

# GAP

## FUNCTION

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

## DESCRIPTION

Gap considers all possible alignments and gap positions between two sequences and creates a global alignment that maximizes the number of matched residues and minimizes the number and size of gaps. A scoring matrix is used to assign values for symbol matches. In addition, a *gap creation penalty* and a *gap extension penalty* are required to limit the insertion of gaps into the alignment. Gap uses the alignment method of Needleman and Wunsch (J. Mol. Biol. **48**; 443-453 (1970)) that has been shown to be equivalent to Sellers (SIAM J. of Applied Math **26**; 787-793 (1974), see the CONSIDERATIONS topic below).

## EXAMPLE

Two haptoglobin genes are aligned with Gap. The alignment from this example is displayed graphically in the example for the GapShow program. The same sequences are compared in the figures included with DotPlot.

```
% gap

GAP of what sequence 1 ? hpr.seq

      Begin (* 1 *) ?
      End   (* 2966 *) ?
      Reverse (* No *) ?

to what sequence 2 (* hpr.seq *) ? hpf.seq

      Begin (* 1 *) ?
      End   (* 2740 *) ?
      Reverse (* No *) ?

What is the gap creation penalty (* 50 *) ?

What is the gap extension penalty (* 3 *) ?

What should I call the paired output display file (* hpr.pair *) ?

Aligning .....
          .....
          .....-
Aligning .....
          .....-
          .....
```



```

1949 TGGCCCCTAGCCCTTTCAATGAATTTTCAGGGAATTGTGAAAATTCCTTTG 1998
      |||||||||||||||||||||||||||||||||||||||||||||||||||
1711 ..GCCCCTAGCCCTTTCAATGAATTTTCAGGGAATTGTGGAAATTCCTTTA 1758

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

2935 GAGGACACCTGGTACGCGGCTGGGATCTTAAG 2966
      ||||||||||||||| ||| |||||||||||||||
2709 GAGGACACCTGGTATGCGACTGGGATCTTAAG 2740

```

**INPUT FILES**

Gap accepts two individual nucleotide sequences or protein sequences as input. The function of Gap 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**

When you want an alignment that covers the whole length of both sequences, use Gap. When you are trying to find only the best segment of similarity between two sequences, use BestFit. PileUp creates a multiple sequence alignment of a group of related sequences, aligning the whole length of all sequences. DotPlot displays the entire surface of comparison for a comparison of two sequences. GapShow displays the pattern of differences between two aligned sequences. PlotSimilarity plots the average similarity of two or more aligned sequences at each position in the alignment. Pretty displays alignments of several sequences. LineUp is an editor for editing multiple sequence alignments. CompTable helps generate scoring matrices for peptide comparison.

**RESTRICTIONS**

Input sequences may not be more than 32,000 symbols long.

**ALIGNING LONG SEQUENCES**

The program attempts to allocate enough computer memory to align the input sequences. In the worst case, where the two sequences being aligned are unrelated, the allocation is proportional to the product of the lengths of the two input sequences. However, in many cases where the sequences being aligned are more closely related, the computer can determine an optimal alignment using less memory. When memory on your computer is limiting and the program cannot allocate all of the memory it needs to align long sequences, it completes the alignment in whatever memory it can allocate and displays the message `*** Alignment is not guaranteed to be optimal ***`. Because the criteria used in the calculation for guaranteeing an optimal alignment are very stringent, the alignment often may be optimal even if this message is displayed.

## Gap

If you know roughly where the alignment of interest for long sequences begins, you can run the program with `-LIMIT`. Then set the starting coordinates for each sequence near the point where the alignment of interest begins and set gap shift limits on each sequence. The program then aligns the sequences from your starting point such that the sequences do not get out of phase by more than the gap shift limits you have set. If you started both sequences at base number one and set the gap shift limit for sequence one to 100 and for sequence two to 50, then base 350 in sequence one could not be gapped to any base outside of the range from 300 to 450 on sequence two. These *limited* alignments often require less computer memory than unlimited alignments.

### EVALUATING ALIGNMENT SIGNIFICANCE

This program can help you evaluate the significance of the alignment, using a simple statistical method, with `-RANDOMIZATIONS`. The second sequence is repeatedly shuffled, maintaining its length and composition, and then realigned to the first sequence. The average alignment score, plus or minus the standard deviation, of all randomized alignments is reported in the output file. You can compare this average *quality* score to the quality score of the actual alignment to help evaluate the significance of the alignment. The number of randomizations can be specified by adding an optional value to `-RANDOMIZATIONS`; the default is 10. You can preserve the dinucleotide or dipeptide composition of the input sequence in the shuffled sequence by using `-PRESERVE=2`. Use `-PRESERVE=3` to preserve the trinucleotide or tripeptide composition of the input sequence.

The score of each randomized alignment is reported to the screen. You can use `<Ctrl>C` to interrupt the randomizations and output the results from those randomized alignments that have been completed.

By ignoring the statistical properties of biological sequences, this simple Monte Carlo statistical method may give misleading results. Please see Lipman, D.J., Wilbur, W.J., Smith, T.F., and Waterman, M.S. (Nucl. Acids Res. **12**; 215-226 (1984)) for a discussion of the statistical significance of nucleic acid similarities.

### CONSIDERATIONS

#### Other Tools May Be Better Than Gap

Gap is capable of ignoring a region of excellent similarity or similarity between two sequences if it can produce an alignment with equal or better quality in some other way. BestFit is a better tool to search for weak or unknown similarity or similarity that you suspect is not coextensive along the sequences. It is extremely important that you think formally about what Gap does. Using Gap rather than BestFit implies that you want an alignment where neither sequence is truncated.

Gap presents you with one member of the family of best alignments. There may be (and usually are) many members of this family, but no other member has a better *quality*. When two sequences are closely related, Gap is a good way to see the relationship between them; however, a gapped alignment obscures, or can even be confounded by, internal repeats. Graphic matrix analysis is more powerful for seeing internally repeated structures and approximating the frame of best alignment between two sequences that have never been previously compared. (See the Compare and DotPlot programs.)

## Scoring Matrices

The modification of scoring matrices is discussed in Appendix VII.

There is considerable evidence that more sensitive nucleic acid alignments may be possible by scoring transitions slightly positive and transversions slightly negative.

Gap 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.) You can use `-GAPweight` and `-LENGTHweight` to specify alternative gap penalties if you don't want to accept the default values.

CompTable helps you create scoring matrices based on a simplification scheme for amino acid differences. There is also a short C program that can be modified to help you write a new scoring matrix quickly. The program is called `cmpvals.c`, and it is located in the public database. You may Fetch and modify `cmpvals.c` if you are comfortable working with the C programming language.

## Forced Pairing

You can get a position in sequence one to pair with some other position in sequence two by choosing a special symbol not used in the rest of the sequences and giving it a very high match value in the scoring matrix. The alphabet of legitimate GCG sequence symbols is defined in Appendix III.

## Needleman-Wunsch Versus Sellers

Gap makes an alignment to find the maximum similarity between two sequences by the method of Needleman and Wunsch (J. Mol. Biol. **48**; 443-453 (1970)) that is similar to finding the minimum difference according to the method of Sellers (SIAM J. of Applied Math **26**; 787-793 (1974)). Smith, Waterman, and Fitch (J. Mol. Evol. **18**; 38-46 (1981)) showed that the methods were precisely equivalent when the Needleman and Wunsch gap creation penalty is equal to the Sellers gap creation penalty - 0.5 and when the end gaps for Needleman and Wunsch are penalized in same way as all the other gaps. `-ENDweight` allows you to penalize the end gaps introduced by Gap.

## Rapid Alignment

When possible, Gap tries to find the optimal alignment very quickly. If this rapid alignment is not unambiguously optimal, Gap automatically realigns the sequences to calculate the optimal alignment. When this occurs, the monitor of alignment progress on your terminal screen (`Aligning...`) is displayed twice for a single alignment.

## ALGORITHM

Gap reads a scoring matrix that contains values for every possible GCG symbol match. Gap finds an alignment with the maximum possible quality where the quality of an alignment is equal to the sum of the values of the matches (each match scored with the scoring matrix) less the *gap creation penalty* times the number of internal gaps and less the *gap extension penalty* times the total length of the internal gaps. The alignment found by Gap is, therefore, sensitive to the scoring matrix values and the gap penalties. There is no penalty if either sequence is shifted to the place where the alignment begins unless *end gaps* are penalized by using `-ENDweight`.

# Gap

## ALIGNMENT METRICS

BestFit and Gap display four figures of merit for alignments: Quality, Ratio, Identity, and Similarity.

The Quality (described above) is the metric maximized in order to align the sequences. Ratio is the quality divided by the number of bases in the shorter segment. Percent Identity is the percent of the symbols that actually match. Percent Similarity is the percent of the symbols that are similar. Symbols that are across from gaps are ignored in the calculation of Percent Identity and Percent Similarity. A similarity is scored when the scoring matrix value for a pair of symbols is greater than or equal to the average positive non-identical comparison value in the matrix, the *similarity threshold*. This threshold is also used by the display procedure to decide when to put a ':' (colon) between two aligned symbols. You can change this threshold by specifying optional values to `-PAIr`. For instance, `-PAIr=10,5` would set the similarity threshold to 5.

*The similarity and identity metrics are not optimized by alignment programs so they should not be used to compare alignments.*

## PEPTIDE SEQUENCES

If your input sequences are peptide sequences, this program uses a scoring matrix, `blosum62.cmp`, with comparison values derived from a study of substitutions between amino acid pairs in ungapped block of aligned protein segments as measured by Henikoff and Henikoff (Proc. Natl. Acad. Sci. USA **89**; 10915-10919 (1992)).

## 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: `% gap [-INfile1=]hpr.seq [-INfile2=]hpf.seq -Default`

Prompted Parameters:

<code>-BEGIN1=1</code>	<code>-BEGIN2=1</code>	sets the beginning of each sequence
<code>-END1=2966</code>	<code>-END2=2740</code>	sets the end of each sequence
<code>-NOREV1</code>	<code>-NOREV2</code>	sets the strand of each sequence
<code>-GAPweight=50</code>		sets the gap creation penalty (8 is protein default)
<code>-LENGTHweight=3</code>		sets the gap extension penalty (2 is protein default)
<code>[-OUTfile1=]hpr.pair</code>		name the alignment output file

Local Data Files:

<code>-MATRIX=nwsgapdna.cmp</code>	assigns the scoring matrix for nucleic acids
<code>-MATRIX=blosum62.cmp</code>	assigns the scoring matrix for proteins

Optional Parameters:

<code>-OUTfile2=hpr.gap</code>	names new file for sequence 1 with gaps added
<code>-OUTfile3=hpf.gap</code>	names new file for sequence 2 with gaps added
<code>-PENalizedlength=12</code>	penalizes gaps longer than 12 sequence characters the same as gaps of length 12

```

-LIMit1=1 -LIMit2=240  limits the surface of comparison
-RANdomizations[=10]  determines average score from 10 randomized
                      alignments
  -PREServe=2          preserves dinucleotide or dipeptide composition
                      in randomized sequence
                      =3          preserves trinucleotide or tripeptide composition
                      in randomized sequence
-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
-ENDWeight          penalizes end gaps like other gaps
-HIGHroad           makes the top alignment for your parameters
-LOWroad            makes the bottom alignment for your parameters
-NOSUMmary          suppresses the screen summary

```

## ACKNOWLEDGEMENTS

Gap and BestFit were originally written for Version 1.0 by Paul Haeberli from a careful reading of the Needleman and Wunsch (*J. Mol. Biol.* **48**; 443-453 (1970)) and the Smith and Waterman (*Adv. Appl. Math.* **2**; 482-489 (1981)) papers.

Limited alignments were designed by Paul Haeberli and added to the Package for Version 3.0. They were united into a single program by Philip Delaques for Version 4.0.

## 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.

Gap reads a scoring matrix from your local directory or the public database with the values for every possible match. The file `nwsgapdna.cmp` (NWS stands for Needleman, Wunsch, and Sellers) has a 10 at every place where the set of bases implied by the alphabetic IUB ambiguity codes (see Appendix III) overlap. All of the other locations have zeros. In the file `blosum62.cmp`, the scores for pairwise amino acid comparisons range from -4 to +11. You can use the Fetch program to copy, view, and possibly modify these scoring matrix files to suit your own needs.

# Gap

## 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.

**-REVerse1** and **-REVerse2**

sets the program to use the reverse strand for the two input sequences.

**-GAPweight=8**

sets the gap creation penalty that is subtracted from the alignment score whenever a gap is created.

**-LENgthweight=2**

sets the gap extension penalty that is subtracted from the alignment score for each gapped symbol.

**-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.

**-OUTfile2=seqname1.gap -OUTfile3=seqname2.gap**

This program can write three different output files. The first displays the alignment of sequence one with sequence two. The second is a new sequence file for sequence one, possibly expanded by gaps to make it align with sequence two. The third, like the second, is a new sequence file for sequence two, possibly expanded by gaps to make it align with sequence one. The program writes only the first file unless there are output file options on the command line. If there are any output files named on the command line, *only* those output files are written. If you add **-OUT** to the command line without an accompanying file name, then the program will write the second and third output files after prompting you for their names.

Aligned sequences (in sequence files) can be displayed with GapShow. Their similarity can be displayed with PlotSimilarity.

**-PENAlizedlength=12**

lets you set the maximum penalty for any gap in the alignment. For instance, if you specify **-PENAlizedlength=12**, then any gap longer than 12 characters is penalized the same as a gap of length 12. Using this parameter, alignments can contain large gaps without incurring large gap extension penalties. This may be useful, for instance, if you are aligning a cDNA sequence with the corresponding genomic DNA sequence containing large introns.



**-LiMit1=20** and **-LiMit2=20**

let you set *gap shift limits* for each sequence. When you already know of a long similarity between two sequences you can "zip" them together using this mode. The beginning coordinates for each sequence must be near the beginning of the alignment you want to see. The alignment continues so that gaps inserted do not require the sequences to get out of step by more than the gap shift limits. You can align very long sequences rapidly. When you set gap shift limits for one or both input sequences, the maximum surface of comparison available to your alignment is 3.5 million. The size of the surface of comparison that your alignment actually requires can be predicted by multiplying the average length of the two sequences by the sum of the two shift limits.

If you add just **-LiMit** to the command line without specifying any value, the program prompts you to enter gap shift limits for each sequence.

**-RANdomizations=10**

reports the average alignment score and standard deviation from 10 randomized alignments in which the second sequence is repeatedly shuffled, maintaining the length and composition of the original sequence, and then aligned to the first sequence. You can use the optional parameter to set the number of randomized alignment to some number other than 10.

**-PRESeRve=2**

preserves the dinucleotide or dipeptide composition of the original sequence in the shuffled sequence. Use **-PRESeRve=3** to preserve the trinucleotide or tripeptide composition.

**-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**.

# Gap

## **-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.

## **-ENDWeight**

causes the end gaps to be penalized in the same way as all other gaps.

