markx=2, the residues in the second sequence are shown only if they differ from the first sequence.

   -markx=3 displays the aligned library sequences without the query sequences; these can be used to build a primitive multiple alignment.

   -markx=4 provides a graphical display of the boundaries of the alignments.

   -markx=5 provides a combination of -markx=4 and -markx=0.

   -markx=6 provides -markx=5 plus HTML formatting.
-markx=9 provides percent identity and coordinates with the initial list of high scores as well as the conventional

-markx=0 alignments.

Use -markx=10 to get aligned sequences in the FastA "parsable" output format.

-native

Output native FastA+ formatted output.

-linesize=60, -lin

Lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

-batch, -bat

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

Printed: February 10, 2006 19:41
TOIG

FUNCTION

ToIG converts GCG sequence file(s) into a single file in IntelliGenetics format.

DESCRIPTION

Any sequence file in GCG format can be converted to a Intelligenetics format with ToIG. ToIG accepts one or more GCG single sequence files as input and creates one output file, containing all the sequences converted to IG format.

EXAMPLE

In this session with ToIG, all of the individual files generated in the sample run of FromIG are put back into IG format.

```
% toig

TOIG of what GCG sequence(s) ? sur*
surphist1 788 bp
surphist2 188 bp

surshist2 682 bp

What should I call the output file (* surphist1.ig *) ? test.ig

%

OUTPUT

Here is part of the output file:

; TOIG of: surphist1 check: 6642 from: 1 to: 788
;
;
; FROMIG of: urchin.nih
;
; definition sea urchin(p.mil.) histone genes; h4 gene. 788bp
; locus surphist1  788 bp updated 11/01/82

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

; (circular sequence)
; surphist1 Length: 788 October 5, 1998 13:04 Type: N Check: 6642

CAACATATTAGAGGAAGGGAGAGAGAGAGAGAGAGAGAGAGAGAGAGAGAGGGGGGGGGGGAGGGAGAAT
INPUT FILES

ToIG accepts multiple (one or more) nucleotide or protein single sequence files in GCG format as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf*; or by using a sequence specification with an asterisk (*) wildcard, for example GenEMBL:. The input files in this example are the output files from the example for FromIG. To create the input files for this example, fetch the file urchin.nih and run FromIG.

RELATED PROGRAMS

The following programs convert sequences between other formats and GCG format: FromEMBL, FromGenBank, FromIG, FromPIR, FromStaden, FromFastA, ToIG, ToPIR, ToStaden and ToFastA.

DataSet creates a GCG data library from any set of sequences in GCG format. GCGToBLAST creates a database that can be searched by the BLAST program from any set of sequences in GCG format.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use \texttt{-CHECK} to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you \textit{must} type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: \texttt{% toig [-INfile=]sur* -Default}

Prompted Parameters: (for single sequences)

-\texttt{-BEGIN=1 -END=444} sets the range of sequence to convert
-\texttt{-REVerse} uses the reverse strand (nucleic acid sequences)
-\texttt{[-OUTfile=}seqname.ig names the output file

Local Data Files: None

Optional Parameters:

-\texttt{-STAden} converts Staden format file(s) into IG format

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-\texttt{-STAden}

Allows input files to be in Staden format instead of GCG format.

Printed: January 9, 2002 13:45 (1162)
TOPIR

FUNCTION

ToPIR writes GCG sequence(s) into a single file in PIR format.

DESCRIPTION

Any sequence file in GCG format can be converted with ToPIR into a format suitable for use with PIR programs. ToPIR accepts one or more GCG sequence files as input and creates one output file, containing all the sequences converted to PIR format.

EXAMPLE

In this session with ToPIR, all of the individual files generated in the sample run of FromPIR are put back into PIR format.

% topir

TOPIR of what GCG sequence(s) ? *.gcg

What should I call the output file (* fwsyg3.pir *) ? sample.pir

FWSYG3 516 characters.
JDVLS 881 characters.
KGRT 179 characters.

% 

OUTPUT

Here is some of the output file:

>P1;FWSYG3
fwsyg3.gcg;2 => FWSYG3
MGKPPFTTLSL SSSLCLLLLSS ACFAITSSKF NECQLNNLNA LEPDHRVESE
GGLIETWNSQ HPHELQACGT VSKRTLNRNG SLHPSYLPPP QMIIVVQQKG
AIGFAPFGCP ETFEKPOQQS SRRGSRSQQQ LDQSHQKIRH FNEGDLVLP
LGVPYWTYNT GDEPVVAISP LDTSNFNQL DNQPRVFYLA GNPIDHEPET
MQQQQQQKSH GGRKQQQRHQ QEEEQGQVLS GFSKHLAQS FNTNEDTAEK
LRSPPDERKQ IVTVEGGQLSV ISPKWQEQED EDEDEDEEEG RTPSYPPRPP
SHGKHDEDED EDEEEDQFPR DHPPQPRPSP EQEPEPRGRC QTRNGVEENI
CTMKLHENA RPSRADYFNP KAGRSTLNS LTLPALRQFG LSAQYVYLYR
NGIYSPDWNL NANSVTMTRG KGRVRVVNCQ GNAVFDGELR RGQLLVVPQN
PAVAEQGGEQ GLEYVPFTH HNAVSYIKD VFRVIPSEVL SNSYNLGQEQ
VRQLKYQGNS GLPN* 
C;P1;FWSYG3 - Glycinin, A3B4 chain precursor - Soybean
C;N;Alternate names: 11S globulin
C;Species: Glycine max (soybean)
C;Accession: A22615
C;R;Fukazawa, C., Momma, T., Hirano, H., Harada, K., and Udaka, K.
INPUT FILES

ToPIR accepts multiple (one or more) nucleotide or protein sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example GenEMBL:*.

The input files in this example are the output files from the example for FromPIR. To create the input files for this example, run FromPIR on qqqss.psq.

RELATED PROGRAMS

The following programs convert sequences between other formats and GCG format: FromEMBL, FromGenBank, FromIG, FromPIR, FromStaden, FromFastA, ToIG, ToPIR, ToStaden and ToFastA.

DataSet creates a GCG data library from any set of sequences in GCG format. GCGToBLAST creates a database that can be searched by the BLAST program from any set of sequences in GCG format.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % topir [-INfile=]seqname(s) -Default

Prompted Parameters:

-BEGIN=1 -END=444 sets the range of sequence to convert
-REVerse uses the reverse strand (nucleic acid sequences)
[-OUTfile=}seqname.pir names the output file

Local Data Files: None

Optional Parameters:

-CIRCular indicates that the input sequence(s) are circular
-NUmbering numbers the output sequence(s)
-NODOCumentation suppresses documentation accompanying each sequence
LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-CIRCular

Treats the sequence(s) as circular. The header line for each circular sequence in the output file begins with >DC;.

-NUMbering

Adds numbering next to each sequence line.

-NODOCumentation

Suppresses writing the sequence documentation in the output file.
**TOSTADEN**

**FUNCTION**

ToStaden writes a GCG sequence into a file in Staden format. If the file contains a nucleotide sequence, the ambiguity codes are converted as shown in Appendix III of the Program Manual.

**DESCRIPTION**

Any sequence file in GCG format can be converted with ToStaden into a format suitable for use in the Staden programs. GCG sequence symbols that aren't recognized by the Staden programs are converted to hyphens (-).

**EXAMPLE**

Here is a session using ToStaden to convert the sequence file test.seq into a Staden-format file:

```plaintext
% tostaden
TOSTADEN of what GCG sequence ? test.seq
    Begin (*) 1 *) ?
    End (*) 389 *) ?
What should I call the output file (* test.sdn *) ?
```

**OUTPUT**

Here is the output file test.sdn:

```
GCTGCCGCAGCGGCNGATGACAATAACRAYTGTTGCTGYGATGACGAYGA
AGAGGARTTTTTCTTYGGTTTGCAGGAGGNNCATCAACCAYATTATCATAA
THAAAGAAARTTGTACTTCTCTACTGTTRCTNYTAYTYRTYRTNATG
ATATAACAYCCTCCCCCACCGCNCAACAGCARCGTCCGCCGACGGGGGAG
AAGGGNGACGACMGGMGGMGMGCTTCTCTCATCAGTAGCTCNAGYWSNA
CTACCACAACGACNGTGTGCAGTGGTGGTNNNTATTACTAYGAAGAG
CAACAGSARTAAATAGIGATATRAATRRAABCD--GH--K-MN---RST-VW
NY------abcd--gh--k-mn---rst-vwny------
```

**INPUT FILES**

ToStaden accepts a single nucleotide or protein sequence as input. Here is the input file for the example above:

```plaintext
!!NA_SEQUENCE 1.0
This sequence contains every symbol in the alphabet of legitimate GCG sequence characters (Appendix III).
```
The function of ToStaden depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

The following programs convert sequences between other formats and GCG format: FromEMBL, FromGenBank, FromIG, FromPIR, FromStaden, FromFastA, ToIG, ToPIR, ToStaden and ToFastA.

DataSet creates a GCG data library from any set of sequences in GCG format. GCGToBLAST creates a database that can be searched by the BLAST program from any set of sequences in GCG format.

CONSIDERATIONS

All documentation and numbering is lost in the Staden-format output file. You should be sure that the Staden program you intend to use is compatible with any ambiguity codes used in your sequence.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `--CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% tostaden [-INfile=]test.seq -Default

Prompted Parameters:

- `BEGin=1 -END=389` sets the range of interest
- `[-OUTfile=]test.sdn` names the output file

Local Data Files: None

Optional Parameters: None

LOCAL DATA FILES

None.

PARAMETER REFERENCE

None.

Printed: February 10, 2006 19:41
TRANSLATE

FUNCTION

Translate translates nucleotide sequences into peptide sequences.

DESCRIPTION

Translate creates a protein sequence by translating nucleic acid sequences that you specify. In addition to translating a single range of a given nucleotide sequence, it will concatenate ranges into a single assembly for translation or translate each range before assembling them. The ranges can be of any length, come from either strand of a sequence, or even from more than one sequence file. Unlike most GCG programs, Translate lets you specify ranges as if the sequence was circular (extending past the end of the sequence and continuing at the beginning).

Translate can be run either interactively or noninteractively. When you specify a single sequence to translate and -Default is not on the command line, it works interactively, prompting you for each segment to translate. To run avoid being prompted, either use -Default on the command line or supply a multiple file specification as input with either a wild card or a list file specification. (See the INPUT FILES topic below for more detailed information.)

Translate supports the IUB-IUPAC character set for the representation of nucleotide ambiguities. See Appendix III for a list of these characters.

EXAMPLE

Here is a session using Translate to translate the G-gamma gene in gamma.seq into the protein sequence for the human fetal beta globin G gamma:

% translate

TRANSLATE from what sequence ? gamma.seq

\* Begin (\* 1 *) \* ? \* 2179
\* End (\* 11375 *) \* ? \* 2270
\* Reverse (\* No *) ?

Range begins ATGGG and ends GGAAG. Is this correct (\* Yes *) ?

That is done, now would you like to:

A) Add another exon from this sequence
B) Add another exon from a new sequence
C) Translate and then add more genes from this sequence
D) Translate and then add more genes from a new sequence
W) Translate assembly and write everything into a file
Please choose one (* W *): a

Begin (* 1 *) ? 2393
End (* 11375 *) ? 2615
Reverse (* No *) ?

Range begins GCTCC and ends TCAAG. Is this correct (* Yes *) ?

That is done, now would you like to:

A) Add another exon from this sequence
B) Add another exon from a new sequence
C) Translate and then add more genes from this sequence
D) Translate and then add more genes from a new sequence
W) Translate assembly and write everything into a file

Please choose one (* W *): a

Begin (* 1 *) ? 3502
End (* 11375 *) ? 3630
Reverse (* No *) ?

Range begins CTCCT and ends ACTGA. Is this correct (* Yes *) ?

That is done, now would you like to:

A) Add another exon from this sequence
B) Add another exon from a new sequence
C) Translate and then add more genes from this sequence
D) Translate and then add more genes from a new sequence
W) Translate assembly and write everything into a file

Please choose one (* W *):

What should I call the output file (* gamma pep *) ? ggamma.pep

% OUTPUT

Here is the output file ggamma.pep:

!!AA_SEQUENCE 1.0
TRANSLATE of: gamma.seq check: 6474 from: 2179 to: 2270
and of: gamma.seq check: 6474 from: 2393 to: 2615
and of: gamma.seq check: 6474 from: 3502 to: 3630
generated symbols 1 to: 148.

Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies, Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.
INPUT FILES

Translate accepts multiple (one or more) nucleotide sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*. If Translate rejects your nucleotide sequence, turn to Appendix VI to see how to change or set the type of a sequence.

Single Sequence Input

If you specify a single sequence on the command line or in response to the first program prompt, and -Default is not on the command line, Translate prompts you for the sequence range and strand. After reading that range, the program prompts you for other ranges in the same or a different sequence or to translate the ranges either before or after assembling them.

Use -Default to translate sequences without prompting you, in accordance with any command-line parameters that are present.

Multiple Sequence Input

When you specify multiple sequences, Translate runs noninteractively. By default, Translate will translate each sequence separately and write out each translation to a separate sequence file without prompting you for the range and strand of each sequence. Use -ONEPPEnde to assemble all of the sequences first and then translate them into a single protein sequence.

If you use a list file to specify multiple sequences as input, you can add begin, end, and strand attributes for each sequence. You can use the join sequence attribute to selectively assemble some of the sequence entries in a list file together before translation. All sequences listed contiguously in the list file that share the same join attribute (i.e. share the same sequence name following the join token) are assembled together before translation and the translated sequence is given the name of the join attribute. All other sequences in the list file are translated separately. Here is an example of an input list file, hsp70dna.list, for Translate.

!!SEQUENCE_LIST 1.0
Example list file of 70kD heat shock coding sequences used as input
for TRANSLATE
..
Using this file as input, Translate writes three output files. The first output file contains a translation of the first sequence entry in this list file. The second output file, hsp70_petunia.pep, is a translation of an assembly of the next two sequence entries. The last output file contains a translation of the last sequence entry in this list file. For more information about list files, see "Using List Files" in Chapter 2, Using Sequence Files and Databases in the User's Guide.

RELATED PROGRAMS

ExtractPeptide can write one or more of the translation frames from the Map program output into protein sequence files. The PepData program translates sequences in all six frames.

RESTRICTIONS

Unknown.

CONSIDERATIONS

Translate allows you to translate sequences where the reading frame is interrupted. This frame-interruption commonly occurs in eukaryotic sequences containing introns. In the example above, a single codon is divided by the first intron. To accommodate frame interruption, Translate allows you to specify ranges to translate (exons) that are not an even multiple of three in length. Translate concatenates the nucleotide ranges that you define and translates them only at the moment you choose a menu item that starts with the word Translate.

If you continue after translating an assembly, you are in effect building a new assembly (gene) and concatenating the protein sequence from the new gene onto the protein sequence you have already created.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % translate [-INfile=]@hsp70dna.list -Default

Prompted Parameters:

[-OUTfile=}hsp70.pep names the output file (single sequences only)
Local Data Files:

-TRANSLate=translate.txt contains the genetic code

Optional Parameters:

-BEGIn=1 -END=100 sets the range of interest
-REVerse specifies the strand for each sequence
-ONEPEPtide translates all concatenated DNA fragments into a single peptide
-NOJOIN ignores all "join" sequence attributes specified in a list file
-LISTfile[=translate.list] writes a list file of output sequence names
-RSF specifies RSF format for output file
-OPEN[=20] only translates open reading frames [minimum peptide length]
-EXTension=.pep sets the file name extension for output sequence files
-NOMONitor suppresses the screen monitor
-NOSUMmary suppresses the summary at the end of the program

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

The translation of codons to amino acids, the identification of potential start codons and stop codons, and the mappings of one-letter to three-letter amino acid codes are all defined in a translation table in the file translate.txt. If the standard genetic code does not apply to your sequence, you can provide a modified version of this file in your working directory or name an alternative file on the command line with an expression like -TRANSLate=mycode.txt. Translation tables are discussed in more detail in Appendix VII.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-TRANSLate=filename.txt

Usually, translation is based on the translation table in a default or local data file called translate.txt. This parameter allows you to use a translation table in a different file. (See Appendix VII for information about translation tables.)
-BEGIN=1

Sets the beginning position for all input sequences. When the beginning position is set from the command line, Translate ignores beginning positions specified for individual sequences in a list file.

-END=100

Sets the ending position for all input sequences. When the ending position is set from the command line, Translate ignores ending positions specified for sequences in a list file.

-REVERSE

Sets the program to use the reverse strand for each input sequence. When -REVERSE or -NOREVERSE is on the command line, Translate ignores any strand designation for individual sequences in a list file.

-ONEPEPTIDE

Concatenates all input sequences together and then translates them all into a single protein sequence.

-NOJOIN

Sets Translate to ignore all join sequence attributes specified in the input list file. All nucleotide sequences specified in the list file are translated into separate output sequence files.

-LISTFILE=translate.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then Translate makes one up using translate for the file name and .list for the file name extension.

-EXTENSION=.pep

This program normally creates output file names by using the original input file name for the base name and the program name for the name extension. Use this parameter to specify some other file name extension.

-MONITOR

This program normally monitors its progress on your screen. However, when you use -DEFAULT to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.
TransMem

FUNCTION

TransMem scans for likely transmembrane helices in one or more input protein sequences.

DESCRIPTION

TransMem builds on the method of Sonnhammer et al. (Proc. of Sixth Int. Conf. on Intelligent Systems for Molecular Biology, 175-182 (1998)) to predict likely transmembrane helices in one or more input proteins. The method is based upon a Hidden Markov Model (HMM) that has been trained on a set of membrane proteins with helical membrane spanning regions.

EXAMPLE

Here is a session using TransMem to generate two predictions for the delta subunit of the mouse GABA(A) receptor.

% transmem sw:gad_mouse

Number of different annotations of each sequence (* 1 *) ? 2

Proximity of feature boundaries to consider annotations equivalent(* 0 *) ?
What should I call the output file (* gad_mouse.transmem *) ?

<table>
<thead>
<tr>
<th>Relative Score</th>
<th>Helix</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAD_MOUSE0</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0.0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GAD_MOUSE1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0.4922</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CPU time: 0.560000
Sequences examined: 1
Sequences written: 2
Results written to "gad_mouse.transmem"

OUTPUT

The output from TransMem is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

!!SEQUENCE_LIST 1.0

TransMem of sw:gad_mouse
-MINHelix = 1
-MMethod=Nbest
-NBest = 2
-PROXimity = 0

August 13, 2001 16:36

Relative Score
.. sw:gad_mouse    !     4     2     3  0.0000
sw:gad_mouse    !     4     2     3  0.4922

\End of List

>>SW:GAD_MOUSE
P22933 mus musculus (mouse). gamma-aminobutyric-acid receptor
delta subunit prec

<table>
<thead>
<tr>
<th></th>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>1</td>
<td>249</td>
</tr>
<tr>
<td>Helix</td>
<td>250</td>
<td>272</td>
</tr>
<tr>
<td>Inside</td>
<td>273</td>
<td>278</td>
</tr>
<tr>
<td>Helix</td>
<td>279</td>
<td>297</td>
</tr>
<tr>
<td>Outside</td>
<td>298</td>
<td>311</td>
</tr>
<tr>
<td>Helix</td>
<td>312</td>
<td>334</td>
</tr>
<tr>
<td>Inside</td>
<td>335</td>
<td>425</td>
</tr>
<tr>
<td>Helix</td>
<td>426</td>
<td>448</td>
</tr>
<tr>
<td>Outside</td>
<td>449</td>
<td>449</td>
</tr>
<tr>
<td>Outside</td>
<td>1</td>
<td>248</td>
</tr>
<tr>
<td>Helix</td>
<td>249</td>
<td>271</td>
</tr>
<tr>
<td>Inside</td>
<td>272</td>
<td>277</td>
</tr>
<tr>
<td>Helix</td>
<td>278</td>
<td>296</td>
</tr>
<tr>
<td>Outside</td>
<td>297</td>
<td>310</td>
</tr>
<tr>
<td>Helix</td>
<td>311</td>
<td>333</td>
</tr>
<tr>
<td>Inside</td>
<td>334</td>
<td>425</td>
</tr>
<tr>
<td>Helix</td>
<td>426</td>
<td>448</td>
</tr>
<tr>
<td>Outside</td>
<td>449</td>
<td>449</td>
</tr>
</tbody>
</table>

1       MDVLGWLLLP  LLLLCTQPHH  GARAMNDIGD  YVGSNLEISW  LPNLDGLMEG
51      YARNFRPGIG  GAPVNVQALI  EVASIDHISE  ANMEYTMVFC  LHQSWRDSDL
101     SYNHTNETLG  LDSRFVDKLA  LPDFIVKNAK  SAWFHDVTVE  NKLIRLQPDG
151     VILYSIRITS  TVACDMDLAK  YPLDEQECML  DLESYGSSE  DIVYVWSENQ
201     EQIHLGDRLQ  LAQFTITSYR  FTTELMNFKS  AGQFPRLSLH  FQILRKNRGVY
251     I1QSYPMSVL  LVAMSVWSEW  ISQAVPARV  SLGITVMIVM  TTTLMSVARSS
301     LPRASAIKAL  DVFYWICYVF  VFAALVEYAF  AHPNADYRKK  RKAUKVTKP
CPU time: 0.560000
Sequences examined: 1
Sequences written: 2

INTERPRETING OUTPUT

The first part of the output file contains a list of all the sequences searched and the predictions generated for a given sequence. When multiple predictions are generated for each sequence, the predictions are listed in order of prediction quality, with the best prediction on top and the sub-optimal predictions below.

Next to each sequence, the file contains the raw counts of how many transmembrane helices, inner loops, and outer loops were found. If you have generated more than one prediction per sequence, there is also a score reported for comparing the quality of the prediction with the best prediction for each sequence. This is a relative measure only and should not be used to compare the quality of predictions between different sequences. In general, a score of 10 or more indicates that the prediction is significantly different from the best prediction.

Following this list of sequences, TransMem displays a table listing the specific boundaries of each feature predicted, followed by the sequence aligned with the predicted labels.

INPUT FILES

TransMem takes any valid GCG specification for one or more protein sequences.

RELATED PROGRAMS

TransMem+ scans for likely transmembrane helices in one or more input protein sequences.

SPScan scans protein sequences for the presence of secretory signal peptides (SPs).

HTHScan scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

HelicalWheel plots a peptide sequence as a helical wheel to help you recognize amphiphilic regions.

PeptideStructure makes secondary structure predictions for a peptide sequence. The predictions include (in addition to alpha, beta, coil, and turn) measures for antigenicity, flexibility, hydrophobicity, and surface probability. PlotStructure displays the predictions graphically.
PepPlot plots measures of protein secondary structure and hydrophobicity in parallel panels of the same plot.

CoilScan locates coiled-coil segments in protein sequences.

SPScan+ scans protein sequences for the presence of secretory signal peptides (SPs).

HTHScan+ scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

CoilScan+ locates coiled-coil segments in protein sequences.

RESTRICTIONS

TransMem only works on protein sequences.

ALGORITHM

TransMem is based upon a Hidden Markov Model (HMM) architecture. The architecture is made up of 7 types of states corresponding to the core of the transmembrane helix, helix caps, cytoplasmic loops, short and long cytoplasmic loop states, and globular domains that are part of each loop.

The states have a close relationship with the biology of membrane proteins; loop states connection to other loops through a helix cap, helix core, and another helix cap. These states correspond to one of three different labels, Inside (cytoplasmic), Helix (membrane spanning helix), and Outside (non-cytoplasmic).

The prediction of transmembrane helices is done by finding an optimal alignment of the sequence with the model using the N-Best algorithm. In the N-Best algorithm, the algorithm uses the model architecture to find the best labeling of the sequence, given the model.

Alternatively, you can run TransMem using the Viterbi algorithm, which finds the optimal alignment of the sequence with the model, then uses that alignment to read the labels. In general, the Viterbi algorithm will give the same results as the N-Best, but in some cases the predictions will differ.

The output of the raw probabilities is based upon the forward-backward algorithm, in which TransMem finds the probability of each labeling (Inside, Outside, or Helix) summed over all the possible alignments of the sequence to the model. Because these values are based upon all possible alignments of the model instead of a single optimal alignment, occasionally the raw probabilities will contradict the final labeling.

CONSIDERATIONS

When no transmembrane helices are predicted, it is not a good idea to treat the Inside/Outside prediction as an accurate measure of whether or not the peptide is
secreted. The inner and outer labeling is only meaningful for integral membrane proteins.

When using the N-Best algorithm, you can also choose to merge predictions with a given overlap. The boundaries of transmembrane helices have an experimental error of a few residues, a fact which was incorporated into the training of the model architecture. By allowing a merging of overlapping predictions, TransMem allows you to blur edges of the predicted helices, which in turn will cause the N-Best algorithm to generate predictions with significant differences.

The N-Best algorithm will always try to find the best labeling of your sequence that matches the parameters of minimum and maximum number of helices, even if this is not the best overall labeling. For example, if you have some other experimental evidence that suggests you are working with a 7 transmembrane protein, yet the algorithm gives you a prediction of 8 transmembrane helices, you can specify a minimum and maximum helix range of 7, which will force the algorithm to find this prediction. If the application is not able to find any matching predictions, try increasing the value of N-Best, which will increase the number of different predictions that the algorithm will consider.

By increasing the number of different predictions generated, you are increasing the number of different predictions that TransMem analyzes. Consequently, you may see weakly predicted helices that would otherwise not be visible, as well as many more false predictions. Additionally, if a helix is visible in a large number of predictions, it is more likely to be an actual helix and not a false positive. Since you are increasing the number of predictions considered, computation time will also increase dramatically with increases in the value for N-Best.

Because of the N-Best algorithm's ability to try to find a prediction that matches the restrictions, it may not be useful for screening protein sequences for a given number of transmembrane helices. Instead, we recommend using the Viterbi algorithm, which is more discriminating and runs faster.

If you have a sequence for which you have experimental evidence of a particular number of transmembrane helices, yet the algorithm does not predict the correct number, specify this number with -MINHelix and -MAXHelix, then try increasing the value for N-Best and the tolerance for merging overlapping predictions. In some cases, this will allow the algorithm to find the helices.

If you are screening large amounts of data for 7 transmembrane proteins, for example, it probably isn't a good idea to limit the search for predictions of only 7 transmembrane regions. Instead, more complete searches can be generated from searching for anything containing 6-8 transmembrane regions.

TransMem only recognizes transmembrane alpha helices. All other types of membrane spanning regions are not recognized.

A Since TransMem will produce a self-consistent topology prediction, if it misses any transmembrane helices, the topology will be wrong.
Predicted transmembrane helices in the n-terminal region sometimes turn out to be signal peptides.

**SCIENTIFIC VALIDITY**

A non-redundant data set of 148 sequences, composed of all known transmembrane proteins (Möller et al, 2001), was used for validation of this program. The data set was run through the public server and through this implementation.

All except 8 sequences showed identical results (95% identical). NB: When the predictions differed, this program found the other prediction as the second best answer.

Of these 8 differences, 4 (COX2_BOVIN, IMMA_CITFR, RCEL_RHOVI, and TCR2_ECOLI) only differed in the exact positions of the helix boundaries. All predicted helices from the two implementations overlapped by at least 16 residues, and the topology predictions were identical.

There were 2 of the 8 proteins (COXH_BOVIN and CYB_RHOSH) where the topology predictions of the two implementations were reversed in addition to minor helix boundary differences. This implementation was correct for COXH_BOVIN and the public server was correct for CYB_RHOSH.

In the final 2 sequences, (CITN_KLEPN and CYOB_ECOLI), the two predictions differed in the presence or absence of a given TM helix. This implementation correctly found an additional helix in CITN_KLEPN. For CYOB_ECOLI, the public server correctly found an additional TM helix that this implementation did not find.

In conclusion, the two implementations are scientifically comparable. Half of the differences could be attributed to minor variation in TM helix boundaries, which are not significant differences, due to the inherent uncertainty in experimental determination of the helix boundaries. When the different implementations gave significant differences, there was an even split between which answer was correct.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use **-CHECK** to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you **must** type in order to use the parameter. Square brackets ([ ] and [ ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % transmem [-INfile=]sw:mouse -Default

Prompted Parameters:

[-OUTfile=]gad_mouse.transmem names the output file

-**NBest=1** Number of different annotations of each sequence
-**-PROXimity=0** Proximity of feature boundaries to consider annotations equivalent

Accelrys GCG 283
Optional Parameters:

- **-RSF[=transmem.rsf]** save predicted domains as features in an RSF file
- **-MEthod=Nbest,Viterbi** selects which method to use to generate the prediction. By default, Nbest is selected. (Selecting Viterbi suppresses -NBest and -PROXimity)
- **-RAWProb** writes out the raw probabilities of each label for each sequence character
- **-MAXHelix=10** only show proteins with at most this many transmembrane helices (default is unlimited)
- **-MINHelix=1** only show proteins with at least this many transmembrane helices (default is 1)
- **-MONitor** displays screen trace of progress
- **-NOSUMmary** suppresses screen summary at the end of the program

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

**-NB**est

Specify how many predictions you want to see. The most likely predictions is the first one listed. Note that larger numbers of predictions can greatly reduce program speed.

**-TOLerance**

When using the N-Best algorithm, merge predictions that have this much overlap or less. This allows you to avoid lists of predictions that are not functionally different.

**-VITerbi**

Use the Viterbi algorithm for predictions instead of the N-Best. This algorithm is faster than the N-Best, but can only generate a single prediction per sequence. If -VITerbi is used, the values for -NBest and for -TOLERance are ignored.

**-RAWProb**

Output the raw probabilities for observing each label at each sequence character. These values are based upon the forward-backward algorithm and may not agree with the final predicted label.
-MAXHelix

Limit the output to include only proteins with this many transmembrane helices or fewer. By default, the maximum number of helices is unlimited.

-MINHelix

Limit the output to include only proteins with this many or more transmembrane helices. If this value is greater than specified with -MAXHelix, the value for MAXHelix is used. By default, the output only includes proteins with one or more helix.

-SUMmary

Writes a summary of the program's work to the screen when you've used -default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.

Printed: February 10, 2006 19:41
TransMem+

FUNCTION

TransMem+ scans for likely transmembrane helices in one or more input protein sequences.

DESCRIPTION

Advantages of Plus “+” Programs:

✓ Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.

✓ Plus programs remove sequence length restriction of 350,000bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.

TransMem+ builds on the method of Sonnhammer et al. (Proc. of Sixth Int. Conf. on Intelligent Systems for Molecular Biology, 175-182 (1998)) to predict likely transmembrane helices in one or more input proteins. The method is based upon a Hidden Markov Model (HMM) that has been trained on a set of membrane proteins with helical membrane spanning regions.

EXAMPLE

Here is a session using TransMem+ to generate predictions for the 17Kda Surface antigens of Rickettsia.

20:16~74> transmem+

Transmem+ is a program that finds the Trans membrane regions for a given protein sequence. It is based upon Hidden Markov Model (HMM) architecture. The architecture is made up of 7 types of states corresponding to the core of the transmembrane helix, helix caps, cytoplasmic loops, short and long cytoplasmic loop states, and globular domains that are part of each loop.

transmem+ of which protein sequence(s) ? uniprot:17kd*
What should I call the output file (* <sequence_name>.transmem+ *) ?

Analyzing sequence '17KD_RICAM' from 'uniprot_sprot:17KD_RICAM'
Processing results...
No helices were found in uniprot_sprot:17KD_RICAM

Analyzing sequence '17KD_RICAU' from 'uniprot_sprot:17KD_RICAU'
Processing results...
No helices were found in uniprot_sprot:17KD_RICAU
Analyzing sequence '17KD_RICCA' from 'uniprot_sprot:17KD_RICCA'
Processing results...
Analyzing sequence '17KD_RICCN' from 'uniprot_sprot:17KD_RICCN'
Processing results...
No helices were found in uniprot_sprot:17KD_RICCN
Analyzing sequence '17KD_RICJA' from 'uniprot_sprot:17KD_RICJA'
Processing results...
No helices were found in uniprot_sprot:17KD_RICJA
Analyzing sequence '17KD_RICMO' from 'uniprot_sprot:17KD_RICMO'
Processing results...
No helices were found in uniprot_sprot:17KD_RICMO
Analyzing sequence '17KD_RICPA' from 'uniprot_sprot:17KD_RICPA'
Processing results...
No helices were found in uniprot_sprot:17KD_RICPA
Analyzing sequence '17KD_RICPR' from 'uniprot_sprot:17KD_RICPR'
Processing results...
No helices were found in uniprot_sprot:17KD_RICPR
Analyzing sequence '17KD_RICRH' from 'uniprot_sprot:17KD_RICRH'
Processing results...
No helices were found in uniprot_sprot:17KD_RICRH
Analyzing sequence '17KD_RICTY' from 'uniprot_sprot:17KD_RICTY'
Processing results...
No helices were found in uniprot_sprot:17KD_RICTY

Results written to transmem+.out

OUTPUT

The output from TransMem+ is a list file, and is suitable for input to any GCG program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-MEthod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec  7 20:16:21 2004

..  
Helix Inside Outside

uniprot_sprot:17KD_RICAM !0  0  1
\End of List

>>uniprot_sprot:17KD_RICAM
P50927 rickettsia amblyommii. 17 kda surface antigen precursor (fragment). 10/2003
Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-MEthod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

Helix  Inside  Outside

uniprot_sprot:17KD_RICAU !0         0         1
\\End of List

>>uniprot_sprot:17KD_RICAU
P50928 rickettsia australis. 17 kda surface antigen precursor (fragment). 10/2003

Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-MEthod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

<table>
<thead>
<tr>
<th>Helix</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniprot_sprot:17KD_RICCA !1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\End of List

>>uniprot_sprot:17KD_RICCA
P29697 rickettsia canada. 17 kda surface antigen (fragment). 10/1996

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>1 11</td>
</tr>
<tr>
<td>Helix</td>
<td>12 30</td>
</tr>
<tr>
<td>Inside</td>
<td>31 80</td>
</tr>
</tbody>
</table>

```
OGOGOGOGO OHHHHHHHH HHHHHHHHH IIIIIIIII IIIIIIIII
1 GSQFGKGKGQ LIGVGAGALL GAILGNQIGA GMDEQDRRLA ELTSQRALET

IIIIIIIIII IIIIIIIII IIIIIIIII
51 TPSGTSIEWR NPDNGNYGYV TPSKTYKNST
```

Sequences examined: 1
Sequences written: 1

!!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-MEthod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

<table>
<thead>
<tr>
<th>Helix</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniprot_sprot:17KD_RICCN !0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\End of List

>>uniprot_sprot:17KD_RICCN
P05372 rickettsia conorii, and rickettsia rickettsii. 17 kda surface antigen precursor. 10/2003

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>1 159</td>
</tr>
</tbody>
</table>

```
OGOGOGOGO OOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
1 MKLLSKIMII ALATSMLQAC NGPGGMNKQG TGTLGGAGG ALLGSQFGKG

OGOGOGOGO OOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
51 KGQILGVGGVG ALLGAVLGGQ IGAGMDEQDR RLAELTSQRA LETAPSGSNNV

OGOGOGOGO OOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
101 EWRNPDNGNY GYVTQNKTYR NSTGQYCQREY TQTQVGGKQ QKAYGNACRQ

