

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., ecoomp1.sample, ecoomp2.sample, ...).

EXAMPLE

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

```
% 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
```

Sample

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 AAACCAAAT CCAGTTTTAG AAAAAGTTTA
     51  TTAAAATTTA CTGCACTTTT ATTTGGCTTG GCGACGGTTG CGCAATCCGT
    101  GCAGTGGATT TGGGATAGCT ATCAAAAACA TCAATGGATT TATCTTGCTT
    151  TTGCTTTAGT CAGTTTGATT ATCATTATAT TGGGTATTAA AGAGATTATT
    201  TGTGAGTGGC GACGTTTAGT TCGTTTAAAA AACGTGAGC AATGGCAACA
    251  ACAAAGTCAG CAGATTTGGT TAGAAAGTGC GGTAAGAAAT 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 GenEMBL:*. 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 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

Sample

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 Wisconsin Package programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you don't 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.

Printed: December 1, 1998 11:23 (1162)

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.

Seg

```
!!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 LFBVATWSDLG LCKKRPKPGG WNTGGSRYPG QGSPGGNRYX
51  XXXXXXXXXXX XXXXXXXXXXX XXXXXXXXXXX XXXXXXXXXXX XXXXTHSQWN
101 KPSKPKTNMK HMXXXXXXXX XXXXXXXXXXX XXXXXRPIIH FGSDYEDRYY
151 RENMHRYPNQ VYYRPMDEYS NQNNFVHDCV NITIKQHXXX XXXXXXXXXXX
201 XDVKMMERVV EQMCITQYER ESQAYYQRGS SMVLFSXXXX XXXXXXXXXXX
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 `GenEMBL:*`. 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 ($\log(\text{base } 2) 20$). If a sequence contained only one character, then each character would add 0.0 bits of information to the sequence ($\log(\text{base } 2) 1$).

The parameters `-LOWcut` and `-HIGHcut` 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

ACKNOWLEDGEMENT

Seg was written by Wootton and Federhen at the National Center for Biotechnology Information (NCBI). Their public-domain program was modified by Scott Rose for distribution with Version 9 of the Wisconsin Package. The document you are now reading was written by John Devereux. We are extremely grateful to Wootton and Federhen for their work on Seg and to NCBI for making this program available to the scientific community.

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.

Seg

-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.

-MINhighlen=0

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.

-EXTension=.seg

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=seg.list`

writes a list file with the names of the output sequence files. This list file is suitable for input to other Wisconsin Package programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you don't specify a file name, then Seg makes one up using `seg` 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: December 1, 1998 11:23 (1162)

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 GenEMBL nucleotide sequence database found in the example session for WordSearch:

```
% segments

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

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

Aligning .....-...
GB_PR2:HUMHBGG      545 bp Gaps: 0 Quality: 4440 / Length: 444
Aligning .....-...
GB_PAT:I42109      443 bp Gaps: 1 Quality: 4377 / Length: 444
Aligning .....-...
GB_PR1:HSGGGPHG    521 bp Gaps: 0 Quality: 3814 / Length: 383

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

%
```

Segments

OUTPUT

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 ATGGGTCATTTTCACAGAGGAGGACAAGGCTACTATCACAAAGCCTGTGGGG 50
  |||
18 ATGGGTCATTTTCACAGAGGAGGACAAGGCTACTATCACAAAGCCTGTGGGG 67
  |||
51 CAAGGTGAATGTGGAAGATGCTGGAGGAGAAACCCTGGGAAGGCTCCTGG 100
  |||
68 CAAGGTGAATGTGGAAGATGCTGGAGGAGAAACCCTGGGAAGGCTCCTGG 117
  |||
```

//

```
ggammacod.seq          check: 2906  from: 1      to: 444
GB_PR1:MMGGLINE        check: 889   from: 2318   to: 11286
X53419 M.mulatta gamma-globin-1(G), gamma-globin-2(A) genes and ...
Gaps: 0  Quality: 2174  Ratio: 9.577  Score: 209  Width: 3  Limits: +/-4
```

```
91 AGGCTCCTGGTTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGG 140
  |||
2409 AGGCTCCTGGTTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGG 2458
  |||
141 CAACCTGTCCTCTGCCTCTGCCATCATGGGCAACCCCAAAGTCAAGGCAC 190
  |||
2459 CAACCTGTCCTCTGCCTCTGCCATCATGGGCAACCCCAAAGTCAAGGCAC 2508
  |||
```

//

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.