## **-LOWroad** and **-HIGHroad**

The insertion of gaps is arbitrary in many cases, and equally optimal alignments can be generated by inserting gaps differently. When equally optimal alignments are possible, this program can insert the gaps differently if you select either the **-LOWroad** or the **-HIGHroad** parameter. Here are examples for the alignment of GACCAT with GACAT with different parameters.

```
For:      Match = 10          MisMatch = -9
          Gap weight = 10     Length Weight = 0
```

```
LowRoad:  1 GACCAT 6
           || |||      Quality = 40
           1 GA.CAT 5
```

```
HighRoad: 1 GACCAT 6
           ||| ||      Quality = 40
           1 GAC.AT 5
```

```
For:      Match = 10          MisMatch = 0
          Gap weight = 30     Length Weight = 0
```

```
HighRoad: 1 GACCAT 6
           |||          Quality = 30
           1 GACAT. 5
```

```
LowRoad:  1 GACCAT 6
           |||          Quality = 30
           1 .GACAT 5
```

Essentially the *low road* shifts all of the arbitrary gaps in sequence two to the left and all of the arbitrary gaps in sequence one to the right. The *high road* does exactly the opposite. When neither *high road* nor *low road* is selected, the program tries not to insert a gap whenever that is possible and uses the high road alternative for all collisions.

## **-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 10:25 (1162)

## GAPSHOW+

### FUNCTION

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

### DESCRIPTION

BestFit and Gap make optimal alignments of sequences by adding gaps to maximize the number of matches. Gap and BestFit normally display the alignments, but they can also write the aligned sequences (with gaps inserted) into new sequence files. GapShow reads these files and plots the distribution of the differences or similarities in the alignment graphically (see figure below). The sequences are represented by horizontal lines. The horizontal lines have openings at points where there are gaps in either sequence. Regions of interest, such as coding regions, can be shown outside these lines (see the LOCAL DATA FILES topic). A large vertical line between the sequences can indicate either a difference or similarity (see the OPTIONAL PARAMETERS topic below). If differences are being shown, gaps are also depicted with short vertical lines to convey that the gap is a difference between the two sequences. You can reproduce the original numbering by using command-line parameters (see below).

When run with the command-line parameter `-OUTfile`, GapShow writes a file like the paired output file from Gap and BestFit. The two sequences are displayed one above the other with vertical bars (|) marking the positions where the symbols are similar. Some people use GapShow with this parameter to see the differences between very similar sequences.

### EXAMPLE

Here is a session using GapShow to display the alignment of `hpr.gap` to `hpf.gap` both graphically and base-by-base:

```
% gapshow -OUTfile=hpr.pair
```

```
GAPSHOW of what sequence 1 ? hpr.gap
```

```
      Begin (* 1 *) ?
      End (* 2982 *) ?
```

```
to what sequence 2 ? hpf.gap
```

```
      Begin (* 1 *) ?
      End (* 2982 *) ?
```

```
When your LaserWriter attached to tty07 is ready, press <Return>.
```

```
Output file: "hpr.pair"
```

```
%
```





# GapShow

## 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.

## 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: % `gapshow [-INfile1=]hpr.gap [-INfile2=]hpf.gap -Default`

### Prompted Parameters:

`-BEGIN1=1 -END1=2982` sets the range of interest for sequence 1  
`-BEGIN2=1 -END2=2982` sets the range of interest for sequence 2

### Local Data Files:

`-MATRIX=swgapdna.cmp` assigns the scoring matrix for nucleic acids  
`-MATRIX=blosum62.cmp` assigns the scoring matrix for proteins  
`-MARK1=hpr.mrk` defines regions of known interest on sequence 1  
`-MARK2=hpf.mrk` defines regions of known interest on sequence 2

### Optional Parameters:

`[-OUTfile1=]hpr.pair` makes a paired output file like the one from GAP  
    `-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  
`-NOPLot` suppresses the plot  
`-BARs=d` plots bars for (s)imilarities or (d)ifferences  
`-CONTinuous` represents the sequences as continuous (unbroken) lines  
`-DENSity=3000` plots 3,000 sequence symbols per page (100 ppu)  
`-NONUMbering` suppresses numbering of both sequences  
`-NOLABELing` suppresses labeling of both sequences

```

-NUm1=1352           starts numbering sequence 1 at 10
-REvNUM1            numbers sequence 1 downwards (default is upwards)
-NUm2=-500          starts numbering sequence 2 at -500
-REvNUM2            numbers sequence 2 downwards (default is upwards)

```

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

```

## ACKNOWLEDGEMENT

The graphics output from GapShow was designed by Professor Oliver Smithies. The program was written by John Devereux.

## 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.

GapShow uses a scoring matrix to define when a symbol in sequence one is similar to a symbol in sequence two. See Appendix VII for more information about scoring matrices. If two symbols have a comparison value of greater than or equal to the average positive non-identical comparison value in the matrix, GapShow puts a bar in the text file and on the plot.

The scoring matrices used by GapShow are the same ones used by Gap and BestFit to make the alignments. If, in your directory, you have the matrix file `swgapdna.cmp` for nucleotide alignments made with BestFit, or `nwsgapdna.cmp` for nucleotide alignments made with Gap, or `blosum62.cmp` for protein alignments made with either BestFit or Gap, your local matrix file is used by GapShow instead of the public versions. GapShow uses `swgapdna.cmp` for nucleotide alignments if the first line of the first sequence file does not contain the string "GAP "; otherwise `nwsgapdna.cmp` is used.

# GapShow

## Marking Files

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. The files `hpr.mrk` and `hpf.mrk` were present when the figure below was made. The coordinates from the marking file are interpreted in the numbering mode chosen.

## 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.

`-MARk=hpr.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.

`-OUTfile=hpr.pair`

makes a text output file showing the alignment in a format similar to the output files from `Gap` or `BestFit`. If you do not supply a name, `GapShow` creates a name for it using the name of the first input sequence with the file name extension `.pair`. The file from the example session is shown above under the `OUTPUT` topic.

These four parameters format the output in the text file.

`-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.

## **-NOPLot**

suppresses the plot.

## **-BARs=d**

By default, GapShow plots bars at the positions where the aligned sequences differ. This parameter lets you choose to show the points of similarity (s) or difference (d).

## **-CONTinuous**

Usually, GapShow represents the sequences as horizontal lines. Where there are gap characters ( `.` and `~` ) in either sequence, GapShow interrupts the line. Use this parameter to make these lines continuous across the gaps.

## **-DENsity=1000**

sets the number of bases or amino acids per 100 platen units (PU). This is usually equivalent to the number of bases or amino acids per page. Output from different GCG graphics programs that are run at the same density can be compared by lining up the plots on a light box.

## **-NONUMbering**

suppresses the numbering of both sequences.

# GapShow

## **-NOLABELing**

suppresses the titles above and below the plot. Look at the Figure program to see a nice way to annotate GCG plots.

These five parameters affect the numbering of each sequence.

## **-NUM1=1352**

begins numbering the first sequence at 1352.

## **-REVNUM1**

numbers the first sequence downwards instead of upwards. If you do not set a beginning number, the first number is equal to the length of the sequence (not counting the gaps).

## **-NUM2=-500**

begins numbering the second sequence at -500.

## **-REVNUM2**

numbers the second sequence downwards instead of upwards. If you do not set a beginning number, the first number is equal to the length of the sequence (not counting the gaps).

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.

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 10:25 (1162)



## GCGToBLAST

### FUNCTION

GCGToBLAST combines any set of GCG sequences into a database that you can search with BLAST.

### DESCRIPTION

BLAST can search only databases that have been compressed into a special format. Such databases must be searched in their entirety. GCGToBLAST is provided to allow you to create a BLAST-searchable database from a group of sequences that interest you.

GCGToBLAST accepts any GCG multiple sequence specification as input and creates the three or four output files necessary for BLAST. These files share a common base name (the database name) and must be kept together in the same directory.

The output is written into your current working directory. If you want your output written into another directory use the command-line parameter `-DIRectory=/usr/user/burgess/seq/`.

### EXAMPLE

Here is a session with GCGToBLAST that converts all the sequences specified by `hsp70.list` into a database suitable for input to BLAST.

```
% gcgtoblast

GCGTOBLAST of what input sequence(s) ? @hsp70.list

What should I call the database ? hsp70

      PIR1:JU0062      675 characters.
      PIR2:A25646      634 characters.

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

      PIR1:S05776      682 characters.
      PIR2:B36590      642 characters.
      PIR1:S29261      638 characters.

GCGTOBLAST complete:

      Sequences: 25
      Symbols: 16,073
      Output files in: .

%
```

### OUTPUT

GCGToBLAST writes three or four files in your current working directory unless you redirect the output with the `-DIRectory` parameter.

# GCGToBLAST

## INPUT FILES

GCGToBLAST accepts multiple sequences of the same type. 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

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

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.

## RESTRICTIONS

*All the sequences compressed by GCGToBLAST must be the same type, that is all nucleotide or all protein!* The output files must be kept together in the same directory.

## SPECIFYING DATABASES TO BLAST

By default BLAST does local searches by reading files from the directory whose logical name is BLASTDB. Each database known to BLAST is named in one of the three local data files: `blast.rdbs`, `blast.ldbs`, and `blast.sdbs`, so if your BLAST-searchable database is in some other directory, you have to name that directory as part of the search set specification to BLAST. For instance you could use a specification like `/usr/user/burgess/seq/mydatabase` that includes both the directory name and the name of the BLAST-searchable database (`mydatabase` in this example).

## CONSIDERATIONS

The compressed representation of nucleotide sequences in the output from GCGToBLAST is not rich enough to represent nucleotide ambiguity codes accurately. So in addition to the compressed form of the database, GCGToBLAST writes an ASCII version of the data in what is becoming known as FastA format. If the sequences in the database are nucleotide, and if they contain ambiguous symbols, GCGToBLAST saves this file. BLAST uses it to display any sequences found in a search that contain ambiguous symbols.

The FastA format file can be large. If the display of the correct original ambiguity codes in your segment pair output is not important to you, you might want to delete this file or use GCGToBLAST with the `-DELEte` parameter so that GCGToBLAST will delete it for you once the database is created. GCGToBLAST automatically deletes this file if the sequences in the data set are proteins, since the compressed amino acid codes can express ambiguity.

## 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: % gcgtoblast [-INfile=]@hsp70.list [-OUTfile=]hsp70 -Default

Prompted Parameters: None

Local Data Files: None

Optional Switches:

-DIRectory=dirname	writes into a directory other than the current directory
-NOMONitor	suppresses the screen monitor
-NOSUMmary	suppresses the screen summary
-OLDblast	create database for pre-2.0 versions of BLAST
-NODElete	don't delete FASTA files when running pre-2.0 BLAST
-BATch	submits the program to run in the batch queue

## 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.

### **-DIRectory=DirName**

This parameter allows you to redirect the output files written by GCGToBLAST to a directory other than your current working directory.

### **-MONitor**

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

### **-SUMmary**

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

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

# GCGToBLAST

## **-OLDblast**

directs GCGToBLAST to create pre-BLAST 2.0-compatible databases. Such databases are incompatible with BLAST versions 2.0 or higher.

## **-DELEte**

directs GCGToBLAST to delete the FastA-format version of a nucleotide sequence database that it creates in addition to the compressed database. You would do this to free up disk space if the display of the correct original ambiguity codes in the output of a BLAST search is not important to you. The FastaA file is deleted by default. If you use the **-OLDblast** parameter, then the FastaA file is deleted by default only if you are creating a protein database. Use **-NODELEte** if you want to retain the FastA file when not building pre-BLAST 2.0 nucleic acid databases.

## **-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.

Printed: December 1, 1998 10:25 (1162)



## GELASSEMBLE

### FUNCTION

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

### DESCRIPTION

See the Fragment Assembly System (FAS) Introduction for an overview of working with the programs within the FAS to assemble sequences in a sequencing project.

GelMerge takes unassembled fragment sequences in a sequencing project database and creates complete assemblies, called contigs. Each contig is a multiple alignment of contiguous, overlapping sequences in the project database. You can edit these sequence alignments in GelAssemble, allowing you to improve the alignments and resolve inconsistencies among the aligned fragments in regions of overlap. With GelAssemble, you can also combine separate contigs into a single assembly, or conversely split a single assembly into more than one contig. By allowing you complete freedom to manipulate the contigs in the project database, without relying solely on the automatic assembly process of GelMerge, you maintain complete control over your sequencing project.

GelAssemble also provides commands to export information from the project database into your local directory. You can copy any fragment or consensus sequence to a file in your current working directory where you can use it as input to other GCG programs. You can write a *Big Picture* schematic representation of the alignment of fragments in a contig. You can also write the contig's aligned sequences into a file that resembles the output displayed by the Pretty program.

### EXAMPLE

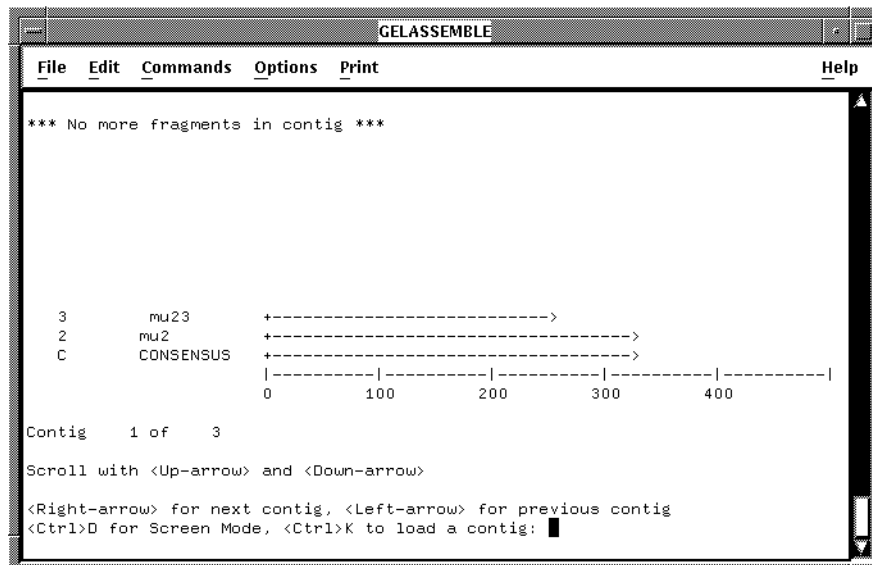
The example screens below are from a session with GelAssemble using "myproject", the fragment assembly project created in the example sessions of GelStart, GelEnter, and GelMerge.