OGOGOGOGO
151 PDGQWQVVPN
```
Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-METHod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

<table>
<thead>
<tr>
<th>Helix</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

uniprot_sprot:17KD_RICJA !0 0 1
\End of List

>>uniprot_sprot:17KD_RICJA
Q52764 rickettsia japonica. 17 kda surface antigen precursor. 10/2003

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>1 159</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>1 159</td>
</tr>
</tbody>
</table>

Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-METHod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

<table>
<thead>
<tr>
<th>Helix</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

uniprot_sprot:17KD_RICMO !0 0 1
\End of List

>>uniprot_sprot:17KD_RICMO
P50929 rickettsia montana. 17 kda surface antigen precursor (fragment). 10/2003

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>1 154</td>
</tr>
</tbody>
</table>
Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot: 17kd*
-MINHelix = 0
-METHod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

<table>
<thead>
<tr>
<th></th>
<th>Helix</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniprot_sprot:17KD_RICPA</td>
<td>!0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
\End of List

>>uniprot_sprot:17KD_RICPA
P50930 rickettsia parkeri. 17 kda surface antigen precursor (fragment).
10/2003

<table>
<thead>
<tr>
<th>Outside</th>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>154</td>
</tr>
</tbody>
</table>

Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot: 17kd*
-MINHelix = 0
-METHod = VITERBI
-nbest = 1
-PROXimity = 0
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside</td>
<td>Begin</td>
<td>End</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>MKLLSKIMII ALAASMLQAC NGQGSMNKQG TGTLGGAGG ALLGSQFGQG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O0000000000 O0000000000 O0000000000 O0000000000 O0000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O0000000000 O0000000000 O0000000000 O0000000000 O0000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KQQLVGVGVLV ALLGAVLGGQ IGASMDEQDR RLLELTSLQA LESAPSGSNI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O0000000000 O0000000000 O0000000000 O0000000000 O0000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EWRNPNDNGNY GYITPNKTYR NSAQGQCREY QVTIVIGGKQ OKTYGNACR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PDGQWQQVNV</td>
<td></td>
</tr>
</tbody>
</table>

Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-METHod = VITERBI
-nbest = 1
-PROXimity = 0

Tue Dec 7 20:16:21 2004
Sequences examined: 1
Sequences written: 1

!!SEQUENCE_LIST 1.0
Transmem of uniprot:17kd*
-MINHelix = 0
-METHod = VITERBI
-nbest = 1
-PROXimity = 0
Tue Dec 7 20:16:21 2004

Helix    Inside    Outside

.. uniprot_sprot:17KD_RICTY !0         0         1
\End of List

>>uniprot_sprot:17KD_RICTY
P22882 rickettsia typhi. 17 kda surface antigen precursor. 10/2003

  Begin     End
Outside
                                 1     159
            OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
                 1 MKLLSKVMIL ALAASMLQAC NGPGGMNKQG TGTLGGAGG ALLGSQFGHG
            OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
                 51 KGQLVGVGVG ALLGAVLGGQ IGASLDEQDR KLELETSQRA LESAPSGSNI
            OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
                 101 EWRNPDNGNH GYVTVPNKTYR NSTGQYCREY TQTVVIGGKQ QTITYGNACRQ
            OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO OOOOOOOOOOO
                  151 PDGQWQVQVN

Sequences examined: 1
Sequences written: 1

INTERPRETING OUTPUT

The first part of the output file contains a list of all the sequences searched and the predictions generated for a given sequence. When multiple predictions are generated for each sequence, the predictions are listed in order of prediction quality, with the best prediction on top and the sub-optimal predictions below.

Next to each sequence, the file contains the raw counts of how many transmembrane helices, inner loops, and outer loops were found. If you have generated more than one prediction per sequence, there is also a score reported for comparing the quality of the prediction with the best prediction for each sequence. This is a relative measure only and should not be used to compare the quality of predictions between different...
sequences. In general, a score of 10 or more indicates that the prediction is significantly different from the best prediction.

Following this list of sequences, TransMem+ displays a table listing the specific boundaries of each feature predicted, followed by the sequence aligned with the predicted labels.

**INPUT FILES**

TransMem+ takes any valid GCG specification for one or more protein sequences.

**RELATED PROGRAMS**

TransMem scans for likely transmembrane helices in one or more input protein sequences.

SPScan+ scans protein sequences for the presence of secretor signal peptides (SPs).

HTHScan+ scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

HelicalWheel plots a peptide sequence as a helical wheel to help you recognize amphiphilic regions.

PeptideStructure makes secondary structure predictions for a peptide sequence. The predictions include (in addition to alpha, beta, coil, and turn) measures for antigenicity, flexibility, hydrophobicity, and surface probability. PlotStructure displays the predictions graphically.

PepPlot plots measures of protein secondary structure and hydrophobicity in parallel panels of the same plot.

SPScan scans protein sequences for the presence of secretor signal peptides (SPs).

HTHScan scans protein sequences for the presence of helix-turn-helix motifs, indicative of sequence-specific DNA-binding structures often associated with gene regulation.

CoilScan+ locates coiled-coil segments in protein sequences.

Coilscan locates coiled-coil segments in protein sequences.

**RESTRICTIONS**

TransMem+ only works on protein sequences.
ALGORITHM

TransMem+ is based upon Hidden Markov Model (HMM) architecture. The architecture is made up of 7 types of states corresponding to the core of the transmembrane helix, helix caps, cytoplasmic loops, short and long cytoplasmic loop states, and globular domains that are part of each loop.

The states have a close relationship with the biology of membrane proteins; loop states connection to other loops through a helix cap, helix core, and another helix cap. These states correspond to one of three different labels, Inside (cytoplasmic), Helix (membrane spanning helix), and Outside (non-cytoplasmic).

The prediction of transmembrane helices is done by finding an optimal alignment of the sequence with the model using the N-Best algorithm. In the N-Best algorithm, the algorithm uses the model architecture to find the best labeling of the sequence, given the model.

Alternatively, you can run TransMem+ using the Viterbi algorithm, which finds the optimal alignment of the sequence with the model, then uses that alignment to read the labels. In general, the Viterbi algorithm will give the same results as the N-Best, but in some cases the predictions will differ.

The output of the raw probabilities is based upon the forward-backward algorithm, in which TransMem+ finds the probability of each labeling (Inside, Outside, or Helix) summed over all the possible alignments of the sequence to the model. Because these values are based upon all possible alignments of the model instead of a single optimal alignment, occasionally the raw probabilities will contradict the final labeling.

CONSIDERATIONS

When no transmembrane helices are predicted, it is not a good idea to treat the Inside/Outside prediction as an accurate measure of whether or not the peptide is secreted. The inner and outer labeling is only meaningful for integral membrane proteins.

When using the N-Best algorithm, you can also choose to merge predictions with a given overlap. The boundaries of transmembrane helices have an experimental error of a few residues, a fact which was incorporated into the training of the model architecture. By allowing a merging of overlapping predictions, TransMem+ allows you to blur edges of the predicted helices, which in turn will cause the N-Best algorithm to generate predictions with significant differences.

The N-Best algorithm will always try to find the best labeling of your sequence that matches the parameters of minimum and maximum number of helices, even if this is not the best overall labeling. For example, if you have some other experimental evidence that suggests you are working with a 7 transmembrane protein, yet the algorithm gives you a prediction of 8 transmembrane helices, you can specify a minimum and maximum helix range of 7, which will force the algorithm to find this prediction. If the application is not able to find any matching predictions, try increasing
the value of N-Best, which will increase the number of different predictions that the algorithm will consider.

By increasing the number of different predictions generated, you are increasing the number of different predictions that TransMem+ analyzes. Consequently, you may see weakly predicted helices that would otherwise not be visible, as well as many more false predictions. Additionally, if a helix is visible in a large number of predictions, it is more likely to be an actual helix and not a false positive. Since you are increasing the number of predictions considered, computation time will also increase dramatically with increases in the value for N-Best.

Because of the N-Best algorithm's ability to try to find a prediction that matches the restrictions; it may not be useful for screening protein sequences for a given number of transmembrane helices. Instead, we recommend using the Viterbi algorithm, which is more discriminating and runs faster.

If you have a sequence for which you have experimental evidence of a particular number of transmembrane helices, yet the algorithm does not predict the correct number, specify this number with -MINHelix and -MAXHelix, then try increasing the value for N-Best and the tolerance for merging overlapping predictions. In some cases, this will allow the algorithm to find the helices.

If you are screening large amounts of data for 7 transmembrane proteins, for example, it probably isn't a good idea to limit the search for predictions of only 7 transmembrane regions. Instead, more complete searches can be generated from searching for anything containing 6-8 transmembrane regions.

TransMem+ only recognizes transmembrane alpha helices. All other types of membrane spanning regions are not recognized.

Since TransMem+ will produce a self-consistent topology prediction, if it misses any transmembrane helices, the topology will be wrong.

Predicted transmembrane helices in the n-terminal region sometimes turn out to be signal peptides.

**SCIENTIFIC VALIDITY**

A non-redundant data set of 148 sequences, composed of all known transmembrane proteins (Möller et al, 2001), was used for validation of this program. The data set was run through the public server and through this implementation.

All except 8 sequences showed identical results (95% identical). NB: When the predictions differed, this program found the other prediction as the second best answer.

Of these 8 differences, 4 (COX2_BOVIN, IMMA_CITFR, RCEL_RHOVI, and TCR2_ECOLI) only differed in the exact positions of the helix boundaries. All predicted helices from the two implementations overlapped by at least 16 residues, and the topology predictions were identical.
There were 2 of the 8 proteins (COXH_BOVIN and CYB_RHOSH) where the topology predictions of the two implementations were reversed in addition to minor helix boundary differences. This implementation was correct for COXH_BOVIN and the public server was correct for CYB_RHOSH.

In the final 2 sequences, (CITN_KLEPN and CYOB_ECOLI), the two predictions differed in the presence or absence of a given TM helix. This implementation correctly found an additional helix in CITN_KLEPN. For CYOB_ECOLI, the public server correctly found an additional TM helix that this implementation did not find.

In conclusion, the two implementations are scientifically comparable. Half of the differences could be attributed to minor variation in TM helix boundaries, which are not significant differences, due to the inherent uncertainty in experimental determination of the helix boundaries. When the different implementations gave significant differences, there was an even split between which answer was correct.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use `-check` to view the summary below and to specify parameters before the program executes. In the syntax summary below, square brackets ([ and ]) enclose parameter values that are optional. For each program parameter, square brackets enclose the type of parameter value specified, the default parameter value, and shortened forms of the parameter name, aliases. Programs with a plus in the name use either the full parameter name or a specified alias. If “Type” is “Boolean”, then the presence of the parameter on the command line indicates a true condition. A false condition needs to be stated as, parameter=false. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

```
%transmem+ -che
```

Transmem+ is a program that performs an HMM search of any number of sequences against an HMM.

**Minimal Syntax:** `% transmem+ [-infile=value] -Default`  
**Minimal Parameters (case-insensitive):**

- **-infile**
  
  [Type: InFile / Default: EMPTY / Aliases: infile1 in]  
  The name of the input file.

**Prompted Parameters (case-insensitive):**

- **-outfile**
  

**Optional Parameters (case-insensitive):**

- **-check**
  
  [Type: Boolean / Default: 'false' / Aliases: che help] Prints out this usage message.

- **-default**
  
  [Type: Boolean / Default: 'false' / Aliases: d def]
Specifies that sensible default values be used for all parameters where possible.

-**documentation** [Type: Boolean / Default: 'true' / Aliases: doc] Prints banner at program startup.

-**quiet** [Type: Boolean / Default: 'false' / Aliases: qui] Tells application to print only a minimal amount of information.


-**method** [Type: String / Default: 'VITERBI' / Aliases: me] Selects which method to use to generate the prediction. Valid values are VITERBI and NBEST.

-**architecture** [Type: InFile / Default: '$GCGROOT/share/hmm/tmhmm.arch' / Aliases: arch] Sets the HMM architecture file.

-**nbest** [Type: Integer / Default: '1' / Aliases: nb] Number of different annotations of each sequence. Only applies when method=NBEST.

-**proximity** [Type: Integer / Default: '0' / Aliases: prox] Proximity of feature boundaries to consider annotations equivalent. Only applies when method=NBEST.

-**rawprob** [Type: Boolean / Default: 'N' / Aliases: raw] Writes out the raw probabilities of each label for each sequence character.

-**maxhelix** [Type: Integer / Default: '2147483647' / Aliases: maxh] Only show proteins with at most this many transmembrane helices (default is unlimited).

-**minhelix** [Type: Integer / Default: '0' / Aliases: minh] Viterbi algorithm: Only show proteins with at least this many transmembrane helices. NBest algorithm: Forces the algorithm to find at least this many helices.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. Shortened forms of the parameter name, aliases, are shown, separated by commas. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- **nbest, -nb**

  Specify how many predictions you want to see. The most likely predictions is the first one listed. Note that larger numbers of predictions can greatly reduce program speed.
-infile, -infile1, -in

The name of the input file.

-outfile, -out

Names the output file.

-check, -che, -help

Prints out this usage message.

-default, -def

Specifies that sensible default values be used for all parameters where possible.

-documentation, -doc

Prints banner at program startup.

-quiet, -qui

This parameter is not supported.

-seqout

Annotated sequence output.

-method, -me

Selects which method to use to generate the prediction. Valid values are VITERBI and NBEST.

-architecture, -arch

Sets the HMM architecture file.

-proximity, -prox

Proximity of feature boundaries to consider annotations equivalent. Only applies when method=NBEST.

-rawprob, -rawp

Output the raw probabilities for observing each label at each sequence character. These values are based upon the forward-backward algorithm and may not agree with the final predicted label.
-maxhelix, -maxh

Limit the output to include only proteins with this many transmembrane helices or fewer. By default, the maximum number of helices is unlimited.

-minhelix, -minh

Limit the output to include only proteins with this many or more transmembrane helices. If this value is greater than specified with -maxhelix, the value for maxhelix is used. By default, the output only includes proteins with one or more helix.
WINDOW

FUNCTION

Window makes a table of the frequencies of different sequence patterns within a window as it is moved along a sequence. A pattern is any short sequence like GC or R or ATG. You can plot the output with the program StatPlot.

DESCRIPTION

Window calculates the frequency of patterns within a window of a set length. A pattern is any short sequence such as GC, or R, or ATG. The output is a table of numbers suitable for input to the StatPlot program. The window is moved along the sequence by a shift increment, and the number of observations of the pattern at every window position is measured. The frequency can be reported as a fraction, a percent, or simply a number of observations. You can also ask to see the difference between the number of observations of the pattern and the expected number of observations for a random sequence of identical composition. This expectation can be based either on the composition within the window (local) or on the composition of the whole sequence range (global). Another statistic lets you see the difference in frequency between two patterns. The pattern frequencies measured by Window are for one strand only.

PARAMETERS

You define the window size and the shift increment. The shift increment is the amount the window is moved between measurements. From a menu of the eight possible measures, you may choose up to six. Each measure you choose makes a column in the output table. After choosing the measurements, you are prompted to enter the pattern you want measured. For each measurement you must designate a pattern when prompted with a question that reminds you of the kind of measurement and the column number.

EXAMPLE

Here is a session using Window to measure the frequency of C, G, CG, and GC in the sequence gamma.seq. You can see from this experiment whether or not the frequency of the dinucleotide CG correlates well with the content of the nucleotides C and G (it doesn't). The output file from this session with Window is plotted as an example in the program StatPlot.

% window

WINDOW on what sequence ? gamma.seq

Begin (* 1 *) ?
End (* 11375 *) ? 500
Reverse (* No *) ?

What window size (* 100 *) ?

What shift increment (* 3 *) ?
What should I call the output file (*gamma.wdw*)?

What functions do you want:

a) number of patterns observed
b) percent of patterns observed
c) fraction of patterns observed
d) number of observed - expected(local) patterns
e) number of observed - expected(global) patterns
f) percent of observed - expected(local) patterns
g) percent of observed - expected(global) patterns
h) percent difference between two patterns

q)uit

Please select up to 6 functions (*ae*):

```
aadad
```

What is the pattern for the "a" stat in column 1 ? c
What is the pattern for the "a" stat in column 2 ? g
What is the pattern for the "a" stat in column 3 ? cg
What is the pattern for the "d" stat in column 4 ? cg
What is the pattern for the "a" stat in column 5 ? gc
What is the pattern for the "d" stat in column 6 ? gc

% 

OUTPUT

Some of the output file is shown below. You can see the data plotted in the figure with the documentation for the StatPlot program.

WINDOW of: gamma.seq  check: 6474  from: 1 to: 500
Window: 100  Shift: 3  MatchType: Subset MisMatch: 0

Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies,  Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.

October 13, 1998 13:06

<table>
<thead>
<tr>
<th>Position</th>
<th>C(obsrv)</th>
<th>G(obsrv)</th>
<th>CG(obsrv)</th>
<th>CG_ob-ex(l)</th>
<th>GC(obsrv)</th>
<th>GC_ob-ex(l)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>17.000</td>
<td>30.000</td>
<td>1.000</td>
<td>-4.049</td>
<td>4.000</td>
<td>-1.049</td>
</tr>
<tr>
<td>53</td>
<td>19.000</td>
<td>29.000</td>
<td>1.000</td>
<td>-4.455</td>
<td>5.000</td>
<td>-0.455</td>
</tr>
<tr>
<td>56</td>
<td>17.000</td>
<td>30.000</td>
<td>1.000</td>
<td>-4.049</td>
<td>5.000</td>
<td>-0.049</td>
</tr>
<tr>
<td>443</td>
<td>31.000</td>
<td>14.000</td>
<td>0.000</td>
<td>-4.297</td>
<td>2.000</td>
<td>-2.297</td>
</tr>
<tr>
<td>446</td>
<td>32.000</td>
<td>14.000</td>
<td>0.000</td>
<td>-4.435</td>
<td>2.000</td>
<td>-2.435</td>
</tr>
<tr>
<td>449</td>
<td>32.000</td>
<td>13.000</td>
<td>0.000</td>
<td>-4.118</td>
<td>2.000</td>
<td>-2.118</td>
</tr>
</tbody>
</table>
INPUT FILES

Window accepts a single sequence file as input. The function of Window depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

StatPlot plots a set of parallel curves from a table of numbers like the table written by the Window program. The statistics in each column of the table are associated with a position in the analyzed sequence.

RESTRICTIONS

The region of the input sequence to be analyzed may not be more than 175,000 symbols long.

No more than six statistics can be tabulated. The shift increment cannot exceed the window size. Numbering in the Position column is for the forward strand even if the reverse strand is chosen.

Pattern definitions can only contain GCG sequence characters (see Appendix III). We could easily modify Window to find patterns using a pattern definition syntax like that used for FindPatterns. Contact us if you think this is a good idea!

ALGORITHM

Each observation of a pattern is stored in a logical array. This array has a true (pattern observed) or false (pattern not observed) value for every position in the original sequence.

After the observation array is assembled, the incidence of each pattern can be found simply by putting down the window as a mask over the array and counting the observations under the window. The window is moved along the array (sequence) by the set shift increment and the observations are counted again.

Window calculates the number of observations per window in the following manner. The fraction of each symbol in the pattern is measured, either in the window (local expectation) or in the whole sequence range (global expectation). The product of the fractions for each symbol in the pattern multiplied by the maximum possible number of patterns in the window is the expected number of observations for the pattern in the window. Four of the measurements report the difference between the actual number of observations and the expected number.

The percentage measures are simply the number of observations divided by the maximum possible number of patterns in the window and multiplied by 100.
Fraction measures are the number of observations divided by the maximum possible number of patterns in the window.

**SUBSET MATCHING**

For nucleic acid sequences, the ambiguity codes in Appendix III are searched for *subset matches*. For instance, if the pattern specified is RR and the sequence contains an AG, an observation is scored at the position of the A. If the pattern specified were AG and the sequence contained an RR, no match would be scored. The sequence symbols must be the same as or a subset of the nucleotides implied by the pattern symbols.

**PERFECT MATCHING**

If the sequence is a peptide sequence or if you have `-PERfect` on the command line, Window scores occurrences of patterns by finding perfect examples of the pattern in the sequence.

**OVERLAPPING SET MATCHING**

If you use the command-line parameter `-ALL` and your sequence is a nucleic acid sequence, the sequence can be an overlapping set of the pattern instead of only a subset. (In other words, ambiguous bases in the sequence can match bases in the pattern even if the sequence's ambiguous base is not a subset of the pattern's base.) Using the same example as in the SUBSET MATCHING topic, the pattern AG would now match the sequence RR. As another example, the pattern RA would match the sequence MK.

**CONSIDERATIONS**

The cost of running Window is very low, but the output files can be very large. You should recognize that Window writes one line in the output file for every position of the window. Running Window on a sequence of length 10,000, with window size 100, shift increment 1, and using five measures will generate an output file with about 10,000 lines and about 60,000 numbers.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use `-CHEck` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Window does not support complete command-line control.

Prompted Parameters:
- `BEGIN=2101 -END=2600` sets the range of interest
- `REVerse` uses the reverse strand
Optional Parameters:

- **-ALL** makes an overlapping-set search
- **-MISmatch** allows mismatches between the pattern and the sequence
- **-PERFect** suppresses ambiguous matches for nucleic acid sequences

**LOCAL DATA FILES**

None.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

**-ALL**

Makes an overlapping-set search for patterns in nucleic acid sequences. If your sequence is rich in ambiguity, you can measure the frequency of potential examples of patterns.

**-MISmatch**

Allows mismatches between the pattern and the sequence. Window will prompt you for the number of mismatches to allow.

**-PERFect**

Normally, Window searches for patterns using subset matching in nucleic acids and perfect matching in peptide sequences. You can override the subset default with the command-line parameter **-PERFect** to suppress all matches between ambiguous base symbols.

Printed: February 10, 2006 19:41
WordSearch identifies sequences in the database that share large numbers of common words in the same register of comparison with your query sequence. The output of WordSearch can be displayed with Segments.

**WORDSEARCH\(^2\)**

**FUNCTION**

WordSearch uses an algorithm similar to the algorithm of Wilbur and Lipman (Proc. Natl. Acad. Sci. (USA) **80**: 726-730 (1983)) to compare one sequence (the *query*) to any group of sequences. You should think of the comparisons as a set of dot-plots with the query as the vertical sequence and the group of sequences to which the query is being compared as the different horizontal sequences (the *search set*). The search finds the registers of comparison (*diagonals*) that have the largest number of short perfect matches (*words*). The best segment of similarity along each diagonal can be viewed with the program Segments.

**What is a Word?**

A word is any short sequence (*n*-mer) where you have set \(n\) to some small constant like six or seven. The word GGATGCGC is one of the 16,384 possible words of length seven that can be created from an alphabet consisting of the four letters G, A, T, and C. The word QQL is one of the 8,000 possible words of length three that you can make with the 20 letters of the amino acid alphabet.

**What is a Word Mask?**

The symbols that match between two words need not be contiguous. You can use the characters + and - to define a word *mask* like +++-++. This mask means that matching words should match at positions 1, 2, 4, 5, 7, and 8 and that positions 3 and 6 may or may not match.

**What is a Diagonal?**

A diagonal is a register of comparison for two sequences -- a path across a surface of comparison where \(X - Y\) for every point is a constant. A series of dots along a diagonal represents a segment of similarity between two sequences. Each diagonal can be defined by the constant \(X - Y\) for that diagonal. The path up from the origin is numbered zero. The paths above the zero diagonal are negative and the paths below the zero diagonal are positive. The diagonals are then numbered between minus the length of the vertical (query) sequence and plus the length of the horizontal (search set) sequence.

**What is the Output?**

WordSearch sorts the scores of all the diagonals in your comparison and shows you a list of the best diagonals where you have restricted the size of the list to some finite number like 50 or 100. You can see optimal alignments of the segments of similarity in the WordSearch output file with the Segments program.
List File

WordSearch compares both strands of your query sequence to any set of sequences you name and shows the best diagonals and the number of symbols within matching words on each of these best diagonals. The diagonals are identified with the coordinate X - Y (described above), the number of symbols within the matching words for that diagonal, the strand of the query sequence, and the name of the search set sequence.

Score Distribution Plot

WordSearch makes a histogram showing the number of diagonals observed for each diagonal score. The histogram shows the distribution of diagonal scores so you can see if a particular diagonal in your list of best diagonals is significant.

EXAMPLE

Here is a session using WordSearch to find sequences in the GenBank nucleotide sequence database with similarities to a human globin coding sequence.

```bash
% wordsearch -PLO -MASk
(Masked) WORDSEARCH with what query sequence ? ggammacod.seq

               Begin (* 1 *) ?
               End (* 444 *) ?

Search for query in what sequence(s) (* GenBank:* *) ?

What word-mask (* +++-++-++ *) ?

List how many best diagonals (* 50 *) ?

Integrate how many adjacent diagonals (* 3 *) ?

What should I call the output file (* ggammacod.word *) ?

1 A16SRRNA       Len:  1,497
101 AB000354     Len:   607
201 AB001715     Len:   876

////////////////////////////////////////////////////////

8-mers found: 2,000,000,000
Diagonals with words: 154,369,323
Total diagonals: 2,000,000,000
Sequences searched: 552,323
CPU time: 17:06.78

Output file: ggammacod.word

When your LaserWriter attached to tty07 is ready, press <Return>.

%
OUTPUT

WordSearch produces a list file containing the names of sequences that contain the best diagonals in your search and optionally can plot the distribution of scores from the search. Here is some of the output file:

!!SEQUENCE_LIST 1.0
(Masked) (Nucleotide) WORDSEARCH of: ggammacod.seq check: 2906
from: 1 to: 444

ASSEMBLE July 27, 1994 11:40
Symbols: 1 to: 92 from: gamma.seq ck: 6474, 2179 to: 2270
Symbols: 93 to: 315 from: gamma.seq ck: 6474, 2393 to: 2615
Symbols: 316 to: 444 from: gamma.seq ck: 6474, 3502 to: 3630

Human fetal beta globins G and A gamma from Shen, Slightom and Smithies, Cell 26; 191-203. . . .

TO: GenBank:* Sequences: 552,323 Total-length: 1,036,534,882
October 19, 1998 12:02

Database Release Information:

Mask: "+++++++"
Word-size: 8 Words: 2000000000 Diagonals: 154,369,323
Total-diagonals: 2,000,000,000
Integral-width: 3 Alphabet: 4 List-size: 50 CPU minutes: 17.11

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Strd</th>
<th>Diag</th>
<th>Score</th>
<th>Width</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB_PR2:HUMHBGG</td>
<td>+</td>
<td>17</td>
<td>442</td>
<td>3</td>
<td>M15386 Human hemoglobin ...</td>
</tr>
<tr>
<td>GB_PAT:I42109</td>
<td>+</td>
<td>-1</td>
<td>440</td>
<td>4</td>
<td>I42109 Sequence 4 fro ...</td>
</tr>
<tr>
<td>GB_PR1:HSGGPHG</td>
<td>+</td>
<td>-20</td>
<td>378</td>
<td>3</td>
<td>X55656 H.sapiens mRNA ...</td>
</tr>
</tbody>
</table>

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Strd</th>
<th>Diag</th>
<th>Score</th>
<th>Width</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB_PR1:HUMHBGG</td>
<td>+</td>
<td>2300</td>
<td>211</td>
<td>3</td>
<td>M32723 Human G-gamma- ...</td>
</tr>
<tr>
<td>GB_PR1:MMGGLINE</td>
<td>+</td>
<td>7197</td>
<td>209</td>
<td>3</td>
<td>X53419 M.mulatta gamm ...</td>
</tr>
<tr>
<td>GB_PR1:MMGGLINE</td>
<td>+</td>
<td>2318</td>
<td>209</td>
<td>3</td>
<td>X53419 M.mulatta gamm ...</td>
</tr>
</tbody>
</table>

308 Program Manual
SCORE DISTRIBUTION PLOT

If you run WordSearch with -PLOT, it plots a histogram showing the number of diagonals observed with each different score. This plot should help you judge which of the diagonals in your output list are significant and whether the output list was large enough to contain all of the significant diagonals. Here is the score distribution plot from the example session:

By looking at a plot like this one, you can conclude that observations with a score of less than about 80 are probably part of the population of diagonals with only random similarity to ggammacod.seq. (The example has an unusual number of significant similarities arising from the fact that many similar globins have been sequenced.)

Bin Size

You can set the resolution of the score distribution plot with -BINsize. By default, each histogram is integrated into bins that are the size of the word length. For words of length 6, the histograms would normally show the frequency of diagonals with scores from 0 to 5, 6 to 11, 12 to 17, and so forth.

The Histogram Shows Scores for Structures

The histogram shows the scores for diagonals after processing into structures. See the ALGORITHM topic below for a description of how scores accumulate on diagonals and the way scores are grouped into structures before becoming eligible to join the list of best diagonals.

List Cutoff

Ideally the list of best diagonals should be large enough to include some diagonals from the high end of the random scores. The list of best diagonals may not have been large.
enough, however, to show all of the diagonals with significant scores. The cutoff or lowest score in the output list is marked on the "Diagonal Scores" axis with an asterisk (*). Notice that the list size was not large enough to include all of the globin sequences in GenBank.

Inset Plot

The end of the histogram with the best observations (highest scores) is magnified into a small plot in the upper-right corner. The inset plot simply expands the vertical axis tenfold so that the number of high-scoring diagonals can be read exactly.

INPUT FILES

WordSearch accepts either a nucleotide or a protein sequence as input. The function of WordSearch depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

Segments aligns and displays the segments of similarity found by WordSearch.

If you run Compare with -WOR, the program calculates the points for a dot plot that shows where common words between two sequences occur.

SSearch does a rigorous Smith-Waterman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). This may be the most sensitive method available for similarity searches. Compared to BLAST and FastA, it can be very slow.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds.

FastA does a Pearson and Lipman search for similarity between a query sequence and a group of sequences of the same type (nucleic acid or protein). For nucleotide searches, FastA may be more sensitive than BLAST. TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

ProfileSearch uses a profile (representing a group of aligned sequences) as a query to search the database for new sequences with similarity to the group. The profile is created with the program ProfileMake. HmmerSearch uses a profile hidden Markov model as a query to search a sequence database to find sequences similar to the family from which the profile HMM was built. Profile HMMs can be created using HmmerBuild.
FastX does a Pearson and Lipman search for similarity between a nucleotide query sequence and a group of protein sequences, taking frameshifts into account. FastX translates both strands of the nucleic sequence before performing the comparison. It is designed to answer the question, "What implied protein sequences in my nucleic acid sequence are similar to sequences in a protein database?" TFastX does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences, taking frameshifts into account. It is designed to be a replacement for TFastA, and like TFastA, it is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?"

FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts.

FindPatterns, StringSearch and Names are other sequence identification programs.

**RESTRICTIONS**

The query sequence may not be more than 30,000 symbols long. You may not select a list size of more than 1,000 "best" diagonals. The word size should be from 1 to 30. Word searching is subject to many limitations and considerations which are discussed further below.

**CONSIDERATIONS**

**The Match Criterion**

The match criterion for two words is that all of the symbols in each word are identical. The symbols that must be identical need not be contiguous if a word mask has been set, but the symbols that must match must be identical, except for case. There is no scoring matrix and no support for the equivalence of nucleic acid ambiguity codes. Lower- and uppercase letters are equivalent however.

**Word Searching Requires some Perfect Identity**

The basic assumption of word comparisons is that patterns of similarity have an unusual number of common words (short perfect matches) along a set of closely spaced diagonals. This is often the case for nucleic acid sequences that have diverged recently, but it may not be true for protein comparisons. You should consider this assumption carefully. When two sequences have diverged sufficiently so that an optimal alignment of them has one mismatch for every six bases, then a word comparison with words of length six may not recognize their similarity.
Sequence Simplification May Increase the Level of Perfect Identity

-SIMplify allows you to map the sequences' symbols into a simpler subset of symbols to find matches between categories of sequence symbols.

Queries Containing Repetitive or Simple Sequences

If you use a query sequence containing a mammalian Alu-family sequence, you are in danger of finding the hundreds of Alu-family sequences that have been published to the exclusion of anything else. The ideal query sequence contains no simple (e.g., polyA) or repeated sequences. Ideally the query should be short enough so that any segment of similarity generates an unusual peak on the histogram. If the query is shorter than 500 bases, most of the diagonals are approximately the same length. Short diagonals of similar length increase the probability that word scores from a small segment of similarity are not lost in the background noise.

SUGGESTIONS

Word Size

You might try a word size of six and an integral width of three for nucleic acid searches as suggested by the program's defaults. You should recognize that when the average word occurs in the query sequence more than zero times, the amount of CPU time rises dramatically. You could start with a word size of two for protein sequence comparisons.

Word Mask

A word mask calling for two matches followed by one uncertainty is more sensitive for recognizing protein coding sequences than a simple contiguous word search. You can set up a word mask by including an expression like -MASx=++++++++, using a plus sign (+) to show the positions where the symbols must match and a minus sign (-) to show the positions where symbols may or may not match. Wobble in the third codon position in the genetic code would make a mask like ++++ more sensitive than +++++ for recognizing similar coding regions.

It does not make sense to define a mask with leading or trailing - characters, and therefore WordSearch removes these. Defining a word mask suppresses the word size query since the word size is inherent in the mask you have chosen. The word size of the mask ++++++++ is eight, even though only six of the eight characters under the mask must match.

List Size

The list should be large enough to cover all of the significant scores with at least 10 scores seeming to arise from the high end of the random scores. The default list size of 50 is large enough for most query sequences, but it is not large enough to include all of the globins in the sample session.
Identifying the Search Set

For information about naming groups of sequences, see Chapter 2, Using Sequence Files and Databases of the User's Guide.

Batch Queue

WordSearch is one of the few programs in the Accelrys GCG (GCG) that can take more than a few minutes to run. Most comparisons should probably be run in the batch queue. You can specify that this program run at a later time in the batch queue by using -BATch. Run this way, the program prompts you for all the required parameters and then automatically submits itself to the batch or at queue. For more information, see "Using the Batch Queue" in Chapter 3, Using Programs in the User's Guide. Very large comparisons may exceed the CPU limit set by some systems.

When WordSearch is run in batch as `% wordsearch -PL0t -BATch`, instructions for plotting the optional histogram is written to a figure file named wordsearch.figure, unless the plot has been directed to a specific file or graphics device from the command line. Please see the entry for the Figure program in the Program Manual for instructions on how to plot a figure file to any graphics device that GCG supports.

Interrupting a Search: <Ctrl>C

You can type <Ctrl>C to interrupt a search and see the results from the part of the search that has already been completed.

LIST REFINEMENT

The database programs LookUp, Names, StringSearch, FindPatterns, FastA, TFastA, FastX, TFastX, SSearch, and WordSearch can be used for list refinement if you are looking for sequences with something in common. For instance, you could identify human globin nucleotide sequences with LookUp. The output list from LookUp could then be refined further with FindPatterns to show only those human globin sequences containing EcoRI sites. If you run FindPatterns with -NAMes, you could then do a FastA sequence search on the FindPatterns list file output to see if a sequence you have is similar to any of these EcoRI-containing human globin sequences.

Adding Lists Together

You can add two lists together by simply appending one of the files to the other. It is better if you use a text editor to modify the heading of the combined list so that the annotation in the list correctly reflects what you have done. Remember to delete the text heading from the second file so that it does not occur in the middle of the list.

Suppressing Items

Suppress any item in a list by typing an exclamation point (!) in front of the item. You can also put comments into a list anywhere on a line by placing an exclamation point before the comment.
OUTPUT FILE FORMAT

WordSearch assembles a list of the best places in your search set to look for similarities to your query sequence. The output is a list file and is therefore suitable for input to any program that allows indirect file specifications. (For information about indirect file specification, see Chapter 2, Using Sequence Files and Databases of the User's Guide.)

The Heading

The first part of the output file contains heading information about the parameters of the search, including a definition of the query sequence, the word size, the window of integration, the size of the desired list, the number of symbols found within matching words (after integration), the number of diagonals on which those words were found, the total number of diagonals in the search, and the size of the alphabet of symbols used. Several lines of the WordSearch output file have a specific format; if these lines are altered, the Segments program will not be able to read the file.

The List of Best Diagonals

The second part of the file contains the list of significant diagonals. These diagonals are defined by the following features: the sequence name, the strand (+ or -), the \( X - Y \) coordinate that identifies the peak diagonal (\( \text{Diag} \)), the number of symbols on the diagonal that were within matching words (\( \text{Score} \)), the width of the structure (\( \text{Width} \)), and a short line of documentation. All of this information is read by the Segments program. (See the ALGORITHM topic below for a further explanation of the information listed with each significant diagonal.)

ALGORITHM

The algorithm described below may be referred to as a hash-table/linked-list search. Wilbur and Lipman searches are an example of a class of comparisons that use direct addressing or k-tuple preprocessing to reduce search time.

You set a word size or define a word mask, which implies a word size. Then WordSearch makes up a dictionary of all of the possible words of that size in the query sequence. A second dictionary is compiled for the opposite strand if the query is a nucleic acid sequence. The dictionary has an entry for every possible word. Imagine each word, such as GGATGG, as a number in base four that corresponds to an entry in the dictionary. At each entry, there is a number telling the positions (coordinates) where the word occurs in the query sequence. If the word does not occur, the number at the entry is zero. Then, for each word in the searched sequences, WordSearch just looks up the word in the dictionary to find out if it occurs in the query sequence.

Score

If the word from a search set sequence does occur in the query sequence, WordSearch adds the length of the word to the score for the diagonal on which the word occurs. If a word match overlaps another one, only the new symbols are added to the score for the diagonal. For instance, two adjacent word matches of length six would contribute a total of seven to the score for their diagonal.
**Alphabet**

The parameter *alphabet* that appears in the output is the number of symbols that could make up each word. For protein sequences, the alphabet is the number of sequence symbols that were actually used in the query sequence. The alphabet should be four for nucleic acids. Notice that nucleic acid ambiguity codes are not supported by this alphabet and that they confound word comparison! *Any word in any search set sequence that contains characters that are not part of the comparison "alphabet" is ignored. U and T are equivalent in nucleic acid sequences however, so DNA patterns may be found in RNA sequences. Uppercase and lowercase sequence symbols are equivalent in all comparisons.*

**The Histogram: Score**

An array of counters, one for the score on each diagonal, is maintained. Each time a word is found in both the horizontal and vertical sequences, the counter for the diagonal on which it was found is incremented by the number of symbols in the word. After each sequence is searched with the dictionary from the query sequence, the result is an array of numbers that tells how many symbols occur within matching words along each diagonal of the comparison. This array of diagonal counters is referred to as the histogram.

**The Histogram is Integrated**

To make the search more tolerant of short length differences (gaps) between the query and the sequences in the database to which it is similar, WordSearch combines the scores of a user-defined number of adjacent diagonals and puts the combined score (rounded up) at the center of this "window of integration." Wilbur and Lipman call this region of adjacent diagonals a *window-space.*

**Finding the N-Best Diagonals: Structures**

After integration, the histogram is searched for a position in which there is a score above the average. A *structure* is defined as a region of diagonal scores in the integrated histogram from the first above-average score to the last; that is, to where the scores fall back to the average again. If the peak score for a structure is better than the worst score in the list of the N-best diagonals observed so far, then the structure is put in the list and the existing worst observation in the list is discarded. The structure is recorded by recording the file and entry being searched, the coordinate of the diagonal at the center of the peak region rounded up, the peak score (after integration), the width of the structure, and whether the top or bottom strand of the query sequence was being used for the comparison. When all of the files in the horizontal search set have been examined, the list of N-best structures is reported, as shown in the output file above.