Segments

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 don't 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 fails 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=]ggamacod.word -Default`

Prompted Parameters:

`[-OUTfile=]ggamacod.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 **-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 don't 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.

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 **-WHOLe** 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.

Segments

-MATRix=mymatrix.cmp

allows you to specify a scoring matrix file name other than the program default. If you don't 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.

-GAPweight=50

lets you designate a gap creation penalty if you don't want to use the default penalty. (See the ALGORITHM topic in BestFit for a description of gap creation penalties.)

-LENgthweight=3

lets you select a gap extension penalty if you don't want to use the default penalty. (See the ALGORITHM topic in BestFit for a description of gap extension penalties.)

-WHOLe

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.

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

-MATch=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. **-MATch** and **-MISmatch** allow you to set values other than 10 for matches and $-20/Alphabet\ size$ for mismatch.

-MISmatch=-5

See **-MATch** for a description of **-MISmatch**.

-PAIr=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 **-PAIr** 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 `x` 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: December 1, 1998 11:23 (1162)

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
....|:.....:|.....i.....:.|.....|.....|.....|.....|...
    0      10      20      30      40      50      60      70

~~~~~^
|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|
    0      10      20      30      40      50      60      70      80      90     100

"sample.seq"      27 nucleotides
```

SeqEd

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.

SeqEd

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] **I**nclude [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.

SeqEd

[s] **HEAding**

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 **-LANes=T,A,G,C** or with the SeqEd command **DIGitizer G,T,C,A**.

Maxam and Gilbert sequencers find it useful to use **-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 **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.

REload

goes into 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 ^ (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 **COMMAND LINE SUMMARY** topic below for more information.

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 **ACcept** an overlap, but the decision is yours.

Help

shows the commands available in the Screen and Command Modes of SeqEd.

[s,f] **EXit** [filename]

works exactly like **write** except that your session with SeqEd ends after the sequence is written out into a new sequence file.

SeqEd

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:

Graf/Bar pin	Computer pin
1	1
7	7
3	2

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 don't 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.

SeqEd

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 DIGitizer 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.

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:

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.

SeqEd

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 don't 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.

SeqEd

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 `-VECTORS` 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 Haerberli and implemented for VAX/VMS by Paul Haerberli 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:

<code>-SINGLEcommand</code>	automatically returns to screen mode after commands
<code>-PROtein</code>	sets sequence type to protein, and sets find to search for perfect symbol matches
<code>-NUCleotide</code>	sets sequence type to nucleotide, and sets find to allow nucleotide ambiguity symbol matches
<code>-PERFect</code>	sets find to search for perfect symbol matches, even if sequence type is nucleotide
<code>-VECTors=GB:SynpBR322</code>	highlights sequences from pBR322
<code>-SITes=GAATTC</code>	highlight GAATTC patterns
<code>-LANes=A,C,G,T</code>	sets the default lane order for digitizer
<code>-MINOverlap=10</code>	sets minimum overlap length for Reload command
<code>-PCTOverlap=95</code>	sets stringency for the Reload command
<code>-TOLerance=0.4</code>	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.

SeqEd

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.

