8. FULL DESCRIPTIONS

TABLE OF CONTENTS

The **Full Description** section of the **User's Guide** gives you the detailed descriptions of the most important and most frequently used HUSAR/GCG programs.

Fetch 8-3

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

SeqEd 8-7

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

Map 8-19

Map displays both strands of a DNA sequence with restriction sites shown above the sequence and possible protein translations shown below.

Gap 8-29

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.

BlastN 8-39

BlastN compares a nucleotide query sequence against a nucleotide sequence database. BlastN is more than an order of magnitude faster as FastA and provides a statistical significance but tends to be less sensitive.

FastA 8-49

FastA does a Pearson and Lipman search for similarity between a query sequence and any group of sequences. For nucleotide database searches, FastA is more sensitive than BLAST.

IRX 8-59

IRx is an Information Retrieval System that identifies sequences. IRx does this by searching for author names, accession numbers, sequence names or for any other keywords within the annotations of the sequence databases.

FETCH

FUNCTION

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

DESCRIPTION

The expression `% fetch *bov*` will retrieve every GCG data file or sequence entry whose name contains the string *bov*. Sequence specification is described in detail in the Chapter 2, **Using Sequences** of the **User's Guide**.

When copying a sequence from a database, Fetch creates a file in GCG format whose name is the entry name and whose extension is the database logical name. For example, `% fetch EMBL:Hsrep2` copies the requested sequence into a file called hsrep2.em_pr. The filename extension is taken from the logical name for the database. In this example, the extension .em_pr indicates that the sequence was copied from the Primate division of the EMBL nucleotide sequence database. (See "Using Database Sequences" in Chapter 2, **Using Sequences** of the **User's Guide** for a complete listing of logical names for all available in HUSAR/GCG databases.) If the file being copied is not from a sequence database, for example enzyme.dat, then its name is not changed.

If your sequence specification contains no logical name, Fetch looks in all the databases and in all the GCG data directories to find all possible entry names. For example, `% fetch hum*` would do almost the same thing as `% fetch GenBank:hum*`, except that if any sequences beginning with *hum* were present in databases other than GenBank or in any GCG data directories, they would also be retrieved.

Special Considerations for Searching

Keep in mind that filenames are case sensitive and database entry names are case *in*sensitive. Because this program searches for both filenames and database entry names, you must take care when you enter the character pattern that makes up your specification.

For example, if you entered **`Gamma*`** as a file specification, this program would find all entries in the databases whose names begin with *Gamma* but no GCG-supplied files would be found. This is because all the files in the Wisconsin Package are named using lowercase letters. Conversely, if you entered **`gamma*`**, this program would find all of the entries in the databases *and* all the GCG-supplied files whose names begin with *gamma*.

(Note that it is often convenient to add `-OUTfile=Term` to the command line so the data are displayed on your terminal screen.)

EXAMPLE

Here is a session using Fetch to retrieve local copies of most of the GenBank human beta globin sequences:

```
% fetch

 FETCH what sequence(s) ?  gb:humhb*

 humhb16aa.gb_pr
 humhb24.gb_pr
 humhba1.gb_pr

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

 humhbvint.gb_pr

%
```

COMMAND-LINE SUMMARY

All parameters for this program may be put on the command line. Use the option **-CHE**ck to view the summary below and to add things to the command line before the program executes. In the summary below, the capitalized letters in the qualifier names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose qualifiers or parameter values that are optional. The "Using Program Parameters" section in the Chapter 3, **Using Programs** of the **User's Guide** describes how to use command lines effectively.

```
Minimal Syntax: % fetch [-INfile=]GenEMBL:Humhb*

Prompted Parameters: None

Local Data Files: None

Optional Parameters:

-OUTfile=FileName   copies file(s)-sequence(s) into one file
-DOCLines=6         copies only the first 6 lines of documentation.
-NOMONitor          suppresses the screen monitor
-REFerence          copies only the documentation
```

LOCAL DATA FILES

None.

OPTIONAL PARAMETERS

The parameters and switches listed below can be set from the command line. Optional parameters available to all programs are described in the "Using Program Parameters" section of Chapter 3, **Using Programs** of the **User's Guide**.

-DOCLines**=6**

sets GCG programs to copy only six non-blank lines of documentation from input data files into the output files. Use the % **doclines** command to set this parameter for your whole session. Usually, Fetch copies all of the documentation from each sequence entry into your new files exactly as it appeared in the original entry.

-OUT`file`**=filename**

copies the sequence(s) and/or data file(s) into one file which you can name. If you leave out the name of the file, Fetch prompts you for one. (Wisconsin Sequence Analysis Package™ programs will not read files containing more than one sequence unless they are in an MSF (multiple sequence format) file.)

It is often useful to use **Term** for the filename so that the data are displayed on your terminal screen.

-REF`erence`

copies only the documentation for the sequence or data file. Unless specified, the name of the output file is the entry name concatenated with `_ref`, followed by the database logical name as the extension.

-MON`itor`

This program normally monitors its progress on your screen. However, when you use the **-D**`efault` option to suppress all program interaction, you also suppress the monitor. You can turn it back on with this option. If your program is running in batch, the monitor will appear in the log file. If the monitor is slowing the program down, suppress it with **-NOMON**`itor`.

SEQED

FUNCTION

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

DESCRIPTION

SeqEd uses the screen of your terminal as a window into a sequence. Changes you make in the sequence take place at the cursor position and are reflected immediately on the screen. You can insert or delete symbols, move the cursor, search for patterns, check sequences by reentering them, and edit documentation and embedded comments.

SetKeys lets you change the positions of the keys on your terminal keyboard to make it more convenient to enter the letters G, A, T, and C.

You can enter a sequence and control SeqEd either from the terminal keyboard or from a Graf/Bar digitizer.

EXAMPLE

When you run SeqEd with a command like `% seqed sample.seq`, your screen will look something like this:

```
sample.seq            ***** K E Y B O A R D *****                seqed
    : Some documentary text about your sequence can be placed          :
    : here in the heading.                                             :
    : You can have as many lines of header comments as you wish, and   :
    : you can edit them in this space with the HEAding command.        :

        1 $The scale below has extra dots where comments occur$
        9 <The number indicates which symbol the comment precedes.<
       20 >There can be as many comments as you like>
       28 $The four comments closest to the cursor are displayed$



      AGTCTTAGTCGATCGTAcTGCATRCGA

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


     ~~~~~~~~~~~~~~~~~~~^

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

   "sample.seq"   27 nucleotides
```

EDIT NEW OR EXISTING SEQUENCES

If you name a sequence file that already exists, SeqEd displays the first four lines of documentation on the top of the screen followed by up to four embedded comments and the base number with which each is associated. SeqEd shows the end of the sequence across the middle of the screen.

If the sequence you name does not exist, SeqEd starts in Heading Mode (see below) to allow you to enter documentation for the new sequence. Use **`<Ctrl>D`** to stop editing the documentation.

SCREEN MODE

Entering a Sequence

In Screen Mode the cursor shows your position in the sequence. You can move around in the sequence, add symbols, delete symbols, and search for patterns. You can insert any valid GCG sequence symbol (see Appendix III) into the sequence by typing the symbol. It is inserted at the cursor.

Deleting a Sequence

The **`<Delete>`** key deletes the symbols to the left of the cursor, one by one.

Moving the Cursor

To move the cursor to the right, use the **`<Right-arrow>`** key; to move to the left, use the **`<Left-arrow>`** key. Movements are confined to the length of the sequence.

If you type a number followed by **`<Return>`**, the cursor moves to that sequence position.

The arrow keys can be preceded by a number indicating how many symbols to move to the left or right. **`10<Right-arrow>`** moves 10 symbols to the right.

Finding Patterns

To search for a pattern, type a **`/`** (slash) in Screen Mode. The cursor moves to the lower-left corner of the screen to let you enter a sequence pattern that you wish to find. You may type in a pattern up to 40 characters long. You can repeat the last search by simply pressing **`/<Return>`**. SeqEd treats all nucleic acid sequences as circular and finds your pattern even if it wraps from the end of the sequence into the beginning. SeqEd uses the same rules for pattern definition and recognition as the programs FindPatterns, MapPlot, Map, and MapSort.

The command-line options **`-PRO`**`tein` and **`-PERF`**`ect` or the **`PRO`**`tein` or **`PERF`**`ect` commands in Command Mode make SeqEd treat the sequence as linear and disable the nucleic acid ambiguity meanings of the GCG sequence symbols (see Appendix III) during pattern searches.

The **`NUC`**`leotide` command in Command Mode tells SeqEd to recognize patterns containing IUB nucleotide ambiguity symbols during searches.

Even if SeqEd thinks your sequence is nucleotide, you can request a perfect-match search by typing **`=`** after the **`/`**. For example, **`/=RCT`** only matches RCT (case does not matter) no matter which kind of sequence SeqEd thinks you have.

Finding a Marked Position

You can *mark* a position in a sequence to which you wish to return. You give the marked position a letter (like giving it a name) using the Command Mode `Mark` command (see below). Then, in Screen Mode, a single quote followed by the letter used to mark the sequence moves the cursor to the position where that mark was defined.

Leaving Screen Mode

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

Screen Mode Summary

Here is the summary of Screen Mode commands in the on-line help:

```
                  Screen Mode

      [n] is an optional numeric parameter.

 G, A, T, . . .   - insert a sequence character
 <Delete>         - delete a sequence character
 /TAACG<Return>   - find the next occurrence of TAACG
                      (last pattern entered is the default)
 <Ctrl>H          - move to start of the sequence
 <Ctrl>E          - move to end of the sequence
 [n]<Right-arrow> - go ahead n characters
 [n]<Left-arrow>  - go back  n characters
 <Up-arrow>       - go up to check sequence
 <Down-arrow>     - go down to original sequence
 'markcharacter   - go to marked position
 37<Return>       - go to position 37 (any positive integer)
 <                - go back  50 characters
 >                - go ahead 50 characters
 <Ctrl>R          - redraw the screen
 <Ctrl>D          - enter command mode
```

COMMAND MODE

Use **`<Ctrl>D`** in Screen Mode to enter Command Mode. The cursor moves down to the lower-left corner of the screen next to a colon prompt after which you can enter any of the commands shown below followed by **`<Return>`**.

Editing SeqEd Commands

SeqEd command editing is modeled on VMS DCL command-line editing. The **`<Left-arrow>`** and **`<Right-arrow>`** keys let you move your cursor around in a command that you have typed so you can insert or delete characters at any position. **`<Ctrl>H`** and **`<Ctrl>E`** move the cursor to the beginning (head) and end of the line, respectively. **`<Ctrl>U`** deletes all the characters from the current cursor position to the start of the line.

Editing Previous SeqEd Commands

SeqEd lets you modify and execute previous commands. The **`<Up-arrow>`** key displays previous commands.

Returning to Screen Mode

If you press **`<Return>`** without entering a command, SeqEd returns to Screen Mode (described above). If you have **`-SING`**`lecommand` on the command line or in your command-line initializing file, SeqEd returns to Screen Mode immediately after executing each command.

Commands May Be Shortened

Only the capitalized portion of the commands described in the documentation below needs to be typed.

Parameters are Used with Commands

Some commands can be preceded with numeric parameters or succeeded with a filename. The square brackets ([and]) in the documentation below show command parameters that are *optional*, meaning you can leave them out.

Command Mode Summary

Here is the summary of Command Mode commands you would see with the **`H`**`elp` command:

```
                        Command Mode

 Commands end with <Return>. [n] indicates an optional parameter.
 s and f are numbers for start and finish of a range of interest
 Only the capitalized part of the command is necessary.

      EDit seqname       - get a new sequence file to edit
[n]   Include [seqname]  - insert another sequence [at position n]
                           (SeqEd prompts for range and strand)
s,f   Delete             - delete a range of bases
[s]   Check [/Blind]     - check a range of bases [beginning at s]
      37                 - go to base 37
      REDraw             - redraw the screen
[n]   COmment comment    - insert a comment [at position n]
[n]   COmment            - enter comment editing mode [at position n]
[n]   HEAding            - edit documentary heading [at line n]
      change             - enter screen mode (<Return> is sufficient)
      screen             - enter screen mode (<Return> is sufficient)
      OVERstrike         - enter overstrike mode
      INSert             - enter insert mode
[n]   Mark markcharacter - mark the sequence [at position n]
      PERFect            - require finds to be perfect matches
      PROtein            - set sequence type to PROTEIN
      NUCleotide         - set sequence type to NUCLEOTIDE
[s,f] Write [seqname]    - write [a part of] the sequence to a file
      DIGitizer          - enter digitizer mode
      RELoad             - enter reload mode
      ACCept             - terminate reload mode
      Help               - show commands in screen and command modes
[s,f] EXit  [seqname]    - write [a part of] the sequence and quit
      Quit               - quit the editor without writing the sequence
```

ED`it` **SeqName**

gets a new sequence from the file you have named for editing with SeqEd. The sequence you are currently editing is lost if you have not written it out before using the **ED**`it` command.

`[s]` **I**`nclude [filename]`

includes another sequence within the sequence being edited at the current cursor position or at the position specified by the optional parameter. SeqEd creates two embedded comments at the start and end of the included section to show what was included. If you do not supply a filename with this command, SeqEd prompts you for one.

s,f **D**`elete`

deletes some or all of the sequence. You must specify a beginning and ending coordinate for the range of symbols you want to delete.

`[s]` **CH**`eck [`**/B**`lind]`

lets you check a sequence entry in Screen Mode. A sequence already entered may be typed in again. If a symbol is typed that disagrees with the first entry, a **^** is printed at the point of disagreement and the terminal bell rings. While checking, the **<Up-arrow>** and **<Down-arrow>** keys move the cursor back and forth between the second entry and the original sequence, allowing you to make changes in either one as mistakes are found. If the optional starting coordinate precedes the command, it specifies where checking begins. If you wish to check your sequence without seeing the original version, type the qualifier **/B**`lind` following the **CH**`eck` command (there must be a blank between the **CH**`eck` command and the **/**).

RED`raw`

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

`[s]` **CO**`mment [comment text]`

allows you to enter, delete or modify embedded comments to document your sequence. In its simplest use, the **CO**`mment` command lets you insert new comments. You simply type the entire comment on the command line. Deletion and modification of existing comments is handled by entering Comment Mode. To do this, you type only the **CO**`mment` command and optional position but no comment text. See the COMMENT MODE topic for more information.

Whenever you enter a comment, SeqEd ensures that comment-delimiting characters are placed around it. A **$**, **<**, or **>** must appear at each end of, and not within, your comment. (SeqEd deletes comment delimiting characters found within a comment when they are the same as the flanking comment delimiting characters.)