**GRAPHICS**

*GCG must be configured for graphics before you run any program with graphics output!* If the `% setplot` command is available in your installation, this is the easiest way to establish your graphics configuration, but you can also use commands like `% postscript` that correspond to the graphics languages GCG supports. See Chapter 5,

**<CTRL>C**

If you need to stop this program, use <Ctrl>C to reset your terminal and session as gracefully as possible. Searches and comparisons write out the results from the part of the search that is complete when you use <Ctrl>C. The graphics device should stop plotting the current page and start plotting the next page. If the current page is the last page, plotters should put the pen away and graphic terminals should return to interactive mode. The function of WordSearch depends on whether your input sequence(s) are protein or nucleotide. Programs determine the type of a sequence by the presence of either Type: N or Type: P on the last line of the text heading just above the sequence. If your sequence(s) are not the correct type, turn to Appendix VI for information on how to change or set the type of a sequence.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % wordsearch [-INfile1=]ggammacod.seq -Default

Prompted Parameters:

- **-BEGin=1 -END=444** sets the range of interest
- **[-INfile2=]GenBank:*** specifies the search set
- **-WORdsize=6 or -MASk=+++++++** sets the word size or mask pattern
- **-LIStsize=50** sets the size of the output list
- **-INTegrate=3** sets the width of integration window
- **[-OUTfile=]ggammacod.word** names the output file

Local Data Files:

- **[-SIMplify=]simplify.txt** assigns an optional simplification table

Optional Parameters:

- **-SIMplify=[filename]** simplifies sequences using the specified file
- **-SINce=6.90** limits search to sequences dated on or after June 1990

316 Program Manual
-LOWscore=10  sets minimum score (from 1 to 100) for diagonal to be listed
-RESORt      sorts output list by name instead of score
-NOSHOWfiles suppresses documentation at the end of each line in the output
-PLOT        makes a plot of the score distribution
-BINsize=6   sets the resolution of the score distribution plot
-NOMONitor   suppresses the screen trace during the search
-NOSUMmary   suppresses the screen summary at the end of the search
-BATch       submits the program to run in the batch queue

All GCG graphics programs accept these and other switches. See the Using Graphics chapter of the USERS GUIDE for descriptions.

-FIGure[=filename] stores plot in a file for later input to FIGURE
-FONT=3       draws all text on the plot using font 3
-COLOR=1      draws entire plot with pen in stall 1
-SCAle=1.2    enlarges the plot by 20 percent (zoom in)
-XPAN=10.0    moves plot to the right 10 platen units (pan right)
-YPAN=10.0    moves plot up 10 platen units (pan up)
-PORtrait     rotates plot 90 degrees

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

If you use -SIMplify, WordSearch reads the local data file simplify.txt to find the symbol equivalences you want to use. You can specify a simplification table with another name using an expression like -SIMplify=mysimplify.txt. There is more on the subject of sequence simplification in the documentation for the Simplify program.

The simplify.txt file in the public data directory is only appropriate for simplifying protein sequences. You must create your own simplify.txt file to define equivalences for nucleic acid simplifications.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
-WORdsize=6

Sets the size of the word, or n-mer, used in the search. Matches between the query sequence and a sequence in the search set are identified by large numbers of identical words shared between the two sequences.

-MASk=++-----++

Specifies the word mask used in the search. With a word mask, you use a plus sign (+) to show those positions of the word where the sequence symbols must match and a minus sign (-) to show the positions where symbols may or may not match. The word size is implicitly defined by the size of the mask.

-LISTsize=50

Sets the number of top-scoring entries to save in the output list.

-INTegrate=3

Specifies the number of adjacent diagonals whose word scores are summed together. By summing the scores of adjacent diagonals, the search is tolerant of small gaps between the query sequence and the sequences being searched.

-SIMplify=filename

Simplifies the sequences before comparison according to a table of equivalences in the local data file called simplify.txt (see the LOCAL DATA FILES topic above). Many investigators feel that protein sequence pattern recognition for word searching is more sensitive if similar amino acids are treated as equivalent. You can name a file other than simplify.txt.

-SINce=6.1990

Limits the search to sequences that have been entered into the database or modified since June 1990. As this is being written, only the EMBL, GenBank, and SWISS-PROT databases support this parameter.

-LOWscore=10

Sets a threshold score, from 1 to 100, at or below which a diagonal cannot be considered.

-RESORt

Causes WordSearch to sort the list of diagonals a second time by sequence name, so that all of the diagonals from the same sequence appear together in the output list. Usually, the diagonal list from WordSearch is shown with the most significant (highest score) diagonal first and diagonals with successively lower
scores following. While this is the obvious order, it slows down the Segments display program that has to read each sequence in the list to make the display.

-NOSHOWfiles

Suppresses the documentation at the end of each line in the output list.

-PLOT

Makes a plot showing the distribution (frequency) of diagonal scores. The score distribution plot is useful for determining if a score in the output list is significant. You must have a plotter or graphic screen to use this parameter. There is a whole paragraph above about the score distribution plot.

-BINsize=6

Sets the resolution of the score distribution plot (how many scores will be reported in each bin of the histogram).

-MONitor

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

-SUMmary

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.

-BATCH

Submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

The parameters below apply to all GCG graphics programs. These and many others are described in detail in Chapter 5, Using Graphics of the User's Guide.
-FIGure=programname.figure

Writes the plot as a text file of plotting instructions suitable for input to the
Figure program instead of sending it to the device specified in your graphics
configuration.

-FONT=3

Draws all text characters on the plot using Font 3 (see Appendix I).

-COLOR=1

Draws the entire plot with the pen in stall 1.

The parameters below let you expand or reduce the plot (zoom), move it in either
direction (pan), or rotate it 90 degrees (rotate).

-SCALE=1.2

Expands the plot by 20 percent by resetting the scaling factor (normally 1.0) to
1.2 (zoom in). You can expand the axes independently with -XSCALE and -YSCALE. Numbers less than 1.0 contract the plot (zoom out).

-XPAN=30.0

Moves the plot to the right by 30 platen units (pan right).

-YPAN=30.0

Moves the plot up by 30 platen units (pan up).

-PORTrait

Rotates the plot 90 degrees. Usually, plots are displayed with the horizontal axis
longer than the vertical (landscape). Note that plots are reduced or enlarged,
depending on the platen size, to fill the page.

Printed: February 10, 2006 19:41
**XNU**

**FUNCTION**

Xnu replaces statistically significant tandem repeats in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

**DESCRIPTION**

The Karlin-Altschul statistics that underlie BLAST assume that the probability of finding a residue at any particular position in a sequence is simply proportional to its composition. Tandem repeats may violate this assumption. Such regions occur frequently in proteins. Query sequences containing such repeats may give significant similarity scores when compared to unrelated proteins containing similar repeats.

Xnu is a program described by Claverie and States in Computers and Chemistry, 17; 191-201 (1993) that is used to mask off tandem repeats in protein sequences. The output sequence is just like the input sequence except that if tandem repeats are found, the amino acid characters comprising such repeats are replaced by X's. Regions containing X's are ignored in a BLAST search.

**EXAMPLE**

Here is a session using Xnu to mask off the repeats in a human major prion protein precursor.

```%
 xnu

 XNU of what input sequence(s) ? PIR:Ujhu

 Begin (* 1 *) ?
 End (* 253 *) ?

 What should I call the output file (* ujhu.xnu *) ?

 PIR1:UJHU  Len:  253
%
```

**OUTPUT**

Each output file contains the input sequence with the amino acid characters that comprise statistically-significant tandem repeats changed into X's. Here is the output file from the session above.

```
!!AA_SEQUENCE 1.0
 XNU of: ujhu  check: 8781  from: 1 to: 253
```
INPUT FILES

You can specify either a single protein sequence or multiple protein sequences as input to Xnu. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf{*}; or by using a sequence specification with an asterisk (*) wildcard, for example GenBank:*.

If Xnu rejects your protein sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

Seg replaces low complexity regions in protein sequences with X characters. If a resulting protein sequence is used as a query for a BLAST search, the regions with X characters are ignored.

Repeat finds direct repeats in sequences. You must set the size, stringency, and range within which the repeat must occur; all the repeats of that size or greater are displayed as short alignments.

RESTRICTIONS

Xnu does not recognize repeats if the width is set much longer than the length of either the repeat or the sequence. Its behavior is not characterized for sequence symbols that
are not among the standard unambiguous IUPAC-IUB amino acid single-letter symbols (ACDEFGHIKLMNPQRSTVWY).

CONSIDERATIONS

Repeat sequences are scored as segment pairs (short gapless alignments). All of the residues in both of the segments of a significant pair are replaced with X's.

Xnu uses a PAM120 scoring matrix for scoring similarities. You cannot select any other scoring matrix. By default, repeats less than five residues long are eliminated unless you set a different maximum repeat length with -\texttt{WID}\texttt{th}.

Many single tandem repeats will not be masked, while triplet repeats of the same kind will be. STUSTU would not be found where STUSTUSTU will be.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -\texttt{CHEck} to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % xnu [-\texttt{IN}file=]pir:ujhu -Default

Prompted Parameters: (for single sequences)

-\texttt{BEG}in=1  -\texttt{END}=253 \hspace{1cm} \text{sets the range of interest}
[-\texttt{OUT}file=]ujhu.xnu \hspace{1cm} \text{names the output file}

Local Data Files: \hspace{1cm} None

Optional Parameters:

-\texttt{BEG}in=1  -\texttt{END}=100 \hspace{1cm} \text{sets the range of interest (for multiple sequences)}
-\texttt{PROb}ability=.01 \hspace{1cm} \text{sets the expectation level for a repeat}
-\texttt{WID}th=4 \hspace{1cm} \text{sets the minimum size of a repeat}
-\texttt{EXTension}=.xnu \hspace{1cm} \text{sets the default output file name extension}
-\texttt{LIST}file=[xnu.list] \hspace{1cm} \text{writes a list file of output sequence names}
-\texttt{NOMONitor} \hspace{1cm} \text{suppresses screen monitor of input sequence names}
-\texttt{NOSUM}mary \hspace{1cm} \text{suppresses the screen summary}

LOCAL DATA FILES

None.
PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

**-BEGIN=1**

Sets the beginning position for all input sequences. When the beginning position is set from the command line, Xnu ignores beginning positions specified for individual sequences in a list file.

**-END=100**

Sets the ending position for all input sequences. When the ending position is set from the command line, Xnu ignores ending positions specified for sequences in a list file.

**-PRObability=.01**

For a repeat to be recognized, it must score high enough so that you would not expect to see a higher score more than once in 100 searches of random sequences of average length and composition. Use this parameter to change that expectation cutoff. Setting this cutoff lower than its default of 0.01 makes the search more stringent and the number of repeats masked off fewer. The minimum and maximum values of this parameter are 0.0001 and 0.1.

**-WIDTH=4**

Sets the maximum size of a repeat. If a repeat were of length five, even if it were significant, it would not be found if this parameter were set to four. When this value is set to zero, Xnu will search for repeats of any size. Very short repeats may not score above the default probability cutoff (see **-PRObability** above). The maximum value of this parameter is 100. The larger it is, the longer the search will take.

**-EXTension=.xnu**

This program normally creates output file names by using the original input file name for the base name and the program name for the name extension. Use this parameter to specify some other file name extension.

**-LISTfile=xnu.list**

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then Xnu makes one up using xnu for the file name and .list for the file name extension.
-MONitor

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

-SUMmary

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.

Printed: February 10, 2006 19:41
PROGRAMS NOT SUPPORTED BY GCG 11.0

CHOPUP

FUNCTION

ChopUp converts a non-GCG sequence file containing lines longer than 511 characters and as long as 32,000 characters into a new file containing lines no longer than 50 characters. The new file can be read by Reformat to create a GCG-format sequence file.

DESCRIPTION

This program converts a file with long lines to a file with lines no longer than 50 characters, splitting long lines into two or more lines. When possible, lines are split at a blank space.

EXAMPLE

Here is a session using ChopUp to convert the file gamma.seq to a file with lines no longer than 50 characters:

% chopup

CHOPUP what file ? gamma.seq

What should I call the chopped file (* gamma.dat *) ?

All of "gamma.seq" chopped up without error!

%  

OUTPUT

Here is part of the output file:

!!NA_SEQUENCE 1.0
Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies, Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.

gamma.seq Length: 11375 June 27, 1994 10:09
Type: N Check: 6474 ..

1  GGATCCTAGA TATTCCTTAG TCTGAGGAGG
   AGCAATTAAG ATTCACTTGT

51  TTAGAGGCTG GGAGTGGTGG CTCACGCCTG
   TAATCCCAGA ATTTTGGGAG
   \\
   \\

326 Program Manual
RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

No line in the input file may be longer than 32,000 characters.

CONSIDERATIONS

Sequence files prepared with a text editor or brought to your computer from other sources may contain lines longer than 511 characters. These sequence files must be converted by ChopUp before being read by Reformat.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use \texttt{-CHECK} to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: \texttt{% chopup [-INfile=]gamma.seq -Default}

Prompted Parameters: None

Local Data Files: None

Optional Parameters:
\texttt{-LENghth=50} \hspace{1cm} sets the maximum line length in the output file
\texttt{-NOMONitor} \hspace{1cm} suppresses the screen monitor

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

\texttt{-LENghth=50} \hspace{1cm} Sets the maximum line length in the output file. \texttt{-LENghth} can have any value from 2 to 132.
This program normally monitors its progress on your screen. However, when you use `-Default` to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.
COMPRESSTEXT

FUNCTION

CompressText removes any or all of the following from files: A) trailing space; B) blank lines; C) extra space between words; D) all space; or E) leading space.

DESCRIPTION

CompressText reduces the white space in text files. It can remove trailing white space or leading white space, delete blank lines, and compress words together so that there is never more than one space between words. It can also remove all non-printing characters from your file. You can select any or all of these possibilities!

EXAMPLE

Here is a session using CompressText to remove trailing white space and delete the blank lines from a GenBank distribution file:

% compress
text

COMPRESSTEXT what file ? genbank.form

How should "genbank.form" be compressed

A) remove trailing white space from each line
B) delete the blank lines
C) compress all word separations to a single space
D) remove ALL space from the file
E) remove leading white space from each line

Choose the desired compressions (* AB *):

What should I call the output file (* genbank.form *) genbank.txt

%

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

Unknown.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you
must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% compressstext [-INfile=]genbank.form -Default`

Prompted Parameters:

- `MENu=ab` selects the type of compressions you want:
  (choose one or more)
  a=trailing white space from each line
  b=blank lines
  c=all word separations to one space
  d=all space from each line
  e=leading white space from each line

[-OUTfile=]genbank.form names the output file

Local Data Files: None

Optional Parameters: None

**LOCAL DATA FILES**

None.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- `MENu=ab`

  Specifies the type of compression and suppresses the option menu.

Printed: January 9, 2002 13:45 (1162)
DETAB

FUNCTION

Detab replaces the tab characters in one or more files with spaces. The files can be written out in card-image format with records of fixed length.

DESCRIPTION

This program replaces all the tab characters in a text file with equivalent blank characters. It assumes that tabs are separated by eight spaces and replaces each tab with one to eight space characters, depending on the position of the tab in the text line.

EXAMPLE

Here is a session using Detab to replace each tab character in the file cmpvals.for with eight spaces.

```
% detab
DETAB what input file(s) ? cmpvals.for

What should I call the output file (* cmpvals.for *) ? temp.txt

  temp.txt   59 lines

%
```

OUTPUT

The contents of temp.txt does not contain any tab characters once Detab is finished. This file could be sent to a printer or text processor that does not support tab characters.

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

Detab is designed to replace tabs set after every eighth column.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHE`ck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.
Minimal Syntax: % detab [-INfile=]cmpvals.for -Default

Prompted Parameters:

[-OUTfile=]cmpvals.for names the output file
(only when the input file name is not ambiguous)

Local Data Files: None

Optional Parameters:

-EXTension=.txt specifies a file name extension for the output files
-DIRectory=DirName specifies a directory for the output files
-CARd makes "card image" files with fixed records of length 80
-NOTRUncate wraps lines longer than 80 characters in card images
-NAMe shows the input file name in the first three lines

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-EXTension=.txt

Sets the file name extension of the output file(s).

-DIRectory=DirName

Writes the output files into a directory other than your current working directory.

-CARd

Makes a card image file by writing all lines in the output file to exactly 80 characters long. This is done by appending spaces to lines that are shorter than 80 characters and truncating lines that are longer. A warning is printed if any lines are truncated. You can keep Detab from truncating lines by placing -NOTRUncate on the command line. The output file has fixed length records.
-NOTRuncate

Usually, Detab truncates lines that are longer than 80 characters when -CARd is on the command line. This parameter sets Detab (run with -CARd) to wrap input lines longer than 80 characters into as many output character records as are needed to save all of the information from the original line.

-NAMe

Sets Detab run with -CARd to write three lines at the top of each output file with the complete original path name of the input file and the date.
EXTRACTPEPTIDE

FUNCTION

ExtractPeptide writes a peptide sequence from one or more of the translation frames displayed in the output from Map. Translate supersedes ExtractPeptide for most applications.

DESCRIPTION

You can see the protein sequence from any of the six possible DNA reading frame translations by running the Map program. You may wish to analyze these protein sequences with other Wisconsin Package programs. ExtractPeptide takes these sequences from the output file of Map and writes them into a new sequence file. If more than one frame of translation is chosen, then the sequences from each frame are concatenated. If any translations from the reverse strand are chosen, then the protein sequences are reversed to maintain the standard amino=>carboxyl orientation.

Translate is a more general way to translate DNA sequences when the start and stop positions are already known.

EXAMPLE

Here is a session using ExtractPeptide to extract the the three forward reading frames of the sequence gamma.seq between bases 2161 and 2600 from the the file gamma.map that was created in the example session for the Map program.

```
% extractpeptide

EXTRACTPEPTIDE from what MAP output file ? gamma.map

What translation frame(s) (A - F) (* ABC *) ?

What should I call the output file (* gamma.pep *) ? test.pep
```

OUTPUT

Here is part of the output file:

```
!!AA_SEQUENCE 1.0
EXTRACTPEPTIDE of frames: ABC from: gamma.map

(Linear) MAP of: gamma.seq  check: 6474  from: 2161  to: 2600
Human fetal beta globins G and A gamma
from Shen, Slightom and Smithies,  Cell 26; 191-203.
Analyzed by Smithies et al. Cell 26; 345-353.
With 216 enzymes: *
```
ExtractPeptide accepts the output file created by the Map program as input. (See the OUTPUT topic in the Program Manual entry for the Map program.)

RELATED PROGRAMS

Translate is a more general way to translate DNA sequences in which the boundaries of the coding regions are known. PepData can translate nucleic acid sequences in all six reading frames.

CONSIDERATIONS

ExtractPeptide writes the output sequence in the standard, single-letter amino acid codes described in Appendix III. If any reading frame is on the opposite strand, the protein sequence from the Map output file is reversed to maintain the standard amino=>carboxyl orientation. You could put the sequence back the way it was in the Map output file with the Reverse program.

COMMAND-LINE SUMMARY

Complete command-line control is not available for this program.

LOCAL DATA FILES

None.

PARAMETER REFERENCE

None.
GETSEQ

FUNCTION

GetSeq reads a sequence from a computer that is acting as a terminal and writes it into a new sequence file in GCG format on the computer running the Wisconsin Package.

DESCRIPTION

GetSeq offers you a quick way to create a GCG sequence file from the keyboard without having to use SeqEd, Reformat, or a text editor. In addition, if you use a microcomputer that can act as a terminal, you can use GetSeq to move sequence files in text format from your microcomputer into your UNIX directory in GCG format. The microcomputer's terminal emulation software must be able to send ASCII text files to the Wisconsin Package (TM) program GetSeq.

When you run GetSeq, it opens a file that you name and then waits for input. You can either type the sequence characters in from the keyboard, or use your microcomputer's terminal emulation software to send a text file that contains sequence data.

GetSeq accepts all of the supported GCG sequence characters you send until it sees a <Ctrl>D. Then it writes those characters out as a sequence file that can be used by any GCG program. Since GetSeq only accepts sequence characters (see Appendix III), the sequence may be in any format; line feeds, carriage returns, spaces, and numbering are ignored. If you are sending a sequence text file to GetSeq, make sure it doesn't contain comments. Any letters in a comment that are valid sequence symbols will be treated as sequence data.

EXAMPLE

Here is a session using GetSeq; the sequence GATTCCGATTG was sent followed by <Ctrl>D:

```
% getseq
GETSEQ into what sequence file? temp.seq

Now start the sequence transfer . . .
End the transfer with a <Ctrl>D

GATTCCGATTG ^D

Bases transferred: 11
```

%
OUTPUT

Here is the output file:

```
!!NA_SEQUENCE 1.0

      Length: 11  September 29, 1998 18:05  Type: N  Check: 4920

1 GATTCCGATT G
```

RELATED PROGRAMS

The UNIX `% cat > temp.txt` command creates a regular text file and puts all of the characters you send from the terminal into it until you send a `<Ctrl>D`. (Typing `<Return>` starts a new record in the file -- it does not terminate text entry.)

Reformat is a utility for changing a text file that contains a sequence into a GCG sequence file.

SeqEd lets you edit a sequence and the documentary heading once you have the sequence as a file in GCG format.

RESTRICTIONS

Your sequence must use the IUB-IUPAC character set for protein or nucleic acid sequences (see Appendix III). Change the sequence with a text editor on the micro if there are any characters in it that are unacceptable to the Wisconsin Package.

The sequence must not be longer than 350,000 characters to be acceptable to Wisconsin Package software.

If there is non-sequence data in the file, such as heading or documentary information, you must have a way to send only the sequence characters. You may need to edit the file on the microcomputer to remove non-sequence sections from the file.

SEQUENCE TYPE

When GetSeq writes GCG sequence files, it assigns the sequence type based on the composition of the sequence characters. This method is not fool-proof, so you may need to change the sequence type of the newly created file. Look on the last line of the text heading just above the sequence itself for `Type: N` or `Type: P`. If the type is incorrect, see Appendix VI for information on how to change or set the type of a sequence.
COMMAND-LINE SUMMARY

Complete command-line control is not available for this program.

LOCAL DATA FILES

None.

PARAMETER REFERENCE

None.

Printed: January 9, 2002 13:45 (1162)
LINEUP

FUNCTION

LineUp is a screen editor for editing multiple sequence alignments. You can edit up to 30 sequences simultaneously. New sequences can be typed in by hand or added from existing sequence files. A consensus sequence identifies places where the sequences are in conflict.

DESCRIPTION

LineUp lets you edit several overlapping or aligned sequences simultaneously. LineUp allows you to edit sequences in the context of an alignment to help you see the effect of your changes on the alignment.

As in SeqEd, you can move the cursor with the arrow keys and insert or delete symbols or gaps in the sequences. In LineUp, the cursor can travel from one sequence to another. You can add new sequences by hand or from existing sequence files, and you can move sequences from one position to another.

LineUp provides a surface on which you can arrange and edit many sequences. This surface resembles a piece of graph paper with 31 rows and as many columns as you need. The screen acts as a window behind which the LineUp surface is scrolled.

Sequences can be placed anywhere on the surface as long as two sequences in the same row do not collide. Several sequences can be placed on the same row.

Sequences placed on the LineUp surface become part of a sequence group. A new sequence group is formed by running LineUp with a new sequence group name. Sequences already stored in files can be placed anywhere on the surface with the Get command. New sequences (not already in sequence files) can be typed in anywhere on the surface.

When you end a session with LineUp, it writes out each sequence in a file and then writes a list file with the name and position of each sequence in the group. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for more information about list files.) When you edit the group again, the sequences reappear on the LineUp surface where you left them.

You can have a consensus sequence display the dominant character at each column where sequences overlap. The consensus uses uppercase where overlapping sequences are in agreement, lowercase to show disagreement, and periods to show where there is no consensus at all.

EXAMPLE

Here is a session using LineUp to edit the same sequence group displayed in the example session for the Pretty program. First use Fetch to copy the files *.frg and picorna.fil to your default directory.
% lineup picorna

R2       Column: 1  Row: 5     No AutoCons  FOSN: PICORNA  Protein

15: ......................................................................ttttgesad.pvtttve....n..yggdt.q....vq
14: ......................................................................ttatgesad.pvtttve....n..ygget.q....vq
13: ......................................................................ttsagesad.pvtttve....n..ygget.q....iq
12: gvenae.kgvtentna.tadfvaqpvylpe.nqt.....kv.affynrs...spi.gaftvks.....
11: gigqmlesmi.dntvretgaatsrdalpnteasgpethskeipaltavetganplvpsdtvqtrhvqv
10: gigqmlesmi.dntvretgaatsrdalpnteasgpahskeipaltavetganplvpsdtvqtrhvqv
  9: gigdmiegav.egitknalvppptstnslpghkpsgpahskeipaltavetganplvpsdtvqtrhvqv
  8: giediiseva.qgal..tlslpkqqdslpdtkasgpahskepaltavetganplapsdtvqtrhvqv
  7: ...gpvedai.......t...aaigr..vadtvgtgpntseaipaltaetgthsqvvpdmtqtrhvkn
  6: gigdeleevivekt.kqtv.asi........ssgpkhtqkvilvnapetgatmpvplpsdiierttym
  5: ...npvenyidevlnevli........vpininssnpttsnsapaldiaetghtsqvqpedvietryvqt

  0         10        20        30        40        50        60

"picorna.fil" successfully loaded.

RELATED PROGRAMS

SeqLab is the graphical user interface for the Wisconsin Package. The SeqLab Editor is a powerful and versatile tool for creating, editing and displaying multiple sequence alignments.

SeqEd is an interactive editor for creating and editing individual sequence files.

PileUp creates a multiple sequence alignment from a group of sequences.

Gap uses the algorithm of Needleman and Wunsch to find the alignment of two complete sequences that maximizes the number of matches and minimizes the number of gaps. If you run Gap with the command-line parameters for sequence output, it writes sequence files with the sequences expanded by the addition of gaps.
Pretty displays multiple sequence alignments and calculates a consensus sequence. It does not create the alignment; it simply displays it. Unlike LineUp, Pretty requires that all your sequences start in the same column.

PlotSimilarity plots the running average of the similarity among the sequences in a multiple sequence alignment.

ProfileMake makes a profile from the multiple sequence alignment. ProfileSearch uses the profile to search a database for new sequences with similarity to the group of aligned sequences. ProfileSegments displays optimal alignments between each sequence in the ProfileSearch output list and the group of aligned sequences (represented by the profile consensus). ProfileGap makes optimal alignments between a sequence and a group of aligned sequences represented as a profile.

STARTING OUT

To create a new sequence group, use the LineUp command with a new group name such as myseqs. If you use % lineup myseqs, LineUp looks in your current directory for the file myseqs.fil. If you use the command % lineup -MSF myseqs, then LineUp looks for the file myseqs.msf. If it doesn't find a file with this name, LineUp starts a new group with one sequence, the consensus, having the same name as the group. (If you do not want to have a consensus sequence in your group, run LineUp with the command-line parameter -NOCONSENSUS.) To construct the group, use the GET command to add sequences from existing sequence files or use the NEW command so LineUp lets you type in a new sequence. LineUp prompts you for a unique name of up to ten characters for each new sequence.

EDITING EXISTING GROUPS

You can start LineUp with the name of an existing group. If you have a file of sequence names called myseqs.fil, which was created in a previous session with LineUp, use % lineup myseqs. If you have a multiple sequence format (MSF) file called myseqs.msf, which was created in a previous session with LineUp, use % lineup -MSF myseqs. You may specify a file name extension if the default extension LineUp adds is not appropriate. You also can use any single or multiple sequence specification as input to LineUp. Multiple sequences can be specified as a list file or as a sequence specification using a wildcard. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for help in specifying sequences.) LineUp loads the sequences into the multiple sequence editor and starts with its window at the left end of the group. You can add more sequences, modify existing ones, delete sequences, rename sequences, and move any sequence to a new position.

SCREEN MODE

In Screen Mode, commands are typically single keystrokes. Except for the search command, Screen Mode commands do not require a <Return>. 
**Entering Sequence Characters**

In Screen Mode, the cursor shows your position in one of the sequences in the group. You can insert any valid GCG sequence symbol (see Appendix III) into the sequence by typing the symbol. It is inserted at the cursor.

**Deleting Sequence Characters**

The <Delete> key and <Ctrl>H delete the symbols to the left of the cursor, one by one. The remainder of the sequence slides over to fill the gap.

**Moving the Cursor Horizontally**

To move the cursor to the right one symbol, use the <Right-arrow> key; to move to the left, use the <Left-arrow> key. Moving the cursor past the end of the sequence moves it to the next sequence on the row.

You can type a number followed by a <Return> and the cursor moves to that position in the current row. If you specify a position that is not occupied by a character, the cursor moves to the nearest occupied position.

The <Left-arrow> and <Right-arrow> keys can be preceded by a number, telling how many symbols to move to the left or right. For example, 10<Right-arrow> moves ten symbols to the right.

You can use the angle brackets to skip 50 characters to the left or right. If you precede the angle bracket by a number, the cursor skips that many characters and continues to do so until you change the number.

**Moving the Cursor Vertically**

The <Up-arrow> and <Down-arrow> keys move the cursor up or down to the next row.

In contrast with the horizontal arrows, if you precede either the <Up-arrow> or the <Down-arrow> with a number, the cursor moves to the row with that number. For example, 10<Up-arrow> moves to row ten, not up ten rows.

**Moving a Whole Sequence**

When the cursor is at the left end of a sequence, you can move the sequence to the right with the space bar and to the left with the <Delete> key or <Ctrl>H. If you want to move a sequence to another row or to the other side of a sequence on the same row, you must use the **Move** command in Command Mode.

**Finding Patterns**

To search for a pattern, type a / (slash) in Screen Mode. You are prompted for the sequence pattern you wish to find. LineUp only searches the current sequence. You can repeat the last search by simply using /<Return>.
The command-line parameter -NUCleotide or the NUCleotide command in Command Mode makes LineUp treat all nucleic acid sequences as circular and finds your pattern even if it wraps from the end of the sequence into the beginning. LineUp uses the same rules for pattern definition and recognition as the FindPatterns, MapPlot, Map, and MapSort programs.

The command-line parameter -PROtein or the PROtein command in Command Mode makes LineUp searches linear and disables the nucleic acid ambiguity meanings of the GCG sequence symbols; they also change the way the consensus sequence is defined (see the topic THE CONSENSUS SEQUENCE below).

Even if LineUp thinks your sequence is a nucleotide sequence, you can request a perfect match by typing an = right after the /. So if you type /=RTC only RTC is matched, whether you have a protein or a nucleotide sequence.

Leaving Screen Mode

Use <Ctrl>D to leave Screen Mode and enter Command Mode.

Screen Mode Summary

Here is the summary of Screen Mode commands you would see in the on-line help:

Screen Mode

[n] is an optional numeric parameter.

G, A, T, C .... - inserts a sequence character
<Delete> - deletes a sequence character, "drags" a sequence to the left if cursor is at its start
<Ctrl>H - deletes a sequence character, "drags" a sequence to the left if cursor is at its start
<Space bar> - "pushes" a sequence to the right if cursor is at its start
/TAA CG<Return> - finds the next occurrence of "TAA CG", last pattern is the default when none is specified
[n]<Right-arrow> - move ahead [n characters]
[n]<Left-arrow> - move back [n characters]
[n]<Up-arrow> - move up to next sequence [or to row specified]
[n]<Down-arrow> - move down to next sequence [or to row specified]
[n]<Return> - move to column n
1<Return> - move to start of current sequence
<Ctrl>E - move to end of current sequence
<Ctrl>R - redraw the screen
<Ctrl>D - enter Command Mode
<Ctrl>I - push over all seqs starting past current column
<Ctrl>P - pull over all seqs starting past current column
[n]< - move 50 [or n] positions to left
[n]> - move 50 [or n] positions to right

COMMAND MODE

In Command Mode, you enter commands followed by a <Return>.

Entering Command Mode

Use <Ctrl>D to leave Screen Mode and enter Command Mode.

Editing LineUp Commands

LineUp command editing is modeled on OpenVMS DCL command line editing. The <Left-arrow> and <Right-arrow> keys let you move your cursor around in a command that you have typed so you can insert or delete characters at any position. <Ctrl>E moves the cursor to the end of the line. <Ctrl>U deletes all the characters from the current cursor position to the start of the line.

Editing Previous LineUp Commands

LineUp lets you modify and execute previous commands. The <Up-arrow> key displays previous commands.

Returning to the Screen Mode

If you simply press <Return>, LineUp returns to Screen Mode described above. If you have -SINGLEcommand on the command line or in your command-line initializing file, LineUp returns to Screen Mode immediately after executing each command.

Commands May Be Truncated

Only the capitalized portion of the commands described in the documentation below must be typed.

Parameters Are Used With Some Commands

Some commands may be preceded or followed by optional numeric parameters or a file name. The square brackets ([ and ]) in the documentation below show optional command arguments: s and f refer to starting and finishing rows or offsets on the surface; x and y refer to offset and row coordinates. When an optional parameter is omitted, some commands prompt you for the value. Others commands make default assumptions that are explained in each command description.

Missing Position Parameters: Spacewalk

Several commands need position parameters to know where to put a sequence. If these parameters are omitted, LineUp enters a mode, called Spacewalk, that allows you to move the cursor anywhere on the surface to select a position for the new sequence. In
Spacewalk Mode, the arrow keys and <Return> can be preceded with numbers as in Screen Mode. <Ctrl>D cancels the command when you are in Spacewalk Mode. If you prefer to provide numeric coordinates rather than position the cursor, you can eliminate Spacewalk Mode by using the command-line parameter -NOSPacewalk or the command NOSPacewalk. You are then prompted for numeric coordinates if you omit them from commands.

Other Missing Parameters

If a required name is omitted or illegal, LineUp prompts you for a name. If you respond with a blank name, LineUp cancels the command.

Working Directory

You must have write privileges in your current working directory to use LineUp; otherwise, LineUp will not accept any name you try to give a sequence.

Default Values for LineUp Prompts

Often when it prompts for sequence or file names, LineUp presents a default value in a manner different from other GCG programs; when the prompt appears, it looks like you have already typed in the default value. You can just press <Return> if you want to accept the default. If you want to change it, proceed as with command-line editing. Make small changes by using the arrow keys to move within the offered response or delete the response with <Ctrl>U and type your desired response.

Command Mode Summary

Here is the summary of Command Mode commands you would see in the on-line help:

Command Mode

x and y represent numbers for column and row.
Only the capitalized part of the command is necessary.

[x,y] Get [filename] - add sequence [at position x,y] [from filename]
[x,y] New [seqname] - add empty sequence [at position x,y] [named seqname]
[x,y] MOve [seqname] - move current or specified sequence [to x,y]
REMove [seqname] - delete current or specified sequence entirely
REName [old] [new] - change sequence name (changing consensus name changes the group)
REDraw - redraw the screen
HEAding [seqname] - edit documentary heading of current or specified sequence
screen - enter screen mode (pressing <Return> is sufficient)
NUCleotide - use nucleotide ambiguity codes in find and consensus
PROtein - do not use nucleotide ambiguity codes
SPacewalk - use spacewalk to position sequences
NOSPacewalk - DO NOT use spacewalk to position sequences
FOSN - use list file format when writing
MSF - use multiple sequence format files when writing
[n] SLide - add n to all sequence columns
[s,f] ROWMove [n] - move a set of rows (s to f) up or down [n rows]
[s,f] PRint [filename] - write the sequence group to a Pretty format file
SUMmary [filename] - write the sequence names and positions in a file or on the terminal screen
GOTO [seqname] - put cursor on start of named sequence
[s,f] CONSensus - calculate consensus [from s to f]
AUtoconsensus - automatically calculate consensus (slow)
NOAUtoconsensus - turn off automatic consensus
FLip - reverse complement the current group
ZIP [filename] - align and gap a sequence to the current group
Write [filename] - write the current sequence group to a file
EXIT [filename] - write the current group to a file and stop
Quit - quit the editor without writing out the group

**COMMAND DESCRIPTIONS**

`: [x,y] Get [filename]`

adds the sequence in the specified file to the group at column x in row y. The screen is erased and you are prompted to enter the range and strand. Unlike the Write and EXIT commands, Get does not assume any file extension. You must type the file name plus any extension it requires.

`: [x,y] New [SeqName]`

adds an empty sequence at column x in row y.

`: [x,y] MOVE [SeqName]`

moves the sequence to start at column x in row y. If the SeqName parameter is omitted, the sequence at the current cursor position is moved.

`: REMOVE [SeqName]`

deletes the entire sequence from the group. If the SeqName parameter is omitted, the sequence at the current cursor position is removed.
:R**E**N**a**me  [OldName]  [NewName]

changes the name of the sequence. If no names are provided in the command, the sequence at the current cursor position is renamed and you are prompted for the new name. If only one name is provided, it is assumed to be the old name and you are prompted for the new name.

:R**E**D**r**aw

redraws your terminal screen. This is useful if noise in the line between your terminal and the computer has changed the screen in some unreasonable way or if a system message appears on your screen.

:[s]  **H**EADING  [SeqName]

enters Heading Mode to let you view and edit the documentary heading. You can modify any part of the heading. Heading Mode is terminated with <Ctrl>D. The optional SeqName parameter specifies which sequence heading you want to edit. If omitted, the sequence at the current cursor position is assumed. The optional numeric parameter specifies which line of the heading you want to start editing.

:screen

returns your session to Screen Mode. Just pressing <Return> also returns you to Screen Mode. (If you prefer to return to Screen Mode after every command is executed, use the command-line parameter **-SINGle** command.)