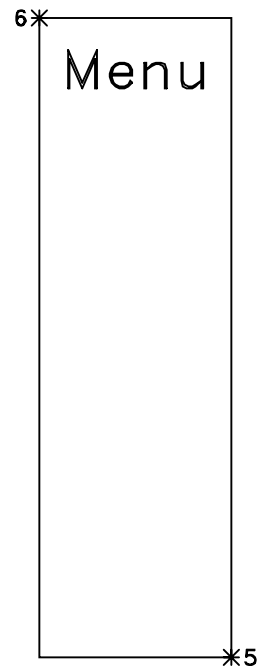
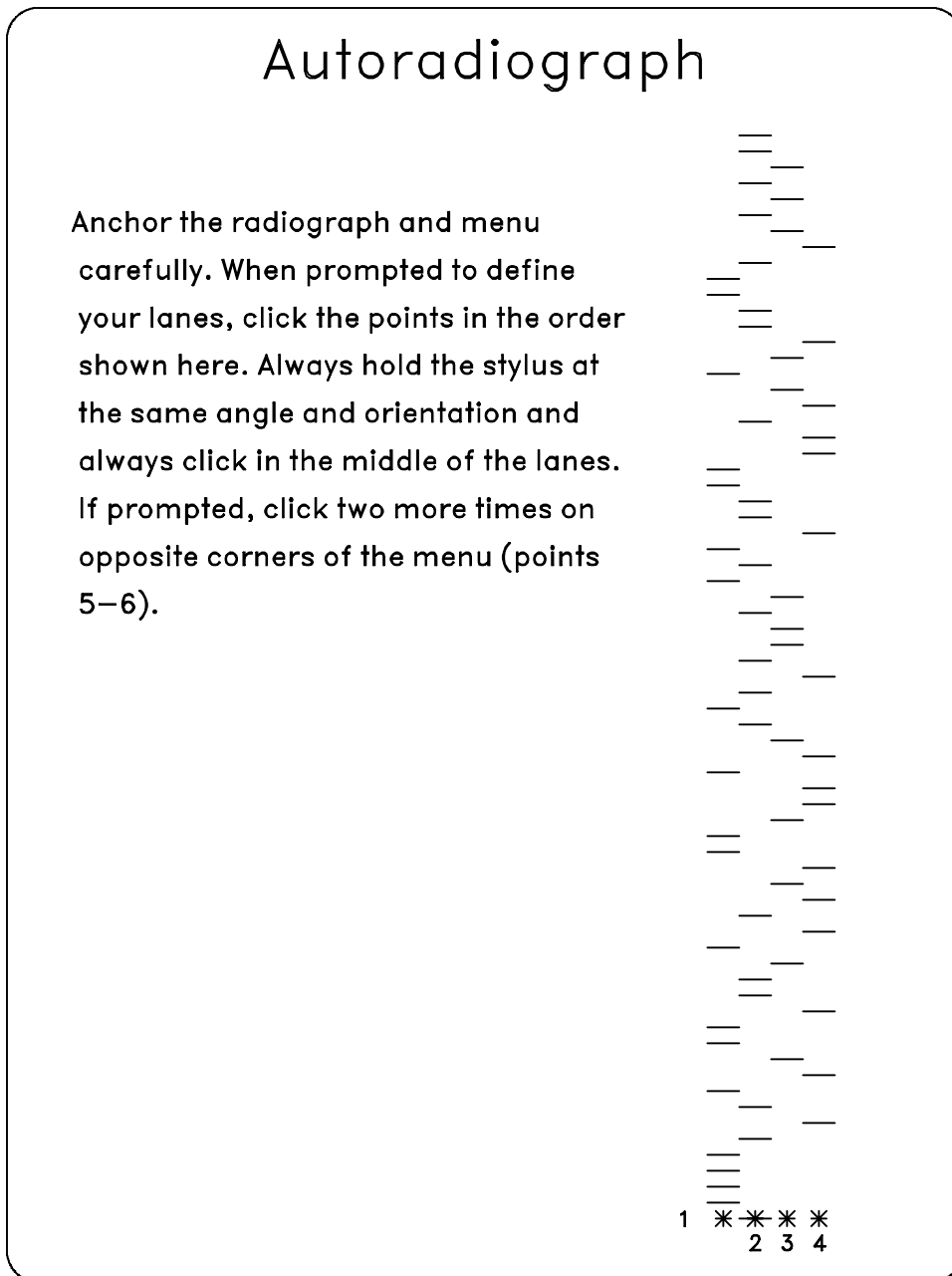
-TOLerance=0.4

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: December 1, 1998 11:23 (1162)



This two-inch "Virtual Menu" controls the Graf/Bar.
Place your autoradiograph and menu outside this area.



7	8	9	↑
4	5	6	↓
1	2	3	←
0	FIND		→
BEGIN		END	
RETURN			
A	C	G	T
R	X	Y	?
DELETE			
NEW LANES			
RELOAD		ACCEPT	
CHECK		HELP	
WRITE FILE			
FRESH SCREEN			
KEYBOARD			

SEQLAB

FUNCTION

SeqLab is the graphical user interface for the Wisconsin Package.

DESCRIPTION

SeqLab is the graphical user interface for the Wisconsin Package. 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 the Wisconsin Package 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 or the online help available from within SeqLab.

EXAMPLE

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

```
% seqlab
```

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

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

ACKNOWLEDGEMENTS

SeqLab was written by Mary Schultz, Philip Delaquess, and Steven Smith. Portions of SeqLab are based on the Genetic Data Environment (GDE) originally developed in the Department of Microbiology, University of Illinois, Urbana-Champaign, Illinois, USA and licensed to GCG.