SeqEd inserts new comments at your current cursor position or at the position specified by the optional position number and then returns to Command Mode.

`[s]` **`HEA`**`ding`

enters Heading Mode, which lets you edit the documentary heading. You can modify any part of the heading. Heading Mode is terminated with **`<Ctrl>D`**. The optional parameter specifies which line of the heading you want to start editing.

`change`

returns your session to Screen Mode. Note that the entire command is optional and a simple **`<Return>`** is equivalent.

`screen`

returns your session to Screen Mode. Note that the entire command is optional and a simple **`<Return>`** is equivalent.

`OVER``strike`

enters overstrike mode. Typing in a new symbol deletes the old symbol at that position and replaces it with the new symbol.

`INS``ert`

enters insert mode. Typing in a new symbol shifts all symbols from the current position to the end of the sequence by one position to the right and adds a new symbol at the current position.

`[n]` **`M`**`ark markcharacter`

You can mark a position in the sequence if you wish to return to it later. If the optional position number is absent, the position marked is the current cursor position. You give the marked position a letter (like giving it a name) using this command. Then, in Screen Mode, a single quote followed by the letter used to mark the sequence moves the cursor to the position where that mark was defined.

`PERF``ect`

makes searches linear and disables the nucleic acid ambiguity meanings of the GCG sequence symbols (see Appendix III).

`PRO``tein`

sets the sequence type to protein. This makes searches linear and disables the nucleic acid ambiguity meanings of the GCG sequence symbols. This also makes SeqEd ignore any set.keys file in your local directory.

`NUC``leotide`

sets the sequence type to nucleotide. This makes searches circular and tells SeqEd to recognize patterns containing IUB nucleotide ambiguity symbols. SeqEd also remaps the keys if a set.keys file is in your local directory.

`[s,f]` **`W`**`rite [filename]`

writes the current form of the sequence into a file. If you supply starting and finishing coordinates, SeqEd only writes the indicated segment. For example, **`1,56 W`**`rite` would write symbols 1 to 56 into a file. If you name a file, SeqEd writes the sequence into a file with that name instead of into the input file.

RELoad

goes into Reload Mode, which is similar to Checking Mode, except that the reloaded sequence grows leftwards from the right end of the main sequence. This is designed to help find the overlap of two loadings of the same reaction. Mismatched bases are marked with **^** (caret) characters, as in Checking Mode. Also, you can use the arrow keys to move around in and edit either the main sequence or the reloaded sequence. When the match becomes especially good, the terminal bell rings. You are free to accept or reject SeqEd's rules of what constitutes a good overlap. See the COMMAND LINE SUMMARY topic below for more information.

ACCept

terminates Reload Mode. The display of the reloaded sequence goes away, leaving you with only the main sequence with the cursor at the end, ready for more input. SeqEd helps you to decide when to **ACC**ept an overlap, but the decision is yours.

Help

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

[s,f] **EX**it [filename]

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

Quit

terminates a session with SeqEd without writing a new sequence file.

COMMENT MODE

Comment Mode allows you to add, change, or delete embedded comments and helps you move quickly to any position in your sequence where a comment is associated. To enter Comment Mode, you must first enter Command Mode with **<Ctrl>D**.

Entering New Comments

If you type the **CO**mment command without any comment text, SeqEd creates a new, empty comment at the position indicated by the optional sequence position number, if present, or at your current position in the sequence. The cursor moves to the part of the screen where embedded comments are displayed. Initially, the cursor is adjacent to a position number followed by an empty comment. You may then type a new comment or move to an existing comment that you wish to modify. Only one new comment can be created each time you enter Comment Mode.

Cursor Movement in Comment Mode

While in Comment Mode you can move around in the comment using the **<Left-arrow>** and **<Right-arrow>** keys, insert text by typing, or delete text using the **<Delete>** key. **<Ctrl>H** and **<Ctrl>E** position the cursor at the beginning (head) or end of the comment. **<Ctrl>U** deletes all characters from the beginning of the comment to the cursor position. You can move from one comment to another using the **<Up-arrow>** or **<Down-arrow>** keys.

Deleting Comments

When you move the cursor off of a comment that is empty, the comment is deleted. You can delete a comment by entering Comment Mode, moving to the end of the comment you wish to delete, and using `<Ctrl>U`. When you move to another comment or leave Comment Mode, the comment disappears. Likewise, the empty comment created when you enter Comment Mode is deleted if you don't type anything at the new comment position.

Comment Delimiters

Comments must start and end with one of the characters `<`, `>`, or `$`. A comment must start and end with the same delimiting character. If you try to move your cursor off of a comment that does not have one of these characters at the ends, or if the delimiters aren't identical, then SeqEd makes sure the delimiters are corrected.

Changing Sequence Position With Comment Mode

As you move to different comments, your position in the sequence in Screen Mode changes to the symbol with which that comment is associated. This allows you to move quickly to any symbol with which a comment is associated when you leave Comment Mode. By marking your place with a comment at the end of one session with SeqEd, you can easily restore your place at the next session.

Leaving Comment Mode

To exit Comment Mode, press `<Return>` or use `<Ctrl>D`.

Comments Are Associated With Sequence Symbols

Comments may be associated with any base. They stay with that base, even though the base's position may change, unless the base is deleted (see below). They can also be associated with either end of the sequence. For example, you may issue the command, `0 CO` to create a comment associated with the left end of the sequence. This comment must be delimited with `<` (SeqEd makes sure of this). Similarly, a comment can be created at the extreme right of the sequence and must be delimited with `>` or `$`.

Comments Can Be Used in Pairs to Bracket Sections of the Sequence.

Comments can document a whole fragment as well as an individual sequence symbol. For example, the `Include` command automatically puts an identifying comment at each end of the included fragment. The characters `<` and `>` were selected as comment delimiters because they imply direction; the comments bracketing the included fragment point at the fragment. A >-comment is associated with the first base of the fragment and a <-comment with the last. When the sequence is saved in a file, all >- and $-comments are written before the base they are associated with and all the <-comments after. This way the bracketing comments surround the entire fragment and point to it.

Between two bases in a sequence file there may be several comments. The <-comments are always associated with the base to the left, the >- and $-comments with the base to the right.

Deleting Comments

The only way to delete a comment is to go into Comment Mode and delete all the characters of the comment. When you move your cursor away from the empty comment, it goes away.

Deleting Bases Associated With Comments

If you delete a base with which a comment is associated, the comments do not go away. They just attach themselves to adjacent bases. To preserve the properties of fragment bracketing comments, the <-comments become associated with the left-hand base, the >- and $-comments with the right-hand base.

HEADING MODE

Heading Mode allows you to edit the documentation that appears above the sequence. When a new sequence is edited, SeqEd goes directly into Heading Mode to let you identify the new sequence.

Entering Heading Mode

SeqEd lets you enter Heading Mode by using the `HEAding` command.

Leaving Heading Mode

Use **`<Ctrl>D`** to return to Command Mode.

Moving the Cursor

You can move around using the arrow keys and make insertions and deletions as you wish. Although the editing window is only four lines high, it scrolls over the heading vertically to let you see and modify any part. **`<Ctrl>H`** and **`<Ctrl>E`** position the cursor at the beginning (head) and end of the current line, respectively.

Editing in Heading Mode

As with many text editors, typing inserts text at the cursor and the **`<Delete>`** key deletes characters to the left of the cursor. **`<Ctrl>U`** deletes everything from the current cursor position to the start of the line; **`<Return>`** creates a new line starting at the current position in the heading.

SYSTEM CRASH OR HANGUP

While you are editing a sequence, SeqEd records your session in a file called seqed.log. This file is automatically deleted when the editor exits normally. If you are accidentally disconnected or the system crashes, your work can be recovered by logging back in, moving to the directory where the crash occurred, and running SeqEd again. SeqEd finds seqed.log and restores the sequence to the state it was in just before you were cut off.

If you do not want SeqEd to restore the session, delete the file seqed.log.

RESTRICTIONS

SeqEd only works on terminals that can provide screen support. Your system manager may be able to help if your terminal is not behaving correctly.

ACKNOWLEDGEMENTS

SeqEd was originally designed by Paul Haeberli and implemented for VAX/VMS by Paul Haeberli and John Devereux. It was completely revised for GCG Version 4 by William Winsborough. The digitizer interface and the `RELoad` command were implemented for Version 5 by Philip Delaquess. We are very grateful for the collaboration of Dr. William Boorstein.

SEQUENCE TYPE

When it opens a new sequence file, SeqEd initially assumes it is nucleic acid. When you write the file, SeqEd examines the sequence to see if it contains any IUB-IUPAC amino acid symbols in the first 300 symbols. If so, it writes the new sequence as a peptide; if not, it writes it as a nucleic acid sequence.

When it opens a pre-existing GCG sequence file, SeqEd obtains the sequence type (nucleotide or protein) from the `Type:` field of the dividing line (the line that contains two successive periods). If the `Type:` field is absent, as in the case of sequence files created prior to Version 7 of the Wisconsin Package, SeqEd infers the type of the sequence from the composition of the sequence characters. When SeqEd writes the edited file, it writes the `Type:` field according to its current understanding of the sequence type.

It is possible for SeqEd to make a mistake. If the `Type:` field of an existing file is incorrect, SeqEd will accept the incorrect type; it doesn't check the composition in this case. For files without a `Type:` field, it is possible for SeqEd to infer the wrong sequence type. For example, a peptide sequence that contains only those amino acids that share IUB-IUPAC symbols with nucleotides will be incorrectly typed as nucleic acid (see Appendix III).

You can override SeqEd's assignment of sequence type in two ways. When you run SeqEd, you can add **`-PRO`**`tein` or **`-NUC`**`leotide` to the command line to tell SeqEd which type of sequence it will be editing. Once SeqEd is running, you can use the Command Mode commands **`PRO`**`tein` and **`NUC`**`leotide` to force the assignment of sequence type.

COMMAND-LINE SUMMARY

All parameters for this program may be put on the command line. Use the option **`-CHE`**`ck` to view the summary below and to add things to the command line before the program executes. In the summary below, the capitalized letters in the qualifier names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose qualifiers or parameter values that are optional. The "Using Program Parameters" section in the Chapter 3, **Using Programs** of the **User's Guide** describes how to use command lines effectively.

```
Minimal Syntax: % seqed [-INfile1=]sample.seq

Prompted Parameters: None

Local Data Files:

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

Optional Parameters:

-SINGlecommand       automatically returns to screen mode after commands
-PROtein             sets sequence type to protein, and sets find to
                        search for perfect symbol matches
-NUCleotide          sets sequence type to nucleotide, and sets find to
                        allow nucleotide ambiguity symbol matches
-PERFect             sets find to search for perfect symbol matches,
                        even if sequence type is nucleotide
-VECtors=EMBL:Pbr322 highlights sequences from pBR322
-SITes=GAATTC        highlight GAATTC patterns
-LANes=A,C,G,T       sets the default lane order for digitizer
-MINOverlap=10       minimum overlap length for Reload command
-PCTOverlap=95       stringency for the Reload command
-TOLerance=0.4       tolerance for digitizing ambiguity (0 to 1)
                        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 the Chapter 4, **Using Local 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 editors SeqEd, GelEnter, LineUp, and GelAssemble 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, and GelAssemble 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 sequence characters 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.

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`**.

OPTIONAL PARAMETERS

The parameters and switches listed below can be set from the command line. Optional parameters available to all programs are described in the "Using Program Parameters" section of Chapter 3, **Using Programs** of the **User's Guide**.

`-SING``lecommand`

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

`-PERF``ect`

makes pattern searches use perfect symbol matches. Normally if you type **`/GARC`** in Screen Mode, the patterns GAAC or GAGC could be found. If you have **`-PERF`**`ect` on the command line, **`/GARC`** would only find the pattern GARC. This also makes SeqEd treat sequences as linear and not find patterns that start at the end and continue into the beginning of the sequence.

`-PRO``tein`

sets the sequence type to be protein, and makes pattern searches use perfect symbol matches. SeqEd treats protein sequences as linear and will not find patterns that start at the end and continue into the beginning of the sequence. Furthermore, **`-PRO`**`tein` causes SeqEd to ignore any set.keys file in your local directory.

-NUCleotide

sets the sequence type to be nucleotide, and makes pattern searches use nucleotide ambiguity symbol matches (unless you force the program to use perfect symbol matches by including **-PERF**ect on the command line or by entering the **PERF**ect command in Command Mode.) SeqEd treats nucleotide sequences as circular and will find patterns that start at the end and continue into the beginning of the sequence. Furthermore, **-NUC**leotide causes SeqEd to use a set.keys file in your local directory.

-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 **DIG**itizer command in Command Mode.

-VECtors=**EMBL:pbr322,EMBL:m13mp18**

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

-SITes=**GAATTC,GAnTC**

tells SeqEd to highlight enzyme recognition sites that interest you.

-MINOverlap=**10**

sets the minimum overlap length regarded as meaningful by the **REL**oad command. SeqEd ignores matches shorter than this, even if they are perfect. However, you are always free to end a reload with the **ACC**ept command.

-PCTOverlap=**95**

sets the minimum percentage of matching bases regarded as meaningful by the **REL**oad 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 **ACC**ept SeqEd's opinion.

-TOLerance=**0.4**

sets the tolerance for digitizing. When digitizing, the program must determine which base lane the sonic pen has touched. Since the gel lane may bend, the program must have some tolerance for deviation. The tolerance value determines how great this deviation can be before you must redefine your lanes. A tolerance of 0 is the least tolerant setting and the slightest deviation would require you to redefine your lanes. A tolerance of 1.0 is the most tolerant setting such that any deviation is accepted. Based on our limited experience, you should not use a tolerance value less than 0.25 or greater than 0.6. The default value (0.4) was chosen because it has seldom made an incorrect assignment and does not require you to redefine the lanes too frequently. The algorithm employed is that of Staden (Nucl. Acids Res., **14**; 217 (1986)).

MAP

FUNCTION

Map displays both strands of a DNA sequence with restriction sites shown above the sequence and possible protein translations shown below.

DESCRIPTION

Map displays a sequence that is being assembled or analyzed intensively. Map asks you to enter the names of those enzymes whose restriction sites should be marked. If you do not answer this question, Map generates a restriction map with a representative isoschizomer from all of the commercially available enzymes. You can choose to have your sequence translated in any of the six possible translation frames. You can also choose to have only the open reading frames translated.

After running Map, you may create a new sequence file with the peptide sequence from any frame of DNA translation by using the ExptractPeptide program with the Map output file.

EXAMPLE

Here is a session using Map to display a portion of gamma.seq, along with a restriction map and six-frame protein translation:

```
% map

 (Linear) MAP of what sequence ?  gamma.seq

                  Begin (* 1 *) ?  2101
                End (* 11375 *) ?  2600

 Select the enzymes:  Type nothing or "*" to get all enzymes. Type "?"
 for help on which enzymes are available and how to select them.