:**N**U**C**leotide

sets the sequence type for each sequence in the sequence group to be nucleotide. This enables the nucleic acid ambiguity meanings of the GCG sequence symbols in pattern searches (with /) and consensus definition (set the topic THE CONSENSUS SEQUENCE below). Also, LineUp treats nucleic acid sequences as circular when searching for a pattern. When the sequences are saved to files with either the Write or Exit command, they are written as nucleotide sequences if their sequence type is nucleotide.

:**P**RO**tein

sets the sequence type for each sequence in the sequence group to be protein. This forces LineUp to treat all sequences as linear in pattern searches and does not interpret any sequence characters as nucleotide ambiguity symbols in pattern searches and consensus definition (see the topic THE CONSENSUS SEQUENCE below). When the sequences are saved to files with either the Write or Exit command, they are written as protein sequences if their sequence type is protein.

:**S**P**a**cewalk

enters Spacewalk Mode, which allows you to move the cursor anywhere on the surface to select a position for a new sequence.
tells LineUp not to use Spacewalk Mode but to prompt for numerical surface coordinates.

:tells LineUp to use the list file format when loading or storing the sequence group.

:tells LineUp to use the multiple sequence format (MSF) file when loading or storing the sequence group.

:SLide

shifts all the sequence starting positions by \( n \). The coordinate ruler appears to slide under the sequences. \( n \) can be either a positive or negative number to shift the sequences to the right or left, respectively.

:ROWMove \([s, f]\) \([n]\)

moves a clump of rows up or down. The sequences on rows numbered from \( s \) to \( f \) are moved up \( n \) rows. Negative values of \( n \) move the sequences down \( n \) rows. This command can be used to open a row in the middle of the surface for another sequence. LineUp will not let you move sequences onto rows containing other sequences not simultaneously being moved.

:PRint \([s, f]\) \([filename]\)

writes a file of the formatted sequence group from position \( s \) to \( f \). The format resembles that of Pretty.

:SUMnary \([filename]\)

writes a list of the names and beginning positions of the sequences loaded into the LineUp editor. This list can go either to a file or to your screen (by typing Term for filename).

:GOto \([SeqName]\)

moves the cursor to the beginning of the named sequence.

:CONsensus

calculates the consensus sequence between positions \( s \) and \( f \). If the optional positions are omitted, the entire consensus is calculated. This command only works when LineUp
is not in the Auto Consensus state. (See the topic THE CONSENSUS SEQUENCE below for further details.)

:**AU**toconsensus

makes LineUp recalculate the consensus sequence each time there is a change in any of the other sequences. When LineUp is in the Auto Consensus state, the consensus is strictly a function of the other sequences and cannot be changed directly. When the sequence group is large, recomputing the consensus uses a lot of machine time and makes LineUp appear sluggish.

:**NOAU**toconsensus

turns off the Auto Consensus state. This allows you to change the consensus directly.

:**FL**ip

makes LineUp reverse and complement all of the nucleic acid sequences in the current group.

:**ZI**p [filename]

aligns a new sequence to the current consensus.

:**Write** [filename]

records the current surface configuration in a list file and saves the current version of each sequence in a file if the program is in FOSN mode (see the FILE NAME CONVENTIONS topic below). If the program is in MSF mode, a multiple sequence format (MSF) file is written. If the filename parameter is omitted, LineUp uses the sequence group name specified when the program is initially run. If you specify a file in another directory, all files are created there.

:**EX**it [filename]

works like the **Write** command but stops the session after writing out the sequences. The filename parameter behaves as in the **Write** command.

:**Quit**

terminates a session with LineUp without saving any changes you’ve made since the last time you used the **Write** command.

:**Help**

shows the commands available in Screen and Command Modes of LineUp.
HEADING MODE

Heading Mode allows you to view and edit the documentation that precedes the sequence in the sequence file. All headings are lost if you write the sequences into a multiple sequence format (MSF) file.

Entering Heading Mode

To enter Heading Mode, use the HEADING command.

Leaving Heading Mode

Use <Ctrl>D to return to Command Mode.

Moving the Cursor

You can move around using the arrow keys. Although the editing window is only twenty lines long, it scrolls over the heading vertically to let you see and modify any part. <Ctrl>E positions the cursor at the end of the current line.

Editing in Heading Mode

Like many text editors, typing inserts text at the cursor and the <Delete> key and <Ctrl>H delete characters to the left of the cursor. <Ctrl>U deletes everything from the current cursor position to the start of the line. Pressing <Return> creates a new line starting at the current position in the heading.

Unlike many text editors, LineUp asks you if you need more storage before letting you edit the Heading. You must enter the maximum number of lines that you expect to have to add. If you are in Heading Mode and find you do not have enough storage for your changes and additions, you can exit Heading Mode and enter it again, specifying some larger number of lines for increased storage.

PROTEIN AND NUCLEOTIDE SEQUENCE GROUPS

LineUp behaves differently depending on whether you are working with a protein or nucleotide sequence group.

If you are working with a nucleotide sequence group, then pattern searches (see "Finding Patterns" under the SCREEN MODE topic) and the consensus definition (see the topic THE CONSENSUS SEQUENCE) assume the IUB nucleotide ambiguity meanings for the GCG sequence symbols. Also, LineUp treats nucleic acid sequences as circular when searching for patterns. When the sequences are saved to files with either the Write or Exit command, they are written as nucleotide sequences if their sequence type is nucleotide.

If you are working with a protein sequence group, then LineUp treats all sequences as linear in pattern searches and does not interpret any sequence characters as nucleotide ambiguity symbols in pattern searches and the consensus definition. When the
sequences are saved to files with either the Write or Exit command, they are written as protein sequences if their sequence type is protein.

By default, if the first sequence entered into the LineUp editor screen is from an existing sequence file, then the type of that sequence determines the type for the entire group. If the first sequence in a sequence group is entered interactively from the keyboard, then LineUp sets the sequence type for the entire sequence group to be protein, by default. LineUp indicates the type of the sequence group (protein or nucleotide) in the upper-right corner of the editor screen.

You can specify the sequence type for the entire group from the command line with the -Protein and -Nucleotide command-line parameters. Once you are viewing the editor screen, you can change the sequence type for the entire sequence group with the Protein and Nucleotide command mode commands.

THE CONSENSUS SEQUENCE

An optional consensus sequence can be generated as a function of the rest of the sequences in a sequence group, or, like any other sequence, typed in by you.

By default, new sequence groups contain an empty consensus at row 0 unless LineUp is run with the -NoConsensus command-line parameter.

If the sequence group has no consensus, you can create one using the New command and giving the new sequence the same name as the sequence group. The Consensus command and the Autoconsensus commands now work on the row you have designated for the consensus with the New command.

When a consensus sequence is generated by LineUp, either by issuing the Consensus command or Autoconsensus command, each consensus character in the consensus sequence is replaced with a character that is a function of the other characters in its column. If all the characters in the column are the same letter and at least one of them is uppercase, the consensus character is the uppercase equivalent of that letter. If there is more than one letter in the column, but one occurs more frequently than any other, or if all letters are the same, but none are uppercase, then the consensus character is the lowercase of that letter. Otherwise, the consensus character is a dot (.)

The consensus definition also depends on whether LineUp is working with a nucleotide or protein sequence group. If a protein sequence group is loaded into the multiple sequence editor, the above description is complete. If a nucleotide sequence group is loaded into the editor, the ambiguity codes are ignored for the purpose of consensus definition. This treats all ambiguity codes as though they were the code 'N.' LineUp indicates the type of the sequence group (protein or nucleotide) in the upper-right corner of the editor screen.

The consensus sequence is distinguished by having the same name as the sequence group. If you rename the consensus sequence with the Rename command, the name of the sequence group changes as well. (You can rename the group even if you have no consensus sequence.) Conversely, if you specify a file name in a Write or Exit
command, this changes the name for the sequence group being saved and also changes the consensus sequence name.

The consensus sequence is unique in that, because it will likely extend to all columns determined by the other sequences, no other sequence may share its row. You can delete the consensus sequence from your group, and you can later create a new consensus sequence. However, an existing sequence cannot become the consensus sequence, either through the `RENAME` or `GET` commands.

**PULL-OVER AND PUSH-OVER**

If you use LineUp with sequence groups that have different sequences starting at several different columns, a problem will arise when you make deletions or insertions to whole columns. For example, suppose you are assembling a group of sequences with LineUp. When a new sequence is added, you may decide a previous sequence reading was incorrect. You may decide to delete a base from an old sequence near the left end of the assembly. The tail of that sequence slides left one column, destroying its alignment with any sequence starting to the right of the deletion site.

The general problem is that insertions or deletions cause shifts of register between sequences. Those sequences that overlap with the changed column appear to need adjustment. But those sequences that start down to the right do not appear misaligned.

LineUp warns you of potential alignment problems by producing a warning sign at the top of the screen. The sign says either `PUSH-OVER WARNING` or `PULL-OVER WARNING`, depending on whether there was an insertion or deletion. The warning is only displayed if there are other sequences that start to the right of your change. To make it easy for you to correct the alignment problem, LineUp provides you with screen mode commands to `PULLOVER` the sequences starting to the right of the cursor (<Ctrl>P for deletion), or to `PUSHOVER` (<Ctrl>I for insertion). You must make the decision whether the deletion or insertion requires adjustments and then ensure that the adjustments are correct. It is not recommended that you blindly trust the warning sign but that you let it remind you of the issue.

**MULTIPLE SEQUENCE FORMAT (MSF) FILES**

By default, LineUp reads and writes individual sequence files, grouped in a list file (FOSN format). Using the command-line parameter `--MSF` causes LineUp to expect a multiple sequence format (MSF) file when reading a sequence group, and to write out an MSF file when storing a sequence group (MSF format). For instance, the command `lineup --MSF hsp70` reads the sequences in the file hsp70.msf into the LineUp editor and names the sequence group hsp70. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for a complete description of MSF files.) When LineUp writes an MSF file, leading gap characters (.) are added to those sequences that do not start at the beginning of the alignment so that all sequences are left-justified in the output file.

The current sequence group format is indicated as either `FOSN:` or `MSF:` on the top line of the screen editor. You can toggle between these two formats using the `FOSN` and `MSF` commands in command mode.
EDITING INDIVIDUAL SEQUENCE FILES

There is no harm in using SeqEd to change a sequence file that has been written by LineUp. Provided the name is the same, the new version is accepted by LineUp. The only restriction on replacing members of a sequence group is that the new members must not overlap with other sequences on the same row. The information where the sequence starts is stored in the list file, so changing the sequence file can only change the length of the sequence. You can change where a sequence starts on the surface by modifying the Offset and Row columns of its entry in the list file using a text editor. If you overlap two sequences on the same row, LineUp refuses to load one of the overlapping sequences.

EMBEDDED COMMENTS

LineUp does not handle embedded comments. LineUp can read files containing embedded comments, but the comments are lost and will not appear in any file written by LineUp.

LINEUP AND PRETTY

If your sequences all start at the same column, you can use Pretty to generate a consensus sequence for a sequence group created by LineUp. Pretty uses a more sophisticated algorithm than LineUp to generate a consensus sequence and you have more control over the consensus calculation. However, Pretty can only handle sequence groups whose left ends are aligned.

Pretty and LineUp both know how to read the other's files of sequence names, so you can use Pretty to get a consensus sequence in Pretty format. Then, % pretty -UGLY makes a file of sequence names that LineUp can read. However, the consensus sequence defined by Pretty will not be recognized as the consensus sequence of LineUp. It is named Consensus by Pretty, whereas LineUp names its consensus sequence with the sequence group name. This is reasonable, since LineUp will not define the consensus in the same way, so the names should be different.

If you alternate between using Pretty and LineUp on a sequence group having a LineUp consensus sequence, you have to preserve the old sequence group name when doing Pretty -UGLY in order to make LineUp recognize the consensus sequence. If you give a new name to the group, the consensus sequence is no longer recognizable, as such, by LineUp.

THE LINEUP DISPLAY

Several indicators for LineUp are displayed on the top row of the screen. The left-most word indicates the name of the sequence on which the cursor currently rests. Next, the cursor's position on the surface is displayed. Then the display shows whether LineUp calculates the consensus automatically every time you add or delete a character. The sequence group name is next, preceded by either FOSN or MSF, indicating the file format to be used for reading and writing the sequence group. Finally, LineUp indicates whether the type of the sequence group is nucleotide or protein.
LineUp frequently displays the **PULL-OVER WARNING** sign (see the PULL-OVER AND PUSH-OVER topic above).

The screen provides a window onto the sequence surface. Through this window, 16 of the 31 surface rows can be viewed at one time. As you move your cursor near the top row of the window, for example, if there are occupied surface rows past the top of the window the surface is scrolled down, letting you see more lines at the top of the window and fewer at the bottom.

When there are more rows in use than can be displayed at once, some rows are hidden above or below the window. When this happens a '+' is displayed next to the top or bottom row number indicating hidden rows in that direction.

Although the window also scrolls horizontally, there is no analogous sign indicating that you cannot see the whole length of the surface.

**FILE NAME CONVENTIONS**

When you save a sequence group using FOSN format, the name given to the FOSN is made up of the sequence group name followed by the extension .fil. The sequence file names are the sequence names used in LineUp and the file extension .frg. When you save a sequence group using MSF format, the name given to the MSF file is made up of the sequence group name followed by the extension .msf.

These file name extensions are the defaults for LineUp, but you can specify your own by using the command-line parameters `-FOSNextension`, `-FRAGextension`, and `-MSFextension` (see below). You can override these choices when you specify an output file name; if you include a file extension, it is used in lieu of that given on the command line or the default.

**SYSTEM CRASH OR HANGUP**

The current version of LineUp cannot recover from a system crash. If you are disconnected from LineUp, you lose everything you have done since the last time you saved the group using the **Write** or **Exit** commands. Therefore, we recommend that you save your work frequently using the **Write** command so that little is lost in the event of a crash.

**ACKNOWLEDGEMENTS**

LineUp was designed and implemented by Dr. William Winsborough. We are very grateful for the collaboration of Drs. William Boorstein and Lynn Manseau of the UW Department of Physiological Chemistry.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use **-Check** to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter
values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % lineup [-INfile1=]picorna

Prompted Parameters: None

Local Data Files:

set.keys (must be in your current working directory to be used)

-MATRix=blosum62.cmp assigns the scoring matrix for Zipping proteins
-MATRix=swgapdna.cmp assigns the scoring matrix for Zipping nucleic acids

Optional Parameters:

-MSF reads and writes sequence groups in MSF format
-SINGlecommnd automatically returns to screen mode after each command
-PROtein sets sequence type to protein, and sets find to search for perfect symbol matches
-NUCleotide sets sequence type to nucleotide, and sets find to allow nucleotide ambiguity code matches
-CONSROW=0 sets the consensus row for a new sequence group
-NOCONsensus starts new sequence groups without a consensus row
-LINesize=50 sets line length for output with the PRint command
-BLOcksize=10 sets block length for output with the PRint command
-FRAGEXtension=frg sets the file extension for each sequence when using FOSN format
-CONSEXtension=con sets the file extension for the consensus when using FOSN format
-FOSNEXtension=fil sets the file extension for the list file when using FOSN format
-MSFEXtension=msf sets the file extension for the multiple sequence file when using MSF format

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.
Customizing Your Keyboard With SetKeys

You can use the program SetKeys to create a set.keys file that tells the SeqEd, GelEnter, LineUp, GelAssemble, and SeqLab sequence editors how to interpret the letters you type at the terminal. When entering gel readings, it is useful to have the symbols for G, A, T, and C under the fingers of one hand in the same positions as the lanes in your gel. SeqEd, GelEnter, LineUp, GelAssemble, and the SeqLab sequence editor automatically read the file set.keys if it is present in your local directory. If set.keys is absent, or if the sequence type is set to Protein (in SeqEd and LineUp only) the terminal keys retain their conventional meanings.

If you have a set.keys file in your directory, SeqEd, GelEnter, LineUp, and GelAssemble only respond to the keys that it redefines. You can edit the file set.keys with a text editor if some of the keys you want to use are not in it. Any keys not mentioned in set.keys appear to be dead in these sequence editors. In the SeqLab sequence editor, keys that are not redefined retain their normal meanings.

Several keys are vital for the control of SeqEd, LineUp, GelEnter, and GelAssemble; this means you are not allowed to redefine the keys for /, [ ], { }, ( ), ::, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, <Ctrl>R, <Ctrl>D, <Ctrl>H, <Return>, and <Ctrl>E.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name MyData; or 3) name a file on the command line with an expression like -MATR ix=mymatrix.cmp. If you do not include a directory specification when you name a file with -MATR ix, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

When you use the :ZIP command to align a new sequence to the current consensus, LineUp reads a scoring matrix file containing values for every possible comparison between sequence symbols. By default, LineUp reads the file swgapdna.cmp for nucleotide sequence alignments and blosum62.cmp for protein sequence alignments.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-MATR ix=mymatrix.cmp
allows you to specify a scoring matrix file name other than the program default.
If you do not include a directory specification when you name a file with -
**Matrix***, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

`-MSF`

Sets LineUp to use MSF format. LineUp reads a sequence group from an MSF (multiple sequence format) file and writes an MSF file when storing a sequence group. The default FOSN format reads and writes individual sequence files, grouped in a list file. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for a complete description of list files and MSF files.)

`-SINGlecommand`

Sets LineUp to return automatically to Screen Mode after every command in Command Mode. `-NOSINGlecommand` is the default.

`-PROtein` and `-NUCleotide`

Sets the sequence type for each sequence in the sequence group to be either protein or nucleotide. By default, if the first sequence in a sequence group is read from an existing sequence file, then the type of that sequence determines the type for the entire group. Also by default, if the first sequence in a sequence group is entered interactively from the keyboard, then LineUp sets the sequence type for the entire sequence group to be protein.

You can change the sequence type for the entire group when you are in LineUp with the `PROtein` and `NUCleotide` commands. `PROtein` tells LineUp to make pattern searches using perfect symbol matches. When LineUp is in the nucleotide state, if you type `/GARC` in Screen Mode, either of the patterns GAAC or GAGC is found. In the protein state, LineUp treats sequences as linear and will not find patterns that start at the end and continue into the beginning. In the nucleotide state, sequences are searched as though they are circular.

The automatic consensus definition is also different in the protein state than in the nucleotide state. In the nucleotide state, ambiguity codes make no contribution to the consensus. They are treated as if they were all Ns and are ignored. In the protein state, all characters have the same status.

`-CONSROW=0`

Tells LineUp on which row to put the consensus in a new sequence group. This command is only in effect if `-NOCONsensus` is *not* on the command line. The default is row 0.

`-CONsensus` and `-NOCONsensus`

Tells LineUp whether new files should start with a consensus sequence in the group. (Remember that you can create or remove the consensus sequence at anytime, so this is only a matter of convenience.) The default is `-CONsensus`. 
-LINesize=n
    sets the line length for pretty-style output created by the PRint command. The
    value must be in the range from 10 to 110. The default value is 50.

-BLOCKsize=n
    sets the block length (number of bases between spaces) for pretty-style output
    created by the PRint command. The range of n must be 1 to line size. The
    default value is 10.

-FRAGExtension=frg
    sets the file extension that LineUp uses when reading and writing sequence files
    while in the FOSN format state. Do not include the dot separating the file from
    the extension. The default value is 'frg'.

-CONSEXtension=con
    sets the file extension that LineUp uses when reading and writing consensus
    sequence files while in the FOSN format state. Do not include the dot separating
    the file from the extension. The default value is 'con'.

-FOSNExtension=fil
    sets the file extension that LineUp uses when reading and writing list files while
    in the FOSN format state. Do not include the dot separating the file from the
    extension. The default value is 'fil'.

-MSFExtension=msf
    sets the file extension that LineUp uses when reading and writing multiple
    sequence format files while in the MSF format state. Do not include the dot
    separating the file from the extension. The default value is 'msf'.

Printed: January 9, 2002 13:45 (1162)
**LISTFILE**

**FUNCTION**

ListFile prints a file on a printer attached to your terminal's pass-through printer port.

**DESCRIPTION**

This utility simply turns on the terminal's pass-through printer port and prints a text file on an ASCII printer (such as a DEC LA-50) connected there.

**EXAMPLE**

Here is a session using ListFile to print the file gamma.seq on our ASCII printer.

```
% listfile
LISTFILE what file(s) ? gamma.seq
%
```

**OUTPUT**

The printer port is turned on, and gamma.seq is printed on the printer.

**RELATED PROGRAMS**

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

**RESTRICTIONS**

ListFile is only for printing text files, not for making plots. The printer must be turned on. The terminal and printer must be set to the same baud rate.

**<CTRL>C**

If you need to stop this program, use <Ctrl>C to reset your terminal and session as gracefully as possible. Searches and comparisons write out the results from the part of the search that is complete when you use <Ctrl>C.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you **must** type in order to use the parameter. Square brackets ( [ and ] ) enclose parameter
values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % listfile [-INfile=]gamma.seq -Default

Prompted Parameters: None

Local Data Files: None

Optional Parameters:

-NOHEAding suppresses the heading at the top of the listing
-PAGe=64 sets the number of lines to print on each page
-FORMFeeds sends a form feed instead of blank lines at end of page
-BLAnklines=4 sets the number of blank lines sent at the end of each page
-NOMARgin doesn't send blank lines or a form feed after each page
-ENDLINes=14 sets the number of blank lines at end of printout
-NOENDFORM suppresses form feed at end of printout
-FILl=80 compresses text and sets the length of the compressed lines
-SPAces=1 sets the number of spaces between words
-NOPASSthrough suppresses the characters that turn the printer port on and off
[-OUTfile=]rrx6: directs output to another terminal or file

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-NOHEAding

suppresses the heading at the top of the print-out.

-PAGe=64

number of lines to print on each page.

-FORMFeeds

sends a form feed instead of several blank lines at the end of each page. Sending blank lines or a form feed at the end of each page positions the paper to begin printing on the next page.
-BLAnklines=4

sets the total number of blank lines to send at the end of each page if -FORMFeeds is not on the command line. Sending blank lines or a form feed at the end of each page positions the paper to begin printing on the next page.

-NOMARgin

doesn't send blank lines or a form feed at the end of each page. This causes printing to be continuous across the page break, without a margin at the top and bottom of each page.

-ENDLINes=14

sets the number of blank lines at the end of the listing. The first page is shortened on the assumption that there were the same number of blank lines at the beginning.

-NOENDFORM

suppresses the form feed at the end of the printout.

-FILl=80

compresses the listing onto as little paper as possible. The words from the file are compressed together and shown on completely filled lines. You can set the length of the compressed lines with the optional parameter value.

-SPAces=1

lets you set the number of spaces separating each word if you are using the -FILl parameter.

-NOPASSthrough

suppresses the characters that turn the printer port on and off. The file appears on your screen.

-OUTfile=/dev/tty11

directs output to another terminal, device, queue, or file. The file will not have standard carriage control.

Printed: January 9, 2002 13:45 (1162)
LPRINT

FUNCTION

LPrint prints text file(s) on a PostScript printer connected to LPrintPort.

DESCRIPTION

We use this utility to list text files on our PostScript laser printers. If you have a PostScript printer, your system manager has defined the logical name LPrintPort to be the port to which that printer is connected.

If the logical name LPrintPort is really your terminal, LPrint turns on the terminal's pass-through printer port before printing the file. You can send the output to a disk file if you need to move the file to another machine before printing it.

EXAMPLE

Here is a session using LPrint to print the file gamma.seq on our PostScript laser printer.

% lprint
   LPRINT what file(s) ? gamma.seq
%

OUTPUT

The output is in the Courier font at 12-point scale.

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

LPrint is only for printing text files, not for making plots. The printer must be turned on. The terminal and printer must be set to the same baud rate.

FORM FEEDS

Programs like Map can keep clusters of related material from spanning page boundaries by putting a form-feed character on a line by itself whenever the next cluster of material will not fit on the current page. If the first character on any line is a form-feed character, LPrint stops printing on this page and continues printing the material from subsequent
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Syntax: LPrint [-INfile=]gamma.seq -Default

Required Parameters: None

Local Data Files: None

Optional Parameters:

-OUTfile=gamma.ps  directs the output to somewhere other than LPrintPort
-COLumns=80  controls the number of characters per line
-NOHEADing  suppresses the heading on each page
-LANdscapae  prints lines lengthwise (eleven-inch width)
-MARgin=5  moves the left margin five characters to the right
-PAGE=60  sets the number of lines per page
-BEGin=1  sets the line number where printing begins
-END=10  sets the line number where printing ends
-NUmbering  prints a line number at the beginning of each line
-CTRLD  adds <Ctrl>D at the end of the output

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-OUTfile=temp.ps

  directs output to a file or another device instead of to LPrintPort.
-**COLumns=80**

Usually, LPrint shrinks the font to fit the number of characters on the longest line in your first input file. You can modify the font size to support between 40 and 255 characters per line with this parameter.

-**NOHEADING**

Usually, LPrint puts a heading that shows the file name, the date and the page number. You can suppress that heading with this parameter.

-**LANdscape**

The text orientation can be rotated to print parallel to the long axis of the paper with this parameter.

-**MARgin=3**

Usually, LPrint prints the first character in the first column (margin = 0). This parameter lets you shift the listing to the right by adding one or more blank characters in the margin. (The number of characters per line of output is the sum of the column and margin settings.)

-**PAGe=60**

Usually, LPrint prints 60 lines per page on regular listings and 42 lines per page in landscape orientation. You can set the number of lines per page with this parameter.

-**BEGIN=1**

You can make LPrint begin printing the file from a line other than the first line with this parameter.

-**END=10**

You can make LPrint stop printing the file at a particular line with this parameter.

-**NUMbering**

makes LPrint print a line number at the beginning of each line.

-**CTRLD**

sends a `<Ctrl>D` (^D, ASCII 4) at the end of the stream of PostScript instructions (some terminal servers require this in order to liberate the terminal for use by others).

Printed: January 9, 2002 13:45 (1162)
ONECASE

FUNCTION

OneCase puts all of the alphabetic characters in a file into lower or UPPER case. It can also capitalize every word.

DESCRIPTION

OneCase accepts a file or a list of file names and makes new files that are either all uppercase or all lowercase. Usually the new files have the same names as the original and are written into your current working directory, but you may rename or redirect them on the command line.

EXAMPLE

Here is a session using OneCase to set all of the files with a .seq file extension into uppercase:

% onecase
ONECASE of what file(s) ? *.seq
Put those file(s) into all
L)ower or all
U)pper
C)apitalize each word
case (* L *) ? u
9b_06.seq
9b_07.seq
9b_15.seq
///////////
%

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

Unknown.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% onecase [-INfile1=]*.seq -Default

Prompted Parameters:

- `-MENu=l` sets lower or upper case conversion:
  `u` = UPPER
  `l` = lower
  `c` = Capitalize Each Word

Local Data Files: None

Optional Parameters:

- `-EXTension=.txt` sets the output file name extension
- `-Directory=dirname` directs output to another directory
- `-NOMONitor` suppresses all screen output

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- `-MENu=u`

  selects the conversion to apply from the following menu: `u`=upper case, `l`=lower case, `c`=capitalize each word.

- `-EXTension=.txt`

  gives the output file a file name extension that is different from the input file's original name extension.

- `-DIRectory=DirName`

  writes the output files into a directory other than your current working directory.
This program normally monitors its progress on your screen. However, when you use `-Default` to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.
PEPDATA

FUNCTION

PepData translates DNA sequence(s) in all six frames, concatenates the translations, and creates a single, protein output file.

DESCRIPTION

It is sometimes necessary to look for protein sequence patterns in nucleic acid sequences where coding regions have not been defined or where there is some suspicion that a coding sequence may contain a frame-shift error. PepData simply translates the entire sequence in all six frames so that every possible protein translation can be examined. PepData translates nucleotide sequences in all six frames and concatenates these six amino acid sequence together into one output sequence. Stop codons appear as '*'s in the output sequence.

If you translate several different nucleotide sequences, the translations are written into separate output files with the same name as the nucleotide sequence but with the file name extension .pdt.

You can find out how to specify a group of sequences that interest you in Chapter 2, Using Sequence Files and Databases in the User's Guide.

EXAMPLE

Here is a session using PepData to translate some of the human globin sequences in GenBank:

% pepdata

PEPDATA from what sequence(s) ? GenBank:Humhb*

Humhbl6aa  1,216 bp   2,428 aa
Humhlaz    483 bp     962 aa
Humh24     2,231 bp   4,458 aa

PEPDATA complete with

Input files:  103
Amino acids: 431,184
Output files:  103
Output file names: *.pdt

%
OUTPUT

Here is the part of the first output file, which is humhb16aa.pdt:

!!AA_SEQUENCE 1.0
PEPDATA from: humhb16aa  check: 8158  from: 1 to: 1,216

M31630 Human cyclic AMP response element-binding protein (HB16) mRNA, 3' end.
6/90

bases 1 to: 1216 translated into: 1 to: 405
bases 2 to: 1216 translated into: 406 to: 810
bases 3 to: 1216 translated into: 811 to: 1214
reverse of bases 1 to: 1216 translated into: 1215 to: 1619
reverse of bases 1 to: 1215 translated into: 1620 to: 2024
reverse of bases 1 to: 1214 translated into: 2025 to: 2428

humhb16aa.pdt  Length: 2428  September 30, 1998 15:35 Type: P Check:
3306 ..

1  VPGFPPLLHL LPNGQTMPV IAASITSSNV HVPAAVPLVR PVTMVPSVPG
51  IPGPSFQPV QSEAKMRLKA ALTQHPPVT NGDVTGHGS GLVRTQSEES
101  RPQSLQQFAT STTETPASPA HTTPQIQSTS GRRRAANED DKEKRRKFLF

INPUT FILES

PepData accepts multiple (one or more) nucleotide sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf{*}; or by using a sequence specification with an asterisk (*) wildcard, for example GenEMBL:*. If PepData rejects your nucleotide sequence, turn to Appendix VI to see how to change or set the type of a sequence.

RELATED PROGRAMS

Translate and Map are other programs that translate nucleotide sequences into protein. BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. TFastA does a Pearson and Lipman search for similarity between a protein query sequence and any group of nucleotide sequences. TFastA translates the nucleotide sequences in all six reading frames before performing the comparison. It is designed to answer the question, "What implied protein sequences in a nucleotide sequence database are similar to my protein sequence?" FrameSearch searches a group of protein sequences for similarity to one or more nucleotide query sequences, or searches a group of nucleotide sequences for similarity to one or more protein query sequences. For each sequence comparison, the program finds an optimal alignment.
between the protein sequence and all possible codons on each strand of the nucleotide sequence. Optimal alignments may include reading frame shifts. FrameAlign creates an optimal alignment of the best segment of similarity (local alignment) between a protein sequence and the codons in all possible reading frames on a single strand of a nucleotide sequence. Optimal alignments may include reading frame shifts.

**RESTRICTIONS**

Since GCG sequences cannot contain more than 350,000 symbols, PepData cannot translate sequences longer than 175,000 bp.

**OUTPUT FILE NAMES**

PepData names output files with the base name of the input file (without the directory) and the file name extension .pdt and writes them in your current working directory.

**COMMAND-LINE SUMMARY**

All parameters for this program may be added to the command line. Use `-CHEck` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% pepdata [-INfile=]genbank:humhb*`

Prompted Parameters: None

Local Data Files:

[-TRANslate=]translate.txt contains the genetic code

Optional parameters:

-EXTension=.pdt lets you specify an output file name extension
-DIRectory=dirname writes output to another directory

**LOCAL DATA FILES**

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like `-DATa1=myfile.dat`. For more information see Chapter 4, Using Data Files in the User's Guide.

The translation of codons to amino acids, the identification of potential start codons and stop codons, and the mappings of one-letter to three-letter amino acid codes are all defined in a translation table in the file translate.txt. If the standard genetic code does not apply to your sequence, you can provide a modified version of this file in your
working directory or name an alternative file on the command line with an expression like -TRANSlate=mycode.txt. Translation tables are discussed in more detail in Appendix VII.

**PARAMETER REFERENCE**

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-TRANSlate= filename.txt

Usually, translation is based on the translation table in a default or local data file called translate.txt. This parameter allows you to use a translation table in a different file. (See Appendix VII for information about translation tables.)

-EXTension= extensionname

allows you to choose a file name extension for the output files other than the .pdt that PepData uses by default.

-DIRectory= DirName

writes the output files into a directory other than your current working directory.
REPLACE

FUNCTION

Replace makes character string replacements in text file(s). You provide a table of
replacements in a file showing each existing string and its replacement.

DESCRIPTION

Replace reads existing strings and replacement strings from a file of replacements.
Replace reads each line of each file you want to modify. It searches for the existing
strings and replaces them with the replacement strings. The existing and replacement
strings can have different lengths, and any character can be put in either string. The
replacements are made to each line in the order of occurrence in the file of
replacements. Replace writes a new version of each modified file in the same directory
as the old files. The new files have the same name as the old files unless you use the
command-line parameter \texttt{-EXTension}. If you use the \texttt{-EXTension} command-line
parameter, the new files are written in your local directory.

EXAMPLE

Here is a session using Replace to convert a file of FORTRAN source code to our new
coding standard:

```
% replace

REPLACE in what file(s) ? cmpvals.for
What replacement file ? GenDocData:procedure.replace
What should I call the output file (* cmpvals.for *) ?
REPLACE complete with:

Files modified: 1
Total Replacements: 29
```

OUTPUT

The output files have the same names and look exactly like the input files except for the
strings that were replaced.

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are
the Wisconsin Package file utilities programs.
RESTRICTIONS

Replace is record oriented -- the strings you want to replace must be found on the same line. The replacements are made in the order of the replacements specified in the file of replacements. You cannot have more than 1,000 different replacements. Lines may be elongated by the replacements, but Wisconsin Package programs cannot read or write lines longer than 511 characters!

INPUT FILE

After an introductory text heading, the strings that you want to replace and the strings that you want to replace them with are enclosed in double quotation marks. The existing and replacement strings must be on the same line. If you want to use a double quotation mark inside an existing or replacement string, simply use two double quotes in a row to represent one. If you want to require that the search for some, but not all, of the existing strings be case sensitive, you can add a 'C' after the replacement string on those lines where you want a case-sensitive search. If you only want to replace the existing string if it occurs in a particular column add the column number after the replacement string.

As in all Wisconsin Package data files, comments can follow an '!' (exclamation) character on any line and blank lines are allowed. Here is a small part of the file of replacements used for example session:

A file of replacements used for the example session with REPLACE and to make old GCG Fortran source code compatible with the C programming language.

"Existing string"       "Replacement string"  ..
" BEEP"                " RingBell"
! changes FPRINTF AND SPRINTF
"PRINTF"               "WriteF"
! changes SSCANF AND FSCANF
"SCANF"                "ReadF"
"WGETCHARVECTORS"      "WGetHersheyVectors"
"GETCH"                "Get_Ch"
"ISBLACK"              "ChIsBlack"
"ISUWGCG"              "ChIsGCG"
! changes STRISUPPER AND STRISLOWER
" ISLOWER"             " ChIsLower"
" ISLOWER"             " ChIsLower"
".ISLOWER"             " .ChIsLower"
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % replace [-INfile=]cmpvals.for \
[-Repfile=]gendocdata:procedure.replace -

Default

Prompted Parameters:

[-OUTfile=]cmpvals.for names output file 
(only when the input file name is NOT ambiguous)

Local Data Files: None

Optional Parameters:

-CASE does a case-sensitive search for all existing strings
-TRACE[=filename] shows lines with replacements on your screen or in a file
-EXTENSION=.replace resets the file name extension on the output files
-NOMONITOR suppresses the screen monitor
-NOSUMMARY suppresses the screen summaries

LOCAL DATA FILES

None.
PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-CAS e

Usually, Replace does a case-independent search for the existing strings that you want to replace. You can restrict that search to find only strings that match in case as well as in characters with this parameter. You can specify a case-dependent search for any particular existing string by adding a 'C' after the replacement string in the replacement file.

-TRA ce=filename

For any line with a replacement, Replace prints the old line and the new line on your terminal screen. This parameter also displays the information from your replacement file. If you supply a value for filename, this parameter will write the tracing information into a file instead of on your terminal screen.

-EXT e nsion=.replace

Usually, Replace uses the input file names as the names for the output files. Use this parameter without the optional value to indicate that the output file name extension should be .replace instead of the original file name extension. To designate a different extension, type it as the value for this parameter. If you use this parameter, the output files are written in your local directory.

-MON itor

This program normally monitors its progress on your screen. However, when you use -default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

-SUM mary

writes a summary of the program's work to the screen when you've used -default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUM mary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.

Printed: January 9, 2002 13:45 (1162)
RED

FUNCTION

Red is a text formatter that creates publication-quality documents on a PostScript printer such as the Apple LaserWriter. You can use 13 different fonts, scaling each font to any size. You can also include figures and graphics from any Wisconsin Package graphics program within the text of the document.

DESCRIPTION

Red is a text formatter somewhat like RUNOFF. You supply a file with text and embedded formatting commands, and Red formats your document and prints it out on a PostScript printer. PostScript is a powerful page description language that can make publication-quality output on printers equipped with a PostScript interpreter, such as the Apple LaserWriter. Red is designed to extend the philosophy of RUNOFF to support some of PostScript's features. Most Wisconsin Package documentation and correspondence is printed using Red.

Formatting is controlled by two basic entities: commands and flags. Commands appear by themselves on a line that starts with a period; flags are special characters that occur within regular text. Typical commands move the carriage or reset the margins; typical flags change the font, move the carriage for subscripts and superscripts, or insert the current date. A complete list of commands and flags appears below.

If you only want part of a document, you can specify the first and last pages to print. The PostScript printer must be attached to the port to which the logical name LPrintPort has been assigned.