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: % seqlab

Optional Parameters:

`-MODE=LIST` Start in the Main List mode of operation

`-MODE=EDITOR` Start in the Editor mode of operation

Printed: December 1, 1998 11:23 (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 don't 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"

%
```

SetKeys

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, GELENER, 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.

Printed: December 1, 1998 11:23 (1162)

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 don't 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:

```
% 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
l 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
TTAGAGGCTG GGAGTGGTGG CTCACGCCTG TAATCCCAGA ATTTTGGGAG
////////////////////////////////////
CCAGGAAAGT GACTGCAGGT CACTTTTCCT GGAGCGGGTG AGAGAAAAGT
GGAAGTTGCA GTAAGTCCG AATTC
```

ShiftOver

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: `% shift [-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: December 1, 1998 11:23 (1162)

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 sequence(s).

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.

Shuffle

Minimal Syntax: % shuffle [-INfile=]gamma.seq -Default

Prompted Parameters:

-BEGin=1 -END=11375 sets the range of interest
-REVerse uses the reverse strand
[-OUTfile=]gamma.shuffle names the output file

Local Data Files: None

Optional Parameters:

-PREServe=2 preserves dinucleotide or dipeptide composition
 in shuffled sequence
 =3 preserves trinucleotide or tripeptide composition
 in shuffled sequence
-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.

-PREServe=2

preserves the input sequence's dinucleotide or dipeptide composition in the output shuffled sequence. Use **-PREServe=3** to preserve the trinucleotide or tripeptide composition.

-NONUMbering

suppresses the numbering in the output file.

Printed: December 1, 1998 11:23 (1162)

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 `GenEMBL:*`. 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.

Simplify

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=]gamma.pep -Default

Prompted Parameters: (for a single sequence)

-BEGin=1 -END=444 sets the range of interest
[-OUTfile=]gamma.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 = L,I,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.

`-LISTfile=simplify.list`

writes a list file with the names of the output sequence files. This list file is suitable for input to other Wisconsin Package programs that support list files (see Chapter 2, Using Sequence Files and Databases in the User's Guide.) If you don't 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: December 1, 1998 11:23 (1162)

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™.

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.

Spew

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.

Printed: December 1, 1998 11:23 (1162)

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

%
```

SPScan

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:

NBRF, Release 57.0, Released on 30Jun1998, Formatted on 18Aug1998
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 `-ADJUSTscores`.

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 `-ADJUSTscores`, 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.

SPScan

INPUT FILES

The input to SPSscan is one or more protein sequences. If SPSscan 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 `GenEMBL:*`.

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 YRYRYR. 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.

CONSIDERATIONS

Under normal circumstances it is likely that SPSscan 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 SPSscan (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. SPSscan 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.

SPScan

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, SPSscan 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:

<code>-BEGIN=1</code>	<code>-END=258</code>	sets the range of interest
<code>-THRESHold=7.0</code>		sets minimum score for SP detection
<code>[-OUTfile=]jh0472.spscan</code>		specifies name of results file

Local Data Files:

<code>-DATA=speuk.dat</code>	assigns the weight matrix for eukaryotic SPs
<code>-DATA=spgpos.dat</code>	assigns the weight matrix for Gram-positive prokaryotic SPs
<code>-DATA=spgneg.dat</code>	assigns the weight matrix for Gram-negative prokaryotic SPs

Optional Parameters:

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

ACKNOWLEDGEMENT

We thank Dr. Gunnar von Heijne and Prof. Duncan McGeoch for helpful information used in writing this program. We are very grateful to Dr. Henrik Nielsen and his group at the Center for Biological Sequence Analysis, The Technical University of Denmark, for the SignalP Server WWW site (<http://www.cbs.dtu.dk/services/SignalP/>) from which we got the data for our weight matrices. We also thank Maureen Colbert of Genetics Institute for her investigations which led to a better choice of default threshold score for Release 10.0.

SPScan was written by Ted Slater.

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`, SPSscan will use the weight matrix file for Gram-positive prokaryotes called `spgpos.dat`. If you use `-GRAMNegative`, SPSscan 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`, SPSscan 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, SPSscan will display all SPs whose scores meet or exceed the threshold.

`-GRAMPositive`

tells SPSscan 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`

tells SPSscan 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.

SPSscan

-ADJustscores

tells SPSscan 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

tells SPSscan 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 SPSscan 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.

-NOPROBabilities

tells SPSscan 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 SPSscan run much faster.

-VERbose

tells SPSscan 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.

-RSF=spscan.rsf

writes an RSF (rich sequence format) file containing the input sequences annotated with features generated from the results of SPSscan. This RSF file is suitable for input to other Wisconsin Package programs that support RSF files. In particular, you can use SeqLab to view this features annotation graphically. If you don't 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.