```
% gelassemble
```

# GelAssemble

## SELECTING A CONTIG

When you first run GelAssemble, your screen looks similar to this:



```
GELASSEMBLE
File Edit Commands Options Print Help
*** No more fragments in contig ***

3      mu23      +----->
2      mu2       +----->
C      CONSENSUS +----->
                        |-----|
                        0       100     200     300     400

Contig  1 of  3

Scroll with <Up-arrow> and <Down-arrow>

<Right-arrow> for next contig, <Left-arrow> for previous contig
<Ctrl>D for Screen Mode, <Ctrl>K to load a contig: █
```

The first contig in the sequencing project is displayed on the screen. Each contig is named after the first fragment in the contig -- the fragment on line 2 of the screen display. All of the available commands are displayed at the bottom of the screen.

### Scrolling Through the Contig List

You can scroll through the list of contigs using the <Left-arrow> and <Right-arrow> keys to display the preceding and following contigs, respectively. If there is not enough room for the entire contig to display on the screen at once, you can use the <Up-arrow> and <Down-arrow> keys to scroll a line at a time.

### Selecting a Contig for Editing

You can choose a new contig to load into the GelAssemble *Edit Screen* by pressing <Ctrl>K when that contig is displayed on the screen. You can move to the Edit Screen without selecting a new contig by pressing <Ctrl>D. If a contig was previously loaded into the Edit Screen, press <Ctrl>D to return to that contig.



## GelAssemble

When the cursor is positioned in the alignment, you can enter *Screen Mode* commands to move the cursor around the alignment, edit the fragment sequences, and modify the alignment. These commands are described below under the SCREEN MODE topic. From Screen Mode, use <Ctrl>D to enter *Command Mode*. The cursor moves down to the lower-left corner of the screen next to the Command: prompt where you can type commands that enable you to: 1) save your changes to the contig to the sequencing project database; 2) write a file containing the contig sequence alignment; 3) export any fragment or consensus sequences out of the Fragment Assembly project database into individual sequence files for use with other GCG programs; and 4) choose a different contig for editing. These commands are described below under the COMMAND MODE topic. There is some overlap in function between Screen Mode and Command Mode commands. In general, Screen Mode commands either move the cursor or edit a sequence, while Command Mode commands manipulate entire sequences or contigs.

### SCREEN MODE

#### Moving the Cursor

In Screen Mode, the cursor shows your position in the current fragment sequence. You can move around in the sequence, add and delete symbols, and search for patterns. Use the <Left-arrow> and <Right-arrow> keys to move the cursor within a single sequence. Use the <Up-arrow> and <Down-arrow> keys to move the cursor to different sequences on different lines in the Edit Screen. The entire alignment is loaded, but only a portion of it may be visible at any time on your terminal screen. If you try to move the cursor near the edge of the screen with the <Left-arrow> and <Right-arrow> keys, the display scrolls to show more of the alignment in that direction. If you try to move beyond the top or bottom of the currently-displayed alignment, the display scrolls to show additional sequences in that direction. Use the < and > keys to scroll the display horizontally, one screen at a time. These keys provide a quick way to review the entire alignment.

You can also quickly move through the alignment. For example, type a number and press <Return> to move the cursor to that position in the current sequence. Use <Ctrl>E to move the cursor to the end of the current sequence. Use <Ctrl>A to move the cursor to the next position in the alignment where at least one of the sequences disagrees with the consensus symbol. Use <Ctrl>R to move the cursor to the next position where the current fragment sequence disagrees with the consensus symbol. Use <Ctrl>V to move the cursor to the next position in the alignment where a gap character is found in the contig consensus.

Other commands for moving the cursor to different parts of the alignment are described under the COMMAND MODE topic, below.

#### Expanding the Alignment Display

A contig may contain so many sequences that overlap at the same position, that the entire alignment at that position cannot be viewed on the screen at once. Use <Ctrl>L to remove the *Big Picture* display from the screen and display additional lines of the sequence alignment. You can toggle the screen to restore the *Big Picture* display by pressing <Ctrl>L again.

## Editing a Sequence

You can insert any valid GCG sequence symbol (see Appendix III) into the sequence by typing the symbol; it is inserted at the cursor. <Ctrl>H or the <Delete> key deletes symbols to the left of the cursor, one by one. Any sequence symbol, <Ctrl>H, and the <Delete> key can be preceded by a number, indicating how many symbols to add or delete. For example, use 10<Delete> to removed the 10 symbols to the left of the cursor. You can edit a different fragment sequence on the screen simply by moving the cursor to the line containing that sequence.

<Ctrl>O toggles between *Insert* and *OverStrike* mode for editing. *OverStrike* mode can be useful for changing sequence symbols because it first deletes a symbol and then inserts a new symbol at the same position. The current editing mode is indicated in the lower-right corner of the Edit Screen.

To delete an entire column of the alignment, position the cursor on that column and press <Ctrl>X. Each time you use <Ctrl>X, the entire contig to the right of the cursor shifts to the left by one position to fill in the deleted column and maintain the appropriate alignment. Use <Ctrl>I to restore an entire column that was just deleted with <Ctrl>X. You only can restore a deleted column if you haven't moved the cursor away from the position of deletion on the screen. If you've deleted adjacent columns in the alignment by sequentially pressing <Ctrl>X several times, you can restore those deleted columns by sequentially pressing <Ctrl>I the same number of times. Up to 10 adjacent columns in the alignment can be restored with <Ctrl>I; if you've deleted more than 10, only the last 10 are restored.

## Removing Sequences from a Contig

You may, at times, wish to remove individual fragment sequences from a contig without beginning the assembly process from scratch in GelMerge. For instance, GelMerge may have included a sequence in a contig that you think should not be aligned as part of that contig. You can remove any sequence from the current contig by placing the cursor on the line containing that sequence and pressing the minus key (-). If the remaining alignment is subsequently written to the sequencing project database as a new contig (see COMMAND MODE for more information), the removed fragment will be written to the sequencing project database as a *single-fragment contig*. If you quit GelAssemble without writing the contig to the sequencing project database, the fragment remains part of the larger contig.

You also can manually assemble separate contigs that do not share sufficient overlap to be automatically assembled by GelMerge. This function is available only in Command Mode (see COMMAND MODE for more information).

## The Consensus Sequence

The consensus sequence on the bottom row of the sequence alignment is not updated automatically when you edit any of the fragment sequences. If you add a new sequence to the current contig (see COMMAND MODE for more information) the existing consensus sequence is not completely recalculated; only that part of the new fragment that extends beyond the ends of the current consensus is added to the consensus sequence. If you remove a sequence from the current contig (see "Removing Sequences from a Contig", above), only that part of the removed fragment that extended beyond the ends of the remaining alignment is removed from the consensus sequence.

## GelAssemble

To completely recalculate the consensus sequence from the fragment sequences currently loaded into the GelAssemble editor, use <Ctrl>G. This replaces what was on row 1 with the computed consensus. For information on recalculating only part of the consensus sequence, see the Command Mode Summary, below.

The consensus function is a simple measure of plurality. IUB nucleotide ambiguity symbols in the fragment sequences are treated as weighted representations of their constituent bases for the purpose of generating a consensus. For example, an R represents half A and half G. If there is no absolute plurality (that is, two or more bases are tied), then those bases tied for plurality are used to generate an IUB nucleotide ambiguity symbol for the consensus. If the most common symbol at a position is a gap character ( . and ~), then the consensus contains a gap character at that position. If there is no unanimity among the loaded fragments at a particular column, then the displayed consensus symbol for that column is in lowercase.

The consensus sequence can be edited directly, just like other fragment sequences.

### Finding Patterns

To search for a pattern, type a / (slash) in Screen Mode. The cursor moves to the lower-left corner of the screen where you can enter the sequence pattern you want to find. You can repeat the last search by simply using /<Return>. The search is case-insensitive, and GelAssemble does not understand nucleotide ambiguity symbols in pattern matching. For instance, /RAC matches RAC, but not GAC.

### Positioning Fragments in the Contig Manually

To shift the entire sequence to the left, position the cursor at the beginning of the sequence and press <Delete>. To move the sequence to the right, place the cursor anywhere within the sequence and press the <Space Bar>. To move two or more positions either way, type the number of positions you want to move and press the appropriate key. For example, to move the sequence three positions to the right, type 3<Space Bar>.

Sequences aligned automatically with GelMerge should rarely require manual positioning.

### Selecting Ranges for Deletion and Insertion

You can delete a sequence range from one position and insert it at another. To delete a sequence range, move the cursor to the beginning of the section you want to delete. Press <Ctrl>B and use the <Left-arrow> and <Right-arrow> keys to select the sequence range. The selected sequence appears in reverse video. Press <Ctrl>N to delete the selected range and temporarily place it in a buffer. **Note:** Only the last deleted range is saved in the temporary buffer.

To insert the last deleted range into the alignment again, position the cursor at the point of insertion and press <Ctrl>P. The sequence range appears at the new position.

### Leaving Screen Mode

Use <Ctrl>D to leave Screen Mode and enter Command Mode.

Remember, to save any changes you make to an alignment, you must explicitly write the contig to the sequencing project database using an appropriate Command Mode command. (See "Writing Contigs" under the COMMAND MODE topic for more information.)

## Screen Mode Summary

Here is the summary of Screen Mode commands you would see by typing `?` in Screen Mode:

### SCREEN MODE

[n] is an optional numeric parameter

Keys Pressed	Action
[n]<Right-arrow>	move ahead [n bases]
[n]<Left-arrow>	move back [n bases]
[n]<Up-arrow>	move up [to row n]
[n]<Down-arrow>	move down [to row n]
>	scroll one screen to the right
<	scroll one screen to the left
1<Return>	move to start of the sequence
<Ctrl>E	move to end of the sequence
165<Return>	move to base 165 in sequence
/GATTC<Return>	find next occurrence of GATTC
<Ctrl>A	move to next ambiguity in alignment
<Ctrl>R	move to next ambiguity in sequence
<Ctrl>V	move to next gap in consensus
<Ctrl>D	enter Command Mode
<Ctrl>L	toggle alignment display enlargement
<Ctrl>W	redraw the screen
<Ctrl>O	toggle INSERT/OVERSTRIKE mode
!	summary of current sequence
?	display these help screens
<Ctrl>G	recalculate the consensus
G A T C ....	add base at the cursor
<Delete>	delete a base, or move sequence left
<Ctrl>H	delete a base, or move sequence left
<Space bar>	move the sequence to the right
<Ctrl>X	delete alignment column
<Ctrl>I	restore alignment column
<Ctrl>B	begin selecting a range for removal
<Ctrl>N	remove the selected range
<Ctrl>P	insert the removed range
-	reject current fragment

## COMMAND MODE

To enter Command Mode from Screen Mode, use `<Ctrl>D`. The cursor moves down to the lower-left corner of the screen next to `Command:` where you can enter any of the commands shown below followed by a `<Return>`.

There is some overlap in function between Command Mode and Screen Mode commands. In general, Screen Mode commands either move the cursor or edit a sequence; Command Mode commands manipulate entire sequences or contigs. *To save changes you've made to the contig currently loaded in the GelAssemble editor, you must enter one of the appropriate Command Mode commands.* (See "Writing Contigs" in this topic for more information.)

# GelAssemble

## Editing GelAssemble Commands

GelAssemble command editing is modeled on the OpenVMS DCL command-line editing. The <Left-arrow> and <Right-arrow> keys let you move your cursor around in a command you've typed; you can insert or delete characters at any position. <Ctrl>E moves the cursor to the end of the line. <Ctrl>U deletes all characters from the current cursor position to the start of the command.

## Returning to Screen Mode

Press <Return> in Command Mode to return to Screen Mode (described above).

## 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 followed with a file name. The square brackets ( [ and ] ) in the documentation below show command parameters that are optional, meaning you can leave them out.

## Anchored, Locked, and Modified Fragments

By default, GelAssemble allows you to move and edit the fragments loaded in the contig editor independently of one another. Also, by default, you can freely edit any fragment sequence on which the cursor is positioned. You may want to override these default settings under certain circumstances.

Using the : **ANCHOR** command, you can link, or anchor together, selected sequences that are loaded into the Edit Screen. Modifications to any anchored fragments, such as insertions, deletions, and sequence reversals, are propagated through all anchored fragment sequences in the current alignment. This preserves the alignment of all anchored fragments. For instance, the insertion of a single base within one anchored fragment causes the same base to be inserted at the same position in all other anchored fragments. An anchored fragment is denoted by an **A** next to the corresponding bar diagram in the Big Picture display. Even if all sequences are unanchored, you can still delete an entire column of the alignment at once with <Ctrl>X (see SCREEN MODE for more information).

When you load the first contig into the GelAssemble editor, all of the fragments in that contig are initially unanchored to one another. If you subsequently load another contig on top of an existing one in the editor using the : **LOAD** command, all of the fragments in the new contig are anchored to one another. Furthermore, all previously anchored fragments become unanchored. This permits easy positioning of the new contig when you are manually attempting to align two existing contigs.

The : **LOCK** command protects the specified sequence(s) from accidental modification in the GelAssemble editor. GelAssemble will not allow insertions, deletions, or reversal of a locked fragment. A locked fragment is denoted by an **L** next to the corresponding bar diagram in the Big Picture display.



An **M** adjacent to a bar in the Big Picture display means that the corresponding fragment sequence has been modified in some way (e.g. insertion, deletion, reversal) in the current editing session.

## Writing Contigs

The *only* way to save modifications you've made to a contig during the current session with GelAssemble is by writing the contig to the sequencing project database.

You can save your modifications to a contig using two commands: **: WRite** and **: EXit**. The **: WRite** command saves the contig alignment as well as the contig consensus sequence to the project database. The contig is named after the left-most sequence in the contig. The **: EXit** command works the same as **: WRite**, but GelAssemble exits after saving the contig to the sequencing project database. To quit GelAssemble without writing the contig to the project database, use **: QUIT**.

The consensus is *not* recalculated automatically when you write a contig to the sequencing project database; you must recalculate the consensus sequence explicitly. Use either <Ctrl>G in Screen Mode or the **: CONsensus** command in Command Mode to update the consensus sequence before writing the contig.

## Manually Assembling Contigs

You can use GelAssemble to assemble contigs manually that do not share sufficient overlap to be assembled automatically with GelMerge. Once you've entered a single contig into the editor, you can use the **: LOad** command to enter other contigs on top of the existing one. You can position the contigs manually (see "Positioning Fragments in the Contig Manually", above) and insert gap characters to create the desired alignment. If you then use the **: WRite** command, the entire alignment is written to the sequencing project database as a single contig. The new contig is named after the left-most fragment sequence in the contig.

Conversely, you can remove individual fragment sequences from a contig as described above (see "Removing Sequences from a Contig", above). The **: REJect** command in Command Mode performs the same function as pressing the minus key (-) in Screen Mode.

To prevent the sequencing project database from being corrupted, you are not allowed to store duplicate entries of any fragment sequence in the database during manual assembly. (Automatic assembly by GelMerge never stores duplicate entries of any sequence in the database.) GelAssemble does not permit you to store a contig containing a single fragment found at more than one position in the alignment. If you accidentally create such an alignment on the screen, you can remove redundant copies of each duplicated sequence with the **: NODUPLICATE** command (see "Command Mode Summary" for more information). If you use the **: LOad** command to purposely place copies of a single fragment at more than one position in the same contig, you can retain the duplicate copies by renaming them. Position the cursor on a duplicated fragment sequence and use the **: SPAWN** command (see "Command Mode Summary" for more information) to rename that fragment. Essentially, you are adding a new fragment sequence to the project database.