You can include a figure from a Wisconsin Package graphics program within the text of a Red document.

EXAMPLES

Here is a session using Red to format the document red.red and send the output to a PostScript printer attached to LPrintPort:

```
% red
RED format what file(s)? Program_Manual:red.red
First page (* 1 *) ?
Last page (* 9999 *) ?
```

OUTPUT

You are reading the output from this session with Red.
RELATED PROGRAMS

Figure makes figures and posters by drawing graphics and text together. You can include output from other Wisconsin Package graphics programs as part of a figure. You can include output from the Figure program (or from any other Wisconsin Package program that produces graphics output) within the text of a Red document with the .PSInclude command, if your graphics configuration was set to produce PostScript output, or .GIFInclude, if your graphics configuration was set to produce GIF output (HTML only).

You can also include output from graphics producing Wisconsin Package programs with the .PNGInclude command if your graphics configuration was set to produce PNG output (HTML only).

RESTRICTIONS

Red directs its output to LPrintPort so this logical name must be assigned to the correct port or queue before you run Red.

COMMANDS

Any line of the input file that starts with a period is interpreted by Red as a command. Commands are shown below as they would appear on a line in the input file. The word in parentheses is the mnemonic for the command. A plus sign (+) following the parenthesis indicates that RUNOFF does not support the command.

Numbers like n1 following commands are integers such as 0, 1, 2. Numbers like f1 and numbers with a decimal point following commands are real numbers such as 1.0, 2.5. All carriage-control parameters (for example, right margin, left margin, spacing) are shown with a decimal point to imply that they can be fractional -- you can skip 1.5 lines.

Points are 1/72 inch. There are 12 points per line (vertical space). The vertical space is the unit used with the top margin, page size, skip, and test page commands. There are six points per character (horizontal space). The horizontal space is the unit used with the left margin, right margin, and set paragraph commands. The horizontal space is constant regardless of the font and scaling being used.

Commands can be in either upper- or lowercase. If there are numbers following the command, there should be a space between the command and the numbers. If more than one number is used, the numbers can be separated from each other by a comma and/or a space.

BEGINSKIP (begin a skip) (+)

starts a section that is skipped over when –ASCii is on the command line. There are often sections of a document (such as figures) that cannot be included in an ASCII output file. This command, together with the .EndSkip command, lets you skip over such sections.
\texttt{.box} \ +5.0 \ -5.0 \ +0.0 \ +0.0 \ (box)

draws a box relative to the current margins settings. The first and second numeric values are the amount to add or subtract to the left and right margins (in 1/12 inch increments). The third and fourth values are the amount to add or subtract to the current carriage position (in 1/6 inch increments).

If the third and fourth values are both +0.0 the box will look like a horizontal line, that is, the box will have no height. If the first and second values are both +0.0 the box will look like a vertical line, that is, the box will have no width.

If the values do not start with a plus or a minus, the box will drawn in the absolute position you specify on the page, regardless of the current carriage position and margin settings.

This command does not move the carriage.

If you do include any parameter values at all, this command draws a box around the current text area. This area is defined by the most recent left margin, right margin, top margin, page size, and shift over commands (.lm, .rm, .tm, .ps, and .so). We use this box to make sure the text is centered on the page or shifted away from the binding sides of each page appropriately.

Before version 9.0, this box extended around the area where the running titles were displayed. In version 9, however, running titles can be anywhere on the page (See .nmfp, .tfp, and .stfp.)

If paragraphs are negatively indented, the box is drawn around the filled part of the text, and the first line of each paragraph will start outside the box.

The file GenDocData:redtest.red makes a little test pattern that exercises this command.

\texttt{.br} \ (break)

stops filling and prints all of the words in the last line as a line of uncertain length, when text is being filled. (Many commands issue a break command before carrying out the rest of their purpose.)

\texttt{.c} \ (center)

centers the text on the next line of the input file between the current left and right margins. If the text being centered will not fit between the current margins, Red centers as much of it as it can and centers the rest on additional lines.

\texttt{.cp 25.0} \ (carriage position)

issues a break and sets the carriage to start printing at line 25. (Lines are measured in 1/6 inch units from the top of the page.) It is probably not a good idea, but you can, at
least in principal, move the carriage backwards to an absolute line with this command. (See .tm.)

**DefineStyle** name (define a style)

begins defining a style with the given name. Styles are discussed in detail below.

**EndNew** (end a section of new emphasis)

issues a break and then ends the section that is being emphasized. (See also .NewStuff.)

**EndSkip** (end a skip)

ends a section that is skipped over when -ASCii is on the command line. (See also .BeginSkip.)

**EndStyle** (end a style definition)

terminates the current style definition. Styles are discussed in detail below.

**EPSInclude** FileSpec 1.0 (Encapsulated PostScript Include)

This command includes a figure at the current carriage position by naming a file of encapsulated PostScript instructions to be sent to the printer.

**f** (fill)

sets Red to fill all of the free-format text in your input file. Within filled text, Red will skip two spaces instead of one after periods, colons, exclamation marks, question marks, semicolons, or right parentheses. Filling may be part of a style, explained below. (See also .nf and .j.)

**fgd 10.0** (figure deferred)

inserts a blank space of 10.0 lines (for a figure). If a space of this size will not fit on the current page, Red leaves the blank space at the top of the next page. Such a pending figure is said to be deferred. You can have up to 32 simultaneously deferred figures. (See also .PSInclude.)

**followon** (follow on)

Normally Red tests the page to see if there is enough room whenever you start a paragraph. However if you have multiple topic lines, as we often do in the documentation for the Procedure Library, you will often start a multi-heading paragraph only to have the page turn before all the headings are shown. This feature will suppress the test page service whenever a paragraph follows a paragraph that is only one line long. (See also .spr and .tp.)
Fonts come in groups of up to eight different fonts. Normally Red uses the first font you have chosen (n1). The number of each available font is shown in the list below. When you use a font flag to begin printing in italics, bold, or bold italics, Red switches to font numbers n2, n3, and n4 respectively. A switch-font-group flag can be used to make a second group of four fonts (n5-n8) work like fonts n1-n4. If you specify only one font (n1), Red uses fonts n1+1 through n1+7 for fonts n2-n8. Tables showing each character in each font appear at the end of this document. Here are the fonts we currently support, scaled to 13 points:

1. New Century Schoolbook
2. New Century Schoolbook Italic
3. New Century Schoolbook Bold
4. New Century Schoolbook Bold-Italic
5. Courier
6. Courier Oblique
7. Courier Bold
8. Courier Bold-Oblique
9. Avant Garde Book
10. Avant Garde Oblique
11. Avant Garde Demi
12. Avant Garde Demi Oblique
13. Symbol

You can define 10-point font groups for fonts n1-n4 with the .SchoolBook, .Courier, and .AvantGarde commands.

This command interrupts execution of the current input file and generates HTML content to display the specified GIF image file. If you include the optional width and height values, the image will be displayed to those size constraints. (See also .PNGInclude.)

Sometimes, such as when you are modifying a big document, you temporarily want to skip over sections. This command sets Red to skip everything in the current input file until it finds a line that starts with .!Below. This target line must occur in the same input file as the .GoTo command -- all included files between the GoTo and its target are ignored. The pagination and absolute position on the page of the text following the target is disrupted.
.If Symbol .AnyCommand (if)

tests the value of Symbol and executes any valid Red command if it is true. (See the .Set and .Unset commands to find out how to set a symbol value.)

.Include FileSpec [string1 string2] (include a file)

This command interrupts execution of the current input file and starts reading text and formatting instructions from an included file until that file is exhausted. Included files can include other files and these can include others to a total depth of 25 files.

If you include the two optional strings, every instance of string1 in the included file is replaced with string2. String1 and string2 cannot contain spaces, but they may contain flags. The scope of the substitution only extends through the included file (and any files that it can, in turn, include).

.j (justify)

sets Red to stretch filled lines so that they extend all the way from the current left margin to the current right margin. This is done by widening the intervening spaces appropriately. The last line of each paragraph is not justified. Justification can be part of a style, explained below. The text in this document is justified. (See also .f and .nj.)

.Letter 1 2 (letter tray1 tray2)

sets Red to print the first page on the paper in tray 1 (see .tray) and continue all subsequent pages from tray 2. If numbering is enabled, the running titles appear at the upper left with the name of the addressee, the date, and page shown on three separate lines.

You should set title font and position with a command like .tfp 1 0.9 4 0 in order to leave enough room.

Red finds out the name of the addressee in one of two ways: 1) you put the name of the addressee on the first line, following a comment line that looks like .Address; or 2) you define the name of the addressee with the title command (.t).

.ListItem 1 (list item)

Red formats a single member of a list of alternatives when your source file contains a string like ^|OpenVMS|UNIX|. This feature is normally disabled, unless you use the .ListItem or .fl list command. If there has been no call to .ListItem, but the .fl list command has enabled the list flag, Red prints the first item of the list. You can have up to 10 items in a list. You can have more than one list of items on a line, but all of the items in any list must occur on the same line. The items may contain other flags and spaces.
.lm 5.0 (left margin)

sets the left margin (the carriage position to the right of which text is printed). The units of horizontal carriage movement are 1/12 inch -- the same as a typewriter using elite type. These units are the same no matter what font and scaling you are using. Usually margins are set at the top of a document and only relative margin adjustments are made within the document. Make relative margin adjustments by using a signed number such as +5.0 or -10.0. For example, your text is centered on a page that is 8.5 by 11 inches when you use margins of 5.0-90.0 or 10.0-85.0. The left margin can be part of a style, explained below. (See also .rm).

Here is approximately what the horizontal carriage ruler looks like:

80  10  20  30  40  50  60  70
80  10  20  30  40  50  60  70
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |

.NEWStuff 2.0 (emphasize new stuff)

identifies new features by adding a dark line to the right of the text from the line where this command occurs down to the point where the next .endnew command occurs. We use these lines to add emphasis to the parts of our document that are new in the current release. The optional value tells how many margin units outside the margin the emphasis line should appear. If you have used .TwoSided then the line appears on the left on even-numbered pages. (See .endnew.)

.nf (no fill)

sets Red to not fill all of the free format text in your input file. (See also .f and .j.)

.nj (no justify)

sets Red to stop justifying filled lines. (See also .j.)

.nm 1 (enable numbering and titling)

sets Red to show page numbers and running titles and subtitles. If you provide a number with this command, Red puts that number on the current page and starts numbering from that number upwards. If you have defined a title or a subtitle, these will also appear. (See also .nmm, .nmr, .Section, .st, .t, .nmfp, .tfp, and .stfp.)

NOTE: When you issue this command, Red resets the title, subtitle, and number positions using the current settings for top margin, page size, left margin, and right margin. (See .tm, .ps, .lm, and .rm.)
.nmr nl (numbering in Roman)\(^{(*)}\)

works like .nm, but pages are numbered with Roman numerals. If your command is uppercase (.NMR), Red displays the numerals in uppercase. (See .nm.)

**NOTE:** When you issue this command, Red resets the title, subtitle, and number positions using the current settings for top margin, page size, left margin, and right margin. (See also .nm, .tfp, .stfp, .nnm, .TwoSided, .ps, .tm, .lm, and .rm.)

.nmfp 1 1.0 2.0 0.0 (number font and position)\(^{(+)}\)

Red will number pages on the right of every page either above or below the main text. When .TwoSided is used, these numbers appear on the left for for even-numbered pages. The first two numbers are the font and scale of these running numbers. The third number is the number of lines above the top or below the bottom line of the text where these numbers should appear. If the third number is positive the number will be above the main text, if negative below.

The fourth number sets the number of margin units outside or inside the outer text margins where the page number will appear. If the fourth number is zero, page numbers are justified at the text margin. If the fourth number is positive, the page number appears outside the margin, if negative inside.

Whenever the subtitle and page number are set to appear together on the same line, the subtitle is justified against the page number with the page number on the outside.

**NOTE:** When you issue this command, Red resets the title, subtitle, and number positions using the current settings for top margin, page size, left margin, and right margin. (See also .nm, .tfp, .stfp, .nnm, .TwoSided, .ps, .tm, .lm, and .rm.)

.nnm (no numbering or titling)

turns off the display of page numbers and running titles. (See also .nm.)

.nofollowon (follow on)

reverses the .followon command. (See .followon.)

.NoSubstitute stringl (nosubstitute)\(^{(+)}\)

stops substitution of stringl. Stringl is any pattern of ASCII printing characters that have been used with a previous .Substitute command. The substitution stops until you use the .Substitute command again to turn it back on.

.OneSided (one sided)\(^{(+)}\)

sets running titles to appear with titles on the left and subtitles and page numbers on the right. (See .TwoSided.)
.PNGInclude PNGImageFileSpec

[width [height]] (include a Portable Network Graphics file, HTML only)

This command interrupts execution of the current input file and generates HTML content to display the specified PNG image file. If you include the optional width and height values, the image will be displayed to those size constraints. (See also .GIFInclude.)

.ps 55.0 (page size)(+)

sets the bottom margin for a page -- the lowest carriage position on which text can be displayed. When text or skipped lines bring the carriage below this position, Red issues a new page command and continues on the next page. (See also .tm.)

.PSInclude FileSpec 1.0 (PostScript Include)(+)

This command includes a figure at the current carriage position by naming a file of PostScript instructions to be sent to the printer (see the FIGURES topic below).

.pg (new page)(+)

ends the current page and prints any additional text on subsequent pages. If the current page has nothing on it, nothing happens. Therefore, this command can appear in the heading at the beginning of an input file.

If you want chapters to start on odd pages, add odd to the parameter. This parameter makes Red print out a blank page if the current page is odd.

If you do not want the next page numbered, use new.

If you want Red to print two pages on each page, use double.

If you want Red to print the text of the next page along the long dimension of the page, use the landscape.

If you want Red to number the first page of your document, use old. (Use this with odd to make Red number the first page of the following chapter.)

.rm 90.0 (right margin)

sets the right margin, which is the carriage position beyond which text will not be printed. (See the .lm command for a discussion of margin control.) The right margin can be part of a style, explained below.
issues a break command and skips 1.0 lines (a line is 1/6 inch). If filling is turned on and if spacing is set to 2.0, the break skips a line and this skip command skips an additional line. The number of lines skipped can be negative. (See also .sp, .sd, and .br.)

.Scale f1 f2 f3 f4 f5 f6 f7 f8 (scale)

Each font in a font group (see .Font) can be scaled independently. The scales are decimal numbers such as 1.0, 1.1. A scale is a factor by which the standard 10-point font is multiplied to make it bigger or smaller. You can specify just one scale factor if you want all eight fonts to be printed with the same scaling. Here is what the New Century Schoolbook font (Font 1) looks like at several different scalings:

- 0.8 prints each font at 8 points
- 0.9 prints each font at 9 points
- 1.0 prints each font at 10 points
- 1.1 prints each font at 11 points
- 1.2 prints each font at 12 points
- 1.3 prints each font at 13 points
- 1.4 prints each font at 14 points
- 1.5 prints each font at 15 points

-section

You can associate page numbers with a section or chapter by defining a string that prints just to the left of the number. In this case the string A− would make the numbering of page three look like A−3. (See also .nm and .nmr.)

.sd 1.0 (skip if dirty)

It is often desirable to skip two lines before starting a figure or a table. On a fresh page, these two lines are wasted. Skip if dirty is a conditional skip that deals with this problem. It issues a break and then skips a line only if something has already been printed on the current page. (A line is 1/6 inch.) The number of lines skipped can be negative.

.Set Symbol (set)

sets the value of a symbol to true. The symbol may be new or may have already been set by previous calls to .Set or .Unset. (See also .Unset and .If.)

.so 2.0 (shift over)

When binding double-sided copies of a manuscript, the binding uses up some of the left margin on odd pages and some of the right margin on even pages. The shift over command staggers the left and right margins depending on the page number. A shift over of 2.0 staggers the odd pages two character widths (2/12 inch) to the right and the
even pages two character widths to the left. The difference between odd and even pages is 4.0 character widths (4/12 inch).

.sp 1.5 (spacing)

sets the number of lines the carriage advances after each line when filling is turned on.

.spr 5.0 1.0 5.0 (set paragraph)

The set paragraph command describes how you want filled paragraphs to be formatted. When filling is turned on, any line that starts with a space is taken to be the start of a new paragraph. The three numeric values for set paragraph determine the amount of indentation (positive or negative), the number of lines to skip between paragraphs, and the number of lines of text that must be left on the page (see .tp). If there are not enough lines of text available, a new page command is issued before the paragraph is printed. The paragraph setting can be part of a style, explained below.

Use a fractional negative indent and turn off justification to make perfect bibliographies. (See also .nj.)

Notice that the units of vertical carriage movement are different in the two parameters for test page and skip. Test page measures the number of lines of text left on this page at the current setting of spacing while skip uses the standard unit of carriage movement, which is 1/6 inch. (See also .tp and .s.)

.Style name (use a style)(+)

start using the previously defined style. Styles are discussed in detail below.

.Subscale 0.5 (subscript scale)(+)

adjusts the size of subscripts in relation to normal text. To print subscripts at the same size as normal text, use .Subscale 1.0. The default setting (.Subscale 0.5), prints subscripts at half the size of normal text. (See the flags @[ and @].)

.Substitute string1 string2 (substitute)(+)

substitutes string2 for every instance of string1. String1 and string2 are any pattern of ASCII printing characters -- they may not contain spaces. String1 and string2 may contain flags. The substitution continues until you use the .nosubstitute command.

.Superscale 0.5 (superscript scale)(+)

adjusts the size of superscripts in relation to normal text. To print superscripts at the same size as normal text, use .Superscale 1.0. The default setting (.Superscale 0.5), prints superscripts at half the size of normal text. (See the flags @[ and @].)
.st ChapterName (subtitle)

defines a running subtitle that appears when numbering is enabled. The font and position of the running subtitles is set with .stfp. (See also .stfp, .tfp, .nm, and .TwoSided.)

.stfp 1 1.0 2.0 0.0 (subtitle font and position)

Red will show a running subtitle on the right of every page either above or below the main text. When .TwoSided is used, these titles appear on the left for even-numbered pages. The first two numbers are the font and scale of these running titles. The third number is the number of lines above the top or below the bottom of the text where the title should appear. If the third number is positive, the title will be above the main text; if negative, below.

The fourth number is the number of margin units outside the outer margin of the page. If this fourth number is positive, the title is outside the margin; if negative, inside.

Whenever the subtitle and page number are set to appear together on the same line, the subtitle is justified against the page number with the page number on the outside (effectively ignoring the fourth parameter of this command).

NOTE: When you issue this command, Red resets the title, subtitle, and number positions using the current settings for top margin, page size, left margin, and right margin. (See also .tfp, .nmfp, .stfp, .nm, .nnm, .TwoSided, .t, .st, .ps, .tm, .lm, and .rm.)

.t BookName (title)

defines a running title that appears when numbering is enabled (see .nm). The font and position of the running titles is set with .tfp. (See also .st, .tfp, .stfp, .nm, and .TwoSided.)

.tf n1 f1 (title font)

This command has been replaced by .tfp. Do not use it and remove it from any Red files you have.

.tfp 1 1.0 2.0 0.0 (title font and position)

Red will show a running title on the left of every page either above or below the main text. When .TwoSided is used, these titles appear on the right for even-numbered pages. The first two numbers are the font and scale of these running titles. The third number is the number of lines above the top or below the bottom of the text where the title should appear. If the third number is positive, the title will be above the main text; if negative, below. The fourth number is the number of margin units outside the margin of the page. If this fourth number is zero, the titles are justified to the margin. If this
fourth number is positive, the title is outside the margin (toward the binding); if negative, inside.

NOTE: When you issue this command, Red resets the title, subtitle, and number positions using the current settings for top margin, page size, left margin, and right margin. (See also .tfp, .nmfp, .stfp, .nm, .nnm, .TwoSided, .t, .st, .ps, .tm, .lm, and .rm.)

.tm 6.0 (top margin)

This command sets the number of lines below the top of the page where the first line of text should appear. The unit of vertical carriage movement is 1/6 inch. For this example, therefore, the first line of regular text would be on line six, one inch below the top of the page. The bottom of each character in the first line of text touches this line. (See also .tfp, .nmfp, .stfp, .ps)

NOTE: Before version 9, this command took two parameters. The first was distance in lines (1/6 inch) from the top of the page to the running titles. The second was the distance from the running titles to the main text area. Red can still interpret these old-format .tm commands. If Red sees two parameters, it infers that you want your titles in the old format and puts the number, title and subtitle all on the line above your main text. The title positions are all reset to this height when such an old-format .tm command is appears in your document.

.tp 5.0 (test page)

issues a break and tests the number of lines of text that can be formatted onto this page (at the current spacing). If you have set spacing to 2.0, then a test page of 3.0 would make sure that 6.0 lines were left on this page. The unit of vertical carriage movement, referred to as a line, is 1/6 inch. (See also .s .sp .tm, and .ps.)

.Tray 1 (set paper tray)

sets Red to use a paper tray other than the default tray. On PostScript printers, such as the LZR1200 series from Dataproducts Corporation, the paper trays are numbered as follows: 0 is the main cassette (the default), 1 is the upper cassette of the multi-cassette sheet feeder, 2 is the lower cassette of the multi-cassette sheet feeder, and 3 is the envelope feeder.

.ts f1 f2 f3 f4 ... (tab stops)

sets tab stops. If you have set up tab stops, put tabs into a line and if filling is not turned on, Red skips ahead to the next tab column if the line has not already printed beyond it. (See .lm for more information about the horizontal carriage movement.)
.TwoSided (two sided)

sets titles, subtitles, and numbering to be mirrored between odd- and even-numbered pages. Numbers and subtitles are on the left and titles are on the right of even-numbered pages while titles are on the left and subtitles and numbers are on the right of odd-numbered pages. (See .nm and .OneSided.)

.Unset Symbol (unset)

sets the value of a symbol to false. The symbol may be new or may have already been set or unset by previous calls to .Set or .Unset. (See also .Set and .If.)

FLAGS

Flags occur within text. They are used to change fonts, to make superscripts or subscripts, or to substitute the current time, date, or input filename. Flags are composed of the infrequently used characters: @, ^, \, _, #, =, [, ], *, &, %, |, and $.

You can get flag characters to print in one of two ways: 1) precede any flag character with an underscore (_) to cause Red to print the flag character instead of interpreting it as a flag; or 2) disable the interpretation of any flag with an .nfl command.

Flags are usually composed of two infrequently used characters. They are shown in the list below exactly as they would appear in your input file. The word in parentheses following the flag is the mnemonic for the flag.

^* (switch font to bold)
changes the font to bold. If italics are currently being used, this flag changes the font to bold-italic.

\* (stop printing in bold)
reverses the action of the switch-font-to-bold flag.

^& (switch font to italic)
changes the font to italic. If text is currently being printed in bold, this flag changes the font to bold-italic.

\& (stop italics)
reverses the action of the switch-font-to-italic flag.

^+ (switch from VMS to Unix case convention)
changes the case from mixed upper and lower to all lower.
\+ (switch to VMS case convention)

reverses the action of the switch-case-to-Unix flag.

^% (switch to alternate font group)
changes the font group (see .Font). Four alternate fonts become available for regular, bold, italic, and bold-italic printing.
\% (switch to regular font group)

reverses the action of the switch-to-alternate-font-group flag.

@I (move carriage up 1/2 line)

moves the carriage up 1/2 line to start a superscript or end a subscript.

@J (move carriage down 1/2 line)

moves the carriage down 1/2 line to start a subscript or end a superscript.

_ (accept)

If you want to print a character normally interpreted as a flag, precede it with an accept flag.

# (space)

If you want two words to always print on the same line (such as Appendix I), put a space flag between them (Appendix#I). You can also add extra space to centered or filled text with the space flag.

When filling, Red skips two spaces after periods, exclamation points, and question marks. If you use a period in an abbreviation (for instance, Dr. Devereux), the extra skipped space is inappropriate. Use a space flag between the period and the next word to avoid the extra skipped space.

$$ (substitute)

You can include the current year, month, day, hour, minute, or filename in a document with the substitute flag. The expression $$Month$$Day, $$Year will print the current date: January 9, 2002. You can include the name of the file being formatted. Here is the syntax of each supported substitution: $$Year, $$Month, $$Day, $$Hours, $$Minutes, $$File.

^| (begin a list of string items)

starts a list of strings, one of which may be selected for output with the .ListItem command. Each string in the list is referred to as an item.
(list separator)

separates one list item from the next within a list.

\| (end the string list)

signifies the end of a list of string items.

DISABLING FLAGS

If you include a vanilla text file within your document, flag characters within it will not print correctly. You can disable each kind of flag or all the flags at once with the `.nfl` command. You can enable any disabled flag with a corresponding `.fl` command. Within a style definition (see below) you may enable all flags, or disable all flags, but there is no way for a style definition to enable or disable particular flags.

`.nfl all` (disable all flags)

disables all flags and prints all the flag characters in your input file. Tabs are not interpreted according to the settings made with the `.ts` command when all flags are disabled. When you reenable the flags with `.fl all`, any flags that were individually disabled remain disabled. This command can be part of a style definition.

`.nfl accept` (disable accept flag)

prints and does not interpret underscore characters (_).

`.nfl alternate` (disable alternate font flag)

prints and does not interpret the two-character patterns `^%` or `%.`.

`.nfl bold` (disable bold flag)

prints and does not interpret the two-character patterns `^*` or `*`.

`.nfl hyphenate` (disable hyphenation flag)

prints and does not interpret equal signs (=). (Red does not currently support hyphenation, but this flag is maintained for compatibility with RUNOFF.)

`.nfl list` (disable list flag)

prints and does not interpret the list flags `|^` and `\`. Red formats a single member of a list of alternatives when your source file contains a string like `^|OpenVMS|UNIX\|`. This feature is normally disabled, unless you use the `.ListItem` command or the `.fl list` command. If there is no call to `.ListItem` and the flag is enabled, Red prints the first item in the list.
.nfl space (disable space flag)

prints and does not interpret pound characters (#).

.nfl substitute (disable substitution flag)

prints and does not interpret the two-character pattern $$.

.nfl superscript (disable superscript flag)

prints and does not interpret the two-character patterns @[ or @].

.nfl underline (disable italic flag)

prints and does not interpret the two-character patterns ^& or \&.

**STYLES**

Starting with Version 8.0, Red supports style definitions. A style is a set of formatting commands grouped together and given a name. Once a style is defined, all of its component formatting commands can be recalled at any time by giving a single command. For example:

```
.DefineStyle Typewriter
   .lm 10
   .mono
   .nf
   .EndStyle

.DefineStyle Normal
   .lm 5
   .rm 90
   .spr 0, 1, 5
   .f
   .prop
   .nofollowon
   .EndStyle
```

The above defines a style named *Typewriter* which uses a left margin of 10, monospaced, and no filling; and a style named *Normal* with left margin 5, right margin 90, proportional, filling, and doesn't suppress the test page service. With these definitions made, Red can easily switch back and forth between the two styles, like this:

```
.style normal
```

This part of the document is formatted in the "normal" style, with proportional font, filling, and so forth.
There are two good reasons for using styles in your documents. First, they reduce noise and clutter by replacing several lines of cryptic format commands by single lines with meaningful style names. Second, they facilitate global format changes. Once you have determined which parts of your document should be in typewriter style, you can then change the style definition, a different left margin for instance, and have the change take effect everywhere the style is used. This is much easier than searching for left margin commands, figuring out which ones are there to effect typewriter styling, and changing only those.

A good practice is to identify those styles commonly required by the documents you usually write, define them in a common header file, and include this file in all your document sources. In this way, you can easily change your mind about the particulars of a style and have the changes affect all of your documents.

Here is a list of the formatting commands that have meaning inside a style definition.

% f  
% fl all  
% j  
% lm  
% mono  
% nf  
% nfl all  
% nj  
% prop  
% rm  
% spr  
% followon

FIGURES

You can include a figure from any Wisconsin Package graphics program within a Red document. This section tells you how.

Setting the Package to Plot With PostScript

First set up the Wisconsin Package to create graphics using PostScript with the SetPlot utility or a command like

% postscript LaserWriter LPrintPort
Creating a File of PostScript Instructions

Run any Wisconsin Package graphics program with the command-line parameter -PSINClude=myfile.ps. The program creates a file (in this case called myfile.ps) with your plot described using PostScript instructions. This file is created instead of the plot that the program would normally create.

Including the PostScript Instruction File Within Your Document

Within your Red input file, insert commands that tell Red to include the file of PostScript instructions you have just created. First test the page to see if there is enough room for your figure and its heading and/or legend, then insert any heading you want. Next, move the carriage down to position the figure and include the PostScript instruction file with the .PSInclude command. Last, move the carriage down so that the legend or the text following the figure prints below the figure. The source file

```
.sd 1
.tp 29
.scale 1.4
.c
Figure 1
.s 14
.PSInclude myfile.ps 1.0
.s 14
```

creates a figure that looks like this
Red rotates the plot (from landscape to portrait), scales it down, and moves the plot up or down so that it is centered at the current carriage position between the current left and right margins.

**Positioning the Figure Horizontally**

You can adjust the left-right position (if the original figure was not perfectly centered) by resetting the left and right margins before the `.PSInclude` statement. (Remember to reset the margins after the `.PSInclude` command.)

**Positioning the Figure Vertically**

You always have to adjust the vertical position of a figure and the text following it with two separate skip (.s) commands. The skip command preceding the `.PSInclude` command positions the plot on the page (by moving the carriage to the middle of the plot). The skip command following the `.PSInclude` command, moves the carriage down so that the legend and the rest of the document continues below the figure.

Start out by skipping 14.0 lines, then issue the `.PSInclude` command and end by skipping 14.0 more lines. Then continue the text of the rest of the document. When you print the first draft, if the figure is too low, skip fewer lines initially; if it is too high, skip more. If the text following the figure collides with the figure, skip more lines after the `.PSInclude` command.

**Scaling the Figure**

The number after the file specification in the `.PSInclude` command is a scaling factor that you can set to some number other than 1.0 if you want to expand or contract the plot.

**Full Size Figures**

GCG graphics usually occupy the whole page in landscape orientation. The `.PSInclude` command rotates them 90 degrees to portrait orientation, moves them up or down to line up with the current vertical carriage position, and reduces their linear size to about 70 percent of the original in each dimension. If you want to maintain the original orientation, scale, and position of the plot, add the word `exact` after the file specification on the same line as the `.PSInclude` command. If you want to maintain the original scale and position of the plot, but rotate it 180 degrees on the page, add the word `rotate` after the file specification on the same line as the `.PSInclude` command.

**DRIVERS**

You can print many different files together in sequence by making up a `driver` file that contains nothing but `.Include` statements. Drivers are a good way to create consistent pagination across a large document. Here is the driver file we use to print the Appendices to the Program Manual:
ACKNOWLEDGEMENTS

Red was written by Philip Delaquess and John Devereux.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `--CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% red [-INfile=]program_manual:red.red` -Default

Promoted Parameters:

- **-BEGIN=1 -END=9999** sets the page range of interest

Local Data Files: None

Optional Parameters:

- **-COPies=2** prints more than one copy
- **-MONitor** shows what page is printing on your terminal screen
- **-TRAY=1** prints document from paper in tray number one
- **-LETter** prints first page from tray 1, the rest from tray 2
- **-DOUBLE** prints two pages on each LaserWriter sheet
- **-LANDscape** rotates text to long dimension of 8.5 x 11 inch paper
- **-A4** moves text field to center text on A4 paper
- **-UP=2.0** moves the whole text field up 2.0/6 inches
- **-OVER=2.0** moves text field to the right 2.0/12 inches
- **-NOJUSTify** doesn't justify
- **-RMLimit=75.0** limits right margin to 75
- **-LMLimit=5.0** limits left margin to 5
- **-NOPSINClude** makes RED ignore all .PSSInclude commands
- **-NOEPSINClude** makes RED ignore all .EPSInclude commands
-OUTfile=temp.ps directs the output to a PostScript file
-BOLDtoupper changes characters within bold flags to uppercase
-CARDimage=filename writes line-by-line text file instead of PostScript
-ASCII=filename writes a formatted text file instead of PostScript
-HTML=filename writes an HTML file instead of PostScript
-MEDIA sets -BEGIN and -END to refer to pages formatted
-CTRLD adds <Ctrl>D at the end of the output

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-COPies=2

makes Red print extra copies of your document.

-MONitor

shows the page that is printing.

-TRAY=1

sets Red to use a paper tray other than the default tray. On PostScript printers, such as the LZR1200 series from Dataproducts Corporation, the paper trays are numbered as follows: tray 0 is the main cassette, tray 1 is the upper cassette of the multi-cassette sheet feeder, tray 2 is the lower cassette of the multi-cassette sheet feeder, and tray 3 is the envelope feeder.

-LETter

sets Red to print the first page using the paper in tray 1 (see -TRAY) and print all subsequent pages using tray 2. If numbering is enabled, the running titles appear in the upper left with the name of the addressee, the date, and page shown on three separate lines. You should set the top margin with a command like .tm 6 6 in order to leave enough room. Red finds out the name of the addressee in one of two ways: 1) you put the name of the addressee on the first line, following a comment line that looks like .address; and 2) you define the name of the addressee with the title command (.t).
-DOUBLE

prints two pages on each sheet of paper.

-LANDscape

rotates the text to the vertical (long) dimension of the paper.

-A4

moves all left margins to the left 9/72 inch and raises all top and bottom margins up by 24/72 inch. This command will center documents on A4 paper without changing their pagination or filling in any way.

-UP=2

raises both the top and bottom margins up by 2/6 of an inch. (1/6 of an inch is the standard unit for the top margin and page size settings.)

-OVER=2

moves both the left and right margins to the right by 2/12 of an inch. (1/12 of an inch is the standard unit for left and right margin settings.)

-NOJUSTify

never justify regardless of what appears in the document. (See .j.)

-RMLimit=70

limit the right margin to never be more than 70 regardless of what appears in the document. (See .rm.)

-LMLimit=5

limit the left margin to never be more than five regardless of what appears in the document. (See .lm.)

-NOPSINclude

makes Red ignore all .PSInclude commands. Included PostScript plots can take a long time to print. You can use this parameter when you want to check the layout and do not need to see the included plots.

-NOEPSINclude

makes Red ignore all .EPSInclude commands. Included encapsulated PostScript plots can take a long time to print. You can use this parameter when you want to check the layout and do not need to see the included plots.
-OUT file=LPrintPort2

lets you direct output to a different port, queue, or disk file. Usually, Red directs its output to a port or queue to which the logical name LPrintPort has been assigned.

-BOLD toupper

changes all the alphabetic characters between the bold flags (^* and \*) to uppercase. Where alternative fonts are not available, this parameter adds emphasis where the original document called for a bold font.

-CARD image=filename

makes a text file instead of writing output to the printer. We use this parameter to quickly make files that show the paging that would have occurred on the printer. The lines are not justified in the card image, but each token appears on the line where it occurs in the fully formatted document.

-ASCII image=filename

makes a text file instead of writing output to a PostScript printer. We use this parameter to make formatted ASCII files for the Wisconsin Package on-line help documents. Some features of Red, such as font modulation, font scaling, page size, titles, subtitles, and numbering are not supported for ASCII output. The margins are absolute character columns; for example, .lm 0 starts printing in column 1 and .rm 75 puts the last character in column 75. Justification is simulated by randomly adding whole spaces between the words on each justified line. Superscripting and subscripting are supported by putting parentheses around offset material. Bold can be supported with the parameter -BOLDtoupper.

-HTML image=filename

makes an HTML-formatted file instead of writing output to a PostScript printer. We use this parameter to make formatted HTML files for the Wisconsin Package on-line help documents. Some features of Red, such as font modulation, font scaling, page size, titles, subtitles, and numbering are not supported nor relevant for HTML output. Superscripting and subscripting are supported by putting parentheses around offset material, and by using the appropriate HTML flag (this is in order to allow text-based HTML viewers to read produced documents).

-MEDIA

When you limit Red to print only part of your document, the first page and last page settings you choose refer to the pages as they are actually numbered. However, in a complex document the numbering may start over several times as it would if there were a title page, a table of contents, and a preface before the main body of a book. Printing pages three to six in such a document will cause Red to start printing as soon as a page number three is found and stop at the first page number six.
With this parameter the first page and last pages you choose refer to the number of pages actually formatted regardless of what pages numbers are printed on them. If Red leaves a blank page so that a new chapter will start on an odd-numbered page, that blank page counts as a face.

WARNING. Do not use this parameter with -CARD or -ASCII parameters.

-CTRLD

sends a <Ctrl>D (^D, ASCII 4) at the end of the stream of PostScript instructions (some terminal servers require this in order to liberate the terminal for use by others).
SPEW

FUNCTION

Spew sends a GCG sequence from the computer that runs the Wisconsin Package to a personal computer acting as a terminal.

DESCRIPTION

Spew identifies a sequence file that you want to transfer to your microcomputer, then it waits until you type <Ctrl>F before it sends the sequence data from your file to your terminal. Before Spew sends the characters, you must open the receiving file on the microcomputer by means of the microcomputer's communications software. Spew sends the sequence as a stream of characters with no characters other than the sequence itself; it does not send the sequence header or documentation. Use the command-line parameter -carriage to have carriage-return characters inserted at a user-defined interval.

EXAMPLE

To send the sequence in the file zein.seq to a waiting microcomputer:

% spew
SPEW out what sequence file ? zein.seq
Type a <Ctrl>F now
TCGCACATATTATTGAGACCAACTAGCAACATAGAAAGCACAATATTGTA...
%

OUTPUT

When you type <Ctrl>F, the sequence in zein.seq is displayed on your terminal.