Printed: December 1, 1998 11:23 (1162)

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:

```
% 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:* *) ?

Don't show scores whose E() value exceeds: (* 10.0 *) :

What should I call the output file (* ggamma.ssearch *) ?

      1 Sequences          105 aa searched      PIR1:CCHU
     501 Sequences        93,217 aa searched   PIR1:IHQFT

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

CPU time used:
  Database scan:  0:10:19.8
Post-scan processing:  0:00: 4.6
  Total CPU time:  0:10:24.6
  Output File:  ggamma.ssearch

%
```

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.)

SSearch

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:
```

```
NBRF, Release 57.0, Released on 30Jun1998, Formatted on 18Aug1998
```

```
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

z-score	obs	exp
	(=)	(*)
< 20	879	0:=====
22	8	0:=
24	15	0:=
26	21	2:*
28	63	25:*
30	153	149:*
32	433	577:====*
34	1130	1565:===== *
36	2412	3213:===== *
38	4595	5310:===== *
40	6860	7408:===== *
42	8728	9055:===== *
44	10204	9988:===== *==
46	10709	10173:===== *===
48	10286	9740:===== *===
50	9525	8888:===== *===
52	8477	7814:===== *===
54	7042	6674:===== *==
56	5774	5575:===== *==
58	4499	4577:===== *
60	3812	3708:===== *==

```

62 2981 2972:=====*
64 2282 2364:=====*
66 1778 1868:=====*
68 1284 1470:=====*
70 1053 1152:=====*
72 791 900:=====*
74 561 702:====*
76 485 546:====*
78 311 424:===*
80 239 330:=*
82 212 252:=*
84 149 200:=*
86 105 155:*
88 87 120:*
90 55 93:*
92 50 72:* :=== *
94 43 55:* :===*
96 31 43:* :==*
98 23 33:* :=*
100 22 26:* :=*
102 17 20:* :=*
104 9 15:* :*
106 7 12:* :*
108 7 9:* :*
110 4 7:* :*
112 5 6:* :*
114 7 4:* :*
116 1 3:* :*
118 1 3:* :*
>120 850 2:*===== :*=====

```

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

```

ggamma.pep
PIR1:HGCZG

```

```

P1:HGCZG - hemoglobin gamma-G chain - chimpanzee
N;Alternate names: hemoglobin gamma-1 chain
C;Species: Pan troglodytes (chimpanzee)

```

SSearch

C;Date: 31-May-1996 #sequence_revision 21-Jan-1997 #text_change 14-Nov-1997
C;Accession: I36939; I61853
R;Slightom, J.L.; Chang, L.Y.; Koop, B.F.; Goodman, M. . . .

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      60
ggamma.pep    MGHFTEEDKATITSLWGKVNVEDAGGETLGRLLVVYPWTQRFDFSFGNLSSASAIMGNPK
                ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
HGCZG         MGHFTEEDKATITSLWGKVNVEDAGGETLGRLLVVYPWTQRFDFSFGNLSSASAIMGNPK
                10      20      30      40      50      60

                70      80      90     100     110     120
ggamma.pep    VKAHGKKVLTSLGDAIKHLDDLKGTFAQLSELHCDKLHVDPENFKLLGNVLVTVLAIHFG
                ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
HGCZG         VKAHGKKVLTSLGDAIKHLDDLKGTFAQLSELHCDKLHVDPENFKLLGNVLVTVLAIHFG
                70      80      90     100     110     120