                                         Enzyme(* * *):

 What protein translations do you want:

      a) frame 1   b) frame 2   c) frame 3
      d) frame 4   e) frame 5   f) frame 6

      t)hree forward frames   s)ix frames   o)pen frames only

      n)o protein translation   q)uit

 Please select (capitalize for 3-letter) (* t *):  s

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

%
```

OUTPUT

Here is part of the output file:

```
 (Linear) MAP of: gamma.seq  check: 6474  from: 2101  to: 2600

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

 With 199 enzymes: *

                             July  1, 1994 14:26  ..

                                                M
                                               Ma   D
                                               ne   d
                                               lI   e
                                               II   I
         AGGAAGCACCCTTCAGCAGTTCCACACACTCGCTTCTGGAACGTCTGAGGTTATCAATAA
    2101 ---------+---------+---------+---------+---------+---------+ 2160
         TCCTTCGTGGGAAGTCGTCAAGGTGTGTGAGCGAAGACCTTGCAGACTCCAATAGTTATT

a        R  K  H  P  S  A  V  P  H  T  R  F  W  N  V  *  G  Y  Q  *   -
b         G  S  T  L  Q  Q  F  H  T  L  A  S  G  T  S  E  V  I  N  K -
c          E  A  P  F  S  S  S  T  H  S  L  L  E  R  L  R  L  S  I  S -
    2101 ---------+---------+---------+---------+---------+---------+ 2160
d           S  A  G  K  L  L  E  V  C  E  S  R  S  R  R  L  N  D  I   -
e            F  C  G  E  A  T  G  C  V  R  K  Q  F  T  Q  P  *  *  Y -
f          P  L  V  R  *  C  N  W  V  S  A  E  P  V  D  S  T  I  L  L -

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

 Enzymes that do cut:

     AccI     AluI    AvaII     BanI     BbvI     BccI   Bce83I     BfaI
     BglI     BpmI    BsaHI    BsaJI    BseRI     BsgI     BslI    BsoFI
 Bsp1286I    BspGI   BstEII   CjePI    CviJI    CviRI     DdeI     DpnI
     DsaI EcoO109I   EcoRII     FokI     HgaI     HphI    MaeII   MaeIII
    MboII     MnlI     MslI  MspA1I     MwoI     NcoI   NlaIII    NlaIV
     NspI   Psp5II    PvuII    RleAI   Sau3AI   Sau96I    ScrFI    SfaNI
     StyI     TseI   Tsp45I     XcmI

 Enzymes that do not cut:

    AatII   AceIII     AciI    AflII   AflIII     AgeI     AlwI   Alw21I
   Alw44I    AlwNI     ApaI    ApaBI     ApoI     AscI     AvaI    AvrII
     BaeI    BamHI    BanII     BbsI    BcefI     BcgI     BcgI     BclI
    BglII   Bpu10I Bpu1102I     BsaI    BsaAI    BsaBI    BsaWI     BsbI

 //////////////////////////////////////////////////////////////////////
```

RELATED PROGRAMS

MapSort, PlasmidMap, and MapPlot display restriction maps in other formats. ExtractPeptide extracts the peptide sequence from any translation frame in the Map output file and puts it into a new sequence file.

FindPatterns searches for short patterns like enzyme recognition sites in one or more sequences. PeptideMap creates a peptide map of an amino acid sequence.

MapSelect selects restriction enzymes by name or by their ability to cut a given sequence, and writes them to a new enzyme file for use in other programs.

RMap computes a restiction map from fragment length data arising from single and double digest. It shows all possible fragment orders within a given error level. This program is VERY experimental. Please contact us if you have any questions, problems etc.

RESTRICTIONS

Map does not treat your sequence as circular unless you use the command line option `-CIRcular`. Map reads the `Type:` field on the dividing line in the sequence file to determine whether your sequence is a nucleic acid or protein. The enzymes you name must be in the enzyme data file or you get an error message. You can have your system manager change the public enzyme data file to contain the enzymes most useful to your group, or you can maintain a private copy for your own use. (See the LOCAL DATA FILES topic below for more information.)

SUBSET, OVERLAP, AND PERFECT SEARCHES

This program normally requires that a sequence pattern be a *subset* of the enzyme recognition site. If the recognition pattern in the enzyme data file were GCRGC, then the pattern GCAGC in your sequence would be found, since A is within the set of bases defined by R (see Appendix III). If the pattern in the enzyme data file were GCAGC, then a GCRGC in your sequence would not be recognized. If your sequence is very ambiguous, as it might be if it were a backtranslated sequence, then it may be better to use the `-ALL` switch to do an *overlap* search. The overlap search would consider an R in your sequence to match an A in the recognition site.

The command-line option `-PERFect` causes this program to look for a perfect symbol match between your sequence and the recognition pattern -- GCRGC in the recognition pattern would only match a GCRGC in the sequence.

All searches are case insensitive (upper- or lowercase) for the letters in either the sequence or the enzyme recognition site.

DISPLAY CONVENTIONS

Collisions

Map identifies patterns by the positions where they occur in sequences. When a pattern cannot be shown at a particular position, it is shown at the next available position in the sequence. A '/' below the enzyme's name indicates that the name of the enzyme has been displaced to the right from the position where it should have been. When the number of finds is very great, the resolution of this kind of display is inadequate. If the display seems too full, you should restrict the number of enzymes chosen. Or, even better, use the `-HORizontal` option to obtain a more readable output.

Potential Sites

When you search for potential restriction sites with either the `-MISmatch` or `-SILent` options, Map differentiates the real sites from the potential sites by capitalizing the enzyme's name at the real sites.

SELECTING ENZYMES

The program presents you with an enzyme selection prompt that lets you enter enzymes individually or collectively. To get help with selecting enzymes, type a **?** at the enzyme prompt. Here is what you see:

```
Select enzymes:

Type "*" to select all enzymes.
Type "**" to select all enzymes including isoschizomers.
Type individual names like "AluI" to select specific enzymes.
Type "?" to see this message and all available enzymes.
Type "??" to see the available enzymes AND their recognition sites.
Type "?A*" to see what enzymes start with "A."
Type "A*" to select all enzymes starting with "A."
Type parts of names like "Al*" to select all enzymes starting with "AL."
Type "~A*" to unselect all selected enzymes starting with "A."
Type "/*" to see what enzymes you have selected so far.
Type "#" to select no enzymes at all.

Press <Return> after each selection.
Press <Return> and nothing else to end your selections.
Spaces are allowed and letter case is ignored.
```

We maintain our enzyme files with a semicolon (;) character in front of all but one member of a family of isoschizomers. (Isoschizomers are restriction endonucleases with the same recognition site.) The isoschizomers beginning with a semicolon are normally not displayed by our mapping programs unless you specifically select them by name or type "******" instead of "*****" at the enzyme prompt.

There is more information on enzyme files in the Chapter 4, **Using Local Data Files** of the **User's Guide**.

A command-line expression like **-ENZ**ymes**=AluI,EcoRII** would choose AluI and EcoRII and suppress interactive enzyme selection.

CHOOSING THE TRANSLATION FRAMES

The translation menu allows several responses. You can name the frames of interest individually with a response like **abcf**. You can use **t** or **s** to mean the three forward or all six possible translation frames. You can make all of the characters in your response uppercase to get three-letter instead of one-letter amino acid symbols in the translation. You can add **o** to your response to get translation only between potential start codons and stop codons (**o** by itself gives open reading frame translation of all six translation frames).

You can use an expression like **-MEN**u**=abcf** to choose translation frames a, b, c, and f from the command line.

OPEN READING FRAMES

You can select translation for open reading frames only. All of the frames are treated as open at the 5' end of each strand; these pseudo-open reading frames run to the first stop codon in that frame (see the **Translation Tables** section in the Chapter 4 **Using Local Data Files** of the **User's Guide**). Thereafter, reading is turned on at each potential start codon and runs to the next stop codon. You can suppress the display of short open reading frames with an expression like **-OPE**n**=20** on the command line which would restrict the display to frames coding for at least 20 residues.

Open reading frames are determined from the beginning and ending of the sequence in the file--not from just the range you have chosen. The potential start codons and stop codons are defined in the data file translate.txt.

POTENTIAL RESTRICTION SITES

To assist scientists doing site-directed mutagenesis, this program searches for places in your sequence where a restriction enzyme recognition site occurs with one or more mismatches. Use the command-line option `-MISmatch=1` to identify positions where recognition could occur with one or fewer mismatches.

Use the command-line option `-SILent` to find the places in your sequence where a restriction site could be introduced without changing the translation. Read more about this at `-SILent` under the OPTIONAL PARAMETERS topic below.

SEARCH FOR ANY SEQUENCE PATTERN

By changing the enzyme data file (see the LOCAL DATA FILES topic below), you can make this program search for any pattern. See the Chapter 4, Using Local Data Files in the **User's Guide** for notes on enzyme data files.

DEFINING PATTERNS

FindPatterns, Map, MapSort, MapPlot, and Motifs all let you search with ambiguous expressions that match many different sequences. The expressions can include any legal GCG sequence character (see Appendix III). The expressions can also include several non-sequence characters, which are used to specify OR matching, NOT matching, begin and end constraints, and repeat counts. For instance, the expression TAATA(N){20,30}ATG means TAATA, followed by 20 to 30 of any base, followed by ATG. Following is an explanation of the syntax for pattern specification.

Implied Sets and Repeat Counts

Parentheses () enclose one or more symbols that can be repeated some number of times. Braces {} enclose numbers that tell how many times the symbols within the preceding parentheses must be found.

Sometimes, you can leave out part of an expression. If braces appear without preceding parentheses, the numbers in the braces define the number of repeats for the immediately preceding symbol. One or both of the numbers within the braces may be missing. For instance, the pattern GATG{2,}A means GAT, followed by G repeated from 2 to 350,000 times, followed by A; the pattern GATG{}A means GAT, followed by G repeated from 0 to 350,000 times, followed by A; the pattern GAT(TG){,2}A means GAT, followed by TG repeated from 0 to 2 times, followed by A. (If the pattern in the parentheses is an OR expression (see below), it cannot be repeated more than 2,000 times.)

OR Matching

If you are searching nucleic acids, the ambiguity symbols defined in Appendix III let you define any combination of G, A, T, or C. If you are searching proteins, you can specify any of several symbol choices by enclosing the different choices in parentheses and separating the choices with commas. For instance, RGF(Q,A)S means RGF followed by either Q or A followed by S. The length of choices need not be the same, and there can be up to 31 different choices within each set of parentheses. The pattern GAT(TG,T,G){1,4}A means GAT followed by any combination of TG, T, or G from 1 to 4 times followed by A. The sequence GATTGGA matches this pattern. There can be several parentheses in a pattern, but parentheses cannot be nested.

NOT Matching

The pattern GC~CAT means GC, followed by any symbol except C, followed by AT. The pattern GC~(A,T)CC means GC, followed by any symbol except A or T, followed by CC.

Begin and End Constraints

The pattern <GACCAT can only be found if it occurs at the beginning of the sequence range being searched. Likewise, the pattern GACCAT> would only be found if it occurs at the end of the sequence range.

PRIMER WALKING

Slightom et al. (Biotechniques, in press (1994)) have shown that some of the 262,144 potential 9-mers prime selectively enough to reduce the custom-primer costs that are usually associated with primer-walking sequencing strategies.

The 9-mers identified by Slightom et al. are available as a kit called OligoArray 1™ from Genosys Biotechnologies, Inc., 1442 Lake Front Circle, Suite 185, The Woodlands, Texas, USA 77380, telephone (800) 234-5362. This program can find where the 9-mers in this kit appear in your sequence. To search for these 9-mers in your sequence, copy the file genosys.dat into your working directory with Fetch and then use the command-line option **-DAT**a=**genosys.dat**

SEQUENCE TYPE

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

COMMAND-LINE SUMMARY

All parameters for this program may be put on the command line. Use the option **-CHE**ck to view the summary below and to add things to the command line before the program executes. In the summary below, the capitalized letters in the qualifier names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose qualifiers or parameter values that are optional. The "Using Program Parameters" section in the Chapter 3, **Using Programs** of the **User's Guide** describes how to use command lines effectively.

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

Prompted Parameters:

-BEGin=2101 -END=2600       range of sequence in which to look for sites
-ENZymes=*[,...]            chooses the enzymes used in the search
-MENu=t                     translation frames s=six, t=three, o=open
[-OUTfile=]gamma.map        output file name

Local Data Files:

-DATa=enzyme.dat          restriction enzyme names and recognition sites
-DATa=proenzyme.dat       peptidases and peptide cleavage reagents
-TRANSlate=translate.txt  the genetic code
```

```
Optional Parameters:

-WIDth=100      sets display width to something other than 60 bp-line
-PAGe[=64]      adds form-feeds to keep clusters on a single page
-OPEn[=20]      translates only in open reading frames [minimum ORF length]
-SIXbase        only finds enzymes with 6 or more bases in recognition site
-ONCe           shows only enzymes that cut once
-MINCuts=2      shows only enzymes that cut at least 2 times
-MAXCuts=2      shows only enzymes that cut no more than 2 times
-EXCLude=n1,n2  doesn't show enzymes that cut between bases n1 and n2
-ALL            finds "overlapping-set" matches
-PERFect        finds only perfect symbol matches between site and sequence
-CIRcular       treats the sequence as circular
-LINear         treats the sequence as linear (default)
-APPend         appends enzyme and genetic code data files to output
-THReeletter    uses three-letter amino acid codes for the translation
-SILent         finds translationally silent potential restriction sites
-MISmatch=1     finds restriction sites with one or fewer mismatches
-HORizontal     displays enzyme names in horizontal direction
-POSition       displays the positions of restriction sites
-RANGE=n1,n2[,n3,n4,...]
                translates the sequence just between bases n1 and n2,
                n3 and n4 etc., respectively
-NOSEQline      suppresses the sequence display
-NOSCALeline    suppresses the scale line
-NOCOMPline     suppresses the complement sequence display
```

ACKNOWLEDGEMENT

The output format of Map was designed by John Schroeder and Frederick Blattner (NAR **10**; 69-84 (1982), Figure 1). Map was written for the first release of the Wisconsin Package™ by Paul Haeberli and John Devereux. It is the most frequently used tool in the GCG Package and has been revised for every release since. The options **-HOR**izontal, **-POS**ition, and **-RANGE** have been implemented by Karl-Heinz Glatting, DKFZ, Heidelberg.