INPUT FILES

Spew accepts a single nucleotide or protein sequence as input.

RELATED PROGRAMS

GetSeq sends sequences from a microcomputer terminal to the computer that runs the Wisconsin Package (TM).

RESTRICTIONS

The sequence is transferred without any carriage control and completely without documentation, numbering, or comments.
When the transfer is complete, the program sends a carriage return and a % prompt to the terminal. If you want, you can also have some end-of-transfer character sent.

DEVICES REQUIRED

A microcomputer running communications software that can receive files with no carriage control is all that you need.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % spew [-INfile=]zein.seq -Default

Prompted Parameters:

Ctrl-F     starts the transmission

Local Data Files: None

Optional Parameters:

-STARtchar=4   sets <Ctrl>F to be the character that starts transmission
-ENDchar=4     puts <Ctrl>D at the end of the transmission
-WAIt=1.0      waits one second after the transmission
-CARRiage=50   puts a carriage return and line feed every 50 characters

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-STARtchar=6

By default, the program sends the sequence after you type <Ctrl>F. Use the -STARtchar parameter to change this behavior. The -STARtchar value is the decimal value of the ASCII character you want to type before transmission. The number may not be 3 since that character (<Ctrl>C) is the program interrupt character on UNIX.
-ENDchar=4

Usually, the program sends no character to imply end of transmission. If you use this parameter, it sends a character, <Ctrl>D (or ^D) in this example, at the end of the transmission. The -ENDchar value is the decimal value of the ASCII character to be sent. If a waiting period is used, the end-of-transmission character is sent before the waiting period begins. If carriage control is used, the end-of-transmission character is sent as the only character on the last record.

-WAIT=1.0

Usually, the program does not wait at the end of transmission. This parameter lets you set a waiting period of up to 15 seconds after transmission is complete. The number (1.0, in this case) is the number of seconds to wait after the sequence is completely sent.

-CARRiage=50

inserts a carriage return and line feed every 50 characters. If you use carriage control, the carriage returns cannot be more than 512 sequence characters apart. The carriage control comes before the end-of-transmission character, if one is used.
**SHIFTOVER**

**FUNCTION**

ShiftOver moves a file to the right or to the left as many columns as you specify.

**DESCRIPTION**

What more can we say? You have text that you need to move to the right or left, and we provide the tool to do that. If you do not want to move the whole file, write out the part you want to shift as a temporary file, shift it over, and then include it in the original file.

**EXAMPLE**

Here is a session using ShiftOver to move the contents of the file gamma.seq over to the left so that the sequence data starts in the first column:

```bash
% shiftover

SHIFTOVER what file ? gamma.seq

How many columns should "gamma.seq" be shifted (* -4 *) ? -10

What should I call the output file (* gamma.seq *) ? temp.txt

495 lines shifted -10 columns

%
```

**OUTPUT**

Here is part of the output file (notice that the heading has been damaged by the shift):

```
NCE 1.0
1 beta globins G and A gamma
  Slightom and Smithies, Cell 26; 191-203.
y Smithies et al. Cell 26; 345-353.

Length: 11375  June 27, 1994 10:09  Type: N  Check: 6474 ..

GGATCCTAGA TATTCCTTAG TCTGAGGAGG AGCAATTAAG ATTCACTTGT
TTAGAGGTGG GAGTTGGGCT CTCAGCGGAA TTACCCAGA ATTTTGGGAG

/---------------------------------------------

CCAGGAAAGT GACTGCAGGT CACTTTTCCT GGAGCGGGTG AGAGAAAAGT
```
RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

Unknown.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % shiftover [-INfile=]gamma.seq -Default

Prompted Parameters:

-Shift=-4  shifts file contents four columns to the left
[-OUTfile=]gamma.seq  names the output file

Local Data Files: None

Optional Parameters: None

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-Shift=-4

specifies the direction of the shift and the distance (number of columns) that the file contents are to be moved. The minus (-) sign designates a leftward shift.

Printed: January 9, 2002 13:45 (1162)
SETKEYS

FUNCTION

SetKeys writes a file in your current directory that redefines your keyboard's keys for easier sequence entry with the SeqEd, LineUp, GelEnter and GelAssemble programs and the SeqLab sequence editor. The output file, called set.keys, can be edited if you want to redefine keys that were not considered by the SetKeys program.

DESCRIPTION

The characters G, A, T, and C are case-shifted keys in three different rows under the left hand. It is difficult to think of a less favorable arrangement for keys that have to be typed quickly and repeatedly. SetKeys addresses this problem by letting you redefine keys to ease nucleic acid sequence entry with SeqEd, LineUp, GelEnter, GelAssemble and the SeqLab sequence editor. SetKeys creates a file named set.keys. It lists nucleotide and ambiguity characters and the keys you wish to use to define them.

Select a key in response to the program prompt for the nucleotides you wish to redefine. If you do not want to redefine a symbol, just answer with <Return>. You can define several keys to have the same meaning.

EXAMPLE

Here is a sample session with SetKeys:

```
% setkeys

Choose key(s) for each nucleotide:

What key(s) should mean G ? j
What key(s) should mean A ? k
What key(s) should mean T ? l
What key(s) should mean C ? ;

Now choose key(s) for the common ambiguity codes:

What key(s) should mean R ? u
What key(s) should mean Y ? i
What key(s) should mean N ? o
What key(s) should mean <Delete> ? p

SetKeys complete: output file is "set.keys"

%
```

OUTPUT

Here is the file set.keys that would be written into your current directory from the session above:
SETKEYS output file for initializing SEQED, LINEUP, GELENTER, GELASSEMBLE, and the SEQLAB editor

September 11, 1996 09:20

-change j into G
-change k into A
-change l into T
-change ; into C
-change u into R
-change i into Y
-change o into N
-change p into ~^?

RELATED PROGRAMS

SeqEd, LineUp, GelEnter, GelAssemble, and the SeqLab sequence editor read a file named set.keys if present in your current directory. You can modify this file with a text editor to redefine any GCG accepted sequence character to another character or SeqEd or LineUp control key. EchoKey is an undocumented tool that shows you each terminal key's decimal value and printing character representation. Use the printing character representation from EchoKey to define the key. Use "~^?" to represent the <Delete> key.

RESTRICTIONS

If you have a file named set.keys in your current directory, then SeqEd, LineUp, GelEnter, and GelAssemble only respond to the keys that you have defined in it; all other keys will be silent. In the SeqLab sequence editor, keys that are not defined in the set.keys file retain their normal definition.

SeqEd, LineUp, GelEnter, and GelAssemble only recognize the GCG sequence character set (see Appendix III). You can also relocate the <Delete> key if you wish.

SUGGESTIONS

Use a text editor to modify set.keys to your own needs.

COMMAND-LINE SUMMARY

Complete command-line control is not available for this program.

LOCAL DATA FILES

None.

PARAMETER REFERENCE

None.
SEQED

FUNCTION

SeqEd is an interactive editor for entering and modifying sequences and for assembling parts of existing sequences into new genetic constructs. You can enter sequences from the keyboard or from a digitizer.

DESCRIPTION

SeqEd uses the screen of your terminal as a window into a sequence. It works like the UNIX vi text editor. Changes you make in the sequence take place at the cursor position and are reflected immediately on the screen. You can insert or delete symbols, move the cursor, search for patterns, check sequences by reentering them, and edit documentation and embedded comments.

SetKeys lets you change the positions of the keys on your terminal keyboard to make it more convenient to enter the letters G, A, T, and C.

You can enter a sequence and control SeqEd either from the terminal keyboard or from a Graf/Bar digitizer.

EXAMPLE

If you are already familiar with the UNIX vi editor, you can learn to use SeqEd quickly. When you run SeqEd with a command like `% seqed sample.seq`, your screen will look something like this:

```
sample.seq            ***** K E Y B O A R D *****
seqed
  : Some documentary text about your sequence can be placed
  : here in the heading.
  :
  : You can have as many lines of header comments as you wish, and
  :
  : you can edit them in this space with the HEAding command.
  :

  1 $The scale below has extra dots where comments occur$
  9 <The number indicates which symbol the comment precedes.$
  20 $There can be as many comments as you like$
  28 $The four comments closest to the cursor are displayed$

AGTCTTAGTCGATCGTAcTGCATRCGA

....|:.........|:.........................|:................|:................|:................|:.............|
 0   10   20   30   40   50   60   70
```
"sample.seq"  27 nucleotides

EDIT NEW OR EXISTING SEQUENCES

If you name a sequence file that already exists, SeqEd displays the first four lines of documentation on the top of the screen followed by up to four embedded comments and the base number with which each is associated. SeqEd shows the end of the sequence across the middle of the screen.

If the sequence you name does not exist, SeqEd starts in Heading Mode (see below) to allow you to enter documentation for the new sequence. Use <Ctrl>D to stop editing the documentation.

SCREEN MODE

Entering a Sequence

In Screen Mode the cursor shows your position in the sequence. You can move around in the sequence, add symbols, delete symbols, and search for patterns. You can insert any valid GCG sequence symbol (see Appendix III) into the sequence by typing the symbol. It is inserted at the cursor.

Deleting a Sequence

The <Delete> key and <Ctrl>H delete the symbols to the left of the cursor, one by one.

Moving the Cursor

To move the cursor to the right, use the <Right-arrow> key; to move to the left, use the <Left-arrow> key. Movements are confined to the length of the sequence.

If you type a number followed by <Return>, the cursor moves to that sequence position.

The arrow keys can be preceded by a number indicating how many symbols to move to the left or right. 10<Right-arrow> moves 10 symbols to the right.

Finding Patterns

To search for a pattern, type a / (slash) in Screen Mode. The cursor moves to the lower-left corner of the screen to let you enter a sequence pattern that you wish to find. You may type in a pattern up to 40 characters long. You can repeat the last search by simply typing /<Return>. SeqEd treats all nucleic acid sequences as circular and finds your pattern even if it wraps from the end of the sequence into the beginning. SeqEd uses the same rules for pattern definition and recognition as the programs FindPatterns, MapPlot, Map, and MapSort.
The command-line parameters `-PROtein` and `-PERFect` or the `PROtein` or `PERFect` commands in Command Mode make SeqEd treat the sequence as linear and disable the nucleic acid ambiguity meanings of the GCG sequence symbols (see Appendix III) during pattern searches.

The `NUCleotide` command in Command Mode tells SeqEd to recognize patterns containing IUB nucleotide ambiguity symbols during searches.

Even if SeqEd thinks your sequence is nucleotide, you can request a perfect-match search by typing `=` after the `/`. For example, `/=RCT` only matches RCT (case does not matter) no matter which kind of sequence SeqEd thinks you have.

**Finding a Marked Position**

You can mark a position in a sequence to which you wish to return. You give the marked position a letter (like giving it a name) using the Command Mode `Mark` command (see below). Then, in Screen Mode, a single quote followed by the letter used to mark the sequence moves the cursor to the position where that mark was defined.

**Leaving Screen Mode**

Use `<Ctrl>D` to leave Screen Mode and enter Command Mode.

**Screen Mode Summary**

Here is the summary of Screen Mode commands in the on-line help:

```
Screen Mode

[n] is an optional numeric parameter.

G, A, T, . . .   - insert a sequence character
<Delete>         - delete a sequence character
<Ctrl>H          - delete a sequence character
/TAACG<Return>   - find the next occurrence of TAACG (last pattern entered is the default)
1<Return>        - move to start of the sequence
<Ctrl>E          - move to end of the sequence
[n]<Right-arrow> - go ahead n characters
[n]<Left-arrow>  - go back n characters
<Up-arrow>       - go up to check sequence
<Down-arrow>     - go down to original sequence
'markcharacter   - go to marked position
37<Return>       - go to position 37 (any positive integer)
<  - go back 50 characters
>  - go ahead 50 characters
<Ctrl>R           - redraw the screen
<Ctrl>D           - enter command mode
```
COMMAND MODE

Use <Ctrl>D in Screen Mode to enter Command Mode. The cursor moves down to the lower-left corner of the screen next to a colon prompt after which you can enter any of the commands shown below followed by <Return>.

Editing SeqEd Commands

SeqEd command editing is modeled on OpenVMS DCL command-line editing. The <Left-arrow> and <Right-arrow> keys let you move your cursor around in a command that you have typed so you can insert or delete characters at any position. <Ctrl>E moves the cursor to the end of the line. <Ctrl>U deletes all the characters from the current cursor position to the start of the line.

Editing Previous SeqEd Commands

SeqEd lets you modify and execute previous commands. The <Up-arrow> key displays previous commands.

Returning to Screen Mode

If you press <Return> without entering a command, SeqEd returns to Screen Mode (described above). If you have -SINGlecommand on the command line or in your command-line initializing file, SeqEd returns to Screen Mode immediately after executing each command.

Commands May Be Shortened

Only the capitalized portion of the commands described in the documentation below needs to be typed.

Parameters are Used with Commands

Some commands can be preceded with numeric parameters or succeeded with a file name. The square brackets ([ and ]) in the documentation below show command parameters that are optional, meaning you can leave them out.

Command Mode Summary

Here is the summary of Command Mode commands you would see with the Help command:

Command Mode

Commands end with <Return>. [n] indicates an optional parameter. s and f are numbers for start and finish of a range of interest. Only the capitalized part of the command is necessary.

EDit seqname       - get a new sequence file to edit
[n] Include [seqname] - insert another sequence [at position n]
   (SeqEd prompts for range and strand)
s,f Delete - delete a range of bases
[s] Check [/Blind] - check a range of bases [beginning at s]
               37 - go to base 37
   REDraw - redraw the screen
[n] COmment comment - insert a comment [at position n]
[n] COmment - enter comment editing mode [at position n]
[n] HEAding - edit documentary heading [at line n]
   change - enter screen mode (<Return> is sufficient)
   screen - enter screen mode (<Return> is sufficient)
   OVERstrike - enter overstrike mode
   INSert - enter insert mode
[n] Mark markcharacter - mark the sequence [at position n]
   PERFect - require finds to be perfect matches
   PROtein - set sequence type to PROTEIN
   NUCleotide - set sequence type to NUCLEOTIDE
[s,f] Write [seqname] - write [a part of] the sequence to a file
   DIGitizer - enter digitizer mode
   RELoad - enter reload mode
   ACCept - terminate reload mode
   Help - show commands in screen and command modes
[s,f] EXit [seqname] - write [a part of] the sequence and quit
   Quit - quit the editor without writing the sequence

EDit SeqName

gets a new sequence from the file you have named for editing with SeqEd. The sequence you are currently editing is lost if you have not written it out before using the EDit command.

[s] Include [filename]

includes another sequence within the sequence being edited at the current cursor position or at the position specified by the optional parameter. SeqEd creates two embedded comments at the start and end of the included section to show what was included. If you do not supply a file name with this command, SeqEd prompts you for one.

s,f Delete

deletes some or all of the sequence. You must specify a beginning and ending coordinate for the range of symbols you want to delete.
[s] **C**heck [/Blind]

lets you check a sequence entry in Screen Mode. A sequence already entered may be
typed in again. If a symbol is typed that disagrees with the first entry, a ^ is printed at
the point of disagreement and the terminal bell rings. While checking, the <Up-arrow>
and <Down-arrow> keys move the cursor back and forth between the second entry and
the original sequence, allowing you to make changes in either one as mistakes are
found. If the optional starting coordinate precedes the command, it specifies where
checking begins. If you wish to check your sequence without seeing the original
version, type /Blind following the C**heck** command (there must be a blank between
the C**heck** command and the /).

**REDraw**

redraws your terminal screen. This is useful if noise in the line between your terminal
and the computer has changed the screen in some unreasonable way or if a system
message appears on your screen.

[s] **C**omment [comment text]

allows you to enter, delete or modify embedded comments to document your sequence.
In its simplest use, the C**omment** command lets you insert new comments. You simply
type the entire comment on the command line. Deletion and modification of existing
comments is handled by entering Comment Mode. To do this, you type only the
C**omment** command and optional position but no comment text. See the COMMENT
MODE topic for more information.

Whenever you enter a comment, SeqEd ensures that comment-delimiting characters are
placed around it. A $, <, or > must appear at each end of, and not within, your
comment. (SeqEd deletes comment delimiting characters found within a comment when
they are the same as the flanking comment delimiting characters.)

SeqEd inserts new comments at your current cursor position or at the position specified
by the optional position number and then returns to Command Mode.

[s] **H**ead

enters Heading Mode, which lets you edit the documentary heading. You can modify
any part of the heading. Heading Mode is terminated with <Ctrl>D. The optional
parameter specifies which line of the heading you want to start editing.

**change**

returns your session to Screen Mode. Note that the entire command is optional and a
simple <Return> is equivalent.

**screen**

returns your session to Screen Mode. Note that the entire command is optional and a
simple <Return> is equivalent.
OVERstrike

enters overstrike mode. Typing in a new symbol deletes the old symbol at that position and replaces it with the new symbol.

INSert

enters insert mode. Typing in a new symbol shifts all symbols from the current position to the end of the sequence by one position to the right and adds a new symbol at the current position.

[n] Mark markcharacter

You can mark a position in the sequence if you wish to return to it later. If the optional position number is absent, the position marked is the current cursor position. You give the marked position a letter (like giving it a name) using this command. Then, in Screen Mode, a single quote followed by the letter used to mark the sequence moves the cursor to the position where that mark was defined.

PERFect

makes searches linear and disables the nucleic acid ambiguity meanings of the GCG sequence symbols (see Appendix III).

PROtein

sets the sequence type to protein. This makes searches linear and disables the nucleic acid ambiguity meanings of the GCG sequence symbols. This also makes SeqEd ignore any set.keys file in your local directory.

NUCleotide

sets the sequence type to nucleotide. This makes searches circular and tells SeqEd to recognize patterns containing IUB nucleotide ambiguity symbols. SeqEd also remaps the keys if a set.keys file is in your local directory.

[s,f] Write [filename]

writes the current form of the sequence into a file. If you supply starting and finishing coordinates, SeqEd only writes the indicated segment. For example, 1,56 Write would write symbols 1 to 56 into a file. If you name a file, SeqEd writes the sequence into a file with that name instead of into the input file.

DIGitizer [G,A,C,T] [*]

enters Digitizer Mode. The lane order from left to right may be redefined. See below for a detailed discussion of digitized input.

The DIGitizer command has two options.
You can specify the left-to-right order of your lanes. The default order is alphabetic (A,C,G,T). You can change the default order with the command-line parameter \texttt{-LANes=T,A,G,C} or with the SeqEd command \texttt{DIGitizer G,T,C,A}.

Maxam and Gilbert sequencers find it useful to use \texttt{-LANes=G,AG,TC,C}. This order is equivalent to G,A,T,C because SeqEd uses the first of multiple letters when assigning a base to a lane. However, specifying pairs of letters causes the screen display of your gel to look more like the autoradiograph.

The second parameter with the \texttt{DIGitizer} command is a simple asterisk (*). This suppresses prompting for lane and menu positions. If you have been digitizing some lanes and return to the keyboard for some reason, you can go back to digitizing the same lanes without having to redefine their positions.

\texttt{RELoad}

go\s to \texttt{Reload Mode, which is similar to Checking Mode, except that the reloaded sequence grows leftwards from the right end of the main sequence. This is designed to help find the overlap of two loadings of the same reaction. Mismatched bases are marked with \texttt{^} (caret) characters, as in Checking Mode. Also, you can use the arrow keys to move around in and edit either the main sequence or the reloaded sequence. When the match becomes especially good, the terminal bell rings. You are free to accept or reject SeqEd's rules of what constitutes a good overlap. See the \texttt{COMMAND LINE SUMMARY} topic below for more information.

\texttt{ACCept}

terminates Reload Mode. The display of the reloaded sequence goes away, leaving you with only the main sequence with the cursor at the end, ready for more input. SeqEd helps you to decide when to \texttt{ACCept} an overlap, but the decision is yours.

\texttt{Help}

shows the commands available in the Screen and Command Modes of SeqEd.

\texttt{[s,f] EXit [filename]}

works exactly like \texttt{Write} except that your session with SeqEd ends after the sequence is written out into a new sequence file.

\texttt{Quit}

terminates a session with SeqEd without writing a new sequence file.

**DIGITIZER MODE**

This part explains how to use the the Graf/Bar GP-7 Sonic Digitizer with SeqEd for sequence entry.
Preparation

We use the Graf/Bar GP-7 Sonic Digitizer with stylus S-7 formerly manufactured by the Science Accessories Corporation which was purchased by GTCO, 7125 Riverwood Dr., Columbia, MD 21046, Phone: (410) 381-6688, Toll-free: (800) 344-4723, URL: http://www.gtco.com. Here is how to set up the GP-7 to work with SeqEd:

Baud Rate

Remove the cover, find the DIP switches, and set the baud rate and parity according to the instructions in the Graf/Bar Operator’s Manual. The parity and the baud rate must match your modem or terminal settings.

Special Cable

We use a special cable when connecting the Graf/Bar to the system, whether it is connected directly or through a data splitter. Build a cable between two 25-pin RS-232 connectors as follows:

<table>
<thead>
<tr>
<th>Graf/Bar pin</th>
<th>Computer pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Connecting the Digitizer to Your Terminal

We connect our Graf/Bar through a modem data splitter to the same line that we use for the terminal. The modem data splitter we use is model 232MDS from B & B Electronics, 1500 Boyce Memorial Drive, Ottawa, Illinois, 61350, USA, Phone: (815) 434-0846

Connecting the Digitizer to Another UNIX Port

The program reads input from the digitizer through a line whose logical name is DigitizerPort. This is assigned to TERM unless you (or your system manager) assign it somewhere else. If you have a port other than your terminal available, put a command like:

```
% name -s DigitizerPort /dev/tty11
```

in your login file. (Such commands are usually made at login time so you do not have to remember them every time you want to use the digitizer.)

Layout

See the figure at the back of this document to get an overview of how to lay out the digitizer.
The Graf/Bar digitizes an active area about 24 inches wide and 18 inches deep, starting 2 inches from the front of the unit. The region of interest in your radiograph must lie entirely within this active area, with the gel lanes perpendicular to the face of the digitizer. The menu must also fit in this area, with its long axis perpendicular to the digitizer. Tape or clamp the radiograph and the menu tightly so that their positions with respect to the digitizer do not change.

Menu

The digitizer controls SeqEd through a printed menu, which you place in your work area next to the radiograph. Cut out one of menus in the figures below or Fetch the file seqedmenu.fig and use the Figure program to draw more copies. The menu may be drawn to whatever absolute size you like, as long as it lies within the active area of the digitizer.

Entering Digitizer Mode

The DIGitizer command switches SeqEd into Digitizer Mode. All input comes from the digitizer stylus in this mode. After you issue this command, SeqEd asks you to define your gel lanes (described below) and to locate opposite corners of the menu. The KEYBOARD key on the menu turns off Digitizer Mode and returns control to the keyboard. A banner across the top of your screen reminds you whether the keyboard or the digitizer is in control. In Digitizer Mode, keyboard input is ignored and in Keyboard Mode, digitizer input is ignored.

Defining Gel Lanes

The basic idea is this: click (with the stylus) in the middle of each lane where you begin (or resume) reading the gel. See the figure below. All of the points should be in the center of the lane and the stylus must always be held at the same angle and orientation. If the program is uncertain of which base to assign a digitized position, it beeps twice and asks you to redefine your lanes. The program uses the command-line value of tolerance as the basis for certainty. A tolerance of 0 is the least tolerant setting and the slightest deviation would require you to redefine your lanes. A tolerance of 1.0 is the most tolerant setting, such that any deviation is accepted. Based on our limited experience, you should not use a tolerance value less than 0.25 or greater than 0.6. The default value (0.4) was chosen because it has seldom made an incorrect assignment and does not require you to redefine the lanes too frequently. The algorithm employed is that of Staden (Nucl. Acids Res., 14, 217 (1986)).

Remember, when answering the prompts for lane position, digitize the centers of the lanes near the first bands you intend to read.

Defining Menu Position

After clicking on the fourth (right-most) lane or your radiograph, you are prompted to click on two opposite corners of the menu.
Using Digitizer Mode

The digitizer replaces the keyboard in Digitizer Mode. You now control all of SeqEd's functions by clicking on the "keys" in the menu with the digitizer stylus. When you click on the KEYBOARD key, control is returned to the keyboard.

Entering a Sequence

Enter bases into the sequence by clicking the bands on the radiograph. SeqEd behaves exactly as though you were typing uppercase letters. You can also click the menu keys labeled A, C, G and T for exactly the same effect. The menu also includes ambiguity codes R, X, and Y, which are entered into your sequence in lowercase. The question mark is like a shift key -- the next base you enter from the gel lanes after clicking the question mark is entered in lowercase.

Deleting a Sequence

Remove bases from the sequence by clicking the DELETE key on the menu. This behaves just like the <Delete> key, removing the base to the left of the cursor.

Moving the Cursor

Move around in your sequence by clicking on the left and right arrow keys. You can click on numbers first in order to move more than one base at a time. You can click on the numbers and then click the RETURN key to move the cursor to any position. The BEGIN and END keys jump to the ends of the sequence. The FIND key lets you move to the beginning of a pattern. You can enter the pattern by clicking the letter keys on the menu ending with a click on the RETURN key.

Loading Second Fragment

RELOAD and ACCEPT function together to help you find the overlap of two loadings of the same sequence. RELOAD prompts you to define new lane positions. (It is not necessary to redefine the menu position.) See the documentation on RELOAD and ACCEPT under Command Mode above.

Checking Entered Sequence

CHECK puts you in Checking Mode so you can re-enter the data and increase your confidence. Use the up and down arrows to move between the main sequence and the check sequence.

Writing Sequences Into Files

WRITE FILE writes your sequence into a file without leaving Digitizer Mode.

Redraw Screen
FRESH SCREEN refreshes your terminal screen if line noise or broadcast messages have interrupted it.

New Lanes

SeqEd should handle slight bends and bulges in your lanes. However, if your bands are not being recorded correctly, it may be necessary to delete the incorrect bases and redefine your lane positions starting with a click on the NEW LANES key.

You can also use this key when you have finished one set of lanes and wish to start another without returning to Keyboard Mode.

Return to Keyboard Control

The KEYBOARD key returns control to the terminal keyboard. You cannot use the digitizer again until you use SeqEd's \texttt{DIG}itizer command.

Help

The HELP key displays the Digitizer Mode help message (below) on your terminal screen. The next click you make, whatever it is, redraws your screen the way it was before the help screen was displayed.

\begin{verbatim}
Digitizer Mode

37<Return> - go to position 37 (any positive integer)
<Up-arrow> - go up to check sequence (Checking Mode)
<Down-arrow> - go down to original sequence (Checking Mode)
[n]<Right-arrow> - go ahead [n] characters
[n]<Left-arrow> - go back [n] characters
FIND TAACG<Return> - find next occurrence of TAACG
Begin - go to beginning of sequence
End - go to end of sequence
A C G T R X Y - insert A C G ... into sequence (at cursor)
? - enter next sequence character in lowercase
<Delete> - delete a sequence character
New Lanes - define new lanes (another loading) for this fragment
Reload - define new lanes and slide new sequence back over old until overlap is "Accepted"
Accept - accept overlap from "Reload"
Check - enter Checking Mode to re-enter and verify sequence
Write File - write out fragment into a file
Fresh Screen - redraw the terminal screen
Keyboard - return control to terminal keyboard

Click the digitizer anywhere to go back to SEQED:
\end{verbatim}
COMMENT MODE

Comment Mode allows you to add, change, or delete embedded comments and helps you move quickly to any position in your sequence where a comment is associated. To enter Comment Mode, you must first enter Command Mode with <Ctrl>D.

Entering New Comments

If you type the comment command without any comment text, SeqEd creates a new, empty comment at the position indicated by the optional sequence position number, if present, or at your current position in the sequence. The cursor moves to the part of the screen where embedded comments are displayed. Initially, the cursor is adjacent to a position number followed by an empty comment. You may then type a new comment or move to an existing comment that you wish to modify. Only one new comment can be created each time you enter Comment Mode.

Cursor Movement in Comment Mode

While in Comment Mode you can move around in the comment using the <Left-arrow> and <Right-arrow> keys, insert text by typing, or delete text using the <Delete> key or <Ctrl>H. <Ctrl>E positions the cursor at the end of the comment. <Ctrl>U deletes all characters from the beginning of the comment to the cursor position. You can move from one comment to another using the <Up-arrow> or <Down-arrow> keys.

Deleting Comments

When you move the cursor off of a comment that is empty, the comment is deleted. You can delete a comment by entering Comment Mode, moving to the end of the comment you wish to delete, and using <Ctrl>U. When you move to another comment or leave Comment Mode, the comment disappears. Likewise, the empty comment created when you enter Comment Mode is deleted if you do not type anything at the new comment position.

Comment Delimiters

Comments must start and end with one of the characters <, >, or $. A comment must start and end with the same delimiting character. If you try to move your cursor off of a comment that does not have one of these characters at the ends, or if the delimiters aren't identical, then SeqEd makes sure the delimiters are corrected.

Changing Sequence Position With Comment Mode

As you move to different comments, your position in the sequence in Screen Mode changes to the symbol with which that comment is associated. This allows you to move quickly to any symbol with which a comment is associated when you leave Comment Mode. By marking your place with a comment at the end of one session with SeqEd, you can easily restore your place at the next session.
Leaving Comment Mode

To exit Comment Mode, press <Return> or use <Ctrl>D.

Comments Are Associated With Sequence Symbols

Comments may be associated with any base. They stay with that base, even though the base's position may change, unless the base is deleted (see below). They can also be associated with either end of the sequence. For example, you may issue the command, 0 CO to create a comment associated with the left end of the sequence. This comment must be delimited with < (SeqEd makes sure of this). Similarly, a comment can be created at the extreme right of the sequence and must be delimited with > or $.

Comments Can Be Used in Pairs to Bracket Sections of the Sequence.

Comments can document a whole fragment as well as an individual sequence symbol. For example, the Include command automatically puts an identifying comment at each end of the included fragment. The characters < and > were selected as comment delimiters because they imply direction; the comments bracketing the included fragment point at the fragment. A >-comment is associated with the first base of the fragment and a <-comment with the last. When the sequence is saved in a file, all >- and $-comments are written before the base they are associated with and all the <-comments after. This way the bracketing comments surround the entire fragment and point to it.

Between two bases in a sequence file there may be several comments. The <-comments are always associated with the base to the left, the >- and $-comments with the base to the right.

Deleting Comments

The only way to delete a comment is to go into Comment Mode and delete all the characters of the comment. When you move your cursor away from the empty comment, it goes away.

Deleting Bases Associated With Comments

If you delete a base with which a comment is associated, the comments do not go away. They just attach themselves to adjacent bases. To preserve the properties of fragment bracketing comments, the <-comments become associated with the left-hand base, the >- and $-comments with the right-hand base.

HEADING MODE

Heading Mode allows you to edit the documentation that appears above the sequence. When a new sequence is edited, SeqEd goes directly into Heading Mode to let you identify the new sequence.
Entering Heading Mode

SeqEd lets you enter Heading Mode by using the `HEAding` command.

Leaving Heading Mode

Use `<Ctrl>D` to return to Command Mode.

Moving the Cursor

You can move around using the arrow keys and make insertions and deletions as you wish. Although the editing window is only four lines high, it scrolls over the heading vertically to let you see and modify any part. `<Ctrl>E` positions the cursor at the end of the current line.

Editing in Heading Mode

As with many text editors, typing inserts text at the cursor and the `<Delete>` key or `<Ctrl>H` delete characters to the left of the cursor. `<Ctrl>U` deletes everything from the current cursor position to the start of the line; `<Return>` creates a new line starting at the current position in the heading.

SYSTEM CRASH OR HANGUP

While you are editing a sequence, SeqEd records your session in a file called `seqed.log`. This file is automatically deleted when the editor exits normally. If you are accidentally disconnected or the system crashes, your work can be recovered by logging back in, moving to the directory where the crash occurred, and running SeqEd again. SeqEd finds `seqed.log` and restores the sequence to the state it was in just before you were cut off.

If you do not want SeqEd to restore the session, delete the file `seqed.log`.

RESTRICTIONS

The total length of all vector sequences specified with the `{command-line parameter may not exceed 100,000 bases. If the total vector sequence length exceeds 100,000 bases, SeqEd notifies you that only the first 100,000 vector bases will be checked.

SeqEd only works on terminals that can provide screen support. Your system manager may be able to help if your terminal is not behaving correctly.

Many terminal conditions must be set if you are using the Graf/Bar sonic digitizer; see the paragraph on Preparation under the DIGITIZER MODE topic.
ACKNOWLEDGEMENTS

SeqEd was originally designed by Paul Haeberli and implemented for VAX/VMS by Paul Haeberli and John Devereux. It was completely revised for GCG Version 4 by William Winsborough. The digitizer interface and the RELoad command were implemented for Version 5 by Philip Delaquess. We are very grateful for the collaboration of Dr. William Boorstein.

SEQUENCE TYPE

When it opens a new sequence file, SeqEd initially assumes it is nucleic acid. When you write the file, SeqEd examines the sequence to see if it contains only IUB-IUPAC nucleotide symbols in the first 300 symbols. If so, it writes the new sequence as a nucleic acid sequence; if not, it writes it as a peptide sequence.

When it opens a pre-existing GCG sequence file, SeqEd obtains the sequence type (nucleotide or protein) from the Type: field of the dividing line (the line that contains two successive periods). If the Type: field is absent, as in the case of sequence files created prior to Version 7 of the Wisconsin Package, SeqEd infers the type of the sequence from the composition of the sequence characters. When SeqEd writes the edited file, it writes the Type: field according to its current understanding of the sequence type.

It is possible for SeqEd to make a mistake. If the Type: field of an existing file is incorrect, SeqEd will accept the incorrect type; it doesn't check the composition in this case. For files without a Type: field, it is possible for SeqEd to infer the wrong sequence type. For example, a peptide sequence that contains only those amino acids that share IUB-IUPAC symbols with nucleotides will be incorrectly typed as nucleic acid (see Appendix III).

You can override SeqEd's assignment of sequence type in two ways. When you run SeqEd, you can add –PROtein or –NUCleotide to the command line to tell SeqEd which type of sequence it will be editing. Once SeqEd is running, you can use the Command Mode commands PROtein and NUCleotide to force the assignment of sequence type.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use –CHEck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % seqed [-INfile1=]sample.seq

Prompted Parameters: None
Local Data Files:

set.keys (must be in your current working directory to be used)

Optional Parameters:

-SINGLEcommand automatically returns to screen mode after commands
-PROtein sets sequence type to protein, and sets find to search for perfect symbol matches
-NUCLEotide sets sequence type to nucleotide, and sets find to allow nucleotide ambiguity symbol matches
-PERFect sets find to search for perfect symbol matches, even if sequence type is nucleotide
-VECTors=gb:synpbr322 highlights sequences from pBR322
-SITES=gaattc highlight GAATTC patterns
-LANes=a,C,G,T sets the default lane order for digitizer
-MINOverlap=10 sets minimum overlap length for Reload command
-PCTOverlap=95 sets stringency for the Reload command
-TOLerance=0.4 sets tolerance for digitizing ambiguity (0 to 1), with 1 being the most tolerant

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like -DATa1=myfile.dat. For more information see Chapter 4, Using Data Files in the User's Guide.

Customizing Your Keyboard With SetKeys

You can use the program SetKeys to create a set.keys file that tells the SeqEd, GelEnter, LineUp, GelAssemble, and SeqLab sequence editors how to interpret the letters you type at the terminal. When entering gel readings, it is useful to have the symbols for G, A, T, and C under the fingers of one hand in the same positions as the lanes in your gel. SeqEd, GelEnter, LineUp, GelAssemble, and the SeqLab sequence editor automatically read the file set.keys if it is present in your local directory. If set.keys is absent, or if the sequence type is set to Protein (in SeqEd and LineUp only) the terminal keys retain their conventional meanings.

If you have a set.keys file in your directory, SeqEd, GelEnter, LineUp, and GelAssemble only respond to the keys that it redefines. You can edit the file set.keys with a text editor if some of the keys you want to use are not in it. Any keys not mentioned in set.keys appear to be dead in these sequence editors. In the SeqLab sequence editor, keys that are not redefined retain their normal meanings.

Several keys are vital for the control of SeqEd, LineUp, GelEnter, and GelAssemble; this means you are not allowed to redefine the keys for /, [ ], { }, ( ), :, , , 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, <Ctrl>R, <Ctrl>D, <Ctrl>H, <Return>, and <Ctrl>E.
PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-SINGLEcommand

    sets SeqEd to return automatically to Screen Mode after every command in Command Mode.

-PROtein

    sets the sequence type to be protein, and makes pattern searches use perfect symbol matches. SeqEd treats protein sequences as linear and will not find patterns that start at the end and continue into the beginning of the sequence. Furthermore, –PROtein causes SeqEd to ignore any set.keys file in your local directory.

-NUCleotide

    sets the sequence type to be nucleotide, and makes pattern searches use nucleotide ambiguity symbol matches (unless you force the program to use perfect symbol matches by including –PERFect on the command line or by entering the PERFect command in Command Mode.) SeqEd treats nucleotide sequences as circular and will find patterns that start at the end and continue into the beginning of the sequence. Furthermore, –NUCleotide causes SeqEd to use a set.keys file in your local directory.

-PERFect

    makes pattern searches use perfect symbol matches. Normally if you type /GARC in Screen Mode, the patterns GAAC or GAGC could be found. If you have –PERFect on the command line, /GARC would only find the pattern GARC. This also makes SeqEd treat sequences as linear and not find patterns that start at the end and continue into the beginning of the sequence.