## Displaying Contig Alignments

You can write the contig alignment to a file in your local directory with the **: PRETTYout** command (see "Command Mode Summary" for more information). You can write the *Big Picture* schematic (lower half of the Edit Screen) to a file in your local directory with the **: BIGPICTure** command.

# GelAssemble

## Analyzing Fragment Sequences with Other GCG Programs

Use the : SEQOUT command to write the fragment and consensus sequences to individual sequence files in your local directory (see "Command Mode Summary" for more information). You can then use other GCG programs to analyze the sequences in these files.

## Selecting Another Contig to Edit

When you first run GelAssemble, you scroll through the list of contigs from which you select one to load into the GelAssemble Edit Screen. Once you move to the Edit Screen, you can use the : CONTIGs Command Mode command to return to this contig list, enabling you to select a new contig for editing. **CAUTION** -- *Selecting a new contig for editing erases all previous work from the Edit Screen. Therefore, use the : WRite command to save the current contig before using the : CONTIGs command.* However, you can restore the current Edit Screen after issuing the : CONTIGs command if you press <Ctrl>D while scrolling through the list of contigs.

## Command Mode Summary

Here is a summary of Command Mode commands you see with the : Help command:

### COMMAND MODE

[a,b] specifies a range of fragments.  
[x,y] specifies a range of bases.  
[n] is an optional numeric parameter.

Edit [ContigName]	replace current contig with a new contig
CONTIGs	select another contig for editing
WRite	write a contig to the database
EXit	write the contig and quit
QUIT	quit without writing
ERASE	delete current contig from the database
238	move to position 238 in the current fragment
[x,y] PRETTYout [FileName]	write the sequence alignment [position x - y]
[a,b] SEQOUT	write fragments [a - b] to sequence files
BIGPIcture [FileName]	write bar schematic to an output file
OVERstrike	select OVERSTRIKE sequence edit mode
NOOVERstrike	select INSERT sequence edit mode
[x,y] CONSeNsus	recalculate the consensus sequence
[a,b] LOCK	lock strands [a through b]
[a,b] Unlock	unlock strands [a through b]
[x,y] SELEct	select bases [x through y]
REMOve	remove the selected bases
[n] INSert	insert the removed bases [at position n]
CANcel	cancel the selection
[x,y] DELEte	delete bases [x through y]
GOTo [FragmentName]	move to strand by name
FIND GAATC	find the next occurrence of GAATC
Differences	show differences from the consensus
MATCHes	show matches with the consensus
NEIther	show neither matches nor differences
REDraw	redraw the screen
Help	display these help screens

	SORT [DEScending]	sorts strands by their offsets in alignment
[a,b]	MOVE	moves a strand [from line a to line b]
	OPen	opens a blank line at the cursor position
[a,b]	ANCHor	anchors strands [a through b]
[a,b]	NOANCHor	unanchors strands [a through b]
	LOad [ContigName]	loads another contig into the Edit Screen
	REVerse	reverse-complement the (anchored) strand(s)
[n]	Offset	shifts the current fragment [to begin at n]
	REJect	removes the current fragment from the screen
	NODuplicate	removes a duplicated fragment from the screen
	SPAWN	renames a duplicated fragment
	SEParate	makes two contigs from anchored and unanchored strands

For all commands requiring row numbers, the consensus strand is row 1.

: **EDIT** [ContigName]

loads the aligned fragment sequences from the specified contig into the Edit Screen, *replacing* any fragment sequences currently on your screen. All unsaved changes to the replaced contig alignment and fragment sequences are lost. Use this command if you know the name of the new contig you wish to edit; otherwise scroll through the list of available contigs displayed with the : **CONTIGs** command.

: **CONTIGs**

returns you to the list of contigs, letting you select a new contig for editing. All modifications to the contig currently in the Edit Screen are lost if you haven't written them to the sequencing project database before selecting a new contig. If you haven't yet selected a new contig, use <Ctrl>D to return to the contig currently in the Edit Screen.

: **WRITE**

writes the contig, including the alignment and all fragment sequences currently loaded into the Edit Screen, to the sequencing project database. All edits made to the contig alignment and the individual fragment sequences are preserved. The contig is named after the left-most fragment in the contig.

**Note:** The contig consensus sequence is not automatically recalculated before writing the contig.

: **EXIT**

writes the contig, including the alignment and all fragment sequences currently loaded into the Edit Screen, to the sequencing project database and then exits GelAssemble. All changes made to the alignment and the individual fragment sequences are preserved. The contig is named after the left-most fragment in the contig.

**Note:** The contig consensus sequence is not automatically recalculated before writing the contig.

# GelAssemble

## : QUIT

exits GelAssemble without saving any of the changes made to the contig loaded in the Edit Screen. All changes made to the alignment and the individual fragment sequences since the last time the contig was written are lost.

## : ERASE

If the fragment on which the cursor is positioned is a single-fragment contig, this command deletes it from the sequencing project database. You are prompted to confirm that you wish to delete the fragment. You cannot delete a fragment if any other fragments are loaded in the Edit Screen or if any fragments have been rejected from the Edit Screen. **CAUTION** -- *If you use the : ERASE command, all copies of the deleted fragment are removed from the database, including the archival copy.*

## : [x,y] PRETTYout [filename]

writes the alignment of all the fragments in the Edit Screen between absolute positions *x* and *y* (inclusive) to an output file in your local directory. If any fragments are anchored, only the alignment of those fragments is written. If you do not specify beginning and ending positions in the alignment, the entire length of the alignment is written.

## : [a,b] SEQOUT [\*]

writes each fragment from row *a* to row *b* (inclusive) to a separate output file in your current working directory. You can use other GCG programs to analyze the sequences in these files. GelAssemble prompts you for a file name for each fragment unless you add an asterisk (\*) after the : SEQOUT command. If you do not specify any row numbers, : SEQOUT writes out the current fragment. If you specify only one row number, : SEQOUT writes out all fragments between that row and the row on which the cursor is positioned. Remember, the consensus row is 1.

## : BIGPICTure [filename]

writes the bar schematic (the lower half of the Edit Screen) to a file in your local directory.

## : OVERstrike

sets OverStrike edit mode. Any sequence symbol typed at the cursor position will replace the existing sequence symbol at that position.

## : NOOVERstrike

sets Insert edit mode. Any sequence symbol typed at the cursor position will be inserted at that position, shifting the entire sequence from that position to the right by one column.

## : [x,y] CONSensus

calculates the consensus sequence from position *x* through position *y* in the alignment, and replaces what is on row 1 with the new consensus. If you don't supply a range, the entire consensus is recalculated.

The consensus function is a simple measure of plurality. IUB nucleotide ambiguity symbols in the fragment sequences are treated as weighted representations of their constituent bases for the

purpose of generating a consensus. For example, an R represents half A and half G. If there is no absolute plurality (that is, two or more bases are tied), then those bases tied for plurality are used to generate an IUB nucleotide ambiguity symbol for the consensus. If the most common symbol at a position is a gap character ( . and ~), then the consensus contains a gap character at that position. If there is no unanimity among the loaded fragments at a particular column, then the displayed consensus symbol for that column is in lowercase.

: [a,b] **LOCK**

locks the fragments on rows *a* through *b* to prevent accidental modification. If you omit the row numbers, the current fragment is locked.

: [a,b] **Unlock**

unlocks the fragments on rows *a* through *b*, once again permitting modification. If you omit the row numbers, the current fragment is unlocked.

: [x,y] **SElect**

selects and highlights bases *x* through *y* of the current fragment. If you omit the range, the selected range begins at the cursor position and is extended using the <Left-arrow> and <Right-arrow> keys.

: **REMove**

deletes the selected range and copies it into a buffer. If the selection was in an anchored fragment, the corresponding bases in the other anchored fragments *are not* removed.

: [n] **INSert**

inserts a copy of the removed bases (using the : **REMove** command) at position *n* in the current fragment. If you omit the position, the bases are inserted at the cursor. If the current fragment is anchored, the corresponding bases *are not* inserted in the other anchored fragments.

: **CAnceL**

cancels the : **SElect** command, unhighlighting the currently selected range.

: [x,y] **DElete**

deletes bases *x* through *y* from the current fragment. If you omit the range, GelAssemble prompts you for one. If the current fragment is anchored, the corresponding bases *are not* deleted from the other anchored fragments.

: **GOTo** [FragmentName]

moves the cursor to the row containing the specified fragment.

: **FI**nd *G*TATTC

finds the next occurrence of *G*TATTC in the current fragment sequence.

# GelAssemble

: **D**Ifferences [attribute]

highlights bases that differ from the consensus symbol at each position in the alignment. This is the default for GelAssemble. The default highlight is reverse-video. The optional attribute lets you specify these other kinds of highlighting: **B** = blinking, **U** = underlining, and **D** = bold.

: **M**Atches [attribute]

highlights bases that match the consensus symbol at each position in the alignment. The default highlight is reverse-video. The optional attribute lets you specify other kinds of highlighting: **B** = blinking, **U** = underlining, and **D** = bold.

: **N**either

turns off all highlighting bases that match or differ from the consensus symbol at each position in the alignment.

: **R**EDraw

redraws your terminal screen. This is useful if line noise between your terminal and the computer has changed the screen in some unreasonable way or if a system message appears on your screen.

: **H**elp

displays the commands available to the Screen and Command Modes of GelAssemble.

: **S**ORT [**D**EScending]

sorts the loaded fragments on the screen in order of increasing offset (beginning position of the fragment in the contig alignment), beginning with the left-most fragment on row 2 of the sequence alignment. If you specify **D**EScending, the sort is in order of decreasing offset, beginning with the right-most fragment on row 2 of the sequence alignment.

: [a,b] **M**OVE

moves a fragment from row *a* to row *b*. If no row numbers are supplied, the fragment on which the cursor is positioned is moved to the next vacant row.

: **O**Pen

creates a vacant row at the cursor by pushing all fragment sequences up one row, including the fragment on which the cursor is positioned.

: [a,b] **A**NChor

anchors rows *a* through *b*. If you omit the row numbers, the current fragment is anchored. (See "Anchored, Locked, and Modified Fragments" for more information.)

**: [a,b] NOAnchor**

unanchors rows *a* through *b*. If you omit the row numbers, the current fragment is unanchored. (See Anchored, Locked, and Modified Fragments for more information.)

**: LOad [ContigName]**

loads the aligned fragment sequences in the specified contig onto the lowest empty rows of the Edit Screen. The left-most sequence in the contig is loaded beginning at the cursor position, and each remaining fragment is loaded so that its alignment within its contig is retained on the screen. The loaded fragments are all anchored, and all fragments already in the Edit Screen are unanchored.

This command is used to assemble contigs manually that do not share sufficient overlap to be assembled automatically by GelMerge.

**: REVerse**

reverse complements the current fragment. If the current fragment is anchored, the entire anchored group is also reversed.

This command is used most often during manual contig assembly to align fragments entered into the Edit Screen with the **: LOad** command.

**: [n] Offset**

shifts the current fragment (fragment on which the cursor is positioned) so that its left end is at position *n* in the sequence alignment. If you omit the position, the fragment is shifted so that its left end begins at the cursor.

This command is used during manual contig assembly to position fragments entered into the Edit Screen with the **: LOad** command.

**: REJect**

removes the fragment on which your cursor is positioned from the screen and stores it in a buffer. If the remaining fragments on the screen are written to the sequencing project database as an aligned contig, each rejected fragment is written to the database as a separate contig.

This command is used during manual contig assembly to remove individual fragments from an existing contig.

**: NODuplicate**

removes the fragment on which your cursor is positioned from the screen if that fragment also occurs on another line in the current Edit Screen.

This command is used during manual contig assembly to remove redundant occurrences of fragments from the Edit Screen before saving a contig to the sequencing project database.

# GelAssemble

: **SPAWN** [NewContigName]

If a fragment is duplicated on another row of the Edit Screen, this command renames the fragment sequence on which the cursor is positioned and stores it in the database as a new fragment. You can then write the duplicated fragments into the same contig since they have different names.

This command is used during manual contig assembly to allow multiple copies of individual fragment sequences to exist in a sequencing project database.

: **SEParate**

divides the fragment sequences loaded into the Edit Screen into two separate contigs that are then written to the sequencing project database. One contig contains the fragment on which the cursor is positioned and all fragments anchored to that fragment. After they are stored in the project database, the fragments in this contig are removed from the Edit Screen. The remaining fragment sequences are written to the sequencing project database as the second contig. The consensus sequence for each contig is recalculated before writing. The consensus sequence on row 1 of the sequence alignment is not recalculated automatically; it is simply truncated to reflect the boundaries of the remaining contig.

This command is used during manual contig assembly to separate contigs created by GelMerge that you believe should not be aligned together.

## RELATED PROGRAMS

GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project. GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files. GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble. GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge. GelView displays the structure of the contigs in a fragment assembly project. GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

## RESTRICTIONS

A contig may not contain more than 1,650 fragments and may not be longer than 200,000 bases. No single fragment may be longer than 2,500 bases.

## CONSIDERATIONS

### Sequence Symbols

GelEnter accepts any valid GCG sequence character (see Appendix III). GelMerge and GelAssemble recognize all IUB nucleotide ambiguity codes (see Appendix III) and the period (.) and tilde (~) as gap symbols for the generation of consensus sequences. All other sequence characters are treated as non-nucleotide symbols in GelMerge and GelAssemble.



## SUGGESTIONS

The number of fragments you can view at any one time and the number of bases from each sequence you can view at once are constrained by the number of lines and columns on your terminal screen. If your terminal can display more than 24 lines and 80 columns, GelAssemble uses the extra space to display additional fragment sequences and larger ranges from each sequence.

## COMMAND-LINE SUMMARY

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

Minimal Syntax: `% gelassemble`

Prompted Parameters: None

Local Data Files:

`set.keys` (must be in your current working directory to be used)

Optional Parameters:

`-CONTIG=ContigName`                    loads the specified contig

## ACKNOWLEDGEMENTS

GelAssemble is derived from the MSE (Multiple Sequence Editor) program written by Dr. William Gilbert at the Massachusetts Institute of Technology, to whom we are very grateful. The screen layout and vertical scrolling are his invention, as are the concepts of anchoring and locking strands. He also designed the mechanism for displaying matches and differences. MSE was converted into GelAssemble by Philip Delaquess, Lisa Caballero, and Irv Edelman. MSE evolved from SeqEd, which was developed by John Devereux and Paul Haerberli.

## 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.

## GelAssemble

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.

**-CONTIG=ContigName**

loads a contig directly into the Edit Screen.

Printed: December 1, 1998 10:25 (1162)

## GELDISASSEMBLE

### FUNCTION

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

### DESCRIPTION

See the Fragment Assembly System (FAS) Introduction for an overview of working with the programs within the FAS to assemble sequences in a sequencing project.

GelDisassemble breaks all assembled contigs in the current fragment assembly project and rebuilds a database consisting of the unjoined, working fragments. All edits made to the fragments in GelAssemble are preserved in the unjoined fragments.

You can rebuild the database from the original gel readings, rather than the working fragments, using **-ARCHIVE**. This recreates the project database exactly as if you had just entered all of the fragments. Any modifications you have made to the sequences in GelMerge and GelAssemble are lost.

### EXAMPLE

Here is a session with GelDisassemble used to disassemble the project "myproject." This project was created in the example sessions of GelStart, GelEnter, and GelMerge.