                130     140
ggamma.pep    KEFTPEVQASWQKMVTGVASALSSRYH
                ||||||||||||||||||||||||||||
HGCZG         KEFTPEVQASWQKMVTGVASALSSRYH
                130     140
```

//

! Distributed over 1 thread.
! Start time: Fri Oct 16 11:56:55 1998
! 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 `GenEMBL:*`. 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

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?"

SSearch

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.

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

The Wisconsin Package™ 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, 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 ($\text{gapweight} + (\text{lengthweight} \times \text{gap length})$). 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 ($\text{gapweight} + (\text{lengthweight} \times (\text{gap length} - 1))$).

SSearch

"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

SSearch is one of the few programs in the Wisconsin Package 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.

ACKNOWLEDGEMENT

The FASTA program family (FastA, TFastA, FastX, TFastX, and SSearch) was written by Professor William Pearson of the University of Virginia Department of Biochemistry (Pearson and Lipman, Proc. Natl. Acad. Sci., USA 85; 2444-2448 (1988)). In collaboration with Dr. Pearson, the programs were modified and documented for distribution with GCG Version 6.1 by Mary Schultz and Irv Edelman, and for Versions 8 through 10 by Sue Olson.

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:

<code>[-INfile2=]PIR:*</code>	specifies the search set
<code>[-OUTfile=]ggamma.ssearch</code>	names the output file
<code>-BEGIN=1 -END=148</code>	sets the range of interest
<code>-EXPECT=2.0</code>	lists scores until E() value reaches 2.0

Local Data Files:

<code>-MATRIX=fastadna.cmp</code>	assigns the scoring matrix for nucleic acids
<code>-MATRIX=blosum50.cmp</code>	assigns the scoring matrix for proteins

Optional Parameters:

<code>-PROCESSORS=2</code>	sets the number of threads devoted to the analysis on a multiprocessor computer
<code>-MINLENGTH=1000</code>	searches only sequences of 1000 or more residues
<code>-MAXLENGTH=5000</code>	searches only sequences of 5000 or fewer residues
<code>-SINCE=6.90</code>	limits search to sequences dated on or after June 1990
<code>-ONESTRAND</code>	searches using only the top strand of nucleotide queries
<code>-GAPWEIGHT=16</code>	sets the gap creation penalty (12 is protein default)
<code>-LENGTHWEIGHT=4</code>	sets the gap extension penalty (2 is protein default)
<code>-LISTSIZE=40</code>	shows the best 40 scores (overrides EXPECT)
<code>-ALIGN=20</code>	shows the best 20 alignments
<code>-NOALIGN</code>	suppresses sequence alignments
<code>-SHOWALL</code>	shows complete sequences in alignment, not just overlaps
<code>-MARKX=3</code>	sets the alignment display mode
<code>-NOHISTOGRAM</code>	suppresses printing the histogram
<code>-LINESIZE=60</code>	sets number of sequence symbols per line of the alignment

SSearch

<code>-NODOCLines</code>	suppresses sequence documentation in the alignment
<code>-BATCh</code>	submits the program to run in the batch queue
<code>-NOMONitor</code>	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 `-MATRIX=mymatrix.cmp`. If you don't 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.

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.

`-MATRIX=mymatrix.cmp`

allows you to specify a scoring matrix file name other than the program default. If you don't 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.

`-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.

-LENGthweight=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.

SSearch

`-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.

`-NOHIS`togram

suppresses printing the histogram.

`-LINES`ize=60

lets you set the number of sequence symbols in each line of the alignment to any number between 60 and 200.

`-NODOC`Lines

suppresses the documentation from the search set sequence accompanying the alignment in the output file. Use `-DOC`Lines=5 to copy only five non-blank lines of documentation.

`-BAT`ch

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.

`-MON`itor=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 `-NOMON`itor.

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.

Printed: December 1, 1998 11:23 (1162)

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:
```

Name	Check	Begin	End	Dir
gamma.seq	6474	1	500	forward

```
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>.
```

StatPlot

```
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(1) GC(obsrv) GC_ob-ex(1) ..
      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

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

      443  31.000   14.000    0.000    -4.297    2.000    -2.297
      446  32.000   14.000    0.000    -4.435    2.000    -2.435
      449  32.000   13.000    0.000    -4.118    2.000    -2.118
```

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

*The Wisconsin Package 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 the Wisconsin Package 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.

StatPlot

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 Wisconsin Package 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.

StatPlot

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).