-VECTors=GB: synpbr322, GB: m13mp18

    tells SeqEd which cloning vector or vectors are of interest to you. SeqEd checks your sequence against them to make sure you are not entering a vector sequence. If it finds that you are entering vector sequence, the terminal bell rings and the vector sequence characters are highlighted with reverse video.

-SITes=GAATTC, GAnTC

    tells SeqEd to highlight enzyme recognition sites that interest you.
-LANes=A, C, G, T

establishes the default left-to-right order of gel lanes. The default may be over-ridden when you issue a DIGitizer command in Command Mode.

-MINOverlap=10

sets the minimum overlap length regarded as meaningful by the RELoad command. SeqEd ignores matches shorter than this, even if they are perfect. However, you are always free to end a reload with the ACCEPT command.

-PCTOverlap=95

sets the minimum percentage of matching bases regarded as meaningful by the RELoad command. In Reload Mode, when the overlap is long enough and good enough, the terminal bell rings to alert you. Again, you have complete freedom to reject or ACCEPT SeqEd’s opinion.

-Tolerance=0.4

sets the tolerance for digitizing. When digitizing, the program must determine which base lane the sonic pen has touched. Since the gel lane may bend, the program must have some tolerance for deviation. The tolerance value determines how great this deviation can be before you must redefine your lanes. A tolerance of 0 is the least tolerant setting and the slightest deviation would require you to redefine your lanes. A tolerance of 1.0 is the most tolerant setting such that any deviation is accepted. Based on our limited experience, you should not use a tolerance value less than 0.25 or greater than 0.6. The default value (0.4) was chosen because it has seldom made an incorrect assignment and does not require you to redefine the lanes too frequently. The algorithm employed is that of Staden (Nucl. Acids Res., 14: 217 (1986)).

Printed: January 9, 2002 13:45 (1162)
Deprecation Programs: Use SeqConv+ to obtain the same functionality

TOFASTA

FUNCTION

ToFastA converts GCG sequence(s) into FastA format.

DESCRIPTION

Sequence files in GCG format can be converted into a format suitable for use by programs that require sequences in FastA format. ToFastA accepts one or more GCG sequences as input and by default creates one output file containing all the sequences in FastA format. However, NCBI's BLAST family of programs accepts only one sequence per input file. Therefore, if you put `-BLAST` on the command line, ToFastA writes your output into separate files, naming each output file with the input sequence's name and the file name extension .tfa.

EXAMPLE

Here is a session using ToFastA to convert the sequence ggamma.pep into FastA format.

```
% tofasta
TOFASTA of what input sequence(s) ? ggamma.pep

Begin (* 1 *) ?
End (* 148 *) ?
Reverse (* No *) ?

What should I call the output file (* ggamma.tfa *) ?

GGAMMA 148 characters.

148 symbols written into "ggamma.tfa".
```

OUTPUT

Here is the output file:

```
>GGAMMA TRANSLATE of: gamma.seq check: 6474 from: 2179 to: 2270
MGHFTEEDKATITSLWGKVNVEDAGGETLGRLLVVYPWTQRFDSFGNLSSASAIMGNPK
VKAHGKVLTSGLDAIKHLDDLKVTFAQLSEELHCDKLHVDPENFKLGNVLTVAIHFG
KEFTPEVQASWQKMVTGVASALSSRYH*
```
INPUT FILES

ToFastA accepts multiple (one or more) nucleotide or protein sequences as input. You can specify multiple sequences in a number of ways: by using a list file, for example @project.list; by using an MSF or RSF file, for example project.msf(*); or by using a sequence specification with an asterisk (*) wildcard, for example GenEMBL:*

If the input is a list file, ToFastA applies any Begin, End, and Strand (if nucleic acid) attributes it finds within that file. However, with one exception, the Command Line qualifiers -BEGIN, -END, -REVerse, and -NOREVerse will override any conflicting attributes found in the list file. The single exception is that if an -END qualifier specified on the Command Line is less than a Begin attribute found in the list file, the output sequence will begin and end at the base indicated by the Begin.

CONSIDERATIONS

To be compatible with NCBI's BLAST server, ToFastA deletes all non-alphabetic characters except periods (.), tildes (~), and asterisks (*). The program changes periods and tildes into hyphen (-) characters to represent gaps. NCBI's BLAST server tolerates asterisk (*) characters in protein sequences to represent the translation of the stop codon. At the time of this writing, we are not aware of the character requirements of other applications using FastA format.

FastA format does not differentiate protein from nucleotide sequences. FastA format is not rigorously defined so there may be different requirements from one application to another. Please contact Technical Support if you find programs that do not work with the output of ToFastA.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % tofasta [-INfile=]ggamma.pep -Default

Prompted Parameters:

-BEGIN=1 -END=148 sets the range of interest (single sequences only)
-REVerse uses the reverse strand (single sequences only)
[-OUTfile=]ggamma.tfa names the output file

Local Data Files: None
Optional Parameters:

- BLASt creates a separate output file for each sequence
- EXTension=.tfa uses .tfa as a file name extension
- NOMONitor suppresses the screen monitor

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

- BLASt

  Creates a separate output file for each sequence in the input file. The output file names consist of the names of the sequences in the input file followed by a .tfa extension.

- EXTension=.tfa

  Changes the default output file name extension if you do not like the default extension .tfa.

- MONitor

  This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.
FROMEMBL

FUNCTION

FromEMBL reformats sequences from the distribution (flat file) format of the EMBL database into individual sequence files in GCG format.

DESCRIPTION

Use FromEMBL when you want to use sequences in EMBL's distribution format with the GCG. Since EMBL maintains many sequences in one file, FromEMBL must write many output files, one for each sequence in the EMBL file. Each output file is named according to the identifier word on the ID line at the beginning of each sequence entry. All documentation from the EMBL input files is preserved in the GCG output files. The nucleic acid ambiguity codes are preserved except that the hyphen (-) symbol in the EMBL sequences is changed to an N in the GCG files.

EXAMPLE

Here is a session using FromEMBL to convert the EMBL distribution file embl.dat into separate files in GCG format:

% fromembl

FromEMBL of what EMBL flat sequence data file ? embl.dat

    a1mvrna2.embl 2593 bp
    a7nifh.embl  1271 bp
    a7nifx.embl  3169 bp
    a7xag.embl  1395 bp
    aagigii.embl 3411 bp

Finished FROMEMBL

Sequences: 5
Bases: 11,839

% 

OUTPUT

Here is part of the first output file, a1mvrna2.embl, from the example above:

ID   A1MVRNA2   standard; RNA; 2593 BP.
AC   X01572;
DT   03-AUG-1987  (an correction)
DT   30-JAN-1986  (author review)
DT   17-JUL-1985  (first entry)
### Alfalfa mosaic virus (A1M4) RNA 2

**Identification**: Alfalfa mosaic virus

**Organism**: Viridae; ss-RNA nonenveloped viruses; Alfamovirus.

**Reference**


**Data Review**

Data kindly reviewed (30-JAN-1986) by J.F. Bol

---

<table>
<thead>
<tr>
<th>Key</th>
<th>From</th>
<th>To</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSCR</td>
<td>1</td>
<td>2593</td>
<td>A1MV RNA 2</td>
</tr>
<tr>
<td>CDS</td>
<td>55</td>
<td>2424</td>
<td>unidentified reading frame (aa 1-790)</td>
</tr>
</tbody>
</table>

**Sequence**

<table>
<thead>
<tr>
<th>Index</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GUUUUURUCU UUUCGCAGUU GAAAAGAUAA GUUUUCAGU UUAUCUUUU</td>
</tr>
<tr>
<td>51</td>
<td>CAAUAUGUUC ACUCUUUUGA GAUGUCAGG AUUCGGUGUU AAUGAACCUA</td>
</tr>
<tr>
<td>2501</td>
<td>UCCUGAUGAG AGAAAUUCUA UAUUGCUUAU AAUUGUGCUU ACGCACAUAU</td>
</tr>
<tr>
<td>2551</td>
<td>AUAAAUGCUC AUGCAAACCU GCAUGAAGC CCCUAAGGGG UGC</td>
</tr>
</tbody>
</table>

**RELATED PROGRAMS**

The following programs convert sequences between other formats and GCG format:
- FromEMBL, FromGenBank, FromIG, FromPIR, FromStaden, FromFastA, ToIG, ToPIR, ToStaden and ToFastA.

DataSet creates a GCG data library from any set of sequences in GCG format. GCGToBLAST creates a database that can be searched by the BLAST program from any set of sequences in GCG format.
INPUT FILES

FromEMBL accepts a single sequence file in EMBL's distribution format as input. Each input file may contain multiple (one or more) sequences. Here is part of the input file used for the example above:

<table>
<thead>
<tr>
<th>ID</th>
<th>A1MVRNA2</th>
<th>standard; RNA; 2593 BP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>X01572;</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>03-AUG-1987 (an correction)</td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>30-JAN-1986 (author review)</td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>17-JUL-1985 (first entry)</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>Alfalfa mosaic virus (A1M4) RNA 2</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KW</td>
<td>unidentified reading frame.</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Alfalfa mosaic virus</td>
<td></td>
</tr>
<tr>
<td>OC</td>
<td>Viridae; ss-RNA nonenveloped viruses; Alfamovirus.</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RN</td>
<td>[1] (bases 1-2593; enum. 1 to 2593)</td>
<td></td>
</tr>
<tr>
<td>RA</td>
<td>Cornelissen B.J.C., Brederode F.T., Veeneman G.H., van Boom J.H.,</td>
<td></td>
</tr>
<tr>
<td>RA</td>
<td>Bol J.F.;</td>
<td></td>
</tr>
<tr>
<td>RT</td>
<td>&quot;Complete nucleotide sequence of alfalfa mosaic virus RNA 2&quot;;</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>Data kindly reviewed (30-JAN-1986) by J.F. Bol</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FH</td>
<td>Key</td>
<td>From     To       Description</td>
</tr>
<tr>
<td>FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FT</td>
<td>TRANSCR</td>
<td>1   2593       A1MV RNA 2</td>
</tr>
<tr>
<td>FT</td>
<td>CDS</td>
<td>55   2424       unidentified reading frame (aa 1-790)</td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ</td>
<td>Sequence 2593 BP; 736 A; 533 G; 547 G; 777 U; GGUUUUAUCCU UUUCCGGAUU GAAAAGAUA AAUUCGAGAUUU UAAACUUUU CAAUAUGUUC ACUCUUUGGA GAUGUCUGCUG AUUCGGUGUU AAUGAACCUA CUAACACUUC CUCACAGAG</td>
<td></td>
</tr>
</tbody>
</table>

When FromEMBL writes GCG sequence files, it assigns the sequence type based on the composition of the sequence characters. This method is not fool-proof, so to ensure that the output files are written with the correct sequence type, use -PROtein or -NUCleotide on the command line when running FromEMBL.
If FromEMBL is run interactively, you can watch the program monitor to see if the sequences are assigned the correct type. As each new file is written, its name and the number of bases (bp) or amino acids (aa) appears on the screen. If the wrong abbreviation appears (for example, bp appears for a protein sequence), the sequence file was assigned the wrong type. The sequence type also appears in the sequence file. Look on the last line of the text heading just above the sequence itself for Type: N or Type: P.

If the sequence type was incorrectly assigned, turn to Appendix VI for information on how to change or set the type of a sequence.

REstrictions

The Wisconsin Package does not accept sequences longer than 350,000 characters. If an EMBL flat file contains a sequence longer than 350,000 characters, FromEMBL truncates the sequence after the 350,000th character.

Each sequence entry in the input flat file must be in EMBL format. In particular, FromEMBL must find these three lines:
1. Each entry starts with "ID SeqName"
2. Each heading ends with "SQ"
3. Each sequence ends with "//"

CommanD-Line Summary

All parameters for this program may be added to the command line. Use -CHeck to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % fromembl [-INfile=]embl.dat -Default
Prompted Parameters: None
Local Data Files: None
Optional Parameters:
-PROtein insists that the input sequences are proteins
-NUCleotide insists that the input sequences are nucleic acids
-LISTfile[=embl.list] writes a list file of output sequence names
-DIRECTory=dirname writes output to another directory
-NOMONitor suppresses the screen trace for each output sequence
-NOSUMmary suppresses the summary at the end of the program
LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-PROtein and -NUCleotide

Set the program to expect protein or nucleic acid sequences, respectively. Normally, FromEMBL determines whether an input sequence is protein or nucleic acid by looking at its composition. If the first 300 alphabetic characters in a sequence are composed entirely of IUB-IUPAC nucleotide codes (see Appendix III), it is reformatted as a nucleic acid sequence in GCG format; otherwise it is reformatted as a protein sequence. Using these command-line parameters, you can insist that your sequences are proteins (-PROtein) or nucleic acids (-NUCleotide).

-LISTfile=fromembl.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then FromEMBL makes one up using fromembl for the file name and .list for the file name extension.

-DIRECTory=DirName

Writes the output files into a directory other than your current working directory.

-MONitor

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

-SUMmary

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.
FROMFASTA

FUNCTION

FromFastA reformats one or more sequences from FastA format into single sequence files in GCG format.

DESCRIPTION

Use FromFastA when you want to convert sequences that are in FastA format into a format suitable for use with programs in the GCG. FastA format may maintain many sequences in one file; in such a case FromFastA writes many output files, one for each sequence in the FastA file. Each output file is named according to the first word (following the > character) on the documentation line just above the sequence data in the FastA file. The documentation line from the FastA input file(s) is preserved in the GCG output file(s).

The command % seqformat FastA sets a global switch to make GCG programs accept sequences in FastA format without running FromFastA. (See "Using Global Switches" in Chapter 3, Using Programs of the User's Guide.) If this switch is set, only the first sequence in a FastA-format file containing multiple sequences is read by GCG programs. Use the FromFastA program when you want to access other sequences in a FastA-format file or to convert sequences that you wish to keep in GCG format.

EXAMPLE

Here is a session using FromFastA to convert the FastA sequence file fasta.aa into separate sequence files in GCG format.

% fromfasta

FROMFASTA of what FastA sequence file(s) ? fasta.aa

egmsmg.pep 1217 aa.
hshual.pep  129 aa.
lcbo.pep 230 aa.
mchu.pep 149 aa.
musplfm.pep 224 aa.
mww.pep  1966 aa.
mwrtcl.pep 428 aa.
gt87.pep 217 aa.
qrhuld.pep 860 aa.

Finished FROMFASTA with 9 files written.
5420 sequence characters were reformatted.

%

OUTPUT

Here is part of the first output file, egmsmg.pep, from the example above:
INPUT FILES

FromFastA accepts multiple (one or more) files containing sequences in FastA format as input. You can specify multiple input files as a list file, for example @fastaseqs.list, or by using a file specification with an asterisk (*) wildcard, for example fasta*.seq. Each input file may contain multiple (one or more) sequences. Here is part of the input file used for the example above:

>EGMSMG Epidermal growth factor precursor - Mouse
MPWGRRPTWL LLAFLVLFKLISILSVTAWQTGNCQPGGLERSERTCAG

When FromFastA writes GCG sequence files, it assigns the sequence type based on the composition of the sequence characters. This method is not fool-proof, so to ensure that the output files are written with the correct sequence type, use -PROtein or -NUCleotide on the command line when running FromFastA.

If FromFastA is run interactively, you can watch the program monitor to see if the sequences are assigned the correct type. As each new file is written, its name and the number of bases (bp) or amino acids (aa) appears on the screen. If the wrong abbreviation appears (for example, bp appears for a protein sequence), the sequence file...
was assigned the wrong type. The sequence type also appears in the sequence file. Look on the last line of the text heading just above the sequence itself for Type: N or Type: P.

If the sequence type was incorrectly assigned, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

The following programs convert sequences between other formats and GCG format: FromEMBL, FromGenBank, FromIG, FromPIR, FromStaden, FromFastA, ToIG, ToPIR, ToStaden and ToFastA.

DataSet creates a GCG data library from any set of sequences in GCG format. GCGToBLAST creates a database that can be searched by the BLAST program from any set of sequences in GCG format.

RESTRICTIONS

FastA format is not rigorously defined, so FastA files from different sources may not have exactly the same format. Please call us at (800) 756-4674 or send us e-mail at support-us@accelrys.com or support-eu@accelrys.com if you encounter problems converting FastA sequences using FromFastA.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use -CHECK to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: % fromfasta [-INfile=]fasta.aa -Default

Prompted Parameters: None

Local Data Files: None

Optional Parameters:

-PROtein insists that the input sequences are proteins
-NUCleotide insists that the input sequences are nucleic acids
-LISTfile=[=fromfasta.list] writes a list file of output sequence names

LOCAL DATA FILES

None.
PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-PROtein and -NUCleotide

Set the program to expect protein or nucleic acid sequences, respectively. Normally, FromFastA determines whether an input sequence is protein or nucleic acid by looking at its composition. If the first 300 alphabetic characters in a sequence are composed entirely of IUB-IUPAC nucleotide codes (see Appendix III), it is reformatted as a nucleic acid sequence in GCG format; otherwise it is reformatted as a protein sequence. Using these command-line parameters, you can insist that your sequences are proteins (–PROtein) or nucleic acids (–NUCleotide).

-LISTfile=fromfasta.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then FromFastA makes one up using fromfasta for the file name and .list for the file name extension.
FROMGENBANK

FUNCTION

FromGenBank reformat one or more sequences in the flat file format of the GenBank database into individual sequence files in GCG format.

DESCRIPTION

Use FromGenBank to convert sequences from GenBank flat file distribution format to GCG format for use with the GCG (TM). Since GenBank maintains many sequences in one file, FromGenBank must write many output files, one for each sequence in the GenBank file. Each output file is named according to the identifier word on the LOCUS line at the beginning of each sequence entry. All the documentation from the GenBank input files is preserved in the GCG output files.

EXAMPLE

Here is a session using FromGenBank to convert the GenBank distribution file genbank.seq into separate sequence files in GCG format:

% fromgenbank

Reformat what GenBank data file? genbank.seq

ecoogt.seq  719 bp.
ecoomega.seq  764 bp.
ecoompa.seq  2270 bp.

reformatted: genbank.seq
total files: 3
total bases: 3753

%

OUTPUT

Here is part of the first output file, ecoogt.seq, from the example above:

LOCUS       ECOOGT        719 bp ds-DNA             BCT       15-
SEP-1989
DEFINITION  E. coli ogt gene for O-6-alkylguanine-DNA-alkyltransferase.
ACCESSION   Y00495
KEYWORDS    DNA repair; 0-6-alkylguanine-DNA-alkyltransferase; ogt gene.
SOURCE      Escherichia coli.
ORGANISM    Escherichia coli
Prokaryota; Bacteria; Gracilicutes; Scotobacteria;
Facultatively anaerobic rods; Enterobacteriaceae.
REFERENCE   1  (bases 1 to 719)
AUTHORS Potter, P.M., Wilkinson, M.C., Fitton, J., Carr, F.J., Brennand, J., Cooper, D.P. and Margison, G.P.
TITLE Characterisation and nucleotide sequence of ogt, the O6-alkylguanine-DNA-alkyltransferase gene of E. coli
JOURNAL Nucleic Acids Res. 15, 9177-9193 (1987)
REFERENCE 2 (bases 1 to 719)
AUTHORS Potter, P.
JOURNAL Unpublished (1988) see COMMENT for author address
STANDARD simple automatic
COMMENT EMBL features not translated to GenBank features:
<table>
<thead>
<tr>
<th>key</th>
<th>from</th>
<th>to</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRM</td>
<td>101</td>
<td>106</td>
<td>-35 region</td>
</tr>
<tr>
<td>PRM</td>
<td>124</td>
<td>132</td>
<td>-10 region</td>
</tr>
<tr>
<td>SITE</td>
<td>137</td>
<td>141</td>
<td>region of transcription initiation</td>
</tr>
<tr>
<td>RBS</td>
<td>174</td>
<td>183</td>
<td>pot. ribosome binding site</td>
</tr>
<tr>
<td>SITE</td>
<td>601</td>
<td>603</td>
<td>pot. alkylgroup acceptor</td>
</tr>
</tbody>
</table>

[Unpublished (1988) see COMMENT for author address] Author address Potter P., Paterson Institute for Cancer Research, Christie Hospital, Wilmslow Road, Manchester M20 9BX, UK.

Submitted (19-JAN-1988) to the EMBL data library.
FEATURES Location/Qualifiers
CDS 187. .702
/note="O-6-alkylguanine-DNA-alkyltransferase"
BASE COUNT 163 a 172 c 196 g 188 t
ORIGIN

ecoogt.seq Length: 719 September 30, 1998 09:21 Type: N Check: 3921 ..

1 TTCCACTGTT TCTTGGATTC CTGCAACGCT ACAAAACCAGA CGCGAAAACTG
51 GGTACTTACT ATTCGTTAGT CTTGCCCTAT CGGCTTATCT TTTTGGTGTT

/~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
651 AGTTTCAGCGA AAAGAGTGGT TATTGCGCCA TGAAGGTTAT CTTTTGCTGT
701 AAACATTAAA CAATTTGTG
## INPUT FILES

From GenBank accepts a single file in GenBank's flat file distribution format as input. Each input file may contain multiple (one or more) sequences. Here is part of the input file used for the example above:

<table>
<thead>
<tr>
<th>LOCUS</th>
<th>ECOOGT</th>
<th>719 bp ds-DNA</th>
<th>BCT</th>
<th>15-SEP-1989</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINITION</td>
<td>E. coli ogt gene for O-6-alkylguanine-DNA-alkyltransferase.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCESSION</td>
<td>Y00495</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORDS</td>
<td>DNA repair; O-6-alkylguanine-DNA-alkyltransferase; ogt gene.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOURCE</td>
<td>Escherichia coli.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORGANISM</td>
<td>Escherichia coli Prokaryota; Bacteria; Gracilicutes; Scotobacteria; Facultatively anaerobic rods; Enterobacteriaceae.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REFERENCE</td>
<td>1 (bases 1 to 719)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTHORS</td>
<td>Potter, P.M., Wilkinson, M.C., Fitton, J., Carr, F.J., Brennand, J., Cooper, D.P. and Margison, G.P.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>Characterisation and nucleotide sequence of ogt, the O6-alkylguanine-DNA-alkyltransferase gene of E.coli</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOURNAL</td>
<td>Nucleic Acids Res. 15, 9177-9193 (1987)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STANDARD</td>
<td>simple automatic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>EMBL features not translated to GenBank features:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>key from to description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRM</td>
<td>101 106 -35 region</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRM</td>
<td>124 132 -10 region</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SITE</td>
<td>137 141 region of transcription</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBS</td>
<td>174 183 pot. ribosome binding site</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SITE</td>
<td>601 603 pot. alkylgroup acceptor</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Unpublished (1988) see COMMENT for author address] Author address Potter P., Paterson Institute for Cancer Research, Christie Hospital, Wilmslow Road, Manchester M20 9BX, UK.

Submitted (19-JAN-1988) to the EMBL data library.

<table>
<thead>
<tr>
<th>FEATURES</th>
<th>Location/Qualifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS</td>
<td>187..702</td>
</tr>
<tr>
<td>/note=O-6-alkylguanine-DNA-alkyltransferase</td>
<td></td>
</tr>
</tbody>
</table>

BASE COUNT | 163 a 172 c 196 g 188 t | ORIGIN
When FromGenBank writes GCG sequence files, it assigns the sequence type based on the composition of the sequence characters. This method is not fool-proof, so to ensure that the output files are written with the correct sequence type, use -PROtein or -NUCleotide on the command line when running FromGenBank.

If FromGenBank is run interactively, you can watch the program monitor to see if the sequences are assigned the correct type. As each new file is written, its name and the number of bases (bp) or amino acids (aa) appears on the screen. If the wrong abbreviation appears (for example, bp appears for a protein sequence), the sequence file was assigned the wrong type. The sequence type also appears in the sequence file. Look on the last line of the text heading just above the sequence itself for Type: N or Type: P.

If the sequence type was incorrectly assigned, turn to Appendix VI for information on how to change or set the type of a sequence.

RELATED PROGRAMS

The following programs convert sequences between other formats and GCG format: FromEMBL, FromGenBank, FromIG, FromPIR, FromStaden, FromFastA, ToIG, ToPIR, ToStaden and ToFastA.

DataSet creates a GCG data library from any set of sequences in GCG format. GCNGToBLAST creates a database that can be searched by the BLAST program from any set of sequences in GCG format.

RESTRICTIONS

The Wisconsin Package does not accept sequences longer than 350,000 characters. If a GenBank flat file contains a sequence longer than 350,000 characters, FromGenBank divides it into more than one output entry. Each extra output entry has a number appended to the input entry's name. Because of this, the number of output entries may be greater than the number of input entries.
COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you must type in order to use the parameter. Square brackets ([ and ]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% fromgenbank [-INfile=]genbank.seq -Default`

Prompted Parameters: None

Local Data Files: None

Optional Parameters:

- **PROtein** insists that the input sequences are proteins
- **NUCleotide** insists that the input sequences are nucleic acids
- **LISTfile**[-fromgenbank.list] writes a list file of output sequence names
- **DIRectory**=dirname writes output to another directory
- **NOMONitor** suppresses the screen trace for each output sequence
- **NOSUMmary** suppresses the summary at the end of the program

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

**-PROtein and -NUCleotide**

Set the program to expect protein or nucleic acid sequences, respectively. Normally, FromGenBank determines whether an input sequence is protein or nucleic acid by looking at its composition. If the first 300 alphabetic characters in a sequence are composed entirely of IUB-IUPAC nucleotide codes (see Appendix III), it is reformatted as a nucleic acid sequence in GCG format; otherwise it is reformatted as a protein sequence. Using these command-line parameters, you can insist that your sequences are proteins (-PROtein) or nucleic acids (-NUCleotide).
-LIST file=fromgenbank.list

Writes a list file with the names of the output sequence files. This list file is suitable for input to other GCG programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you do not specify a file name, then FromGenBank makes one up using fromgenbank for the file name and .list for the file name extension.

-DIRECTORY=DirName

Writes the output files into a directory other than your current working directory.

-MONitor

This program normally monitors its progress on your screen. However, when you use -Default to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

-SUMmary

Writes a summary of the program's work to the screen when you've used -Default to suppress all program interaction. A summary typically displays at the end of a program run interactively. You can suppress the summary for a program run interactively with -NOSUMmary.

You can also use this parameter to cause a summary of the program's work to be written in the log file of a program run in batch.
APPENDICES

The Appendices contain reference tables, forms, and a complete list of literature citations for references used throughout the set of GCG documentation.

APPENDIX I

Character Fonts

GCG graphics programs plot text using any of several different character fonts. Programs like Figure let you select the fonts by choosing the number of the font from the tables below. All GCG graphics programs print their text using one specific font if you put an expression such as -FONT=3 on the command line.

Font 0 is the plotting device’s own character set and is referred to as a firmware character set. The firmware font is the default for most programs since it usually plots faster than any other font. Font 1 is the GKS (Graphic Kernel Standard) standard font and is designed for applications where speed and accurate positional control are required.

If several fonts appear to be similar in the tables below, the fonts with the higher numbers in that series are more complex. Therefore, they require more computing and more output to the plotter. Usually the higher numbered fonts in a series are more attractive for finished work, such triplex fonts often slow down.

There are two tables below. The first shows the upper- and lowercase letters and the numbers from each font. It is designed to give you an appreciation for the appearance of each font. Each font has 94 characters, which means that 32 of the available characters from each font are not shown in Table 1. The second table is complete and shows all of the characters in each font and their equivalent firmware character.

Characters for computers are usually numbered according to the American Standard Code for Information Interchange, or ASCII. The numbering for Table 2 is according to ASCII and goes from 33 to 126.

The tables below were drawn by the GCG program Fonts using a Hewlett Packard HP 7550. Some institutions with GCG software will choose not to support all of the fonts, as they require more than a mega-byte of disk storage. Run the program plottest -SHOwfonts to show you what fonts are supported at your site. The program Fonts makes a complete display of the fonts available on your system for your device.

Acknowledgement

The fonts available to GCG graphics programs originate with the work of Dr. A. V. Hershey. They were first reported in Technical Report Number 2101, August 1, 1967, The U.S. Naval Weapons Laboratory, Dahlgren, Virginia 22338, USA. The fonts we use are described in “ Contribution To Computer Typesetting Techniques: Tables of Coordinates for Hershey’s Repertory of Occidental Type Fonts and Graphic Symbols,” U.S. National Bureau of Standards, Special Publication 424, which is available from the US Government Printing Office. A tape with
the vectors for each character is available from the National Bureau of Standards. Hershey fonts were implemented for GCG by Philip Delaquess and John Devereux in collaboration with William Gilbert and Verne Luckow.
APPENDIX II

Sequence Symbols

Accelrys GCG (GCG) programs allow all upper- and lowercase letters, periods (.), asterisks (*), tildes (~), ampersands (&), and at (@) symbols in biological sequences. Nucleotide symbols, their complements, and the standard one-letter amino acid symbols are shown below in separate lists. The meanings of the symbols &, and @ have not been assigned at this writing (October, 1996).

GCG supports two gap characters: the period (.) and the tilde (~). GCG programs run from the command line or from the Main List mode of SeqLab treat the two gap characters identically in input sequences. GCG programs run from the Editor mode of SeqLab remove any tilde gap characters from the right end of each input sequence before performing their analyses.

In the future, programs run from either the command line or from SeqLab may differentiate the two gap characters in their analyses. The period gap character will increasingly be used as a space holder that may represent a missing character in a sequence. For example, the period gap character may represent a missed base call in a contig alignment in fragment assembly. The tilde gap character will increasingly be used as a simple place holder that never represents an actual character in a sequence. For example, two tildes may be used in a translated sequence to align each codon in a nucleotide sequence with its corresponding single-letter amino acid symbol. As another example, gaps at the ends of sequences in an alignment may be written as tildes when those gaps are due to differences in input sequence lengths rather than missing characters in the input sequences.

GCG uses the letter codes for amino acid codes and nucleotide ambiguity proposed by IUPAC-IUB. These codes are compatible with the codes used by the EMBL, GenBank, and PIR databases.

Nucleotides

The meaning of each symbol, its complement, and the Cambridge equivalents are shown below. Cambridge files can be converted into GCG files and vice versa with the programs FromStaden and ToStaden.

<table>
<thead>
<tr>
<th>IUB/GCG</th>
<th>Meaning</th>
<th>Complement</th>
<th>Staden/Sanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>T</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>G</td>
<td>C</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>C</td>
<td>G</td>
</tr>
<tr>
<td>T/U</td>
<td>T</td>
<td>A</td>
<td>T</td>
</tr>
<tr>
<td>M</td>
<td>A or C</td>
<td>K</td>
<td>M</td>
</tr>
<tr>
<td>R</td>
<td>A or G</td>
<td>Y</td>
<td>R</td>
</tr>
<tr>
<td>W</td>
<td>A or T</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>S</td>
<td>C or G</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Y</td>
<td>C or T</td>
<td>R</td>
<td>Y</td>
</tr>
<tr>
<td>K</td>
<td>G or T</td>
<td>M</td>
<td>K</td>
</tr>
<tr>
<td>V</td>
<td>A or C or G</td>
<td>B</td>
<td>V</td>
</tr>
</tbody>
</table>
The uncertainty and frame ambiguity codes used by Staden are not supported by the GCG and are converted by FromStaden to the lowercase single base equivalent.

<table>
<thead>
<tr>
<th>Staden Code</th>
<th>Meaning</th>
<th>GCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>probably C</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>probably T</td>
<td>t</td>
</tr>
<tr>
<td>3</td>
<td>probably A</td>
<td>a</td>
</tr>
<tr>
<td>4</td>
<td>probably G</td>
<td>g</td>
</tr>
<tr>
<td>5</td>
<td>A or C</td>
<td>m</td>
</tr>
<tr>
<td>6</td>
<td>G or T</td>
<td>k</td>
</tr>
<tr>
<td>7</td>
<td>A or T</td>
<td>w</td>
</tr>
<tr>
<td>8</td>
<td>G or C</td>
<td>s</td>
</tr>
</tbody>
</table>

Amino Acids

Here is a list of the standard one-letter amino acid codes and their three-letter equivalents. The synonymous codons and their depiction in the IUB codes are shown. You should recognize that the codons following semicolons (;) are not sufficiently specific to define a single amino acid even though they represent the best possible backtranslation into the IUB codes! You can redefine all of the relationships in this list in a local data file as described in Appendix VII.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>3-letter</th>
<th>Meaning</th>
<th>Codons</th>
<th>IUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Ala</td>
<td>Alanine</td>
<td>GCT,GCC,GCA,GCG</td>
<td>!GCX</td>
</tr>
<tr>
<td>B</td>
<td>Asp,Asn</td>
<td>Aspartic,</td>
<td>GAT,GAC,AAT,AAC</td>
<td>!RAY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Asparagine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Cys</td>
<td>Cysteine</td>
<td>TGT,TGC</td>
<td>!TGY</td>
</tr>
<tr>
<td>D</td>
<td>Asp</td>
<td>Aspartic</td>
<td>GAT,GAC</td>
<td>!GAY</td>
</tr>
<tr>
<td>E</td>
<td>Glu</td>
<td>Glutamic</td>
<td>GAA,GAG</td>
<td>!GAR</td>
</tr>
<tr>
<td>F</td>
<td>Phe</td>
<td>Phenylalanine</td>
<td>TTT,TTC</td>
<td>!TTY</td>
</tr>
<tr>
<td>G</td>
<td>Gly</td>
<td>Glycine</td>
<td>GGT,GGC,GGA,GGG</td>
<td>!GGX</td>
</tr>
<tr>
<td>H</td>
<td>His</td>
<td>Histidine</td>
<td>CAT,CAC</td>
<td>!CAY</td>
</tr>
<tr>
<td>I</td>
<td>Ile</td>
<td>Isoleucine</td>
<td>ATT,ATC,ATA</td>
<td>!ATH</td>
</tr>
<tr>
<td>K</td>
<td>Lys</td>
<td>Lysine</td>
<td>AAA,AAG</td>
<td>!AAR</td>
</tr>
<tr>
<td>L</td>
<td>Leu</td>
<td>Leucine</td>
<td>TTG,TTA,CTT,CTC,CTA,CTG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>!TTR,CTX,YTR;YTX</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Met</td>
<td>Methionine</td>
<td>ATG</td>
<td>!ATG</td>
</tr>
<tr>
<td>N</td>
<td>Asn</td>
<td>Asparagine</td>
<td>AAT,AAC</td>
<td>!AAP</td>
</tr>
<tr>
<td>P</td>
<td>Pro</td>
<td>Proline</td>
<td>CCT,CCC,CCA,CCG</td>
<td>!CCX</td>
</tr>
<tr>
<td>Amino Acid</td>
<td>One Letter</td>
<td>Three Letter</td>
<td>Codons</td>
<td>Additional Information</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>--------------</td>
<td>--------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Q</td>
<td>Gln</td>
<td>Glutamine</td>
<td>CAA, CAG</td>
<td>!CAR</td>
</tr>
<tr>
<td>R</td>
<td>Arg</td>
<td>Arginine</td>
<td>CGT, CGC, CGA, CGG, AGA, AGG</td>
<td>!CGX, AGR, MGR, MGX</td>
</tr>
<tr>
<td>S</td>
<td>Ser</td>
<td>Serine</td>
<td>TCT, TCC, TCA, TCG, AGT, AGG</td>
<td>!TCX, AGY, WSX</td>
</tr>
<tr>
<td>T</td>
<td>Thr</td>
<td>Threonine</td>
<td>ACT, ACC, ACA, ACG</td>
<td>!ACX</td>
</tr>
<tr>
<td>V</td>
<td>Val</td>
<td>Valine</td>
<td>GTT, GTC, GTA, GTG</td>
<td>!GTX</td>
</tr>
<tr>
<td>W</td>
<td>Trp</td>
<td>Tryptophan</td>
<td>TGG</td>
<td>!TGG</td>
</tr>
<tr>
<td>X</td>
<td>Xxx</td>
<td>Unknown</td>
<td></td>
<td>!XXX</td>
</tr>
<tr>
<td>Y</td>
<td>Tyr</td>
<td>Tyrosine</td>
<td>TAT, TAC</td>
<td>!TAY</td>
</tr>
<tr>
<td>Z</td>
<td>Glu, Gln</td>
<td>Glutamic, Glutamine</td>
<td>GAA, GAG, CAA, CAG</td>
<td>!SAR</td>
</tr>
<tr>
<td>*</td>
<td>End</td>
<td>Terminator</td>
<td>TAA, TAG, TGA</td>
<td>!TAR, TRA, TRR</td>
</tr>
</tbody>
</table>

Printed: November 30, 2004 12:27 (1162)
APPENDIX III

Form for Reporting Errors

You can have a file with an empty form by using the command % fetch bug.report. It can then be filled it out with a text editor or by hand. Mailing instructions are included on the form.

Please include the complete command line that you used and any local data files in which you have made changes.

bug.report

GCG BUG REPORT

Please complete this form to report any problem with the Accelrys GCG (GCG) Package. In addition, please send all information necessary to reproduce the problem. This may include but is not limited to commandline qualifiers, program options, and input files.

E-mail this form along with any related files to support-us@accelrys.com, support-japan@accelrys.com, or support-eu@accelrys.com. If you cannot use E-mail, you may fax this form and related files to us at (858) 799-5102.