```
% geldisassemble

Are you sure you want to disassemble your project (* No *) ? Yes

1) Emptying "relation" directory....

2) Emptying "consensus directory....

3) Copying "working" to "consensus"....

4) Creating "relation"....

Gel Project Disassembled

%
```

### RELATED PROGRAMS

GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project. GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files. GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble. GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge. GelView displays the structure of the contigs in a fragment assembly project. GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

# GelDisassemble

## 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: `% geldisassemble -Default`

Prompted Parameters: None

Local Data Files: None

Optional Parameters:

<code>-ARCHive</code>	rebuilds database from the original gel readings
<code>-NOMONitor</code>	suppresses screen monitor of program progress

## 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.

`-ARCHive`

rebuilds the database from original gel readings rather than from the working fragments.

`-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 10:25 (1162)

## GELENTER

### FUNCTION

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

### DESCRIPTION

See the Fragment Assembly System (FAS) Introduction for an overview of working with the programs within the FAS to assemble sequences in a sequencing project.

GelEnter enters fragment sequences into the Fragment Assembly System. GelEnter accepts sequence data from the following: 1) your terminal keyboard; 2) a digitizer; or 3) existing sequence files using **-ENTER**.

GelEnter and SeqEd are essentially the same program. However, unlike SeqEd, GelEnter writes its output file to the fragment assembly project database rather than your local directory. For a complete description of GelEnter commands, see SeqEd.

### EXAMPLE

Here is a session using GelEnter to enter previously existing sequence files into the project database created in the example session for GelStart:

```
% gелenter -ENTER=mu*.seq

Entering mu*.seq into the database.
"mu10"  361 nucleotides
"mu18"  42 nucleotides

//////////

"mu6"   296 nucleotides
"mu9"   39 nucleotides

%
```

### RELATED PROGRAMS

GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project. GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files. GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble. GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge. GelView displays the structure of the contigs in a fragment assembly project. GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

GelEnter is SeqEd customized to run in the fragment assembly environment.

# GelEnter

## RESTRICTIONS

A contig may not contain more than 1,650 fragments and may not be longer than 200,000 bases. No single fragment may be longer than 2,500 bases.

## CONSIDERATIONS

GelEnter doesn't allow you to enter two different fragments with the same name. This protects you from overwriting existing fragments in the database by accidentally reusing a name. If two files share the same file name but have different file extensions, GelEnter considers them to have the same name. (See Chapter 1, Getting Started in the User's Guide for more information on naming files.)

The heading documentation that appears above the sequence and the sequence comments are not preserved when fragments are entered into the fragment assembly database.

### Sequence Symbols

GelEnter accepts any valid GCG sequence character (see Appendix III). GelMerge and GelAssemble recognize all IUB nucleotide ambiguity codes (see Appendix III) and the period (.) and tilde (~) as gap symbols for the generation of consensus sequences. All other sequence characters are treated as non-nucleotide symbols in GelMerge and GelAssemble.

## SUGGESTIONS

If you are repeatedly entering an over-cloned fragment or cloning vector into the database, you can have GelEnter highlight the redundant fragment each time you begin to enter it. To do this, edit the redundant fragment with GelAssemble. Extract the sequence using GelAssemble's : **SEQOUT** command and then run GelStart with **-VECTOR**, using the file specification of the extracted sequence. Similarly, GelStart's **-SITES** parameter causes GelEnter to highlight restriction sites that you have specified. This can help you to detect instances of rejoined sticky ends. GelEnter only highlights vector sequences and restriction sites when you are entering sequence data from either your terminal keyboard or a digitizer; it does not highlight sequences in existing sequence files entered using **-ENTER**.

Do not use GelEnter's : **WRITE** command to enter a sequence in the database that you wish to continue editing. Once the sequence is entered into the database, any further editing cannot be saved because you are not permitted to reenter a fragment already in the database. If you wish to modify a fragment that is already in the database, use GelAssemble. If you wish to replace a fragment in the database, first delete that fragment from the database using GelAssemble's : **ERASE** command, and then reenter the new version using GelEnter.

Because GelEnter does not reedit a fragment that has already been entered, it is advisable to enter, without interruption, as much of the fragment sequence as possible. If you exit GelEnter in the middle of entering a sequence, you can either enter the remaining sequence as a different fragment or use GelAssemble to enter the remaining sequence. GelAssemble can modify existing sequences only from the keyboard.

## 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: % gelenter [-INfile1=]mu\*.seq

Prompted Parameters: None

Local Data Files:

set.keys (must be in your current working directory to be used)

Optional Parameters:

-ENTER=mu*.seq	enters existing files into the database
-STAden	enters existing Staden format files into the database
-FASTA	enters existing FASTA format files into the database
-SINGlecommand	automatically returns to screen mode after each command
-PERFect	sets find to search for perfect symbol matches
-VECTors=gb:synpbr322	highlights sequences from pBR322
-SITes=gaattc	highlights GAATTC patterns
-LANes=g,A,T,C	sets lane order for digitizer
-MINOverlap=10	sets minimum overlap length for Reload command
-PCTOverlap=95	sets stringency for the Reload command
-TOLerance=0.4	sets tolerance for digitizing ambiguity (0 to 1), with 1 being the most tolerant

## LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like **-DATA1=myfile.dat**. For more information see Chapter 4, Using Data Files in the User's Guide.

## Customizing Your Keyboard With SetKeys

You can use the program SetKeys to create a set.keys file that tells the SeqEd, GelEnter, LineUp, GelAssemble, and SeqLab sequence editors how to interpret the letters you type at the terminal. When entering gel readings, it is useful to have the symbols for G, A, T, and C under the fingers of one hand in the same positions as the lanes in your gel. SeqEd, GelEnter, LineUp, GelAssemble, and the SeqLab sequence editor automatically read the file set.keys if it is present in your local directory. If set.keys is absent, or if the sequence type is set to Protein (in SeqEd and LineUp only) the terminal keys retain their conventional meanings.

If you have a set.keys file in your directory, SeqEd, GelEnter, LineUp, and GelAssemble only respond to the keys that it redefines. You can edit the file set.keys with a text editor if some of the keys you want to use are not in it. Any keys not mentioned in set.keys appear to be dead in these sequence editors. In the SeqLab sequence editor, keys that are not redefined retain their normal meanings.

Several keys are vital for the control of SeqEd, LineUp, GelEnter, and GelAssemble; this means you are not allowed to redefine the keys for /, [, ], {, }, (, ), :, ,, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, <Ctrl>R, <Ctrl>D, <Ctrl>H, <Return>, and <Ctrl>E.

# GelEnter

## 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.

### **-ENTER=filename**

enters an already existing sequence file into the database. You can enter several sequences into the database at once by specifying a list file or sequence specification using an asterisk (\*) wildcard. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for help in naming a group of sequences.)

### **-STAden**

allows input sequence files to be in Staden format instead of GCG format. You use this parameter only with **-ENTER=filename**

### **-FASTA**

allows input sequence files to be in FastA format instead of GCG format. You use this parameter only with **-ENTER=filename**

### **-SINGlecommand**

sets GelEnter to return automatically to Screen Mode after every command in Command Mode.

### **-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 use **-PERFect**, **/GARC** would only find the pattern GARC. This also makes GelEnter treat sequences as linear and not find patterns that start at the end and continue into the beginning of the sequence.

### **-VECTors=GB:SynpBR322**

tells GelEnter which cloning vector or vectors are of interest to you. When you are entering a sequence from your terminal keyboard or a digitizer, GelEnter checks the sequence against the vectors of interest to make sure you are not entering a vector sequence. If GelEnter finds that you are entering vector sequence, the terminal bell rings and the vector sequence characters are highlighted with reverse video.

Normally, you specify in GelStart the cloning vectors you want to highlight. Specifying them in GelEnter allows you to override those choices.

### **-SITes=GAATTC,genetic**

tells GelEnter to highlight enzyme recognition sites that interest you when you are entering a sequence from your terminal keyboard or a digitizer.

Normally, you specify in GelStart the enzyme recognition sites you want to highlight. Specifying them in GelEnter allows you to override those choices.



The following parameters affect entering sequences in GelEnter using a digitizer. For a complete description of digitizer use and commands, see SeqEd.

**-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. GelEnter 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** GelEnter'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 10:25 (1162)



## GELMERGE

### FUNCTION

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

### DESCRIPTION

See the Fragment Assembly System (FAS) Introduction for an overview of working with the programs within the FAS to assemble sequences in a sequencing project.

GelMerge takes unassembled fragment sequences in your sequencing project database and creates complete assemblies called contigs. Each contig is a multiple alignment of contiguous, overlapping sequences in the project database. These contigs are saved in your project database, where you can review and edit them with GelAssemble. As you add sequences that connect separate contigs to the project database, GelMerge aligns the separate contigs into larger assemblies. Ultimately, all fragments may be assembled into a single contig that represents the underlying genomic sequence from which all the fragments were derived.

You can use GelMerge to assemble the fragments in your sequencing project database into aligned contigs in several ways. You may choose to: 1) preserve the relationships among all the fragments in existing contigs as you align those contigs into larger assemblies; 2) reassemble the entire project from the original gel readings, disregarding all sequence edits and assemblies you've previously created; or 3) reassemble the entire project from the individual edited fragments, retaining all edits you have made (base insertions, deletions, and substitutions) but disregarding all of the assemblies you've previously created.

In GelStart, you can specify cloning vectors used to isolate the sequenced fragments. In GelMerge, you can identify and optionally remove vector sequences from the working copies of the fragment sequences in your project database; the original sequences entered into the project database are unaffected.

### EXAMPLE

Here is a session using GelMerge to assemble contigs from the fragments in "myproject", which was created in the example sessions of GelStart and GelEnter.

```
% GelMerge -MINIdentity=13

What word size (* 7 *) ?

What fraction of the words in an overlap must match (* 0.80 *) ?

What is the minimum overlap length (* 14 *) ?

  Reading .....
Comparing .....

  Aligning .....