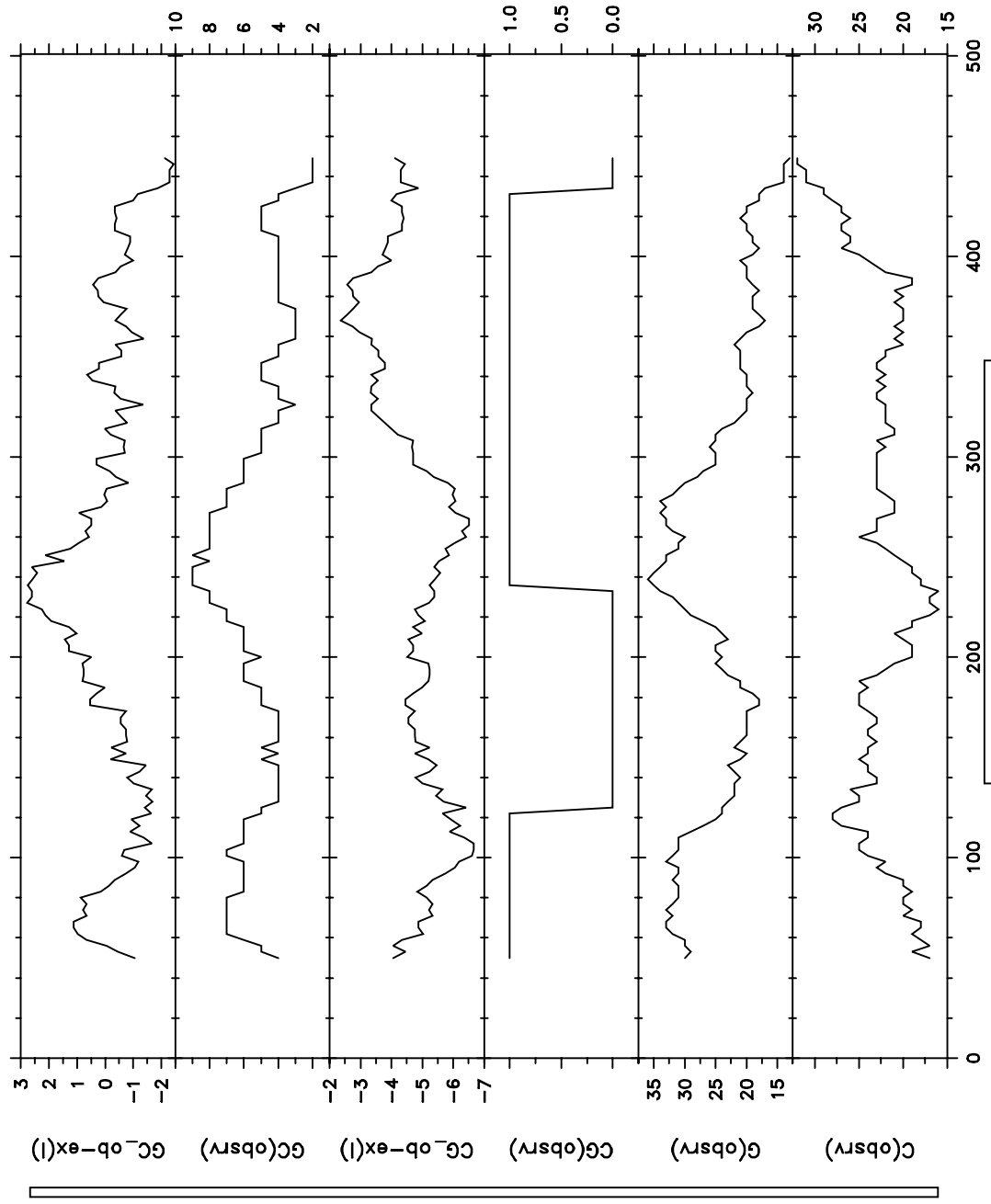
-YPAN=30.0

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: December 1, 1998 11:23 (1162)



STATPLOT on WINDOW of: Gamma,Seq check: 6474 from: 1 to: 500
Window: 100 Shift: 3 MatchType: Subset Mismatch: 0
Density: 23.15 October 24, 1996 16:27