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either; 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like **-DAT**a1=**myfile.dat**. For more information see the Chapter 4, **Using Local Data Files** in the **User's Guide**.

This program reads the public or local version of enzyme.dat to get the enzyme names, recognition sites, cut positions, and overhangs. You can use mapping programs to search for any sequence pattern by adding the pattern to the enzyme data file. If you use the command line option **-APP**end, this program appends the enzyme data file to the output file. (See the "Restriction Enzymes" in the Chapter 4, **Using Local Data Files** of the **User's Guide** for more information about enzyme data files.)

Note Use the program MapSelect to create your own enzyme tables.

If Map finds `Type: P` on the dividing line in the sequence file, it reads proteolytic cleavage data in the local data file proenzyme.dat.

The translation of codons to amino acids, the identification of potential start codons and stop codons, and the mappings of one-letter to three-letter amino acid codes are all defined in a translation table in the file translate.txt. If the standard genetic code does not apply to your sequence, you can provide a modified version of this file in your working directory or name an alternative file on the command line with an expression like `-TRANSlate=mycode.txt`. Translation tables are discussed in more detail in the **Data Files** manual. If you use the command line option `-APPend,` this program appends the enzyme data file to the output file.

OPTIONAL PARAMETERS

The parameters and switches listed below can be set from the command line. Optional parameters available to all programs are described in the "Using Program Parameters" section of Chapter 3, **Using Programs** of the **User's Guide**.

`-OPEn=20`

restricts the display of translations to open reading frames (ORFs). If you supply a number like 20 with this qualifier, the ORF would only be displayed if it coded for at least 20 residues.

`-CIRcular`

tells Map to treat your sequence as circular. If a possible recognition site starts at the end and continues into the beginning of the sequence, the site is marked at the point where a circular molecule would be cut. For instance if your sequence ends in GAA and starts with TTC, Map shows an EcoRI cut two bases before the end of the sequence. The sequence is only circularized at the ends found in the file, so if you want a subrange to be treated as circular you have to create a file in which the subrange is the entire sequence (see the Assemble program).

`-LINear`

is the opposite of `-CIRcular`. If you have defined a command that runs Map with `-CIRcular` as the default, use the `-LINear` switch to make Map treat your sequence as linear.

`-PAGe=64`

When you print the output from this program, it may cross from one page to another in a frustrating way -- especially when you print on individual sheets. This option adds 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 the `-PAGe` qualifier.

`-WIDth=60`

allows you to choose the number of bases shown on each line of output. The standard is 60, which can be shown on a terminal screen nicely, but 100 sequence symbols per line is very convenient for estimating the size of fragments between cuts.

`-THReeletter`

sets the translation to show three-letter amino acid codes instead of the one-letter codes. Normally the case of the translation menu is sufficient to make the three-letter/one-letter distinction. However, when you run Map from the command line, you must add `-THReeletter` to get three-letter amino acid codes.

`-MISmatch=1`

causes the program to recognize sites that are like the recognition site but with one or fewer mismatches. If you allow too many mismatches, you may get ridiculous results. The output from most mapping programs distinguishes between sites with no mismatches and sites with mismatches.

`-SILent`

shows the places where restriction sites can be introduced (by site-directed mutagenesis) without changing the peptide translation of the sequence. The `-SILent` switch assumes that the range you have chosen defines a coding region and reading frame precisely. Sites may be found that have any number of bases changed as long as the changes do not alter the translation. The silent frame is implied by the beginning coordinate you specify. The output from most mapping programs distinguishes between real sites and sites with one or more mismatches. The data file translate.txt defines the genetic code.

`-PERFect`

sets the program to look for a perfect alphabetic match between the site and the sequence. Ambiguity codes are normally translated so that the site RXY would find sequences like ACT or GAC. With this switch the ambiguity codes are not translated so the site RXY would only match the sequence RXY. This switch is *not* the same as `-MISmatch=0`!

`-ALL`

makes an overlap-set map instead of the usual subset map. If your sequence is very ambiguous (for instance, as a back-translated sequence would be) and you want to see where restriction sites could be, then an overlap-set map is for you. Overlap-set and subset pattern recognition is discussed in more detail in the **Program Manual** entry for Window.

`-APPend`

appends the input enzyme data file to your output file.

`-HORizontal`

displays enzyme names in horizontal direction.

`-POSition`

displays the position of restriction sites.

`-RANGE=n1,n2[n3,n4,...]`

translates the sequence just between bases n1 and n2, n3 and n4 etc., respectively.

The options `-SIXbase`, `-ONCe`, `-MINCuts`, `-MAXCuts`, and `-EXCLude` all suppress the display of undesired enzymes. The list of excluded enzymes in the program output includes both enzymes that cut within excluded ranges and enzymes that do not cut the right number of times.

-SIX`base`

searches only for enzymes with six or more bases in the recognition site. You can display the cuts from any enzyme in the enzyme data file that you take the trouble to name individually, but when you use ***** (meaning all), the program uses all of the other enzymes whose recognition sites have six or more non-N, non-X bases.

-ONC`e`

excludes, from the set you have chosen, those enzymes that cut your sequence more than once.

-MINC`uts`**=2**

excludes enzymes that do not cut at least two times.

-MAXC`uts`**=2**

excludes enzymes that cut more than two times.

-EXCL`ude`**=n1,n2**`[n3,n4,...]`

excludes enzymes that cut anywhere within one or more ranges of the sequence. If an enzyme is found within an excluded range, then the enzyme is not displayed. The list of excluded enzymes includes enzymes that cut within excluded ranges. The ranges are defined with sets of two numbers. The numbers are separated by commas. Spaces between numbers are not allowed. The numbers must be integers that fall within the sequence beginning and ending points you have chosen. The range may be circular if circular mapping is being done. Exclusion is not done if there are any non-numeric characters in the numbers or numbers out of range or if there is not an even number of integers next to the qualifier.

-TRANS`late`**=filename.txt**

Usually, translation is based on the translation table in a default or local data file called translate.txt. This option allows you to use a translation table in a different file. (See the **Data Files** manual for information about translation tables.)

The center of the Map display is the sequence, a scale, and the sequence's complement. These three switches let you suppress any of these lines.

-NOSEQ`line`

suppresses the sequence display.

-NOSCAL`eline`

suppresses the scale line between the sequence and its complement.

-NOCOMP`line`

suppresses complement sequence display.

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 and creates the alignment with the largest number of matched bases and the fewest gaps. You provide a *gap creation penalty* and a *gap extension penalty* in units of matched bases. In other words, Gap must make a profit of *gap creation penalty* number of matches for each gap it inserts. If you choose a gap extension penalty greater than zero, Gap must, in addition, make a profit for each gap inserted of the length of the gap times the gap extension penalty. Typical values to use as a point of departure for the gap creation and gap extension penalties are 5.0 and 0.3, respectively, for nucleic acid sequence comparisons, and 3.0 and 0.1, respectively, for protein sequence comparisons. 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 (see note below).

EXAMPLE

Two very long operons of 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

  (Limited) 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 (* 5.00 *) ?

  What is the gap extension penalty (* 0.30 *) ?

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

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

```
           Gaps:    13
        Quality: 2442.6
  Quality Ratio: 0.891
   % Similarity: 94.897
         Length:  2982

%
```

OUTPUT

Here is the output from this session:

```
 GAP of: hpr.seq  check: 8102  from: 1  to: 2966

Haptoglobin related sequence
HindIII fragment sequenced 12/27/83
  (partially from hpf sequence)

 to: hpf.seq  check: 2624  from: 1  to: 2740

Haptoglobin alpha2
HindIII fragment , region equivalent to hp1f

Symbol comparison table:  Gencoredisk:gcgcore/data/rundata/nwsgapdna.cmp
CompCheck: 6876

         Gap Weight:  5.000      Average Match:  1.000
      Length Weight:  0.300   Average Mismatch:  0.000
            Quality: 2442.6             Length:   2982
              Ratio:  0.891               Gaps:     13
 Percent Similarity: 94.897   Percent Identity: 94.897

 hpr.seq x hpf.seq         February 21, 1991  13:37  ..

                  .         .         .         .         .
       1 AAGCTTGGTATGCTCAGAAGCAGCTAAAGCGTGTATGTGGGGCGGAGGGT 50
         ||||||||||||||||||||| ||||||| ||||||| |  |    | ||
       1 AAGCTTGGTATGCTCAGAAGCTGCTAAAGTGTGTATGGGCAG....GTGT 46

    ////////////////////////////////////////////////////////////
                  .         .         .         .         .
    1749 TTCCTCTTTCTTCAGAGATGATGAATTATTGTAGCTCCTAGCCCTTTCTT 1798
         ||| |||||||| |||||  |||||||||||||
    1678 TTCATCTTTCTTTAGAGAGAATGAATTATTGTA................. 1710
                                  .
                                  .
                                  .
                  .         .         .         .         .
    1949 TGGCCCCTAGCCCTTTCAATGAATTTCAGGGAATTGTGAAAATTCCTTTG 1998
           |||||||||||||||||||||||||||||||||||| ||||||||||
    1711 ..GCCCCTAGCCCTTTCAATGAATTTCAGGGAATTGTGGAAATTCCTTTA 1758

    ////////////////////////////////////////////////////////////
                  .         .         .
    2935 GAGGACACCTGGTACGCGGCTGGGATCTTAAG 2966
         |||||||||||||| ||| |||||||||||||
    2709 GAGGACACCTGGTATGCGACTGGGATCTTAAG 2740
```

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.

Similarity finds k best non-intersecting alignments between two sequences or within one sequence.

MultAlign does a simultaneous alignment for two or more DNA or protein sequences. It introduces a certain number of gaps into either pairwise aligned sequences or groups of sequences to find a minimal global distance. The user can influence the result by defining the order in which the sequences will be aligned. The program is based on a generalization of the algorithm of Waterman, Smith and Beyer by Krueger and Osterburg.
If you want to use the output of this program as input to other programs (like ClusTree, ToPhylip, LineUp, PrettyPlot or BoxAlign) you have to create a 'Multiple Sequence File' (MSF-File) by using the parameter -msf on the command line.

ClustAl calculates a multiple alignment of nucleic acid or protein sequences according to the method of Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994). This is part of ClustalW.
If you want to use the output of this program as input to other programs (like ClusTree, ToPhylip, LineUp, PrettyPlot or BoxAlign) you have to create a 'Multiple Sequence File' (MSF-File) by using the parameter -msf on the command line.

MAlign calculates a multiple global alignment of nucleic acid or protein sequences. It is especially suited for the alignment of sequences of various lengths.

RESTRICTIONS

Input sequences may not be more than 34,001symbols long. This program cannot evaluate a surface of comparison larger than 20 million elements. A 800 x 25,000 comparison is possible, as well as a 2,300 x 2,300 comparison. See the ALIGNING LONG SEQUENCES topic for help in aligning long sequences that would normally exceed the maximum surface of comparison. You can also ask your system manager to increase the maximum surface of comparison if your system has enough virtual memory.

ALIGNING LONG SEQUENCES

This program can align very long sequences if you know roughly where the alignment of interest begins. Run the program with the command line option `-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. If you omit `-LIMit` on the command line, the program automatically sets gap shift limits if they are needed to allow the alignment of long sequences to proceed. In this case, the program limits the total length of gaps that can be inserted into each sequence and calculates the best alignment within this incomplete, or *limited*, surface of comparison. The program then performs a calculation to determine whether the alignment could possibly be improved if there were no restriction on the total length of gaps in each sequence. If the program cannot rule out this possibility, it 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, a limited alignment often may be optimal even if this message is displayed. In any event, the program continues to completion.

EVALUATING ALIGNMENT SIGNIFICANCE

This program can help you evaluate the significance of the alignment, using a simple statistical method, with the `-RANdomizations` command line option. 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 along with the `-RANdomizations` command line qualifier; the default is 10.

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 the "Scoring Matrices" section in the Chapter 4, **Using Local Data Files** of the **User's Guide**.

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

In general, you should try to normalize your matrix so that good matches are worth about 1 and bad matches about 0 to -1 so that Gap treats the gap creation and gap extension penalties in a manner that is consistent with your experience using a matrix with 1s and 0s.

CompTable helps you create scoring matrices based on a simplification scheme for amino acid differences.

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 G. 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. The command line option `-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 the command line option `-ENDWeight`.

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. A similarity is scored when the scoring matrix value for a pair of symbols is greater than or equal to 0.50, 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 reset it from the command line with the second optional parameter of **-PAI**r. For instance, the expression **-PAI**r**=1.0,0.5** would set the similarity threshold to 0.5.

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 with matches scored as 1.5 and mismatches scored according to the evolutionary distance between the amino acids as measured by Dayhoff and normalized by Gribskov (Gribskov and Burgess Nucl. Acids Res. **14(16)**; 6745-6763 (1986)).

SEQUENCE TYPE

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.

COMMAND-LINE SUMMARY

All parameters for this program may be put on the command line. Use the option **-CHE**ck to view the summary below and to add things to the command line before the program executes. In the summary below, the capitalized letters in the qualifier names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose qualifiers or parameter values that are optional. The "Using Program Parameters" section in the Chapter 3, **Using Programs** of the **User's Guide** describes how to use command lines effectively.

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

Prompted Parameters:

-BEGin1=1  -BEGin2=1    beginning of each sequence
-END1=2966 -END2=2740   end of each sequence
-NOREV1    -NOREV2      strand of each sequence
-GAPweight=5.0          gap creation penalty    (3.0 is protein default)
-LENgthweight=0.3       gap extension penalty   (0.1 is protein default)
[-OUTfile1=]hpr.pair    output file for alignment

Local Data Files: -DATa=nwsgapdna.cmp scoring matrix for nucleic acids
                  -DATa=nwsgappep.cmp scoring matrix for peptides

Optional Parameters:

-OUTfile2=hpr.gap       new file for sequence 1 with gaps added
```

```
-OUTfile3=hpf.gap       new file for sequence 2 with gaps added
-LIMit1=1 -LIMit2=240   limit the surface of comparison
-RANdomizations[=10]    determine average score from 10 randomized
                           alignments
-PAIr=1.0,0.5,0.1       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 Delaquess for Version 4.0. Default gap penalties for protein alignments were modified according to the suggestions of Rechid, Vingron and Argos (CABIOS **5**; 107-113 (1989)).

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 the Chapter 4, **Using Local 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 1.0 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. The file nwsgappep.cmp has 1.5 for perfect symbol matches and values less than 1.5 (depending upon the evolutionary distance) for non-matches. You can use Fetch to copy these files and and modify them to suit your own needs.