  Writing ...
```

# GelMerge

```
Input Contigs:      12
Output Contigs:     3

CPU time:          02.29 (seconds)
```

\$

## OUTPUT

GelMerge modifies your sequencing project database. The contigs assembled by GelMerge are saved into your project database where you can review and modify them using the other Fragment Assembly programs.

FAS never modifies the files in the project archival directory. This allows you to always recover the original gel readings.

## RELATED PROGRAMS

GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project. GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files. GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble. GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge. GelView displays the structure of the contigs in a fragment assembly project. GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

## RESTRICTIONS

The total length of all fragment sequences entered into GelMerge cannot exceed 380,000 bases. If you want to assemble your sequencing project with GelMerge, the total length of all consensus sequences in the project database is therefore limited to 380,000. Contigs longer than 100,000 bases cannot be assembled with any other contigs. However, two contigs that are each shorter than 100,000 bases can be assembled into a contig that is longer than 100,000 bases.

You can enter a maximum of 1650 fragments into GelMerge. No fragment entered into GelMerge can be longer than 2500 bases. In creating alignments, no fragment can have more than 100 gap insertions in a single session with GelMerge.

In an overlap, the part of each contig with fragments that at least partially span the overlap can be no longer than 5,000 bases. This limit is set high enough that you should never encounter it in normal sessions with GelMerge, regardless of the size of the sequencing project.

The *surface of comparison* (see the CONSIDERATIONS topic for an explanation) is limited to 500,000. Again, this limit is set high enough that you should never encounter it in normal sessions with GelMerge.

## ALGORITHM

### General Method

Each fragment in the sequencing project database is part of a contig. A contig contains either a group of aligned sequences associated together or a single, unaligned fragment called either a *contig-of-one* or a *single-fragment contig*. GelMerge finds the two contigs with the longest overlap and then aligns them to assemble a single contig. The program then finds the next two contigs with the longest overlap (possibly involving the just-assembled contig) and aligns them to assemble a single contig. GelMerge repeats this process of overlap determination and contig assembly until there are no remaining overlaps among the contigs in the project database. The result of GelMerge may either be a single contig (if all contigs overlap to form a contiguous assembly) or several contigs (if none of the remaining contigs share significant overlap). As you add new fragments to the sequencing project database, they may connect separate contigs to form larger assemblies.

GelMerge allows you to modify several parameters that affect the speed and sensitivity of the overlap determination. Additionally, you can modify parameters affecting the alignment process in contig assembly and the speed and sensitivity of vector recognition. Most of these modifiable parameters are discussed in the remaining algorithm discussion.

### Finding Overlaps

The following discussion uses the concept of a *diagonal*, which is a register of comparison between two sequences. For any two positions in the two sequences, you can calculate their diagonal as the position in the second sequence minus the position in the first sequence. For instance, if a block of 5 sequential identities begins at position 412 in the second sequence and position 1 in the first sequence, then the block is located on diagonal 411 ( $412 - 1 = 411$ ,  $413 - 2 = 411$ , ...,  $416 - 5 = 411$ ). The *comparison surface* includes all possible diagonals in a comparison of two sequences.

In finding overlaps among the contigs, GelMerge represents each contig by its consensus sequence. GelMerge uses a modification of the approximate alignment procedure of Wilbur and Lipman (SIAM J. Appl. Math. **44**; 557-567 (1984)) to determine the amount of overlap between any two consensus sequences. This procedure creates alignments from short blocks of contiguous sequence identities. The program places gaps between blocks, but not within blocks. You can set the minimum length for each short block of identities in response to the `What word size` program prompt (default length is 7).

In GelMerge, each alignment must contain at least one long block of contiguous sequence identities. You can set the minimum length for each long block with `-MINIdentity` (default length is 14). The requirement for at least one long block of identities allows the program to exclude trivial overlaps from consideration. This requirement also limits the extent of gapping permitted in the approximate alignment. The alignment cannot get out of phase by more than 10 registers of comparison from diagonals containing long blocks of sequence identities. (Use `-MAXGap` to adjust this default of 10 registers of comparison.) This effectively prevents the alignment from wandering too far away from registers of comparison with significant sequence identity.

GelMerge determines all of the possible distinct alignments between the two contig consensus sequences being compared. Each alignment corresponds to a different overlap between the same two contig consensus sequences. An alignment does not necessarily extend over the entire length of the corresponding overlap. For example, the length of the alignment in the following figure is the length from *B* to *C* while the length of the overlap is the length from *A* to *D*.

# GelMerge



After finding all of the distinct alignments between a pair of contigs, GelMerge counts the number of exact nucleotide matches in the best alignment (alignment with the most nucleotide matches). By default, if this number of identities is at least 80% of the length of the corresponding overlap, the program saves the position and length of this overlap. (You can change this default in response to the `What fraction of the words in an overlap must match` program prompt.) If this overlap does not meet the identity criterion, the next best alignment is checked. For example, in the figure below, although the alignment for overlap *A* has the greatest number of sequence identities, it does not meet the default identity criterion. The alignment for overlap *B*, containing fewer sequence identities, does meet the default identity criterion. In this case, GelMerge selects *B* as the overlap between these two contigs.

A



B



If no alignment of two contig consensus sequences meets the identity criterion, then the two contigs do not overlap. After GelMerge finds overlaps among all the contigs, it precisely aligns the two contigs with the longest overlap to assemble a single contig. The approximate alignments used to determine overlaps are not used to create final contig assemblies. More precise alignments, involving all of the sequences in both contigs, and without gap size limitations, are used to create the actual assemblies from overlapping contigs.

## Aligning Contigs

Once GelMerge determines the pair of contigs with the longest overlap (see above), it aligns them using the method of Needleman and Wunsch (*J. Mol. Biol.* **48**; 443-453 (1970)). This method, originally used to align individual sequences, has been extended for use with contigs of aligned sequences. For a pairwise alignment of individual sequences, the comparison score between any two sequence symbols is found in a scoring matrix (see the `LOCAL DATA FILES` topic for more information). For a pairwise alignment of contigs of aligned sequences, the comparison score between any two positions in those contigs is the arithmetic average of the scores for all possible symbol comparisons at those positions. When the program inserts gaps into a contig to produce an alignment, they are inserted at the same position in all of the sequences of the contig.

## Recognizing (and Optionally Removing) Vector Sequences

GelMerge searches for vector sequences in single-fragment contigs using a two-step approach. First, GelMerge finds approximate alignments between vector and contig sequences using a modification of the method of Wilbur and Lipman (described above in "Finding Overlaps"). You can fine tune the vector searching by adjusting some parameters of the approximate alignment procedure. For vector searching, the minimum length for each short block of sequence identities

is the same as the length used to find overlaps among the contigs; you can set the minimum length in response to the `What word size` program prompt (default length is 7). The minimum length of each long block of sequence identities in vector searching can be set with `-VECTORMINIdentity` (default length of 12). The alignment cannot get out of phase by more than 5 registers of comparison from diagonals containing long blocks of sequence identities. (This default of 5 registers of comparison can be adjusted with `-VECTORMAXGap`.)

Each of the approximate alignments indicates the position of possible vector sequences in the contig. In the second step of vector recognition, GelMerge refines these alignments using the method of Smith and Waterman (*Advances in Applied Mathematics* **2**; 482-489 (1981)). By default, if the aligned portions of the contig and vector sequences share greater than 80% sequence identity, the contig bases in the alignment become candidates for excision. (You can adjust this value using `-VECTORStringency`.) Additionally, by default, the vector sequences must begin within 12 bases of either end of the contig in order to be excised. (This value is the same as the minimum length for each long block; you can adjust it with `-VECTORMINIdentity`.)

## CONSIDERATIONS

### Sequence Symbols

GelEnter accepts any valid GCG sequence character (see Appendix III). GelMerge and GelAssemble recognize all IUB nucleotide ambiguity codes (see Appendix III) and the period (.) and tilde (~) as gap symbols for the generation of consensus sequences. All other sequence characters are treated as non-nucleotide symbols in GelMerge and GelAssemble.

### Consensus Sequences

Each contig in a sequencing project database is associated with a contig consensus sequence. GelMerge creates a new consensus sequence for each assembled contig and stores it in the project database. In a contig consensus sequence, the consensus symbol at any position in the contig is the most common symbol among all of the contig fragments at that position. GelMerge treats IUB nucleotide ambiguity symbols (described in Appendix III) in the fragment sequences as weighted representations of their constituent bases in order to generate a consensus. For example, an R represents half A and half G. If there is no absolute plurality at a position (that is, two or more unambiguous bases are tied), then those bases tied for plurality are used to generate an IUB nucleotide ambiguity symbol for the consensus. If the most common symbol at a position is a gap character (.) or (~), then the consensus contains a gap character at that position. (The automatic consensus generation function in GelAssemble follows these same rules.)

You can edit the consensus sequence associated with each contig in the database with GelAssemble. GelMerge does not use these consensus sequences to find overlaps, but rather determines a new consensus sequence for each contig as described below in "Sequence Simplification". Therefore, any edits you've made to any consensus sequences in GelAssemble are ignored in subsequent sessions with GelMerge.

### Sequence Simplification

In finding overlaps among the contigs, GelMerge represents each contig by a simplified consensus sequence, containing only the symbols G, A, T, and C. The consensus symbol at any position in the contig is simply the most common unambiguous sequence symbol among all of the fragments at that position. As previously described, GelMerge treats IUB nucleotide ambiguity symbols in the fragment sequences as weighted representations of their constituent bases in

# GelMerge

order to generate a consensus. If there is no absolute plurality at a position in the consensus, meaning that two or more unambiguous bases are tied, then the consensus symbol chosen is the one that is present at the highest frequency in GenBank. The relative frequency of bases in GenBank is A>T>G>C. For example, if T and C were tied, the consensus symbol chosen for that position would be T. Gap characters ( . and ~) are ignored so they are never found in the simplified consensus sequence.

Similarly, in the first step of vector recognition, GelMerge represents each vector and fragment sequence as a simplified sequence containing only the symbols G, A, T, and C. The simplification of ambiguous bases is achieved as described above.

Simplified sequences are used only in the preliminary steps of both contig assembly and vector recognition. Once overlaps are found, the final alignment of two overlapping contigs into a single assembly uses all of the sequence information from all of the fragments in both contigs. In vector recognition, the final alignments, which are both listed in the Report file and used to determine vector sequence excisions, are created using the original vector and fragment sequences.

## Surface of Comparison

GelMerge performs a series of pairwise alignments between clusters of fragments (contigs) to create the final contigs. Normally, each pairwise alignment requires enough computer memory for a surface of comparison proportional to the product of the lengths of the two contigs being aligned. However, since GelMerge limits the alignment to the region of overlap between the contigs, a much smaller surface of comparison is required. The amount of memory allocated for the surface of comparison in GelMerge should be sufficient for the automated assembly of almost all sequencing projects.

## Memory Requirement

GelMerge is shipped with the Wisconsin Package™ so that you can run the program if you have access to 18 MB of virtual memory.

## SUGGESTIONS

### Creating Assemblies

By default, GelMerge assembles existing contigs together. As you add new sequences to your project databases, you may want to reassemble the entire project "from scratch." With **-ARChive**, GelMerge reassembles the entire project from the original gel readings, ignoring any sequence edits you have made and contig relationships that you have previously created. This is equivalent to first using GelDisassemble with **-ARChive** and then using GelMerge; however, using GelMerge with **-ARChive** as a single step is much faster. With **-WORKing**, GelMerge assembles the entire project from the individual edited fragments, retaining all edits you've made (base insertions, deletions, and substitutions) but disregarding all of the assemblies you've created. **-WORKing** removes all gap characters from the edited sequences before they are reassembled.

GelMerge recognizes overlaps based upon the fraction of sequence identity across the entire overlap (see the ALGORITHM topic for more information). Because of the requirement for similarity across the entire overlap, GelMerge may not recognize overlaps between cDNAs and the corresponding genomic sequences if the genomic sequences contain introns. Furthermore, GelMerge may not recognize overlaps between overlapping genomic sequences if they are flanked by vector sequences that are not removed using **-EXCise**.



Using the default program parameters, GelMerge may fail to recognize weak overlaps among fragments whose sequences are poorly or ambiguously determined. Initially, you should consider using the default parameters with GelMerge to quickly assemble contigs from the fragments with strong overlaps. If you believe that GelMerge has missed some existing overlaps among the contigs in your sequencing project, you may then consider modifying some of these parameters in subsequent sessions with the program.

For example, you might first reduce the minimum fraction of required matching words in an overlap with `-STRIngency` or in response to the program prompt. Reducing this requirement won't greatly increase the time required for GelMerge to complete. However, greatly reducing this requirement may result in incorrect assemblies.

If you believe that some overlaps may not contain even one block of 14 contiguous sequence identities, you might reduce this requirement with `-MINIidentity`. Reducing this requirement may significantly increase the time required for GelMerge to complete.

If you believe that fragment sequences are so poorly determined that overlaps may not contain mostly runs of at least 7 identical bases in a row, you might reduce this requirement in response to the `What word size` program prompt. Reducing this requirement may greatly increase the time required for GelMerge to complete and is not recommended.

## Removing Vector Sequences

GelMerge only checks for vector sequences in unaligned, single-fragment contigs. While it recognizes and reports vector sequences located at any position within the fragment, GelMerge can automatically remove only vector sequences found near the ends of a fragment. By default, GelMerge removes vector sequences from a fragment if they begin within 12 bases from either end of the fragment. For instance, let's say the alignment between fragment and vector begins at position 5 and ends at position 40 in the fragment. With `-EXCise` GelMerge would remove the first 40 bases in the fragment.

If you've specified more than one vector sequence in GelStart, GelMerge finds matches between all vector sequences and the project fragments. However, GelMerge excises only those vector sequences found near the ends of the fragments. For instance, if the order of vector sequences in a fragment is

vector A	vector B	genomic sequences
----------	----------	-------------------

you could remove only *vectorA* sequences from the fragment in a single session with GelMerge. To remove all vector sequences, you must first excise the *vectorA* sequences using `-EXCise` and `-NOMerge`. Then, in the next session with GelMerge, you would find *vectorB* sequences that you could excise near the end of the fragment.

You can preview which vector sequences would be removed from single-fragment contigs in your project database without actually removing those sequences or aligning the contigs. If you use `-REPortfile` and `-NOMerge`, GelMerge generates a file of matches between the vector and fragment sequences without making any modifications to your sequencing project database. The excisions that GelMerge would have made if you had used `-EXCise` are clearly marked above the appropriate alignments in the Report file.

# GelMerge

After previewing the GelMerge decisions for vector sequence removal, you could alter the parameters for vector removal if you disagree with those decisions. For instance, the Report file may indicate that GelMerge would remove a 10 base sequence at the beginning of a fragment. If you don't want GelMerge to remove these bases, you can choose to increase the minimum run of contiguous identities that must be found at least once in a match with `-VECTORMINIdentity` (see the ALGORITHM topic above). Alternatively, instead of allowing GelMerge to automatically remove vector sequences from your project fragments, you could use the alignments listed in the Report file as a guide to make manual excisions in GelAssemble.

If a project fragment contains only vector sequences, GelMerge writes a zero length contig into the project database after removing all bases from the contig. While this zero length fragment provides a record of the original fragment, you may eventually want to remove it from your sequencing project database. You can do this in GelAssemble with the `: ERASE` command (see the GelAssemble entry in the Program Manual). GelMerge notifies you if your project contains any zero length contigs in the screen summary displayed at the end of the program.

## Execution Speed and the Batch Queue

GelMerge uses an algorithm in which computation time is approximately proportional to the number of contigs multiplied by the total length of all contigs. For example, using the default program parameters, it takes a DEC 5000/300 about one minute of CPU time to assemble a project containing 300 fragments with an average length of 400 bases (119,994 total bases) into a single contig of length 21,139. As another example, it takes about 25 seconds of CPU time to assemble a project containing 200 fragments with an average length of 300 bases (60,018 total bases) into five contigs. In a subsequent session with GelMerge, by reducing the required fraction of word matches in an overlap from 0.8 to 0.7, it takes about 7 seconds of CPU time to assemble the five contigs into a single contig of length 10,716.

You may want to consider running GelMerge in the batch queue for large sequencing projects. 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 assemblies may exceed the CPU limit set by some systems.

## 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: `% gelmerge -Default`

Prompted Parameters:

<code>-WORDsize=7</code>	sets word size for overlap determination
<code>-STRIngency=0.8</code>	sets minimum fraction of matching words in overlap
<code>-MINOverlap=14</code>	sets minimum length of overlap

## Local Data Files:

-MATRix1=gelmergedna.cmp      assigns the scoring matrix for contig assembly  
 -MATRix2=gelmergelocaldna.cmp      assigns the scoring matrix for vector recognition

## Optional Parameters:

-MINIdentity=14      sets minimum run of identical bases found at least  
    once in an overlap between two contigs  
 -MAXGap=10      sets maximum gap size for overlap determination  
 -GAPweight=8      sets gap creation penalty in contig assembly  
 -LENgthweight=2      sets gap extension penalty in contig assembly  
 -ARChive      creates contigs from the original gel readings  
 -WORKing      creates contigs from individual working  
    fragment (with gaps removed)  
 -REPortfile[=Filename]      writes report of recognized vector sequences  
 -EXCise      removes vector sequences from single-fragment  
    contigs  
 -VECTORSTrigency=0.8      sets minimum fraction of matches in vector recognition  
 -VECTORMINIdentity=12      sets minimum run of identical bases found at least  
    once in a match between vector and fragment  
 -VECTORMAXGap=5      sets maximum gap size in first step of vector  
    recognition  
 -VECTORGAPweight=30      sets gap creation penalty in vector recognition  
 -VECTORLENgthweight=3      sets gap extension penalty in vector recognition  
 -NOMERge      suppresses contig assembly  
 -NOMONitor      suppresses screen trace of program progress  
 -NOSUMmary      suppresses screen summary at the end of the  
    program  
 -BATCh      submits program to the batch queue

**ACKNOWLEDGEMENTS**

GelMerge was written by Irv Edelman.

**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.

# GelMerge

## 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.

GelMerge uses the scoring matrix file `gelmergedna.cmp` to align overlapping contigs into a single assembly. GelMerge uses the scoring matrix file `gelmergelocaldna.cmp` to align fragment and vector sequences as the final step in vector sequence recognition. Neither scoring matrix contains values for the comparison of IUB nucleotide ambiguity symbols, but nucleotide ambiguity symbols are completely supported in these GelMerge alignments. Any ambiguity symbol is converted into a weighted representation of its constituent bases within GelMerge. For instance, GelMerge treats an R as half A and half G for the purpose of alignment.

## PARAMETER REFERENCE

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

`-WORDsize=7`

sets the word size for overlap determination. This is the minimum contiguous length of matching nucleotides that is needed to extend an overlap between two sequences.

`-STRIngency=0.8`

sets the minimum fraction of matching words in an overlap to be considered for assembly into a contig.

`-MINOverlap=14`

sets the minimum length of an overlap to be considered for assembly into a contig.

`-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.

Use `-MATRix1` to specify a different scoring matrix for contig assembly and `-MATRix2` to specify a different scoring matrix for vector recognition.

**-MINIdentity=14**

sets the minimum size for the long block of contiguous sequence identities that must be found at least once in an overlap between two contig consensus sequences (see the ALGORITHM topic for more information).

**-MAXGap=10**

In overlap determination, sets the maximum amount by which an approximate alignment can deviate from registers of comparison containing long blocks of sequence identities (see the ALGORITHM topic for more information).

**-GAPweight=8**

sets the gap creation penalty for the alignment of contigs during the assembly.

**-LENGthweight=2**

sets the gap extension penalty for the alignment of contigs during the assembly.

**-ARCHive**

reassembles the entire project from the original gel readings. All current contig relationships and edits to fragments are lost.

**-WORKing**

reassembles the entire project from the individual, edited fragments. GelMerge saves all your edits to the fragment sequences, but all current contig relationships are lost. Before reassembly, the program removes all gap characters from the project sequences.

**-REPortfile=myproject.report**

writes an output file with sequence alignments of matching regions between vector and fragment sequences. The output file uses the project name as the file name and .report as the file name extension unless you set it to something else.

**-EXCise**

removes vector sequences from the ends of unaligned project fragments (single-fragment contigs). This parameter also writes an output file with sequence alignments of matching regions between vector and fragment sequences unless you suppress it by also specifying **-NOREPortfile**. The output file uses the project name as the file name and .report as the file name extension unless you set it to something else with **-REPortfile**.

**-VECTORSTringency=0.8**

sets the minimum fraction of sequence identity in a match between fragment and vector sequences that is required for vector sequences to be excised from the fragment.

# GelMerge

## **-VECTORMINIdentity=12**

sets the minimum size for the long block of contiguous sequence identities that must be found at least once in a match between fragment and vector sequences (see the ALGORITHM topic for more information).

## **-VECTORMAXGap=5**

In the first step of vector recognition, sets the maximum amount by which an approximate alignment can deviate from registers of comparison containing long blocks of sequence identities (see the ALGORITHM topic for more information).

## **-VECTORGAPweight=30**

sets the gap creation penalty for the alignment of vector and contig sequences during the second step of vector recognition (see the ALGORITHM topic for more information).

## **-VECTORLENgthweight=3**

sets the gap extension penalty for the alignment of vector and contig sequences during the second step of vector recognition (see the ALGORITHM topic for more information).

## **-NOMERge**

suppresses contig assembly.

## **-MONitor**

shows the progress of GelMerge on your screen. Use this parameter to see this same monitor in the log file for a batch process. If the monitor slows down the program because your terminal is connected to a slow modem, suppress it by including **-NOMONitor**.

## **-SUMmary**

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

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

## **-BATch**

submits the program to the batch queue for processing after prompting you for all required user inputs. Any information that would normally appear on the screen while the program is running is written into a log file. Whether that log file is deleted, printed, or saved to your current directory depends on how your system manager has set up the command that submits this program to the batch queue. All output files are written to your current directory, unless you direct the output to another directory when you specify the output file.

Printed: December 1, 1998 10:25 (1162)

## GELSTART

### FUNCTION

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

### DESCRIPTION

See the Fragment Assembly System (FAS) Introduction for an overview of working with the programs within the FAS to assemble sequences in a sequencing project.

GelStart initializes the fragment assembly environment, *locking in* the GCG Fragment Assembly System to the project you name. FAS continues to operate on this sequencing project during the current session until you rerun GelStart with a different project name. You must run GelStart each time you use FAS on a different project.

When you name a project, GelStart determines if you are naming a new or existing project. GelStart searches for an existing project of the same name in several places. The order of the search is as follows: 1) the current working directory; 2) a directory defined with the logical name FASPROJECTS; and 3) all of your directories, beginning with the top (or home) directory. (See Chapter 1, Getting Started in the User's Guide for help on defining logical names.) If GelStart finds the project you named, it checks the project database for integrity and then sets the other programs in the fragment assembly package (GelEnter, GelMerge, GelAssemble, GelView and GelDisassemble) to operate on that project. For GelStart to recognize an existing project, it not only must find a root directory with the same name as the project, but also must find the appropriate data subdirectories below the root directory. (See the Introduction to the Fragment Assembly System essay that precedes the FAS program entries in the Program Manual for a more complete description of the fragment assembly database.)

If GelStart does not find the project you have named, it offers to create a new project with that name. New projects are created in your current working directory.

With a new project, GelStart asks if there are any vector sequences that you want highlighted. You can respond with any valid GCG sequence specification. GelEnter then highlights the indicated sequences as you enter them from a digitizer or keyboard. GelMerge reports and optionally excises the indicated sequences from unaligned fragments before it aligns those sequences into assemblies, called contigs. You are not restricted to highlighting only vector sequences. For example, you could specify a fragment that has been over-cloned. GelStart also prompts you to enter any restriction sites that you want highlighted. You must respond with the actual sequence patterns, separated with commas; do not respond with GCG sequence specifications. If you highlight restriction sites, GelEnter can warn you of possible rejoined sticky fragment ends.

### EXAMPLE

Here is a session using GelStart to create a new project named "myproject":