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 ...
```

StemLoop

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
- 4) Choose new parameters,
- 5) 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 AGGCTGCAGTG AGCCGTGAT 11, 25
    ||||| |||
257 TCCGGCCTCAC GTCACCGCG 19
                               C
```

```
135 TAGCCGGGCGT GG 11, 22
    ||| || |||
160 GTCCGCGCGCG 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:

Loop	Start	End	Size	Quality
1	217	257	11	25
2	135	160	11	22
3	139	160	8	20
4	69	95	7	20
5	4	25	9	20
6	213	247	8	19
7	221	248	7	18
8	35	54	8	18

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.

StemLoop

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 `-PAIR`. *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 `-MATRIX`, 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`

Prompted Parameters:

<code>-BEGIN=1 -END=290</code>	sets the range of interest
<code>-STEMlength=6</code>	sets the minimum stem length
<code>-BONds=12</code>	sets the minimum bonds per stem
<code>-MINLoopsize=3</code>	sets the minimum loop size
<code>-MAXLoopsize=20</code>	sets the maximum loop size (distance to furthest inverted repeat)
<code>-MENu1=1</code>	specifies output type: 1=See stems, 2=See coordinates, 3=File, 4=DotPlot file
<code>-MENu2=1</code>	sorts by: 1=Position, 2=Quality, 3=Size
<code>-MAXSTems=25</code>	sets the maximum number of stems to show (quality or size sorts only)
<code>[-OUTfile=]alucons.stem</code>	names the output file

Local Data Files:

<code>-MATRix=stemloop.cmp</code>	assigns the scoring matrix for finding bonds/stem
-----------------------------------	---

Optional Parameters:

<code>-PAIr=1</code>	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 don't 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

StemLoop

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).

`-MENu1=1`

indicates the type of output. `-MENu1=1` means display the stems; other values for `-MENu1` are: 2) display the coordinates, 3) save the stems to a file, and 4) save the stem coordinates to a DotPlot file.

`-MENu2=1`

indicates how to sort the stems in the output. `-MENu2=1` means sort by position; other values for `-MENu2` are: 2) sort by quality and 3) sort by size.

`-MAXSTems=25`

sets the maximum number of stems to show (only applies when stems are sorted by quality or size).

-MATRix=mymatrix.cmp

allows you to specify a scoring matrix file name other than the program default. If you don't 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.

-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.

Stem structure nibbling also uses the threshold value set by this parameter to decide what pairs should be nibbled away from the structure. *You can set a pairing threshold high enough so that all stems are nibbled away!*

Printed: December 1, 1998 11:23 (1162)

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. GenEMBL 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 Wisconsin Package™ 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 Wisconsin Package programs. Sequence specification is described in detail in Chapter 2, Using Sequence Files and Databases of the User's Guide.

StringSearch

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 the Wisconsin Package are named using lowercase letters. Conversely, if you entered `gamma*`, this program would find all of the entries in the databases *and* all the 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) (* GenEMBL:* *) ?
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 (* genembl.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: genembl.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: GenEMBL:* October 22, 1998 11:38

! searching for: "pseudo" ..

Gb_ba:Ab000361 AB000361 Pseudomonas cichorii gene for D-Tagatose ...
Gb_ba:Ab001577 AB001577 Pseudomonas sp. DNA for low specificity ...
Gb_ba:Ab001722 AB001722 Pseudomonas stutzeri carbazole catabolic ...

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

Gb_sts:Ppu85462 U85462 Pseudomyrmex pallidus clone Psd2421AAG tri ...
Gb_sts:Ppu85463 U85463 Pseudomyrmex pallidus clone Psd2427AAG tri ...
Gb_sts:Ppu85464 U85464 Pseudomyrmex pallidus clone Psd2523CAC tri ...
! Sequences searched: 552323
```

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 `GenEMBL:*`; or by using a list file, for example `@project.list`, that contains the names of database sequences.

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`

StringSearch

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 *psuedo* appears 14 times in the definitions for GenBank Release 108.0!

Hyphenation is particularly prone to inconsistent usage. A search for *pseudogene* would only be complete if *pseudo-gene* (or *pseudo gene*) were never used to refer to pseudogenes.

Using a nonspecific pattern such as *pseudo* to find sequences of pseudogenes will result in many false matches. (In the example session, there were 1570 instances of *Pseudomonas*, 321 instances of *pseudoobscura*, and 42 instances of *pseudoautosomal* out of 6237 total matches.) But restricting the search by setting `-MATCH=AND` and searching for `pseudo,gene` will miss pseudogene sequences whose definitions use terms like *pseudoexon* or the prefix *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 and not what is not 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 `Primate:hum*` to look only at the group of sequences in which you really expect the text pattern to be found.

Use the expression `% typedata primate:hum*` to see some examples of sequence annotations.

StringSearch

-STRings=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: December 1, 1998 11:23 (1162)

SYMBOL

FUNCTION

Symbol creates, changes, deletes, or displays GCG symbol(s) from the GCG symbol table.

DESCRIPTION

Programs of the Wisconsin Package™ use *symbols* to refer to various optional values that control operation of the Wisconsin Package. 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 the Wisconsin Package. 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 the Wisconsin Package 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, Wisconsin 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.

Symbol

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 `.gcgrc` file. (See "Customizing Your Login" in Chapter 1, Getting Started of the User's Guide for information about the `.gcgrc` 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`

Prompted Parameters: None.

Optional Parameters:

<code>-Set identifier value</code>	changes the value of the symbol or adds a new symbol
<code>-Unset identifier-spec</code>	deletes all symbols matching identifier-spec
<code>-Name=identifier</code>	lets you enter a symbol identifier anywhere on the command line
<code>-Value=value</code>	lets you enter a symbol value anywhere on the command line
<code>-Quiet</code>	suppress all messages
<code>-KILL</code>	stops the symbols service for the session
<code>-List identifier-spec</code>	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.

Symbol

-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 the Wisconsin Package 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: December 1, 1998 11:23 (1162)