OPTIONAL PARAMETERS

The parameters and switches listed below can be set from the command line. Optional parameters available to all programs are described in the "Using Program Parameters" section of Chapter 3, **Using Programs** of the **User's Guide**.

`-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. The surface of comparison is still limited to 3.5 million. The size of a comparison can be predicted by multiplying the average length of the two sequences by the sum of the two shift limits.

If you add **`-LIMit`** to the command line without any qualifier 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.

`-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 any qualifying filename, 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.

`-PAIr=1.0,0.5,0.1`

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 0.50, and a period is put between symbols whose comparison value is greater than or equal to 0.10. You can change these *match display thresholds* from the command line. The three parameters for `-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 parameter. (See the **Data Files** manual for more information about scoring matrices.)

`-PAGe=64`

When you print the output from this program, it may cross from one page to another in a frustrating way -- especially when you print on individual sheets. This option adds 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 the `-PAGe` qualifier.

`-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.

-NOBIGG`aps`

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.

-ENDW`eight`

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

-LOW`road` and **-HIG**`hroad`

The insertion of gaps is, in many cases, arbitrary, 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 **-LOW**`road` or the **-HIG**`hroad` options. Here are examples for the alignment of GACCAT with GACAT with different parameters.

```
For:       Match = 1.0        MisMatch = -0.9
      Gap weight = 1.0   Length Weight =  0.0

LowRoad:   1 GACCAT 6
             || |||      Quality = 4.0
           1 GA.CAT 5

HighRoad:  1 GACCAT 6
HighRoad:  1 GACCAT 6
             ||| ||      Quality = 4.0
           1 GAC.AT 5

For:       Match = 1.0        MisMatch =  0.0
      Gap weight = 3.0   Length Weight =  0.0

HighRoad:  1 GACCAT 6
             |||           Quality = 3.0
           1 GACAT. 5

LowRoad:   1 GACCAT 6
                |||        Quality = 3.0
           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.

-SUM`mary`

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

Use this qualifier also to include a summary of the program's work in the log file for a program run in batch.

BLASTN

FUNCTION

BlastN compares a nucleotide query sequence against a nucleotide sequence database. BlastN is more than an order of magnitude faster as FastA, but tends to be less sensitive.

DESCRIPTION

BlastN is based on the BLAST (Basic Local Alignment Search Tool) - Algorithm (see Stephen F. Altschul et al., J. Mol. Biol. **215**; 403-410 (1990)). The unit of BLAST algorithm output is the High-Scoring Segment Pair (HSP), where a segment is an arbitrarily long run of contiguous residues. A HSP is a pair of segments, one from the query sequence and one from a database sequence, where the score of their ungapped alignment meets or exceeds a parametrized, positive-valued cutoff. A set of zero or more HSPs is thus defined by two sequences, a scoring scheme, and a cutoff score. With nucleotide sequences, the scoring scheme is very simple: identical nucleotides are counted as +5 (**`-MATCH`**), different ones as -4 (**`-MISM`**`atch`).

A Maximal-Scoring Segment Pair (MSP) is defined by two sequences and a scoring scheme and is the highest-scoring of all segment pairs on all diagonals. Depending on the parameters of a BLAST sequence comparison, there may be a non-zero probability of not finding one or more HSPs of which the MSP is a member.

EXAMPLE

Here is a session using BlastN to find sequences in the GeAll nucleotide sequence data library with similarities to a potato spindle tuber viroid RNA sequence:

```
% blastn

 BLASTN with what query sequence ?  embl:ptvseqc

                  Begin (* 1 *) ?
                End (*   361 *) ?

 Search for query in what database (* geall *) ?

 What should I call the output file (* ptvseqc.blastn *) ?

 Run BlastN in Batch-Mode ? (* y *)

 Run BlastN in what queue?

 a) short      queue
 b) long       queue
 c) verylong   queue
 d) blastshort queue
 e) blastlong  queue

 What Queue ? (* D *)

 ** blastn will run as a batch or at job.

 ** blastn was submitted using the command:
    "  qsub  -q blastshort   "
```

```
     Request 8032.cvx12 submitted to queue: blastshort.
```

OUTPUT

Here is some of the output file:

```
BLASTN of: PTVSEQC   from: 1 to: 361  July 30, 1996 14:06
compared to database: geall   both strands     ..

     Observed Numbers of Database Sequences Satisfying
    Various EXPECTation Thresholds (-EXP parameter values)

        Histogram units:      = 6 Sequences     : less than 6 sequences

 EXPECTation Threshold
 (-EXP parameter)
    |
    V   Observed Counts-->
  10000 1025 353 |==========================================================
   6310  672 209 |===================================
   3980  463 148 |=========================
   2510  315  96 |=================
   1580  219  50 |========
   1000  169  40 |======
    631  129  26 |====
    398  103  18 |===
    251   85   8 |=
    158   77   1 |:
    100   76   3 |:
   63.1   73   6 |=
   39.8   67   2 |:
   25.1   65   2 |:
   15.8   63   0 |
 >>>>>>>>>>>>>>>>>>>>>  Expect = 10.0, Observed = 63  <<<<<<<<<<<<<<<<<
   10.0   63   2 |:
   6.31   61   0 |
   3.98   61   1 |:
   2.51   60   0 |
   1.58   60   0 |
   1.00   60   1 |:
   0.63   59   0 |
   0.40   59   0 |
   0.25   59   1 |:


                                                                 Smallest
                                                                   Sum
                                                          High  Probability
Sequences producing High-scoring Segment Pairs:          Score  P(N)      N

>>>emvrl:PTVSEQC   M88678 Potato spindle tuber viroid RNA...  1805  8.8e-145  1
>>>emvrl:PTVMCGA   M14814 Potato spindle tuber viroid mil...   846  9.7e-143  5
>>>emvrl:PSTVM     X76844 Potato spindle tuber viroid (M)...   852  1.7e-142  5
>>>emvrl:PSTVA     X52036 Potato spindle tuber viroid gen...   846  5.3e-142  5
>>>emvrl:PTVSEQB   M88677 Potato spindle tuber viroid mRN...   837  5.3e-142  5
>>>emvrl:PSTVS     X52039 Potato spindle tuber viroid gen...  1084  1.6e-141  2
>>>emvrl:PSTVI4    X76848 Potato spindle tuber viroid (I-...  1043  2.0e-141  4
>>>emvrl:PTVSEQA   M88681 Potato spindle tuber viroid mRN...   828  2.9e-141  5
>>>emvrl:PTVAAA    M93685 Potato spindle tuber viroid mRN...   817  2.3e-140  5
```

```
>>>emvrl:S54933    S54933 {viroid} [potato spindle tuber ...   817  2.3e-140  5
>>>emvrl:PSTVS23   X76846 Potato spindle tuber viroid (S-...   583  2.8e-140  4
>>>emvrl:PS23058   U23058 Potato spindle tuber viroid (PS...  1057  5.2e-140  3
>>>emvrl:PSTVD440  X58388 Potato spindle tuber viroid RNA...  1057  5.2e-140  3
>>>emvrl:PTVCOMPL  M36163 Potato spindle tuber viroid com...  1057  6.3e-140  3

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

>>>>emvrl:PTVMCGA M14814 Potato spindle tuber viroid mild strain, complete
            genome. 7/89
            Length = 359

  Plus Strand HSPs:

 Score = 846 (233.8 bits), Expect = 9.7e-143, Sum P(5) = 9.7e-143
 Identities = 170/171 (99%), Positives = 170/171 (99%), Strand = Plus / Plus

Query:   141 CCTAGCGGCCGACAGGAGTAATTCCCGCCGAAACAGGGTTTTCACCCTTCCTTTCTTCGG 200
             || |||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct:   138 CCCAGCGGCCGACAGGAGTAATTCCCGCCGAAACAGGGTTTTCACCCTTCCTTTCTTCGG 197

Query:   201 GTGTCCTTCCTCGCGCCCGCAGGACCACCCCTCGCCCCCTTTGCGCTGTCGCTTCGGCTA 260
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct:   198 GTGTCCTTCCTCGCGCCCGCAGGACCACCCCTCGCCCCCTTTGCGCTGTCGCTTCGGCTA 257

Query:   261 CTACCCGGTGGAAACAACTGAAGCTCCCGAGAACCGCTTTTTCTCTATCTT 311
             |||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct:   258 CTACCCGGTGGAAACAACTGAAGCTCCCGAGAACCGCTTTTTCTCTATCTT 308

 Score = 313 (86.5 bits), Expect = 9.7e-143, Sum P(5) = 9.7e-143
 Identities = 69/77 (89%), Positives = 69/77 (89%), Strand = Plus / Plus

Query:    49 AAAAAGAAAAAAGATAGGCGGCTCGGAGGAGCGCTTCAGGGATCCCCGGGGAAACCTGGA 108
             | |||  |||||   |||||||||||||||||||||||||||||||||||||||||||||
Sbjct:    48 AGAAAAGAAAAAAGAAGGCGGCTCGGAGGAGCGCTTCAGGGATCCCCGGGGAAACCTGGA 107

Query:   109 GCGAACTGGCAAAAAAG 125
             |||||||||||| || |
Sbjct:   108 GCGAACTGGCAATAAGG 124

 Score = 301 (83.2 bits), Expect = 9.7e-143, Sum P(5) = 9.7e-143
 Identities = 61/62 (98%), Positives = 61/62 (98%), Strand = Plus / Plus

Query:     1 CGGAACTAAACTCGTGGTTCCTGTGGTTCACACCTGACCTCCTGAGCAAAAAAGAAAAAA 60
             |||||||||||||||||||||||||||||||||||||||||||||||| |||||||||||
Sbjct:     1 CGGAACTAAACTCGTGGTTCCTGTGGTTCACACCTGACCTCCTGAGCAGAAAAGAAAAAA 60

Query:    61 GA 62
             ||
Sbjct:    61 GA 62

 Score = 224 (61.9 bits), Expect = 9.7e-143, Sum P(5) = 9.7e-143
 Identities = 48/52 (92%), Positives = 48/52 (92%), Strand = Plus / Plus

Query:   310 TTCTTGCTTCCGGGGCGAGGGTGTTTAGCCCTTGGAACCGCAGTTGGTTCCT 361
             |||||   | ||||||||||||||||||||||||||||||||||||||||||
Sbjct:   308 TTCTTTGCTTCGGGGCGAGGGTGTTTAGCCCTTGGAACCGCAGTTGGTTCCT 359
```

```
 Score = 106 (29.3 bits), Expect = 9.7e-143, Sum P(5) = 9.7e-143
 Identities = 22/23 (95%), Positives = 22/23 (95%), Strand = Plus / Plus

Query:   120 AAAAAGGACGGTGGGGAGTGCCC 142
             || ||||||||||||||||||||
Sbjct:   118 AATAAGGACGGTGGGGAGTGCCC 140

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

>>>>emhum1:HSHOMEC L32607 Human homeobox-like gene. 12/95
            Length = 4283

  Minus Strand HSPs:

 Score = 122 (33.7 bits), Expect = 9.1, P = 1.0
 Identities = 34/46 (73%), Positives = 34/46 (73%), Strand = Minus / Plus

Query:   110 GCTCCAGGTTTCCCCGGGGATCCCTGAAGCGCTCCTCCGAGCCGCC 65
             ||| |  ||| || |||||||||||| || | |||   |||||| |
Sbjct:  3177 GCTTCCCGTTCCCGCGGGGATCCCTGGAGAGGTCCGGAGAGCCGGC 3222

Parameters:


  Query                       -----  As Used  -----    -----  Computed  ----
  Strand MatID Matrix name     Lambda    K       H      Lambda    K       H
   +1      0   +5,-4           0.192   0.173   0.357    same    same    same
   -1      0   +5,-4           0.192   0.173   0.357    same    same    same

  Query
  Strand MatID  Length  Eff.Length   E     S  W   T  X     E2  S2
   +1      0      361       361      10. 119 11 N/A 73   0.021 77
   -1      0      361       361      10. 119 11 N/A 73   0.021 77

Statistics:
  Query          Expected          Observed           HSPs       HSPs
  Strand MatID  High Score        High Score       Reportable  Reported
   +1      0    126 (34.8 bits)  1805 (498.8 bits)    203        203
   -1      0    126 (34.8 bits)   122 (33.7 bits)       2          2

  Query         Neighborhd  Word      Excluded    Failed   Successful  Overlaps
 Strand MatID   Words      Hits        Hits    Extensions Extensions  Excluded
   +1      0       354       84852       11564       70583     2856        19
   -1      0       354       78046        8422       67306     2514         0