```
% gelstart
```

```
What is the name of your fragment assembly project? myproject
```

```
GELSTART cannot find this project. Is it a new one? (* No *) Yes
```

# GelStart

You have a new project named "myproject".

Which vector sequence(s) would you like highlighted? **GB:M13mp18,GB:SynpBR322**

Which restriction site(s) would you like highlighted? **GAATTC,GGATCC**

Project MYPROJECT has 0 fragments in 0 contigs.

You are ready to run the other fragment assembly programs.

⌘

## OUTPUT

In this example, you would find a new file in your current working directory named myproject.dir. (See the Introduction to the GCG Fragment Assembly System preceding the FAS program entries in the Program Manual for a more complete description of the fragment assembly database.)

## RELATED PROGRAMS

GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project. GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files. GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble. GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge. GelView displays the structure of the contigs in a fragment assembly project. GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

## RESTRICTIONS

A contig may not contain more than 1,650 fragments and may not be longer than 200,000 bases. No single fragment may be longer than 2,500 bases.

## CONSIDERATIONS

The project name must consist only of characters that can be used in a normal file specification. You cannot have two projects with the same name.

## SUGGESTIONS

You can potentially work on any fragment assembly project by setting your default directory to the directory where the gel project is located. However, *make sure that only one person works on the project at a time!*

GelStart prompts you for highlighted vector sequences and restriction sites only when you create the project. To reenter or change these specifications, you must use **-VECTors** and **-SITes**.

Some of the fragment assembly programs (GelEnter, GelMerge, and GelAssemble) make use of command-line initialization files in the project root directory. For a description of command-line initializing files, see Chapter 3, Using Programs in the User's Guide. If these files don't exist, GelEnter creates them. If a command line initialization file is corrupted, it is best to delete it and allow GelStart to recreate a new one.



## 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: % gelstart [-NAME=]MyProject -Default

Prompted Parameters:

<b>-NEW</b> project	begins a new sequencing project
<b>-VECT</b> ors= <b>GB:M13mp18,GB:SynpBR322</b>	highlights specified sequences in GELENTER
<b>-SIT</b> es= <b>GAATTC,GGATCC</b>	highlights specified patterns in GELENTER

Local Data Files: None

Optional Parameters:

<b>-DELE</b> te	deletes a whole project!
<b>-NOMON</b> itor	suppresses the screen monitor

## LOCAL DATA FILES

None.

## PARAMETER REFERENCE

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

**-NEW**project

creates a new sequencing project if one doesn't already exist with the same name.

**-VECT**ors=**GB:M13mp18,GB:SynpBR322**

selects vector sequences to highlight when you enter sequences in GelEnter. In GelMerge, you can use **-EXC**ise to remove these vector sequences from the ends of single-fragment contigs.

**-SIT**es=**GAATTC,GGATCC**

selects patterns to highlight when you enter sequences in GelEnter.

**-DELE**te

deletes a whole project and all of its files. Do not use this parameter unless you want to delete your entire database. Make copies of the sequences you want to save in another directory before using the **-DELE**te option.

# GelStart

## `-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 10:25 (1162)

## GELVIEW

### FUNCTION

GelView displays the structure of the contigs in a fragment assembly project.

### DESCRIPTION

See the Fragment Assembly System (FAS) Introduction for an overview of working with the programs within the FAS to assemble sequences in a sequencing project.

GelView makes a display of your fragment assembly project. A list of the fragments in each contig, along with a bar diagram representing the position, length, and orientation of each fragment, allows you to review the current status of your sequencing project.

### EXAMPLE

Here is a session using GelView to look at the contigs in "myproject." This project was created in the example sessions of GelStart, GelEnter, and GelMerge.

```
% gelview
```

```
What should I call the output file (* myproject.view *) ?
```

```
MYPROJECT has 12 Fragments in 3 Contigs
```

```
%
```

### OUTPUT

```
GELVIEW Fragment Assembly contig display of Project: myproject
September 24, 1998 09:05
```

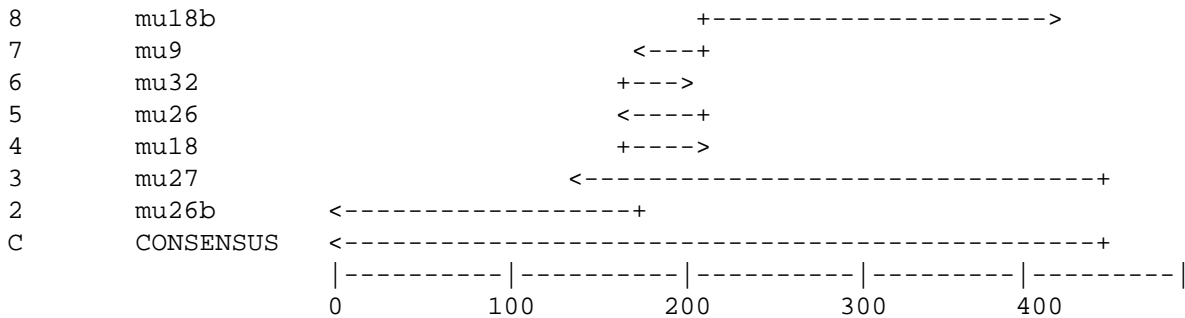
```
Contig: mu2
```

```

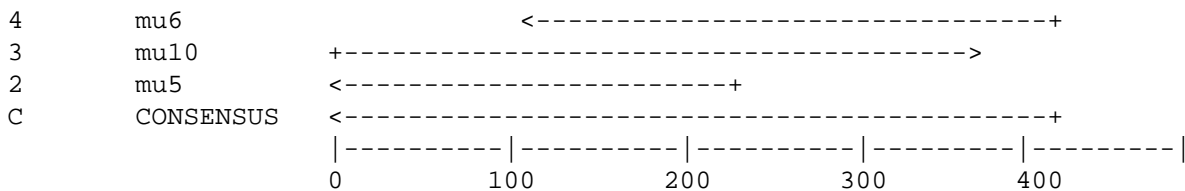
3      mu23      +----->
2      mu2       +----->
C      CONSENSUS +----->
          |-----|-----|-----|-----|
          0         100       200       300       400
```

# GelView

Contig: mu26b



Contig: mu5



12 Fragments in 3 Contigs

## RELATED PROGRAMS

GelStart begins a fragment assembly session by creating a new fragment assembly project or by identifying an existing project. GelEnter adds fragment sequences to a fragment assembly project. It accepts sequence data from your terminal keyboard, a digitizer, or existing sequence files. GelMerge aligns the sequences in a fragment assembly project into assemblies called contigs. You can view and edit these assemblies in GelAssemble. GelAssemble is a multiple sequence editor for viewing and editing contigs assembled by GelMerge. GelView displays the structure of the contigs in a fragment assembly project. GelDisassemble breaks up the contigs in a fragment assembly project into single fragments.

## RESTRICTIONS

A contig may not contain more than 1,650 fragments and may not be longer than 200,000 bases. No single fragment may be longer than 2,500 bases.

## 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: % gelview -Default

Prompted Parameters:

[-OUTfile=]myproject.view names the output file

Local Data Files: None

Optional Parameters:

-NOBAR suppresses the fragment bar diagrams

### **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.

**-NOBAR**

suppresses the bar diagram associated with each fragment name.

Printed: December 1, 1998 10:25 (1162)



## GETSEQ

### FUNCTION

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

### DESCRIPTION

GetSeq offers you a quick way to create a GCG sequence file from the keyboard without having to use SeqEd, Reformat, or a text editor. In addition, if you use a microcomputer that can act as a terminal, you can use GetSeq to move sequence files in text format from your microcomputer into your UNIX directory in GCG format. The microcomputer's terminal emulation software must be able to send ASCII text files to the Wisconsin Package™ program GetSeq.

When you run GetSeq, it opens a file that you name and then waits for input. You can either type the sequence characters in from the keyboard, or use your microcomputer's terminal emulation software to send a text file that contains sequence data.

GetSeq accepts all of the supported GCG sequence characters you send until it sees a <Ctrl>D. Then it writes those characters out as a sequence file that can be used by any GCG program. Since GetSeq only accepts sequence characters (see Appendix III), the sequence may be in any format; line feeds, carriage returns, spaces, and numbering are ignored. If you are sending a sequence text file to GetSeq, make sure it doesn't contain comments. Any letters in a comment that are valid sequence symbols will be treated as sequence data.

### EXAMPLE

Here is a session using GetSeq; the sequence GATTCCGATTG was sent followed by <Ctrl>D:

```
% getseq

GETSEQ into what sequence file ? temp.seq

Now start the sequence transfer . . .
End the transfer with a <Ctrl>D

GATTCCGATT G^D

Bases transferred: 11

%
```

# GetSeq

## OUTPUT

Here is the output file:

```
!!NA_SEQUENCE 1.0
GETSEQ from Edelman, October 20, 1996 10:13.

      Length: 11   September 29, 1998 18:05   Type: N   Check: 4920   ..

1  GATTCCGATT G
```

## RELATED PROGRAMS

The UNIX % `cat > temp.txt` command creates a regular text file and puts all of the characters you send from the terminal into it until you send a <Ctrl>D. (Typing <Return> starts a new record in the file -- it does not terminate text entry.)

Reformat is a utility for changing a text file that contains a sequence into a GCG sequence file.

SeqEd lets you edit a sequence and the documentary heading once you have the sequence as a file in GCG format.

## RESTRICTIONS

Your sequence must use the IUB-IUPAC character set for protein or nucleic acid sequences (see Appendix III). Change the sequence with a text editor on the micro if there are any characters in it that are unacceptable to the Wisconsin Package.

The sequence must not be longer than 350,000 characters to be acceptable to Wisconsin Package software.

If there is non-sequence data in the file, such as heading or documentary information, you must have a way to send only the sequence characters. You may need to edit the file on the microcomputer to remove non-sequence sections from the file.

## SEQUENCE TYPE

When GetSeq writes GCG sequence files, it assigns the sequence type based on the composition of the sequence characters. This method is not fool-proof, so you may need to change the sequence type of the newly created file. Look on the last line of the text heading just above the sequence itself for `Type: N` or `Type: P`. If the type is incorrect, see Appendix VI for information on how to change or set the type of a sequence.



**COMMAND-LINE SUMMARY**

Complete command-line control is not available for this program.

**LOCAL DATA FILES**

None.

**PARAMETER REFERENCE**

None.

Printed: December 1, 1998 10:25 (1162)



## GROWTREE+

### FUNCTION

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

### DESCRIPTION

GrowTree reconstructs a phylogenetic tree from a distance matrix such as the one created by Distances. Two methods are available for reconstructing the tree: UPGMA (unweighted pair group method using arithmetic averages) and neighbor-joining.

### EXAMPLE

Here is a session using GrowTree to reconstruct a tree from the distance matrix created in the sample session with Distances and create both a text representation and a graphical plot of the tree.

```
% growtree -FIGure

What is the distance matrix ? hum_gtr.distances

Which method to use ?

    1 Neighbor-joining
    2 UPGMA

Choose the method to use: (* 1 *)

What should I call the trees file (* hum_gtr.trees *) ?

3 internal, 5 terminal nodes

The minimum density for a one-page plot is 3.3 taxa/100 platen units.
What density do you want (* 3.3 *) ?