Site name:
User name:
Phone or fax number (in case we need more information):
Operating System Version:
GCG Package Version:
X Windows Display Device (if using SeqLab):

Description of problem:

Example showing how to reproduce the problem:

Input files (if possible):

Additional Comments:

Please send all correspondence regarding problems to:
E-mail: support-us@accelrys.com, support-japan@accelrys.com, or support-eu@accelrys.com
Fax : (858) 799-5102
Or call us at (800) 756-4674.
APPENDIX IV

References

The information in this appendix lists the citations used throughout the set of Accelrys GCG (GCG) Package documentation.


APPENDIX V

Sequence Typing

As you work with the Accelrys GCG (GCG), you will find that some programs accept only nucleotide sequences while others accept only proteins. Many programs allow both nucleotide and protein sequences as input but perform their analysis differently depending on the input sequence type.

You can determine the type of a sequence by looking at the sequence file. Sequences in GCG format contain a dividing line between an optional text heading and the sequence data. Consider the following example of a typical dividing line:

Gamma.Seq Length: 11375 January 1, 1997 10:09 Type: N Checksum: 6474 ..

The sequence type should appear on the dividing line as either Type: N for nucleotide or Type: P for protein. (See "Types of Sequence Files" in Chapter 2, Using Sequence Files and Databases of the User's Guide for a complete description of sequence file formats.) Sequences created before version 7.0 of the GCG (April 1991) do not have this Type: field on the dividing line. If the dividing line doesn't contain a Type: field, the GCG infers the sequence type from the characters in the sequence. This inference may not always be correct.

In previous versions of the GCG, you could ensure that programs inferred the correct sequence type by specifying the sequence type on the command line when you ran a program. However, starting with Version 8.0 of the Package, the sequence type is now an inherent part of the sequence; it cannot be changed from the command line.

If the Type: field of any sequence is incorrect or missing, you can correct it with the Reformat program. Type

% reformat /NUCleotide filename or
% reformat /PROtein filename

For more information, see the Reformat documentation in the Program Manual. ("Specifying Sequence Type" in Chapter 2, Using Sequence Files and Databases of the User's Guide also details how to change the sequence type.)

Printed: February 10, 2006 19:41
APPENDIX VI

Data Files

This appendix contains descriptions of the following types of data files used by Accelrys GCG (GCG) programs:

- Restriction Enzymes
- Scoring Matrices
- Proteolytic Enzymes and Reagents
- Protein Analysis Data Files
- Transcription Factor Database (TFD)
- Codon Frequency Tables
- Translation Tables
- PROSITE
- Profiles
- Version 2.0 Profiles

OVERVIEW

Most GCG programs analyze nucleic acid or protein sequences stored in files or in sequence databases. Additionally, many programs require nonsequence information, or data files, which they use to analyze the sequences. For example, the nucleic acid mapping programs require two data files: enzyme.dat, which contains restriction enzyme names and their corresponding recognition sites; and translate.txt, which associates codons with their corresponding amino acids.

All programs that require a data file have a default file they use, so as a new user, you do not need to worry about supplying one. These default files are public data files. Public data files are located in the public directory with the logical name GenRunData and may be accessed by everyone who uses the package. When you run a program that requires a data file, it automatically finds the appropriate default file in this directory without you having to specify the directory and file name.

GCG also supplies alternative public data files you can have a program use instead of the default. These files are located in the directory with the logical name GenMoreData. There may be times when you want to use an alternative public data file rather than the default file. For example, if you're using the CodonPreference program to analyze a Drosophila sequence, you may want to use the alternative codon frequency table drosophila_high.cod, rather than the default table, ecohigh.cod, which is more appropriate for bacterial sequences.

In each of the following data file descriptions, we provide the names of the default data files used by programs as well as alternative public data files you can specify separately. You will find the following subtopics in each data file's description:

**Default data file.** You can find all default public data files in the directory with the logical name GenRunData.
**Alternative data file.** You can find alternative public data files in the directory with the logical name GenMoreData.

You also can create your own data file or personalize a public data file by copying it to your working directory and modifying it. These files are known as local data files. For instance, you could copy the restriction enzyme data file called enzyme.dat to your directory and delete all of the enzymes in it that are not available in your laboratory. Or, let's say you're working with the FindPatterns program and you create a data file of patterns specific to your research. This personal data file, then, would be available only to you.

**VIEWING OR MODIFYING DATA FILES**

To view a public data file online, use the TypeData program, for example `% typedata enzyme.dat`. To copy a public data file to your directory, use the Fetch program, for example, `% fetch enzyme.dat`. Then, open the file in the text editor of your choice to view or modify the file to your needs. For information on how to use an alternative data file with a program, see Chapter 4, Using Data Files in the User's Guide.
RESTRICTION ENZYMES

Function

Nucleotide mapping programs read the list of available restriction enzymes along with their recognition sites, cut positions, and overhangs from an enzyme data file.

Programs that use this file

Map, MapSort, and MapPlot.

Default data file

enzyme.dat

Alternative data files

None.

Format

Heading: An enzyme data file consists of an optional documentary heading. A divider of two adjacent periods (..) separates the heading from the enzymes.

Name: The first field on each line contains the name of the restriction enzyme; the name should have no more than 132 characters. Only one enzyme should appear per line.

Offset: The name is followed by an offset number, which tells the mapping programs where to cut the top strand when the recognition site is found.

Recognition site: The offset is followed by the enzyme recognition sequence. Nucleic acid recognition sequences, like all nucleotide sequences, are represented in 5’ to 3’ orientation. The recognition sequences should be shorter than 350 characters. They may contain any IUPAC-IUB alphabetic nucleotide character. See Appendix III of the Program Manual for a complete list of supported sequence symbols.

Nonsequence characters in the recognition site: Mapping programs read the offset and overhang fields to find out where each enzyme actually cuts, but the recognition sequences contain non-sequence characters to help humans see the cut points. An apostrophe (’) indicates the cut point on the top strand; an underscore ( _ ) indicates the cut point on the bottom strand (when the enzyme does not leave a blunt end). These apostrophes and underscores are ignored by mapping programs and may therefore be absent.

Overhang: The fourth field in the list of enzymes tells the number of bases (positive or negative) from the cut point on the top strand to the point where the bottom strand is cut. A 0 (zero) would leave a blunt end; a 3 would give a 5’ overhang of 3 bases; a -3 would leave a 3’ overhang of 3 bases. If the recognition site is a palindrome, the overhang field is ignored. If the overhang field is
absent or is a non-numeric character (? or . are most often used), the bottom strand is not searched.

**Display of isoschizomers:** The public file has a semicolon in front of all but one member of each family of isoschizomers. (Isoschizomers, in this context, are restriction endonucleases with the same recognition sequence.) Mapping programs normally ignore isoschizomers whose names are preceded by a semicolon. These isoschizomers are available if you select them individually by name or if you type ** in response to the enzyme prompt.

**Isoschizomers, suppliers, and literature:** Any information on the line to the right of an exclamation point (!) is documentary and is ignored by mapping programs. The documentary information on each record of the public file contains the names of other isoschizomers, if any are known, along with the commercial suppliers and literature references for the enzyme. (See "Restriction Enzyme Suppliers" and "Restriction Enzyme Literature" below.)

**Format requirements:** The exact column for each field on a line does not matter; only the order of the fields is important. Each field should be separated from all other fields on the same line by at least one blank space. Blank lines are tolerated. Most GCG programs ignore information to the right of an exclamation mark (!) so you can add comments to the data file.

**Asymmetric recognition sequences:** If the forward and reverse recognition sites are not the same, then there are two records, one showing the forward and the other the reverse strand. These records must be adjacent to one another in the enzyme file. (See BcgI for an example.) You can give several recognition sites with the same name, but you must put all entries with the same name on adjacent lines of the enzyme data file.

**Suggestions**

You can put semicolons in front of the enzymes to which you do not have access so that they are not displayed when you create restriction enzyme maps. Because GCG mapping programs using the default data file display only one member of each family of isoschizomers, these programs find all possible recognition sites but not all possible cut points. If you find an enzyme displayed near a point of interest, you might want to examine the enzyme file to see if another cut point is available.

**Restriction Enzyme Suppliers**

Many of the restriction enzymes displayed by the GCG mapping programs are available commercially. The file enz_sources.txt shows the main suppliers of restriction enzymes together with the enzymes they make available. This file is for your information only; it is not read by any GCG program.

You can use Fetch to copy this file to your working directory and then search it with a text editor.

**Restriction Enzyme Literature**

Most of the restriction enzymes displayed by the GCG mapping programs are described in the scientific literature. The citations for each enzyme are the numbers that appear last on each record of the enzyme data file enzyme.dat. You can find these citations in the file enz.refs.txt. This file is for your information only; it is not read by any GCG program.
You can use Fetch to copy this file to your working directory and then search it with a text editor.

Acknowledgments

Dr. Richard Roberts at New England Biolabs developed and maintains REBASE, the restriction enzyme database from which the enzyme data in the GCG are drawn.
PROTEOLYTIC ENZYMES AND REAGENTS

Function

Peptide mapping programs read enzyme and reagent names, recognition patterns, and cut positions from an enzyme data file.

Programs that use this file

PeptideMap, MapSort, MapPlot, and PeptideSort.

Default data files

<table>
<thead>
<tr>
<th>Program</th>
<th>Data file</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeptideMap, MapSort, and MapPlot</td>
<td>proenzyme.dat</td>
</tr>
<tr>
<td>PeptideSort</td>
<td>proenzall.dat</td>
</tr>
</tbody>
</table>

Note: Proenzall.dat is a more complete list of proteolytic agents, containing several agents that cut at the same place.

Alternative data files

None.

Format

**Overhang:** The fourth field is the overhang which is used in nucleotide restriction enzyme data files. It has no function for proteolytic reagents.

**Display of isoschizomers:** Mapping programs normally ignore enzymes whose names are preceded by a semicolon (;). These enzymes are available if you select them individually by name or if you type ** in response to the enzyme prompt.

**Documentation:** Any information on the line to the right of an exclamation point (!) is documentary and is ignored by mapping programs.

**Format requirements:** The exact column for each field on a line does not matter; only the order of the fields is important. Each field should be separated from all other fields on the same line by at least one blank space. Blank lines are tolerated. Most GCG programs ignore information to the right of an exclamation mark (!) so you can use these marks to create comments within the data.

**Multiple specificities:** You may include more than one occurrence of an enzyme name if the enzyme has more than one specificity. All records with the same name must appear on adjacent lines of the enzyme data file. If you want to distinguish specificities (for instance trypsin cutting when arginines are blocked), you can create a unique name the distinguishes trypsin cutting at lysine from trypsin cutting at arginine.
Suggestions

You can put semicolons in front of all the enzymes and reagents that you do not have access to or that you do not want to use. GCG programs will ignore those enzymes and reagents. GCG programs PeptideMap, MapSort, and MapPlot search for every point of specific cleavage but not every cleavage pattern. PeptideSort tries to identify each known single-digest cleavage pattern. Send us suggestions for other specificities and cleavage patterns that you think these files should include.
TRANSCRIPTION FACTOR DATABASE (TFD)

Function

This data file provides a list of the recognition sequences for eukaryotic sequence-specific transcription factors from the Transcription Factor Database (TFD).

Programs that use this file

FindPatterns. (Map, MapSort, and MapPlot can also read this file.)

Default data file

None.

Alternative data files

tfsites.dat

Format

Heading: tfsites.dat consists of an optional documentary heading. A divider of two adjacent periods (..) separates the heading from the transcription sites.

Name: The first field on each line contains the name of the site; the name should have no more than 132 characters. Only one site should appear per line.

Offset: The name is followed by an offset number, which tells programs where to mark the top strand when the recognition site is found.

Recognition site: The offset is followed by the recognition sequence. Nucleic acid recognition sequences are represented in 5' to 3' orientation. The recognition sequences should be shorter than 350 characters. They may contain any IUPAC-IUB alphabetic nucleotide character. See Appendix III of the Program Manual for a complete list of supported sequence symbols.

Overhang:

The fourth field should be set to zero to signal that both strands should be searched.

Display of isoschizomers: The public file has a semicolon (;) in front of frequently found sites. Mapping programs normally do not display sites whose names are preceded by a semicolon. If you want to use any of these sites, use the Fetch program to copy tfsites.dat to your working directory and use a text editor to remove the semicolons you want.

Literature: Any information on the line to the right of an exclamation point (!) is documentary and is ignored by mapping programs. The documentary information on each record of the public file contains a common name as well as a literature reference for the site.
Format requirements: The exact column for each field on a line does not matter; only the order of the fields is important. Each field should be separated from all other fields on the same line by at least one blank space. Blank lines are tolerated. Most GCG programs ignore information to the right of an exclamation mark (!) so you can add comments to the data file.

Suggestions

You can use Fetch to copy tfsites.dat to your working directory and then rename it pattern.dat. FindPatterns will then read it automatically and use it as the default data file. Also note that you should always search both strands (FindPatterns does this by default) as most transcription factor sites are strand specific.

Acknowledgments

Dr. David Ghosh developed and maintains TFD.
CODON FREQUENCY TABLES

Function

Codon frequency tables reflect the known codon preferences of an organism.

Programs that use these tables

BackTranslate, CodonPreference, and Frames.

Default data file

deco_high.cod

Alternative data files

drosophila_high.cod
human_high.cod
maize_high.cod
yeast_high.cod
celegans_high.cod
celegans_low.cod

Format

Heading: A codon frequency table consists of an optional documentary heading. A divider of two adjacent periods (..), separates the heading from the table. For example

AmAcid  Codon  Number     /1000     Fraction  ..
Gly     GGG    13.00       1.89      0.02

AmAcid: The first field of information on each line of the table contains a three-letter code for an amino acid.

Codon: The second field contains an unambiguous codon for that amino acid.

Number: The third field lists the number of occurrences of that codon in the genes from which the table is compiled.

/1000: The fourth field lists the expected number of occurrences of that codon per 1,000 codons in genes whose codon usage is identical to that compiled in the codon frequency table.

Fraction: The last field contains the fraction of occurrences of the codon in its synonymous codon family.
Each field of information is separated from every other field by at least one blank space.
Suggestions

You can use the CodonFrequency program to create a codon frequency table from a set of input nucleotide sequences and/or existing codon frequency tables. You also can create or modify a codon frequency table with a text editor. If you choose to use a text editor, you need provide only the first three fields of information on each line of the table. The lines can be in any order; only codons whose use is greater than zero need be present. You should then generate the complete codon usage table -- five fields of information, one line for each codon, and all lines ordered by amino acid -- by using the table you created as the input to the CodonFrequency program.
TRANSLATION TABLES

Function

Translation tables are used by GCG programs for three purposes:

1. To define the relationships between codons and amino acids.
2. To define the relationships between one-letter and three-letter amino acid codes.
3. To identify potential start codons and stop codons.

Programs that use these tables

BackTranslate, CodonFrequency, CodonPreference, Diverge, Frames, Map, MapPlot, MapSort, PepData, Publish, Reformat, and Translate.

Default data file

translate.txt

Alternative data files

<table>
<thead>
<tr>
<th>Data file</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>transmitodros.txt</td>
<td>drosophila mitochondrial translation table</td>
</tr>
<tr>
<td>transl_table_02.txt</td>
<td>vertebrate mitochondrial translation table</td>
</tr>
<tr>
<td>transl_table_03.txt</td>
<td>yeast mitochondrial translation table</td>
</tr>
<tr>
<td>transl_table_04.txt</td>
<td>mold, protozoan, and coelenterate mitochondrial</td>
</tr>
<tr>
<td></td>
<td>and mycoplasma/spiroplasma translation table</td>
</tr>
<tr>
<td>transl_table_05.txt</td>
<td>invertebrate mitochondrial translation table</td>
</tr>
<tr>
<td>transl_table_06.txt</td>
<td>ciliate, dasycladacean, and hexamita translation table</td>
</tr>
<tr>
<td>transl_table_09.txt</td>
<td>echinoderm mitochondrial translation table</td>
</tr>
<tr>
<td>transl_table_10.txt</td>
<td>euplotid translation table</td>
</tr>
<tr>
<td>transl_table_11.txt</td>
<td>bacterial translation table</td>
</tr>
<tr>
<td>transl_table_12.txt</td>
<td>alternative yeast translation table</td>
</tr>
<tr>
<td>transl_table_13.txt</td>
<td>ascidian mitochondrial translation table</td>
</tr>
</tbody>
</table>
transl_table_14.txt  flatworm mitochondrial translation table
transl_table_15.txt  blepharisma mitochondrial translation table
transl_table_16.txt  chlorophycean mitochondrial translation table
transl_table_21.txt  trematode mitochondrial translation table
transl_table_22.txt  scenedesmus obliquus mitochondrial translation table
transl_table_23.txt  thraustochytrium aureum mitochondrial translation table

To specify an alternative translation data file, add the parameter -T filename.txt on the command line.

Format

Heading: A translation table consists of an optional documentary heading. A divider of two adjacent periods (..), separates the heading from the table. For example
Symbol 3-letter Codons IUPAC .. A Ala GCC GCA GCG !GCX

Symbol: The first field of information on each line of the table is a single-letter amino acid sequence symbol.

3-letter:
The second field is the three-letter amino acid code for that sequence symbol.

Codons: The third field must contain a list of all unambiguous codons for the amino acid; this list must come before the exclamation point (!).

IUPAC: In the fourth field, the exclamation point delimits where the unambiguous codons end and where the ambiguous codons start. The ambiguous codons are provided for documentary purposes only and are completely ignored by GCG programs. Each field is separated from every other field by at least one blank space. Any of the 31 GCG sequence symbols (see Appendix III of the Program Manual) may be associated with a three-letter code and one or more unambiguous codons. Each codon and each sequence symbol may be used only once.

Output

Potential start codons are written only in lowercase letters. Stop codons are translated as the asterisk (*) symbol.
SCORING MATRICES

(formerly Symbol Comparison Tables)

Function

Many sequence comparison programs make comparisons between pairs of sequence symbols by looking up a value in a scoring matrix. The matrix assigns an integer value for the match quality of every possible pair of symbols. If you are comparing nucleotides, the matrix might contain 1's for matching symbols and 0's (zeros) for mismatching symbols. However, if you are comparing amino acids, a number could be assigned that is based on chemical similarity or evolutionary distance. The number might be negative if two residues were very dissimilar.

Programs that use these files

BestFit, Compare, FastA, FrameAlign, FrameSearch, Gap, GapShow, GelMerge, PileUp, PlotSimilarity, Pretty, Prime, ProfileMake, Repeat, Segments, StemLoop, TFastA, and the Consensus operation (in the Edit menu) in the Editor mode of SeqLab.

Default data files

For nucleotides:

<table>
<thead>
<tr>
<th>Program</th>
<th>Default data file</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestFit</td>
<td>swgapdna.cmp</td>
</tr>
<tr>
<td>Compare</td>
<td>compardna.cmp</td>
</tr>
<tr>
<td>FastA</td>
<td>fastadna.cmp</td>
</tr>
<tr>
<td>Gap</td>
<td>nwsgapdna.cmp</td>
</tr>
<tr>
<td>GapShow</td>
<td>swgapdna.cmp or nwsgapdna.cmp</td>
</tr>
<tr>
<td>GelMerge</td>
<td>gelmergedna.cmp and gelmergeoldna.cmp</td>
</tr>
<tr>
<td>PileUp</td>
<td>pileupdna.cmp</td>
</tr>
<tr>
<td>PlotSimilarity</td>
<td>plotsimdna.cmp</td>
</tr>
<tr>
<td>Pretty</td>
<td>prettydna.cmp</td>
</tr>
<tr>
<td>Prime</td>
<td>prime.cmp</td>
</tr>
<tr>
<td>ProfileMake</td>
<td>profiledna.cmp</td>
</tr>
</tbody>
</table>
For proteins:

All analysis programs, except FastA and TFastA, use blosum62.cmp as the default data file. FastA and TFastA use blosum50.cmp. The Consensus operation (in the Edit menu) in the Editor mode of SeqLab uses identpep.cmp.

**Alternative data files for nucleotides**

To specify an alternative scoring matrix file, add the parameter `-MATRix=filename.txt` on the command line.

**Global alignments with Segments, ProfileGap, and ProfileSegments**

By default, Segments creates *local* alignments, analogous to those created by BestFit. You can direct Segments to create *global* alignments, analogous to those created by Gap, by using the command-line parameter `-WHOle`. Segments then uses the scoring matrix seggapdna.cmp, containing no negative values for mismatches. ProfileGap and ProfileSegments can be directed to create *global* alignments by using the command-line parameter `-GLObal`. If you want to create *global* alignments using these programs, you might want to create the profile in ProfileMake using the alternative scoring matrix profilegapdna.cmp.

**randomdna.cmp**

This matrix is most appropriate for programs creating *local* alignments (BestFit, Segments, ProfileGap, and ProfileSegments). Since all mismatches between IUPAC-IUB nucleotide symbols are given a value of -3 and all matches are given a value of +10, *local* alignments created using this matrix will be extended further than those created with any of the default scoring matrices for these programs.

**Alternative data files for proteins**

To specify an alternative scoring matrix file, add the parameter `-MATRix=filename.txt` on the command line.

**BLOSUM matrices**

GCG provides a set of BLOSUM matrices for the comparison of peptide sequences, derived from substitutions observed in more than 2,000 blocks of aligned sequences (Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks (Proceedings of the National Academy of Sciences USA 89; 10915-10919) are provided as alternative peptide scoring matrices in the files blosum30.cmp, blosum35.cmp, blosum40.cmp, blosum45.cmp, blosum55.cmp, blosum60.cmp, blosum65.cmp, blosum70.cmp, blosum75.cmp, blosum80.cmp, blosum85.cmp, blosum90.cmp, and blosum100.cmp. To complete this set, blosum50.cmp and blosum62.cmp are also provided as the default scoring matrices for some analysis programs in the GCG.
These matrices are the log odds form of the mutation data matrix for 120 PAMs and 250 PAMs (Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. [1979] in Atlas of Protein Sequence and Structure, and Dayhoff, M. O. Ed, pp. 345-352 (Figure 84), National Biomedical Research Foundation, Washington D.C., respectively).

This matrix, described by Risler, et al. (Journal of Molecular Biology 204; 1019-1029), is derived from an analysis of amino acid substitutions after superposition of homologous protein structures. To construct this matrix the authors converted only substitutions whose alpha carbon atoms are very close to one another after superposition of the structures. Based on results from test alignments using Gap and BestFit, the authors suggest that this scoring matrix may prove superior to others in finding weak similarities in distantly related proteins.

An alternative peptide scoring matrix in the file oldpep.cmp can be provided to GCG programs as a local data file. This matrix was derived from the default peptide scoring matrix in Version 8 of the GCG. Each value in the Version 8 matrix of floating point values was multiplied by 10 and rounded to the nearest integer to determine the comparison values in oldpep.cmp. Perfect matches in oldpep.cmp have a comparison value of 15, and no matches in the matrix have a higher value than perfect matches.

A scoring matrix file consists of a documentary heading, a dividing line with two adjacent periods (..), an optional auxiliary data block that specifies the default gap creation and extension penalties associated with the scoring matrix, and the matrix itself. GCG nucleotide and amino acid symbols are described in Appendix III of the Program Manual.

GCG programs can use two different types of scoring matrices: BLAST format and GCG format. BLAST-format scoring matrices
If you have a native BLAST-format scoring matrix, for example BLOSUM62, it can be used directly by GCG programs without converting it to GCG format. However, one advantage to converting native BLAST-format scoring matrices to GCG format is that you can explicitly set gap creation and gap extension penalties within the file (see "Auxiliary Data Block: Setting Gap Creation and Extension Penalties" below). GCG-format scoring matrices
GCG-formatted scoring matrices can be of two forms: rectangular or "equals." You can use either form of scoring matrix with GCG programs with no difference in program performance or results.

Rectangular scoring matrices. The rectangular form organizes the sequence symbols along an x axis (columns) and y axis (rows), where each symbol along the x axis is compared with each symbol along the y axis. There is a row and column for every sequence symbol that has at least one non-zero comparison value. The value of each pair of symbols compared is placed at the intersection of the appropriate row and column. All relationships that are not explicitly defined in the matrix are assigned a value of 0. Every comparison value is separated from every other value by at least one blank space. Blank lines are tolerated.

Consider the example below:
The intersection of row D with column D has a value of 6, which represents an identical match for a D-D pairwise comparison. However, the pairwise comparison between non-identical symbols often is given a lower value, for example a C-D comparison is -3.

Notice that the values are identical at the C-D comparison and at the D-C comparison: -3. Previous versions of the package supported only triangular forms of scoring matrices to eliminate this repetition. However, to make publicly available scoring matrices, which are in a rectangular format, easier to use, the GCG now supports only rectangular-format scoring matrices. See "Converting Scoring Matrices" later in this section for converting pre-Version 9 scoring matrices to the new format.

Equals-form scoring matrices. The second form of GCG-format scoring matrix supported is "equals" form, so named because within the matrix, each pairwise comparison equals a value. For instance, in the example below, a A-A symbol comparison is assigned, or equals, a value of 4.

\[
\begin{array}{cccccccc}
AA & = & 4 & AB & = & -2 & AD & = & -2 \\
AF & = & -2 & & & & & \\
AH & = & -2 & AI & = & -1 & AK & = & -1 \\
AM & = & -1 & & & & & \\
AN & = & -2 & AP & = & -1 & AQ & = & -1 \\
AS & = & 1 & & & & & \\
AW & = & -3 & AX & = & -1 & AY & = & -2 \\
BB & = & 6 & & & & & \\
BC & = & -1 & BD & = & 6 & BE & = & 2 \\
BG & = & -1 & & & & & \\
\end{array}
\]

All relationships that are not explicitly defined in the matrix are assigned a value of 0. Every comparison value is separated from every other value by at least one blank space. Blank lines are tolerated. Some people find the equals form of scoring matrix easier to read than the rectangular form.

Auxiliary Data Block: Setting Gap Creation and Extension Penalties

You can specify gap creation and gap extension penalties within a scoring matrix to ensure that programs reading the scoring matrix use those values as defaults. If you do not specify these penalties, the program calculates reasonable defaults based on the values in the matrix.

Gap creation and gap extension penalties must follow a specific format within a scoring matrix. These penalties must appear in an auxiliary data block, which appears after the dividing line with
the two adjacent periods (..) and before the line of sequence symbols in the scoring matrix, as shown below:

```
..
{
    GAP_CREATE 12
    GAP_EXTEND 4
}
```

A  B  C  D  E  F  G  H ...

If you create your own scoring matrix, or if you modify an existing one, you must maintain this format for specifying gap creation and extension penalties.

Note that even though gap creation and extension penalties may be set within a scoring matrix, you can override them on the command line. To do so, use the parameters -GAPweight and -LENgthweight on the command line when you run a program that uses scoring matrices.

Suggestions

Creating new scoring matrices

Use the CompTable program to create scoring matrices. You also can use a text editor to create a scoring matrix; if you do so, use the Reformat program with the command-line parameter -COMparison to rewrite the file into GCG format. Both CompTable and Reformat round the values in the matrix to the nearest integer.

Modifying existing scoring matrices

Several programs may use the same default scoring matrix. However, although the matrices may be identical, the default matrix for each program is contained in a separate file. This allows you to modify a local version of the matrix for one program without affecting the matrix used by another program.

If you make modifications to a matrix, use the Reformat program with the command-line parameter -COMparison to rewrite your scoring matrix data file into GCG format.

Converting scoring matrices

Converting pre-Version 9 scoring matrices to the new format

In Version 9 all scoring matrices provided with the package in GenRunData and GenMoreData are already converted to the new format. However, you must convert all of the scoring matrices in your personal directories, including your personal directory with the logical name MyData, to the new rectangular format. When you do so, you will need to specify the scoring matrix as either nucleotide or protein. GCG programs will not accept pre-Version 9 scoring matrices, and they will display the following error message if you try to use one:
*** ERROR, READSCOREMAT cannot read the scoring matrix in the file "filename"!

If this is a scoring matrix created before Version 9, try converting it with "% reformat /OLDCMPformat /PROtein" or "% reformat /OLDCMPformat /NUCleotide"

To convert pre-Version 9 scoring matrices to the new format, type
% Reformat -OLDCMPformat -NUCleotide scoring_matrix
or
% Reformat -OLDCMPformat -PROtein scoring_matrix

Converting scoring matrices to make them more readable

GCG programs can accept two forms of GCG-format scoring matrix files: rectangular and "equals." There is no difference in analysis or performance between the forms. However, some people find "equals" format easier to read, and the package provides a way to convert between the two forms.

To convert rectangular scoring matrices to the more readable "equals" format, type
% Reformat -COMParison -EQUALSformat scoring_matrix
To convert "equals" format scoring matrices to rectangular format, type
% Reformat -COMParison scoring_matrix

Converting BLAST-format scoring matrices to GCG-format

GCG also works with native BLAST-formatted scoring matrices. Although converting BLAST-formatted scoring matrices to GCG-format is unnecessary, you may find it useful to do so. GCG-formatted scoring matrices allow you to specify gap creation and extension penalties within the scoring matrix file.

To convert BLAST-formatted scoring matrices to GCG-format, type
% Reformat -COMParison -NUCleotide scoring_matrix
or
% Reformat -COMParison -PROtein scoring_matrix
PROTEIN ANALYSIS DATA FILES

Function

These data files enable programs to locate motifs in protein sequences and to make predictions about peptide isolation, secondary structure, hydrophobicity, and antigenicity.

Programs that use these tables

PeptideSort, Isoelectric, PepPlot, HelicalWheel, CoilScan, SPScan, and HTHScan.

Default data file

<table>
<thead>
<tr>
<th>Program</th>
<th>Default data file</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeptideSort</td>
<td>aminoacid.dat</td>
<td>amino acid residue properties</td>
</tr>
<tr>
<td></td>
<td>extinctcoef.dat</td>
<td>extinction coefficients for amino acids</td>
</tr>
<tr>
<td></td>
<td>isoelectric.dat</td>
<td>residue-specific pK values for the prediction of a peptide's isoelectric point</td>
</tr>
<tr>
<td>Isoelectric</td>
<td>isoelectric.dat</td>
<td>residue-specific pK values for the prediction of a peptide's isoelectric point</td>
</tr>
<tr>
<td>PepPlot</td>
<td>pepplot.dat</td>
<td>residue-specific values for the prediction of protein secondary structure, hydrophobicity, and helical hydrophobic moment</td>
</tr>
<tr>
<td></td>
<td>ges.dat</td>
<td>residue-specific values for identifying nonpolar transbilayer helices</td>
</tr>
<tr>
<td></td>
<td>garnier.dat</td>
<td>residue-specific values for secondary structure prediction using the method of Garnier</td>
</tr>
<tr>
<td>HelicalWheel</td>
<td>helicalwheel.dat</td>
<td>residue-specific attributes for the display of a peptide sequence as a helical wheel</td>
</tr>
<tr>
<td>CoilScan</td>
<td>mtidkcoils.dat</td>
<td>weight matrix of amino acid</td>
</tr>
</tbody>
</table>
coiled-coil propensities

SPScan    speuk.dat  weight matrix for eukaryotic signal peptides

            spgpos.dat  weight matrix for Gram-positive bacterial signal peptides
            spgneg.dat  weight matrix for Gram-negative bacterial signal peptides

HTHScan    htharac.dat  weight matrix for AraC family H-T-Hs
            hthlysr.dat  weight matrix for LysR family H-T-Hs
            hthhomeobox.dat  weight matrix for Homeobox family H-T-Hs

Alternative data files

    CoilScan   mtkcoils.dat

Format

All data files consist of an optional documentary heading, a dividing line with two adjacent periods (..), and the data. The exact column for each field on a line does not matter; only the order of the fields is important. Each field should be separated from all other fields on the same line by at least one blank space.
PROSITE

Function

You can search protein sequences for motifs that are represented in the PROSITE Dictionary of Protein Sites and Patterns.

Programs that use this file

Motifs

Default data file

prosite.patterns

Alternative data files

None.

Format

The format of GCG pattern files is described in the documentation for programs that use these files.

The exact column for each field on a line does not matter; only the order of the fields is important. Each field should be separated from the other fields on the same line by at least one blank space. Blank lines are tolerated. Most GCG programs ignore information to the right of an exclamation mark (!), so you can use these marks to create comments within the data. You cannot edit prosite.patterns unless your text editor can handle very large records.

Heading: This data file has an optional documentary heading, followed by a dividing line with two adjacent periods (..).

Name: The first field on each line contains the name of the restriction enzyme; the name should have no more than 132 characters. Motifs prefixed by a semicolon ( ; ) are short patterns which are expected to occur in most protein sequences by chance alone. Such frequently found patterns are not displayed by the Motifs program unless you run Motifs with the command-line parameter -FREquent. Only one motif should appear per line.

Offset: The name is followed by an offset number, which tells Motifs where to mark the sequence when the motif expression is found.

Pattern: Patterns should be shorter than 350 characters. They may contain any alphabetic amino acid character. See Appendix III of the Program Manual for a complete list of supported sequence symbols.

For a complete description of the syntax in which motifs are represented, see the topic DEFINING PATTERNS in Motifs in the Program Manual.
Note that some motifs require multiple patterns to identify them. If this is so, these patterns will have the same name and must appear on adjacent lines.

**PDoc:** The fourth field tells the name of the PROSITE abstract for the pattern. You can copy this file to your directory with the Fetch command, or you can display it with the TypeData command.

**Suggestions**

prosite.seqcat contains a short description of each motif in prosite.patterns. Use the Fetch command to copy the prosite.seqcat file to your directory or use the TypeData command to view the file online.

The use of Motifs is so straightforward that there are few occasions when you will need to modify this file.

**Acknowledgments**

Dr. Amos Bairoch of the University of Geneva publishes and maintains the *PROSITE Dictionary of Protein Sites and Patterns*. PROSITE is distributed by the European Bioinformatics Institute in Cambridge, England.
**PROFILES**

**Function**

This database contains validated profiles derived from the motifs in the PROSITE Dictionary of Protein Sites and Patterns.

**Programs that use this file**

ProfileScan

**Default data file**

profilescan.fil

**Alternative data file**

oldprofilescan.fil

**Format**

**Heading:** The optional heading documents the contents of each column. A divider of two adjacent periods (..) separates the heading from the profiles.

**Name:** The first column contains the location and name of each profile (see SUGGESTIONS below). These names correspond to the names of the patterns in the prosite.patterns file. The profile name must contain fewer than 255 characters.

**High and Intrst:** By default, ProfileScan reports only alignments with normalized scores greater than the HIGH value. If you add the -INTEResting parameter to the command line, ProfileScan will report alignments that score higher than the INTRST value.

**Gap and Len:** These values specify, respectively, the gap creation and extension penalties used to align the motif profile to the query sequence.

**A, B, C, AVE, and SD:** These values specify the parameters for length-dependent normalization of the alignment scores. See ProfileSearch in the Program Manual for a description of the derivation of these values and their use in normalizing the alignment scores.

**Suggestions**

Individual profile files are maintained in the directory with the logical name ProfileDir. To view a profile's documentation, use the Fetch command to copy a profile file to your directory, for example `-% fetch apple.prf`, or use the TypeData command to view the file online.

**Acknowledgments**

Dr. Michael Gribskov of the San Diego Supercomputing Center prepared and validated these profiles. Dr. Amos Bairoch of the University of Geneva publishes and maintains the PROSITE Dictionary of Protein Sites and Patterns.
VERSION 2.0 PROFILES

Function

This database contains validated profiles derived from the motifs in the PROSITE Dictionary of Protein Sites and Patterns. Profiles are a special kind of scoring matrix used by several different programs. The addition of MEME and MotifSearch to the GCG required the introduction of a new format of profile that allows multiple profiles to be kept in one file.

Programs that use these files

MEME generates version 2.0 profiles, while MotifSearch is intended to process them. ProfileSearch, ProfileGap and ProfileSegments can all read ONLY THE FIRST profile from a version 2.0 file.

Default data file

Not Applicable

Alternative data file

Not Applicable

Format

Heading: The file should begin with a line containing either "!!AAPROFILE 2.0" or "!!NAPROFILE 2.0". Thereafter, you may include any information you like, concluding the heading section with a divider of two adjacent periods (..)

Auxiliary Data Block: The ADB begins with a line having nothing but a "{", and ends with a line having nothing but a "}". These MUST appear in the first column of their respective lines.

The ADB must contain four parsable data lines. The first gives the Length of the profile (sometimes thought of as the width !), in the form "Length: <value>". The next two lines control the gap creation and extension penalties for the profile, and the fourth gives the labels of the columns used in the profiles. The column labels should be separated by blank spaces. The first label should always be "Cons" (for Consensus), and this should appear at the beginning of the line -- no indentation please.

Here is an example of a simple ADB, with some of the column labels replaced by an ellipsis:

```
{ Length: 9
  Gap: 1.00       Len: 1.00
  GapRatio: 0.0   LenRatio: 0.0
  Cons   A      C      D      E      F    . . .      W      Y
  Gap    Len }
```

The ADB may contain any number of "Comment" lines, indicated by a "!" in the first column

Profile The profile itself is made up of rows of log-odds values, with each row corresponding to a
position in the profile and (with three exceptions) each column corresponding to a valid symbol for that position. The exceptions are the first column (which contains a letter identifying the consensus symbol for the row) and the last two columns, which give the multiplying factor for the gap creation and extension penalties for the row. (Note that MEME's output profiles are always ungapped, and thus will always have 100 (the maximum value) in the last two columns). The last row in a profile does NOT correspond to a position in the profile. Instead it contains counts for the number of appearances of each letter at any position in the sequences from which the profile was derived. This information is not used by any programs at this time, but it nonetheless must be there. Note that this dummy row is NOT included in the Length count given in the Auxiliary Data Block.