  Database:  geall
  # of letters in database:  634350332
  # of sequences in database:  948922
  # of database sequences satisfying -EXP:  63
  No. of states in DFA:  197 (197 KB)
  Total size of DFA:  206 KB (256 KB)
  Time to generate neighborhood:  0.01u 0.03s 0.04t  Real: 00:00:00
  Time to search database:  38.59u 37.51s 76.10t  Real: 00:06:26
  Total cpu time:  39.21u 38.24s 77.45t  Real: 00:06:28
```

What is the Output

The output is categorized into four sections:
The first part shows a histogram of all scores found in comparison to the database. The second part is a one-line description of the database sequences that yielded to one or more HSPs. The third part shows the HSPs themselves, including more information about the sequence. The output of these sections can be selectively arranged by using the optional parameters **`-NOHIST`**, **`-LIST`** and **`-ALIGN`**. The fourth part shows the chosen parameters and contains statistical information concerning the database, number of hits etc..

SUM STATISTICS

Whereas the previous version of BLAST programs use Poisson statistics to ascribe significance to multiple HSPs, the new version retains Poisson statistics as an option, but use Karlin and Altschul (Proc. Natl. Acad. Sci. **90**: 5873-5877 (1993)) Sum statistics by default instead. Sum statistics tends to rank database matches in a more intuitive order than Poisson statistics and, in many cases, yields markedly increased sensitivity. The Sum P-value for a set of HSPs is a function of the sum of the information scores of the HSPs (expressed in bits) and the number of HSPs in the set.

POISSON STATISTICS

The occurrence of two or more HSP's involving the query sequence and the same database sequence can be modeled as a Poisson process by specifying the **`-POISS`**`onp` option. An important result of applying Poisson statistics is that an HSP having a low score and high Expect value (low statistical significance) may be ascribed a statistically significant Poisson P-value when the HSP appears in the context of additional match(es) of equal or greater score with the same database sequence.

The Poisson P-value for any given HSP is a function of its expected frequency of occurrence and the number of HSPs observed against the same database sequence with scores at least as high. The Poisson P-value for a group of HSP events is the probability that at least as many HSPs would occur by chance alone, each with a score at least as high as the lowest-scoring member of the group. HSPs which appear on opposite strands of a nucleotide query or database sequence are considered to be independent, distinguishable events, and are counted separately.

RELATED PROGRAMS

TBlastN compares a peptide query sequence against a dynamically translated database. BlastP compares a protein query sequence against a protein sequence database. BlastX compares a nucleotide query sequence against a protein sequence database. TBlastX compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

Pam generates a point accepted mutation matrix (PAM) for a given distance and arbitrary scaling between 0 and 1000. PressDB makes a user defined GCG nucleotide sequence database available for BlastN and TBlastN.

FastA does a Pearson and Lipman search for similarity between a query nucleotide sequence and any group of nucleotide sequences. TFastA does a Pearson and Lipman search for similarity between a query peptide sequence and any group of nucleotide sequences.

RESTRICTIONS

The **-WORD**size parameter is now adjustable by the user. By default, this parameter is set to 11 (12 in earlier version), so with it's default settings, the new version is slightly more sensitive than the previous one. Smaller values for **-WORD**size lead to an increase in computing time, so you are strongly recommended not to use small values for large databases (like GeAll). Setting **-WORD**size**=10**, for instance, may triple computing time, **-WORD**size**=7** may increase the time by a factor of 7 or 8.

ALGORITHM

In a first step, BlastN searches for all identical subsequences of length **-WORD**size (11 by default) between your query sequence and every particular database sequence. A very simple scoring scheme is used with BlastN: a match is counted as 5 (**-MATCH**) and a mismatch as -4 (**-MISM**atch), respectively, so an identical 11mer yields a scoring value of 55. Afterwards, these initial hits are extended in both directions until the scoring value falls off by the quantity **-EXT**ension (by default 73) from its maximum achieved value or goes below zero. Finally the program displays all sequences scoring as high or higher than a predefined cutoff score (**-CUT**off). By default this cutoff score correponds to an expectation value of about 10, which means that with a random sequence of the same length as the query sequence and a random database of the same size as the databases involved, one would expect 10 sequences scoring as high or higher as the default cutoff score simply by chance.

CONSIDERATIONS

The results of BlastN are comparable to those of other database searching programs (like FastA) but, nonetheless, there are some differences. In general, BlastN (with its default parameter settings) is well suited for finding nearly identical sequences rapidly, but poorly suited to finding moderately- or distantly-related sequences. For this purpose, you should use programs like TBlastX or FastA or you could try to decrease the **-WORD**size parameter at the expense of speed (see RESTRICTIONS above). The major strengthes of BlastN are speed and statistics. Compared to FastA, BlastN is more than an order of magnitude faster and provides you with a statistical assessment of the generated alignments. Additionallly, all regions of similarity between your query sequence and a particular database sequence are displayed.

BlastN supports many optional parameters. Before changing their default values you should get familiar with the algorithm and the statistical basics of the program. Otherwise you might obtain doubtful results! To understand what BlastN really does you are recommended to read: Stephen F. Altschul et al., Journal of Molecular Biology **215**: 403-410 (1990).

SUGGESTIONS

Batch Queue

This program is one of the few programs in the HUSAR/GCG package that can take more than a few minutes to run. Therefore you might want to submit it as a batch-job to leave your screen free for other work.

In HUSAR, just press <rtn> when the program asks:

```
    Run ProgramName in Batch Mode (* y *):
```

Afterwards, you are prompted for the type of queue in which you want to run your program (short, long or verylong queue). Then the program is submitted to the specified batch queue. The "Batch Queue" section in the chapter 6, **Using Batch Queues** of the **User's Guide** describes the batch system and the specifications of the different queues.

ACKNOWLEDGEMENTS

BlastN was written by Stephen F. Altschul(1), Warren Gish(1), Webb Miller(2) Eugene W. Myers(3) and David J. Lipman(1). ((1) National Center for Biotechnology lnformation, National Library of Medecine, National lnstitutes of Health, Bethesda, MD 20894, U.S.A. (2) Department of Computer Science, The Pennsylvania State University, University Park, PA 16802, U.S.A. (3) Department of Computer Science, University of Arizona, Tucson, AZ 85727, U.S.A.).

BlastN was adapted to GCG and implemented to HUSAR by Karl-Heinz Glatting (DKFZ Heidelberg).

COMMAND-LINE SUMMARY

All parameters for this program may be put on the command line. Use the option **-CHE**ck to view the summary below and to add things to the command line before the program executes. In the summary below, the capitalized letters in the qualifier names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose qualifiers or parameter values that are optional. The "Using Program Parameters" section in the Chapter 3, **Using Programs** of the **User's Guide** describes how to use command lines effectively.

```
Minimal Syntax: % blastn [-INfile1=]embl:ptvseqc -Default

Prompted Parameters:

-BEGin=1 -END=229             range of interest
[-INfile2=]geall              database
[-OUTfile=]ptvseqc.blastn     output file name
-BATch[=short]                submits the program to run in the batch queue

Optional Parameters:

-ONEstrand         searches only the top strand of the sequence
-EXP=10.           expected number of maximal scoring segment pairs
-CUToff=97         cutoff score for reporting high scoring segment pairs
-EXP2=0.15         expected number of high scoring segment pairs when
                   comparing two sequences of length 300 (second pass)
-LOWCUToff=35      cutoff score for second pass of search
-WORDsize=11       word size for finding initial hits against the database
-THRESHold=55      threshold for generating neighborhood words
-EXTension=73      maximum permissible drop-off of the cumulative segment
                   score during word-hit extension.
-MATCH=5           score for a single-letter match
-MISMatch=-4       score for a single-letter mismatch
-HSPMax=100        maximum number of reported HSP's per database sequence
-SPAN=a            defines criteria for judging whether one HSP spans another
                      a = display HSP only if it is not spanned by another HSP
                      b = display HSP even if one segment is spanned (and the
                          other one not)
                      c = turns off detecting and discarding of spanned HSP's
-NOCONSISTency     turns off the determination of consistent HSP's
-OLFraction=0.125 defines the maximum fractional length of an HSP that can
                   overlap another HSP and are considered as consistent
-PRUNE             eliminates HSP's not involved in achieving statistical
                   significance from the output
-POISSonp          uses Poisson statistics (instead of Sum statistics) for
                   assessing statistical significance of multiple HSP's
-GAPDEcayrate=0.5 defines a penalty imposed on the gap between each HSP
                   to compute Poisson probabilities
```

```
-SORT=a           defines sort order:
                     a = sort by p-value
                     b = sort by number of HSP's
                     c = sort by highest score
                     d = sort by sum of all scores
-NOHIST           suppresses the histogram
-ALIGN=250        maximum number of sequence alignments to be displayed
-LIST=500         maximum number of sequence descriptions to be displayed
```

OPTIONAL PARAMETERS

The parameters and switches listed below can be set from the command line. Optional parameters available to all programs are described in the "Using Program Parameters" section of Chapter 3, **Using Programs** of the **User's Guide**.

-ONEstrand

searches only the top strand of the sequence.

-EXP=10

establishes a statistical significance threshold for reporting database sequence matches. **-EXP** is interpreted as the upper bound on the expected frequency of chance occurrence of one or more high scoring segment pairs (HSPs). It may be thought of as the number of matches one expects to observe alone during the database search.

-CUToff=97

represents the score at which a single HSP would by itself satisfy the significance threshold **-EXP**. **-CUT**off is calculated from **-EXP** if not explicitly set on the command line.

-EXP2=0.15

is interpreted as the expected number of HSPs that will be found when comparing two nucleotiode sequences of length 1,000.

-LOWCUToff**=35**

sets cutoff score which defines HSPs. It may be thought of as the score expected for the maximum scoring segment pair (MSP) between two nucleotide acid sequences of length 1,000. If not set on the command line, this parameter will be calculated from **-EXP2**.

-WORDsize**=11**

sets the word size for finding initial hits against the database. Smaller values for this parameter lead to an increase in computing time, so your are strongly recommended not to use small values for large databases (like GeAll). Setting **-WORD**size**=10**, for instance, may triple computing time, **-WORD**size**=7** may increase the time by a factor of 7 or 8.

-THREShold**=55**

sets the threshold value for generating neighborhood words

-EXTension**=73**

is the maximum permissible drop-off of the cumulative segment score during word-hit extension. Raising the value of **-EXT**ension may decrease the chance that the program overlooks an high-scoring segment pair (HSP), but it may significantly increase the computing time.

-MATCH=5

sets the score for a single-letter match (Must be a positiv integer).

-MISMatch**=-4**

sets the score for a single-letter mismatch (Must be a negative integer).

-HSPMax**=100**

sets the maximum number of reported high scoring segment pairs (HSP's) per database sequence.

-SPAN=a

This parameter defines criteria for judging whether one HSP spans another. (This option was previously called **-OVER**lap in the earlier version of BLastN.)

a = display HSP only if it is not spanned by another HSP
b = display HSP even if one segment is spanned (and the other one not)
c = turns off detecting and discarding of spanned HSP's

-NOCONSISTency

This parameter turns off the determination of consistent HSP's.

-OLFraction**=0.125**

This option defines the maximum fractional length of an HSP that can overlap another HSP and is considered as consistent.

-PRUNE

This parameter eliminates HSP's not involved in achieving statistical significance from the output.

-POISSonp

This parameter uses Poisson statistics (instead of Sum statistics) for assessing statistical significance of multiple HSP's.

-GAPDEcayrate**=0.5**

This parameter defines a penalty imposed on the gap between each HSP to compute Poisson probabilities.

-SORT=a

This parameter defines sort order as below.

a = sort by p-value
b = sort by number of HSP's
c = sort by highest score

d = sort by sum of all scores

`-NOHIST`

suppresses the histogram.

`-ALIGN=250`

regulates the display of alignments (high-scoring segment pairs). The default is 250 and thus the maximum number of database sequences for which an alignment (high-scoring segment pair) will be reported. This may be much smaller than the actual number of high-scoring segment pairs reported, since any given database sequence may yield several HSPs. With **`-ALIGN=0`** no HSPs are reported.

`-LIST=500`

sets the maximum number of database sequences for which one-line descriptions will be reported. The default value is 500. A warning message is prominently displayed at the end of the one-line descriptions section when more HSPs are found. With **`-LIST=0`** no one-line descriptions are reported and no warning is given.

FASTA

FUNCTION

FastA does a Pearson and Lipman search for similarity between a query sequence and any group of sequences. For nucleotide database searches, FastA is more sensitive than BLAST.

DESCRIPTION

FastA uses the method of Pearson and Lipman (Proc. Natl. Acad. Sci. USA **85**; 2444-2448 (1988)) to search for similarities between one sequence (the *query*) and any group of sequences. In the first step of this search, the comparison can be viewed as a set of dot plots, with the query as the vertical sequence and the group of sequences to which the query is being compared as the different horizontal sequences. This first step finds the registers of comparison (*diagonals*) having the largest number of short perfect matches (*words*) for each comparison. In the second step, these "best" regions are rescored using a scoring matrix that allows conservative replacements, ambiguity symbols, and runs of identities shorter than the size of a word. In the third step, the program checks to see if some of these initial highest-scoring diagonals can be joined together. Finally, the search set sequences with the highest scores are aligned to the query sequence for display.

What is a Word?

A word is any short sequence (n-mer or k-tuple) where you have set n to some small constant less than or equal to six. The word GGATGG is one of the 4,096 possible words of length 6 that can be created from an alphabet consisting of the four letters G, A, T, and C. The word QL is one of the 400 possible words of length 2 that you can make with the 20 letters of the amino acid alphabet.

EXAMPLE

Here is a session using **FastA** to find sequences in the GeAll nucleotide sequence data library with similarities to a human globin coding sequence:

```
% fasta

 FASTA with what query sequence ?  ggammacod.seq

                    Begin (* 1 *) ?
                  End (*   444 *) ?

 Search for query in what sequence(s) (* geall:* *) ?

 What word size (* 6 *) ?

 List how many best scores and alignments (* 40 *) ?

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

 Run Fasta in Batch-Mode ? (* y *)

 Run Fasta in what queue?
 a) short    queue
 b) long     queue
 c) verylong queue
 d) s10 (special (t)fasta) queue

 What Queue ? (* d *)

 ** fasta will run as a batch or at job.

 ** fasta was submitted using the command:
    "  qsub  -q s10   "

Request 4932.cvx12 submitted to queue: s10.

%
```

OUTPUT

Here is some of the output file:

```
(Nucleotide) FASTA of: ggammacod.seq  from: 1 to: 444  June 13, 1993  18:30

Coding sequence for Human fetal beta globin G-gamma.
 ASSEMBLE    April 30, 1987  11:04
Symbols:    1 to: 92   from: Gamma.Seq  ck: 6474,  2179  to: 2270
Symbols:   93 to: 315  from: Gamma.Seq  ck: 6474,  2393  to: 2615
Symbols:  316 to: 444  from: Gamma.Seq  ck: 6474,  3502  to: 3630

 TO: geall:*  Sequences:    116,881  Symbols: 145,836,961  Word Size: 6

Score Init1 Initn
<  4   557   557:==================================================
   8     0     0:
  12     1     1:=
  16     0     0:
  20     5     5:===
  24  5296  5296:==================================================
  28 25618 25618:==================================================
  32 52168 52168:==================================================
  36 54104 54104:==================================================
  40 37288 37288:==================================================
  44 22915 22915:==================================================
  48 16546 16546:==================================================
  52  9491  9360:==================================================
  56  4908  4666:==================================================
  60  2652  2448:==================================================
  64   984   886:==================================================
  68   475   429:==================================================
  72   194   171:==================================================
  76    88    81:=========================================---
  80    30    45:===============++++++++
  84    19   174:==========++++++++++++++++++++++++++++++++++++++++
  88    19   206:==========++++++++++++++++++++++++++++++++++++++++
  92     6   142:===+++++++++++++++++++++++++++++++++++++++++++++++
  96     2    92:=+++++++++++++++++++++++++++++++++++++++++++++
 100     7    54:====+++++++++++++++++++++++
 104     3    17:==+++++++
 108     3    19:==++++++++
 112     0     8:++++
 116     8    20:====++++++
 120     0    14:+++++++
 124     6    25:===++++++++++
 128     8    16:====++++
 132     0     8:++++
 136     7    12:====++
 140     1     6:=++
 144     3     4:==
 148     6     6:===
 152    17     6:===------
 156    10     9:=====
 160     1     1:=
>160   316   339:==================================================
 mean initn score:  35.3 (5.49)
 mean init1 score:  35.3 (5.49)

The best scores are:                                        init1 initn opt..

empri:hshbgg  M15386 Human glycine-gamma-globin, 3’ end. ...1776  1776  1776
empri:hsgggphg  X55656 H.sapiens mRNA for gamma-G globin ...1525  1525  1641
empri:gggm12glb  M92295 Gorilla gorilla gamma-1 and gamma... 900  1382   901

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

ggammacod.seq
empri:hshbgg

ID   HSHBGG     standard; RNA; PRI; 545 BP.
AC   M15386;
DT   16-JUL-1988 (Rel. 16, Created)
DT   06-JUL-1989 (Rel. 20, Last updated, Version 1)
DE   Human glycine-gamma-globin, 3’ end.
KW   gamma-globin; globin. . . .