That will take 1 page. Is this all right (* yes *) ?

FIGURE instructions are now being written into growtree.figure.

%
```

# GrowTree

## OUTPUT

Here is the output trees file in NEXUS format:

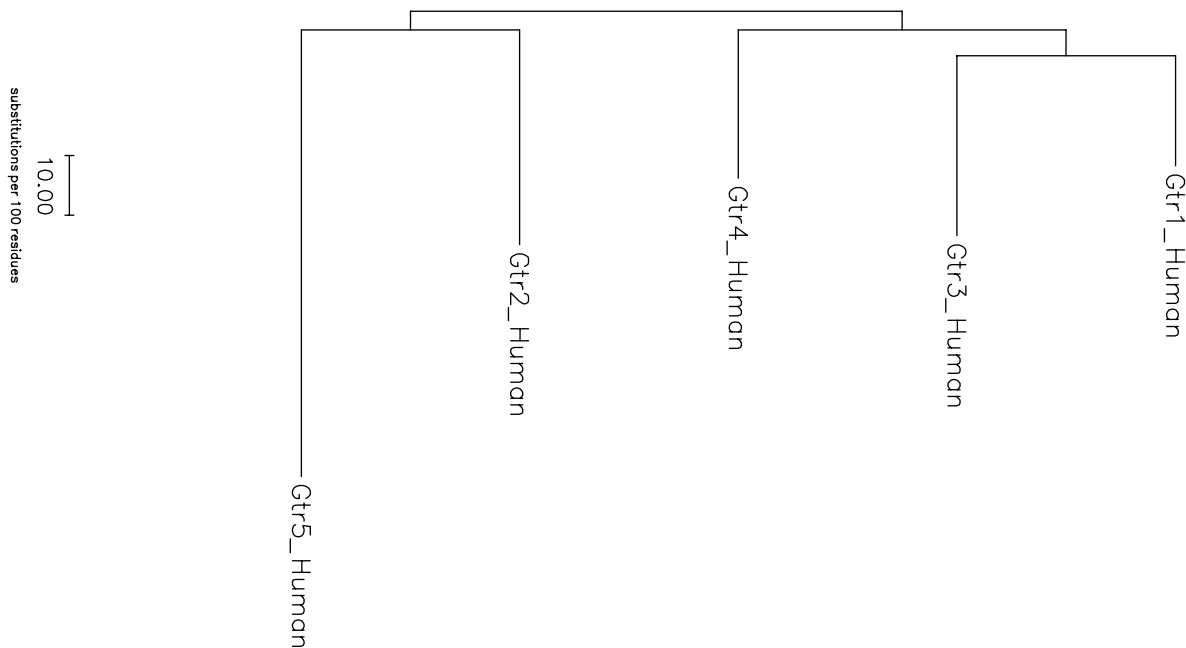
```
#NEXUS

[ Trees from file: hum_gtr.distances ]

begin trees;
utree Tree_1 =
  ((('Gtr1_Human':18.43,'Gtr3_Human':30.18):4.34,'Gtr4_Human':24.87)
:3.19,('Gtr2_Human':35.98,'Gtr5_Human':74.88):3.19):0.00;
endblock;
```

Here is the output Figure file:

Growtree Phylogram of: Hum\_Gtr.Distances, Tree Tree\_1  
October 26, 1998 14:38



## INPUT FILES

GrowTree accepts a distance matrix in the format produced by Distances and Diverge.

## RELATED PROGRAMS

PileUp creates a multiple sequence alignment from a group of related sequences using progressive pairwise alignments. It can also plot a tree that shows the clustering relationships used to create the alignment. LineUp can be used to create and edit multiple sequence alignments. Pretty displays multiple sequence alignments and calculates a consensus sequence. It does not create the alignment; it simply displays it.

The Wisconsin Package includes several programs for evolutionary analysis of multiple sequence alignments. Distances creates a matrix of pairwise distances between the sequences in a multiple sequence alignment. Diverge measures the number of synonymous and nonsynonymous substitutions per site of two or more aligned protein coding regions and can output matrices of these values. GrowTree reconstructs a tree from a distance matrix or a matrix of synonymous or nonsynonymous substitutions. PAUPSearch reconstructs phylogenetic trees from a multiple sequence alignment using parsimony, distance, or maximum likelihood criteria; PAUPDisplay can manipulate and display the trees output by PAUPSearch and can also plot the trees output by GrowTree.

## RESTRICTIONS

Unknown.

## ALGORITHM

### UPGMA

This method (Sneath and Sokal, *Numerical Taxonomy*, Freeman, San Francisco (1973)) can be used to estimate a species tree or gene tree when the expected rate of gene substitution is constant and the distance measure is linear with evolutionary time (for example, distance is measured as amino acid substitutions). The distances *must* be ultrametric to obtain a correct tree using this method.

The two sequences that have the smallest distance in the distance matrix are combined to form a cluster. That cluster replaces the original sequence pair as a single entry in the distance matrix (reducing the dimension of the matrix by one), and distances between the cluster and the other entries are calculated. The entries in the new matrix that have the smallest distance are combined to form a new cluster, and the process continues until only a single cluster remains. The resulting tree is a rooted tree.

Instead of using a simple average, the UPGMA method calculates the distances between a new cluster and the other entries in the distance matrix based on the total number of sequences in the cluster. If the new cluster  $C$  was formed by combining two clusters  $a$  and  $b$ , cluster  $a$  representing  $N_a$  total sequences and cluster  $b$  representing  $N_b$  total sequences, the distance between the new cluster  $C$  and another entry  $k$  is:

$$\text{distance}(k,C) = [ \text{distance}(k,a) * N_a + \text{distance}(k,b) * N_b ] / (N_a + N_b)$$

### Neighbor-Joining

This method is designed to find an approximation to the minimum evolution tree for a set of aligned sequences, using less computer time than the full algorithm for determining a minimum evolution tree. It works best when the distances are additive. The algorithm is that of Saitou and Nei, *Mol. Biol. Evol.* **4**; 406-425 (1987), simplified by Studier and Keppler, *Mol. Biol. Evol.* **5**; 729-731 (1988), and modified by Swofford, Olsen, Waddell, and Hillis in *Molecular Systematics*,

## GrowTree

second edition, ed. Hillis, Moritz, and Mable, Sinauer Associates, Inc., 1996, Ch. 11, "Phylogenetic Inference."

The neighbor-joining method clusters the sequences in a pairwise fashion. However, instead of picking the next pair to cluster by looking for the smallest distance in the distance matrix, this method seeks to form pairs that minimize the sum of the branch lengths for the entire tree. Therefore at each round of clustering, all possible pairs of entries are considered one at a time and the sum of the branch lengths for the resulting tree is calculated. The pairing that results in the smallest sum is the one that will be used to form the new cluster. This new cluster replaces its two constituent entries in the distance matrix (reducing the dimension of the distance matrix by one), and distances are calculated between the new cluster and the remaining entries in the distance matrix. The process continues until only two entries remain. The resulting tree is an unrooted tree. Because this method attempts to build an additive tree from the data, negative branch lengths may result if the distance data are not exactly additive (see the CONSIDERATIONS topic for more information on this).

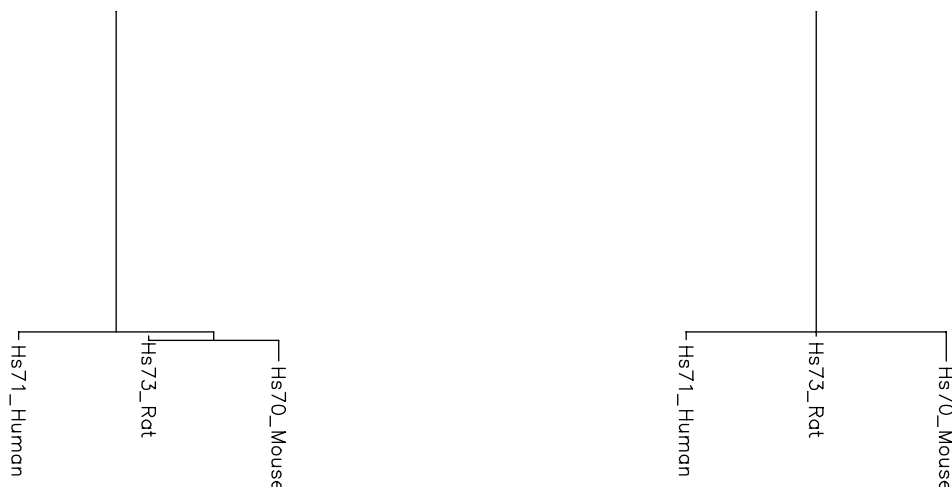
### CONSIDERATIONS

In order for these methods to produce correct trees, the steps leading up to this analysis (sequence alignment and calculation of pairwise distances) must be done carefully. For example, the alignment used to produce the distance matrix may have to be adjusted manually, especially for nucleic acid sequences that are coding regions. The proper distance correction method should be used to create the distance matrix from the alignment, depending on the characteristics of the sequences (whether the base composition or amino acid composition is skewed, whether the substitution rate varies greatly from site to site, etc.) and on the assumptions made by each of the distance correction methods.

#### Negative Branch Lengths and Small Branch Lengths

You may notice some tree branches pointing "up" in the tree plot and negative branch lengths in the corresponding trees file. This can occur with the neighbor-joining method because the algorithm tries to represent the data by an additive tree. If the distances are not perfectly additive, negative branch lengths can result.

The appearance of negative branch lengths is not necessarily a problem. There is usually some error in calculating pairwise distances, so the distances are seldom perfectly additive. Short negative branch lengths may result from this. Short negative (or positive) branch lengths may also suggest that a *polytomy* exists. The neighbor-joining method reconstructs a tree by considering pairs of distances and has no method of dealing with a situation where more than two taxa branch off at the same point (a polytomy). If the branch lengths resulting from a neighbor-joining tree reconstruction are very small (either negative or positive), it may be an indication that the taxa with the short branch lengths should be represented at the same level as neighboring taxa instead of at different levels. For example, in the figure below, the tree on the right (with the trisomy) is probably a better representation than the binary tree on the left. In order to resolve a polytomy, more data will be needed.



On the other hand, long negative branch lengths often indicate a problem with the distance data or with the sequence alignment from which the distance matrix was calculated. You should double-check the alignment to make sure it is the best possible alignment for the data, and examine the distance matrix used to reconstruct the tree (see the **CONSIDERATIONS** topic in the documentation for the Distances program). If the distance matrix contains a lot of infinite distances (represented by 999.99), the tree built from the matrix may be incorrect.

In some cases, you can create a better distance matrix. For example, if the matrix was created without correcting the distances for multiple substitutions at a single site, recreate the matrix using an appropriate correction method.

Another correction you can make is to eliminate the effects of "randomization" of the third position of codons in coding regions. Because the third position can change without altering the amino acid that the codon specifies, this position often has a much higher substitution rate than the other two positions. This contributes noise to the distance matrix. To eliminate the noise, recreate the distance matrix using only the first two positions of the codon (Distances with **-POSITION=4**). Alternatively, translate the coding regions, align the resulting amino acid sequences, and use the protein alignment to create the distance matrix.

**-NONEG**ative resets negative branch lengths to zero after the tree is constructed. An appropriate use of this parameter is to neaten the plot of a tree that contains negative branch lengths that are short. *Do not use this parameter to disguise the fact that your tree has long negative branches!* If long negative branches are present, you should be examining your data and rethinking your strategy of tree reconstruction, not worrying about visual esthetics.

### Branch Length Units

When **-TREEFORMat=1**, a phylogram is produced and distances along the branches of the tree are indicated by a bar labeled "X substitutions per 100 residues." If the distance matrix was created by Distances, this label is correct. If the distances in your distance matrix use a metric other than substitutions per 100 residues, this label will not be correct. (In particular, distances derived from OldDistances are *not* expressed as substitutions per 100 amino acids.) Make sure

## GrowTree

that the label on the plot is changed to the proper units before publishing the phylogram. This can be done by running GrowTree with `-FIGure=growtree.figure` to create a figure file named `growtree.figure`, editing this file to change the label for the distance bar, and running Figure on this edited figure file to generate the tree.

### 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: `% growtree [-INfile=]hum_gtr.distances -Default`

Prompted Parameters:

```
[-OUTfile=]hum_gtr.trees  names the output file
                           (tree information in NEXUS format)
-MENu=1                    uses neighbor-joining to reconstruct the tree
      2                    uses UPGMA method to reconstruct the tree
-DENsity=20.0              sets number of sequences/100 pu in the tree plot
```

Local Data Files: None

Optional Parameters:

```
-NONEGative    resets negative branch lengths to zero
-NOBRanch      suppresses reporting branch lengths in trees file
-ROUND         reports branch lengths in trees file to nearest integer
-NOPLot        suppresses graphical display of tree

-ORDER=1       orders sequences in tree display using "standard" order...
      =2       reverse standard order
      =3       alphabetically by name (ascending)
      =4       alphabetically by name (descending)
      =5       "laddered" to the left by number of descendents
      =6       "laddered" to the right by number of descendents

-TREEFORMat=1  draws the tree as a phylogram
      =2       draws the tree as a cladogram
```

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**

None.

**PARAMETER REFERENCE**

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

**-MENu=1**

sets the method used to reconstruct the tree: (1) neighbor joining (default), (2) UPGMA.

**-DENsity=20.0**

sets the number of sequences to plot per 100 platen units in the graphical output.

**-NONEGative**

resets any negative branch lengths to 0.0 after the tree is built.

**-NOBRanch**

suppresses the reporting of branch lengths in the trees file.

**-ROUND**

reports branch lengths in the trees file to the nearest integer.

**-NOPLot**

suppresses the output of a graphical representation of the tree.

**-ORDER=0**

sets the method for ordering sequences (as far as is possible) in the tree display: "standard" order (1), reverse standard order (2), order alphabetically by ascending name (3) or by descending name (4), "ladder" the taxon groups to the left (5) or to the right (6) according to the number of descendants of each internal node.

**-TREEFORMat=1**

sets the format to use in drawing the tree: phylogram (1) is drawn with branch lengths proportional to calculated distances; cladogram (2) is drawn with constant branch lengths.

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.

# GrowTree

**-FONT=3**

draws all text characters on the plot using Font 3 (see Appendix I).

**-COLor=1**

draws the entire plot with the pen in stall 1.

The parameters below let you expand or reduce the plot (*zoom*), move it in either direction (*pan*), or rotate it 90 degrees (*rotate*).

**-SCAlE=1.2**

expands the plot by 20 percent by resetting the scaling factor (normally 1.0) to 1.2 (zoom in). You can expand the axes independently with **-XSCAlE** and **-YSCAlE**. Numbers less than 1.0 contract the plot (zoom out).

**-XPAN=30.0**

moves the plot to the right by 30 platen units (pan right).

**-YPAN=30.0**

moves the plot up by 30 platen units (pan up).

**-PORtrait**

rotates the plot 90 degrees. Usually, plots are displayed with the horizontal axis longer than the vertical (landscape). Note that plots are reduced or enlarged, depending on the platen size, to fill the page.

Printed: December 1, 1998 10:25 (1162)