SCORES     Init1: 1776 Initn: 1776 Opt: 1776
           100.0% identity in 444 bp overlap

                                 10        20        30        40
ggamma                  ATGGGTCATTTCACAGAGGAGGACAAGGCTACTATCACAAGCC
                        |||||||||||||||||||||||||||||||||||||||||||
hshbgg CTCCTAGTCCAGACGCCATGGGTCATTTCACAGAGGAGGACAAGGCTACTATCACAAGCC
               10        20        30        40        50        60

            50        60        70        80        90       100
ggamma TGTGGGGCAAGGTGAATGTGGAAGATGCTGGAGGAGAAACCCTGGGAAGGCTCCTGGTTG
       ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
hshbgg TGTGGGGCAAGGTGAATGTGGAAGATGCTGGAGGAGAAACCCTGGGAAGGCTCCTGGTTG
               70        80        90       100       110       120

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

 CPU time:  0:25:34
 Output File: ggammacod.fasta
```

What is the Output?

The first part of the output file contains a histogram showing the number of overlapping regions between the query and search set sequences that were observed for each score. The histogram is integrated into bins that are of size 2 (for proteins) or 4 (for nucleic acids). For a nucleic acid query sequence, the histogram would normally show the frequency of overlapping regions with scores of 1 to 4, 4 to 8, 9 to 12, and so forth.

The top score for each bin is listed in the leftmost column of the histogram. The second and third columns list the number of *init1* and *initn* scores that fall within each bin. (See the ALGORITHM topic for an explanation of *init1* and *initn* scores.) In the histogram itself, each symbol represents two sequences. The *init1* and *initn* scores are represented by minus (-) and plus (+) symbols, respectively. If the *init1* and *initn* scores are the same in a bin, or if both scores exceed the limit that the histogram can display (100 scores), they are both represented by equals (=) symbols. The mean scores for the entire search are displayed at the bottom of the histogram, along with their standard deviations in parentheses.

Below the histogram, FastA displays a listing of the best scores. `/rev` after the sequence name in this list indicates that the search set sequence overlaps with the bottom (reverse-complement) strand of the query sequence.

Following the list of best scores, FastA displays the alignments of the regions of best overlap between the query and search sequences. A `/rev` following the query sequence name indicates that the search sequence is aligned with the bottom strand of the query sequence.

This program displays only the region of overlap between the two aligned sequences unless you put **-SHOW**`all` on the command line. The display of identities and conservative replacements between the aligned sequences depends on the value of the **-MARK**`x` command-line option. By default (**-MARK**`x`**=3**), the pipe character (`|`) is used to denote identities and the colon (:) to denote conservative replacements.

RELATED PROGRAMS

BlastN compares a nucleotide query sequence against a nucleotide sequence database. BlastN is more than an order of magnitude faster as FastA and provides a statistical significance but tends to be less sensitive. BlastP compares a protein query sequence against a protein sequence database. BlastX compares a nucleotide query sequence against a protein sequence database. TblastN compares a peptide query sequence against a dynamically translated nucleotide sequence database. TBlastX compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

WordSearch identifies sequences similar to a query sequence using a Wilbur and Lipman search. WordSearch answers the question, "What sequences in the database are similar to my sequence?" The output is a list of significant diagonals whose alignments can be displayed with Segments.

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

If you run Compare with the command line option **-WOR**`d`, it calculates the points for a a dot-plot that show where common words between two sequences occur.

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

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

IRx, FindPatterns, StringSearch and Names are other programs for identifying sequences.

RESTRICTIONS

The query sequence cannot be longer than 32,000 symbols. You cannot select a list size of more than 1,000 best scores nor view more than 1,000 alignments. The word size must be from 1 to 6 for nucleic acid queries, and from 1 to 2 for protein queries.

Gaps in the alignments cannot be indefinitely large; one sequence cannot shift out of register with the other by more than 32 symbols.

ALGORITHM

FastA uses the method of Pearson and Lipman (Proc. Natl. Acad. Sci. USA **85**; 2444-2448 (1988)) to search for similarities between one sequence (the *query*) and any group of sequences.

This method first identifies the ten best regions of similarity between the query sequence and each sequence from the search set using a modification of the algorithm of Wilbur and Lipman (Proc. Natl. Acad. Sci. USA **80**; 726-730 (1983)).

The algorithm for this step of the search may be referred to as a *hash-table look-up search*. Wilbur and Lipman searches (including FastA) belong to a class of comparisons that use what is becoming known as *direct addressing* or *k-tuple preprocessing* to increase efficiency.

You set a word size that is then used by FastA to make up a dictionary of all of the possible words of that size in the query sequence. A second dictionary is compiled for the opposite strand if the query is a nucleic acid sequence. The dictionary has an entry for every possible word. Imagine each word, such as GGATGG, as a number in base 4 that corresponds to an entry in the dictionary. At each entry, there is a number telling the positions (coordinates) where the word occurs in the query sequence. If the word does not occur, the number at the entry is zero. Then, for each word in the searched sequences, FastA just looks up the word in the dictionary to find out if it occurs in the query sequence.

If a word from a search set sequence does occur in the query sequence, FastA adds a score for the word to the score of the diagonal on which the word occurs. This added score is equal to the sum of the scoring factors (see below) for each symbol in the matching word. If a word match overlaps another word on the same diagonal, only the scoring factor for the non-overlapping symbol is added to the score of the diagonal.

The default scoring factors for a protein query sequence are the identical match scores from the scoring matrix used. Thus, symbols in a matching word that represent relatively immutable amino acids contribute a correspondingly large scoring factor to the word score. The default scoring factor for a nucleic acid query sequence is a single, constant value for all symbol matches. These defaults can be overridden with the `-PAMfactor` command-line qualifier.

Secondly, the highest scoring regions from each comparison in the first step are rescored using a scoring matrix that allows conservative replacements and runs of identities shorter than the size of a *word*. These scores are saved as the *init1* scores.

Thirdly, FastA determines if several of the highest-scoring regions from different diagonals of a single comparison may be joined together. Only non-overlapping regions may be joined. A single highest score, *initn*, is saved from each comparison if that score exceeds the previously determined joining threshold value.

Finally, the best segment of similarity between the query sequence and each of the highest scoring search set sequences is determined, using the alignment procedure described in Chao, Pearson, and Miller (CABIOS **8**; 481-487 (1992)). The score for this alignment is reported as the *opt* score.

By default, the *initn* scores are used as the basis for keeping a user-set number of the best scores, and the alignments and determination of the *opt* scores are not done until all of the search set sequences have been scanned. The best scores are sorted and reported according to the *initn* score.

Alternatively, by means of the `-OPTall` command-line qualifier, you can direct the program to immediately align the sequences when the *initn* scores are greater than a given threshold and use the *opt* score as the basis for keeping and reporting the list of best scores. This increases the sensitivity of the search, but at the cost of speed. At its most sensitive setting, this option will slow the search down about five-fold. The program calculates a default threshold from the length of the query sequence and the ktup setting. You can override this threshold by adding a positive non-zero number after the command-line qualifier, for example: `-OPTall=20`. A threshold of 1 is the most sensitive setting.

Another option affecting the order of the scores kept is the `-SCAle` command-line switch. When this is specified, all scores are scaled by the factor ln(n0) / ln(n1), where n0 is the length of the query sequence and n1 is the length of the database sequence. This has the effect of increasing the scores of matches with database sequences that are shorter than the query sequence and lowering the scores of matches found with sequences that are longer than the query sequence. The longer the database sequence, the more likely it is that a high score can be attained by chance. Therefore, scaling the score with respect to length can increase the selectivity of the search by making it less likely that high scores resulting from chance will appear near the top of the list.

The FastA algorithm is described in detail by the author of the program, Dr. William Pearson, in *Methods in Enzymology*, **183**; 63-98, Academic Press, San Diego, California, USA, 1990.

CONSIDERATIONS

If there is a database entry that overlaps your query in several places, only the best overlap appears in the alignment display.

The Wisconsin Package™ version of FastA searches both strands of nucleic acid queries unless you put `-ONEstrand` on the command line. Dr. Pearson's FastA searches only the top strand.

There is a difference in program behavior depending on whether FastA is run interactively or non-interactively (on the batch queue or with `-Default` on the command line). When the program is run interactively, it will display the number of scores that you requested, then ask if you want to see more. If you see that the scores for the last few sequences in the list are still high, you can direct the program to display more scores (up to the maximum of 1000) to ensure that you will see all of the high-scoring matches. This in effect allows you to increase `-LIStsize` on-the-fly (and concommitantly, the value for `-ALIGN`, since the number of alignments is also incremented unless `-NOALIGN` was on the command line). When the program is run non-interactively, you have no way of examining the scores during the run, so the program attempts to do it for you. It will continue to report scores that exceed a certain score, even if there are more than were originally requested. (The number of alignments is increased also unless `-NOALIGN` was on the command line.) This may save you from having to repeat the search if `-LIStsize` was initially set too low. If you are certain that you don't want the number of scores and alignments to increase, use `-NOINCrease` on the command line.

SUGGESTIONS

Word Size

By default, FastA uses the maximum word size permitted. Use of smaller word sizes increases the amount of CPU time required to run the program.

Identifying the Search Set

If you want to search a single database division instead of an entire database, see the "Using Database Sequences" and the "Database Logical Names For The HUSAR/GCG Package" section in the Chapter 2, **Using Sequences** of the **User's Guide** for a list of the logical names used for the databases and the divisions of each database. The search set can also consist of a group of sequence files that are not in a database. Use a multiple sequence specification to name these. See the Chapter 2, **Using Sequences** of the **User's Guide** for information about naming groups of sequences for the search set.

Batch Queue

This program is one of the few programs in the HUSAR/GCG package that can take more than a few minutes to run. Therefore you might want to submit it as a batch-job to leave your screen free for other work.

In HUSAR, just press <rtn> when the program asks:

```
    Run ProgramName in Batch Mode (* y *):
```

Afterwards, you are prompted for the type of queue in which you want to run your program (short, long or verylong queue). Then the program is submitted to the specified batch queue. The "Batch Queue" section in the chapter 6, **Using Batch Queues** of the **User's Guide** describes the batch system and the specifications of the different queues.

Interrupting a Search: <Ctrl>C

You can type **`<Ctrl>C`** to interrupt a search and see the results from the part of the search that has already been completed.

COMMAND-LINE SUMMARY

All parameters for this program may be put on the command line. Use the option **`-CHE`**`ck` to view the summary below and to add things to the command line before the program executes. In the summary below, the capitalized letters in the qualifier names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose qualifiers or parameter values that are optional. The "Using Program Parameters" section in the Chapter 3, **Using Programs** of the **User's Guide** describes how to use command lines effectively.

```
Minimal Syntax: % fasta [-INfile1=]ggammacod.seq -Default

Prompted Parameters:

[-INfile2=]GeAll:*             search set (all of GeAll)
[-OUTfile=]ggammacod.fasta     output file name
-BEGin=1 -END=444              range of interest
-WORdsize=6                    word size
-LIStsize=40                   number of scores and alignments to show
-BATch[=long]                  submits the program to run in the batch
```

```
                              queue

Local Data Files:

-DATa=fastadna.cmp            scoring matrix for nucleic acids
-DATa=fastapep.cmp            scoring matrix for peptides

Optional Parameters:

-GAPweight=12.0    gap creation penalty
-LENgthweight=4.0  gap extension penalty
-SINce=6.90        limits search to sequences dated on or after June 1990
-ONEstrand         searches only the top strand of nucleotide sequences
-PAMfactor         uses scoring matrix to calculate initial diagonal scores
-OPTall[=20]       immediately computes opt score if initn above threshold
-SCAle             scales scores by ln(n0) divided by ln(n1)
-SHOWall           shows complete sequences in alignment, not just overlaps
-MARKx=3           determines the alignment display mode
-NOALIGN           suppresses sequence alignments
-NOHIStogram       suppresses printing the histogram
-LINesize=60       number of sequence symbols per line of the alignment
-NODOCLines        suppresses sequence documentation in the alignment
-NOMONitor         suppresses the screen trace for each search set sequence
-NOINCrease        suppresses increase to LIStsize when not interactive
```

ACKNOWLEDGEMENT

FastA and TFastA were written by Professor William Pearson of the University of Virginia Department of Biochemistry (Pearson and Lipman, Proc. Natl. Acad. Sci., USA **85**; 2444-2448 (1988)). In collaboration with Professor Pearson, they were modified and documented for distribution with GCG Version 6.1 by Mary Schultz and Irv Edelman, and for Version 8 by Sue Olson.

SEQUENCE TYPE

The function of FastA 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.

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 the Chapter 4, **Using Local Data Files** in the **User's Guide**.

FastA reads a scoring matrix containing the values for every possible match from your working directory or the public database. The files fastadna.cmp (for nucleic acid sequences) and fastapep.cmp (for protein sequences) contain the default values for matches. fastapep.cmp is the same as Dayhoff's PAM250 matrix. You can use the Fetch program to obtain a copy of these files in order to modify them to suit your own needs.

OPTIONAL PARAMETERS

The parameters and switches listed below can be set from the command line. Optional parameters available to all programs are described in the "Using Program Parameters" section of Chapter 3, **Using Programs** of the **User's Guide**.

`-SINce=6.90`

limits the search to sequences that have been entered into the database or modified since June 1990. As this is being written, only the EMBL, GenBank, and SWISS-PROT databases support this feature.

`-ONEstrand`

searches only the top strand of nucleotide sequences.

`-PAMfactor`

uses a scoring matrix for the calculation of initial diagonal scores. Instead of using a constant factor for each match in a *word*, the identical match scores from the scoring matrix are used. This is the default for protein sequences, while `-NOPAMfactor` is the default for nucleic acid sequences.

`-GAPweight=12.0`

is subtracted from the alignment score whenever a gap is created.

`-LENgthweight=4.0`

is subtracted from the alignment score for each residue added to an existing gap.

`-OPTall=20`

immediately performs an alignment and calculates the *opt* score when the *initn* score is greater than a threshold score. Scores are saved and sorted by *opt* score instead of by *initn* score. You can override the default threshold calculated by the program by typing a number after the `-OPTall` qualifier.

`-SCAle`

scales scores by ln(n0) / ln(n1), where n0 is the length of the query sequence and n1 is the length of the search set sequence.

`-SHOWall`

shows entire sequences in the alignment display, instead of just the best region of overlap.

`-MARKx=3`

determines the alignment display mode -- especially the symbols that identify matches and mismatches. The default value, 3, uses a pipe character (`|`) to show identities and a colon (:) to show conservative replacements. `-MARKx=0` uses a colon to show identities and a period (.) to show conservative replacements. `-MARKx=1` will not mark identities; instead, conservative replacements are connected with a lowercase x, and non-conservative substitutions are connected with an uppercase X. If `-MARKx=2`, the residues in the second sequence are shown only if they differ from the first sequence.

-NOALIGN

suppresses the sequence alignments in the output file. The resulting output file can be used as a list file (previously called a file of sequence names) for input to other Wisconsin Package programs. Use **-ALIGN=10** to display the alignments of the top 10 scoring regions in the output file.

-NOHIStogram

suppresses printing the histogram.

-LINesize**=60**

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

-NODOCLines

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

-NOINCrease

if the program is run noninteractively, it will report more than the requested number of best scores (**-LIS**tsize) if the last scores in the list are still fairly high. **-NOINC**rease suppresses this automatic increase in number of reported scores.

-MONitor**=100**

monitors this program's progress on your screen. Use this option to see this same monitor in the log file for a batch process. If the monitor is slowing down the program because your terminal is connected to a slow modem, suppress it with **-NOMON**itor.

The monitor is updated every time the program processes 100 sequences or files. You can use the optional parameter to set this monitoring interval to some other number.

IRX

FUNCTION

IRx is an Information Retrieval System that identifies sequences. IRx does this by searching for author names, accession numbers, sequence names or for any other keywords within the annotations of the sequence databases.

DESCRIPTION

IRx is designed to be a self-prompting system with menu options displayed at each stage of the retrieval process. Users are encouraged to learn the system by using it and relying on the Help information that is associated with each menu.
IRx matches your request against the sequence annotations, which is done probabilistically and therefore the set of retrieved documents is not guaranteed to answer precisely the user's question, but is likely to contain relevant information. The retrieved documents are ordered according to their expected relevance and so a user typically needs to review only a few documents even when the retrieval set is large.

Using IRx is a simple, three-step procedure:

- Enter a question
- Receive a list of documents which is ranked in order of expected relevance.
- Select documents from the list for reading or storing

Most IRx activities are carried out by typing one-letter commands. You often do not need to press RETURN after entering a command. At each screen, the most common commands are listed at the top. Although they appear in upper case, you may type them in lower case. To see a list of all commands available at a specific screen, type **??** for online help.

One **warning** at the beginning : IRx does **not** support any cursor(arrow)-keys! Use the appropriate one-letter commands to move on the screen.

TERMINAL TYPE

To run IRx your terminal or PC terminal program must be able to emulate VT100. Furthermore, you must set the terminal type properly during the login procedure. With terminal types like "network", IRx will **not** work!

EXAMPLE

Here is an example session using IRx to retrieve sequences of human interleukin-receptors in the EMBL data library :

```
%irx
```

Calling IRxwill put you into the first screen, where you may obtain a short help, introducing the program. Here we refrain from this possibility :

```
================================================================================
                             IRx at DKFZ Heidelberg

                       Information Retrieval Experimental
                     Workbench on Convex with Husar Output

                                 Version 3.0.37
                                  6 March 1990

                             [B-Tree access method]

                        Lister Hill National Center for
                           Biomedical Communications


                       Do you want help? [Type Y or N]: N
================================================================================
```

The following screen lists the available databases for selection. Here we choose the EMBL data libary (in this case simply by pressing RETURN). To switch to other databases, use **D**(down line) or **U**(up line). Remember, **don't** use cursor keys!

```
================================================================================
SELECT ONE OF THE FOLLOWING DATABASES:
        F(forward screen), B(backward screen), U(up line), D(down line),
       RETURN selects an item, ESCAPE returns
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

         ==>    1). EMBL Nucleotide Sequence Annotations
                2). EmNew Sequence Annotations
                3). GenBank Sequence Annotations
                4). SWISS-PROT Protein Sequence Annotations
                4). GbOnly Sequence Annotations
                5). SWISS-PROT Protein Sequence Annotations
                6). Protein Sequence Database (PIR) Annotations
                7). PirOnly Sequence Annotations
                8). Kabat Nucleotide Database Annotations
                9). Kabat Protein Database Annotations
                10). Reference Library
                11). Papillomavirus Nucleotide Database Annotations
                12). Papillomavirus Protein Database Annotations
                13). Sequence Analysis Bibligraphic Reference Data Bank

================================================================================
```

After loading the database, IRx puts you into the QUESTION INPUT screen. Here you may enter your question. In this example we ask for information on human interleukin-receptors, using a specific query technique to retrieve only relevant documents (see **HOW TO ENTER A QUESTION**, **Boolean Operators**) :

```
================================================================================
QUESTION INPUT:                    EMBL Nucleotide Sequence Database (31, 6/92)
  Enter your question in the window below and press RETURN when completed.
  Press ESCAPE for the command menu.  Type ?? at any time for help.
-------------------------------------------------------------------------------
  Database embl loaded.
                      Enter question 1 in the window below
-------------------------------------------------------------------------------
human AND interleukin AND receptor








================================================================================
```

Nearly all **IRx-screens** are divided into three sections separated by dashed lines: The *first* part shows the screen title and the commands, the *second* one gives explanation or displays the summary of search results and the *third* one accepts input or lists the search results in relevant order (see below).
After starting the retrieval process by pressing RETURN, IRx will show you the results. If no documents match your question, you will get a message.

Here is the DOCUMENT SELECTION screen showing the results of the sample question indicating that 61 documents were retrieved and displaying information about the first three ones. The matched words in the documents are highlighted.

```
================================================================================
DOCUMENT SELECTION                 EMBL Nucleotide Sequence Database (31, 6/92)
R(Read current document), D(move pointer Down), U(move pointer Up)
F(move Forward a screen), Q(enter a Question), W(Write), ??(Help), E(Exit IRX)
-------------------------------------------------------------------------------

           61 documents contain one or more words from your question.
-------------------------------------------------------------------------------
 ==>  1: [Weight=31, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2RB
[DE] Human interleukin-2 receptor mRNA (short form), complete cds.

      2: [Weight=31, 3 words; interleukin receptor human]
[IRXID] embl:HSIL1RFT
[DE] Human mRNA for interleukin-1 receptor (fibroblast type)

      3: [Weight=31, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2REC
[DE] Human mRNA for interleukin-2 receptor

================================================================================
```

Here is a short description of the one letter commands listed in the upper part of the screen:

D(move pointer Down) and **U**(move pointer Up) move within the list of documents from one to another.
F(move Forward a screen) moves to the next page of the list.
Q(enter a Question) returns to the QUESTION INPUT screen, to modify the previous question or to ask a new one (see **HOW TO ENTER A QUESTION** and for modifying **GENERAL PURPOSE COMMANDS**).
W(Write) stores retrieved information (detailed description in **HOW TO STORE INFORMATION**).
??(Help) will put you into the appropriate HELP FILE SELECTOR screen, where **all** available commands at this screen are listed. Only the most important commands are listed at the top, in most cases there are many others available (see also **HOW TO SELECT DOCUMENTS**).
E(Exit IRX) leaves IRx and return to your HUSAR-session.

R(Read current document) puts you into the DOCUMENT READER screen and displays the annotation of the corresponding sequence. The screen looks like this:

```
================================================================================
DOCUMENT READER:                   EMBL Nucleotide Sequence Database (31, 6/92)
  F(move Forward), B(move Backward), S(Search for input terms), L(return
  to document List), Q(enter a Question), ??(Help), W(Write), E(Exit IRX)
-------------------------------------------------------------------------------
-- Document 1 of 68 (55 lines) --
[IRXID] embl:HSIL2RB
[DE] Human interleukin-2 receptor mRNA (short form), complete cds.
-------------------------------------------------------------------------------
[ID] HSIL2RB    standard; RNA; PRI; 1563 BP.

[AC] Accession number
     K03122;

[DT] Date
     26-JUL-1991 (Rel. 28, Created)
     26-JUL-1991 (Rel. 28, Last updated, Version 1)

[KW] Keywords
     alternate splicing; interleukin; interleukin receptor;
     T-cell growth factor.

================================================================================
```

At this screen you find some additional commands listed at the top (remember, to see all available commands, type **??**) :

B(move Backward) moves in one step to the previous page of the list (opposite to **F**(move Forward)).
S(Search for input terms) searches for matched query terms within the document, the query terms are highlighted.
L(return to document List) returns to the actual document list.

ONLINE HELP

You will find most topics of this description as online help in the correponding screen by typing **??**. Invoking the HELP FILE SELECTOR when you are in the QUESTION INPUT mode, lists all available online information about question input. The HELP FILE SELECTOR screen looks like this:

```
================================================================================
HELP FILE SELECTOR:               EMBL Nucleotide Sequence Database (31, 6/92)
           Select one of the following menu items for additional
           information.  ESCAPE returns to the previous screen.
-------------------------------------------------------------------------------
  Database embl loaded.
-------------------------------------------------------------------------------
                        How to Enter Questions in IRX

        Normally, questions for IRX are entered from the keyboard in
        plain English.  When the question is complete, press RETURN.
        [Do not use RETURN to move to the next input line; IRX does
        this automatically when necessary.]

        Type the topic number for more information on any topic below:

          1 - Basics                     6 - Ambiguous Pattern Matching
          2 - Editing Questions          7 - Proximity Searching
          3 - Natural Language Questions 8 - Direct Access to Documents
          4 - Boolean Operators          9 - Stemming
          5 - Field Restriction
          -------------------------------------------------------------
          ! - User manual              ESC - Return to previous screen

================================================================================
```

HOW TO ENTER QUESTIONS

At the QUESTION INPUT screen you are asked to enter your question (see **EXAMPLE**). The following information might help you to use an IRx specific syntax and editing functions to improve and simplify your query technique. Using these tools will accelerate the retrieval of those documents you are interested in.

Basics

Questions to IRx can be stated in plain English. Such questions tend to retrieve many documents, including many irrelevant ones. However, IRx's ranking alogrithm moves those most likely to be relevant to the top of the list. The easiest way to get good results is to restrict the question to a few but specific words ("keywords"). Advanced search features are also available for experienced users.

To enter a question, simply type it into the QUESTION INPUT screen. When done, press RETURN to initiate the document retrieval process. (Do not use RETURN to advance lines during input, this is done automatically.)

Until you press RETURN, you may edit your question using the commands given in the QUESTION INPUT screen. You can also press ESCAPE to invoke the command menu (see also **GENERAL-PURPOSE COMMANDS**), where you will find the database **vocabulary** and the command **history**. You can use these features and lateron, you may return to the question input for further editing of your question.

Editing Questions

The following HELP FILE SELECTOR screen lists all available functions for editing and correcting question input.

```
================================================================================
HELP FILE SELECTOR:                EMBL Nucleotide Sequence Database (31, 6/92)
           Select one of the following menu items for additional
           information.  ESCAPE returns to the previous screen.
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
                          (2)  Editing Questions

        The following editing functions are available for correcting
        input before initiating retrieval using the RETURN key.  The
        form ^X means "hold down the control key and press X".  Normal
        keys (letters, punctuation, etc.) are simply inserted at the
        cursor position.  Type the command for more information.

         ^H     Delete before cursor.      ^D   Delete after cursor.
         ^W     Delete current word.       ^U   Clear the edit buffer.
         ^A     Move to start of input.    ^E   Move to end of input.
         ^F     Move the cursor forward.   ^B   Move the cursor backward.
         ^L     Redraw the screen.         ^R   Give numeric argument.
         ^X     Insert history item.       ^P   Load previous question.
        RETURN  Initiate retrieval.      ESCAPE Invoke the command menu.
        ----------------------------------------------------------------
          ! - User Manual   DEL - to main menu   ESC - to previous menu

================================================================================
```

Natural Language Questions

The most natural way to use IRx is to enter questions in plain English. Usually, this causes a large number of documents to be retrieved, only some of which are relevant. However, IRx uses a ranking alogrithm to rank the documents in order of expected relevance. Despite ranking, questions in plain English are often not a quite effective method for searching the database.

It should be understood that IRx does not "understand" the question being asked. Instead, the words in the documents are matched statistically against the words in the question. All documents containing any significant word in the question are retrieved, and the retrieved set of documents is ranked based upon the number of matches and the discriminating power of the words matched. Therefore, questions which are rich in terms specific to the desired topic give the best results.

Boolean Operators

IRx supports three Boolean logic operators, OR (|), AND (&), and NOT (~). When a user intends to include a Boolean operator in a question, it must be entered in all capital letters or you may use the symbols shown below in parentheses. A Boolean operator must appear between two terms in the IRx question:

OR (|)	Returns a match if either term matches the document.
AND (&)	Returns a match only if both terms match the document.
NOT (~)	Returns a match if the first term matches but the second term does not. (Think of this as "and not")

Terms in a question which are not separated by a Boolean operator are treated as if an OR operator is present. A term is defined as a word, a proximity expression (phrase in quotes), a regular expression or a Boolean expression (implicit or explicit) enclosed in parentheses. The terms may or may not have a field restriction.

The example question `interleukin AND human AND receptor` tells IRx to retrieve only those documents where **all** three words are found.

Field Restrictions

For IRX, all the database annotation for the many sequences is broken down into named sections called "fields". Each field contains text which is specific to that field. For example, one field might contain the document's title, another its description and a third a reference list. The names of the fields are more or less database independent but their contents are document specific.

Any term in an IRx question can be restricted to a field or set of fields using the form of **`term [field list]`**. Where **`[field list]`** is a list of one or more field identifiers separated by lines. A term can be a word, a regular expression, a proximity expression or a parenthesized expression containing the above.

In most databases (like EMBL) you will find the following fields:

[ID]	Sequence Identification (sequence name, e.g.: HSIL2RB)
[IRXID]	IRX-identification (unique IRx-identifier, e.g.: embl:HSIL2RB)
[AC]	Accession number
[DT]	Date of creation and updates
[DE]	Description of the document (e.g. name of the protein)
[KW]	Keywords summarize common information in a few words, belonging to the research field (useful for further query)
[OS]	Organism source
[OC]	Organism classification in the natural system
[REF]	References (e.g. authors, titles, literature, journals)
[DR]	Crossreferences
[FT]	Features of the sequence (e.g. introns, mutations, repeats, CDS)
[SQ]	Sequence statistics (e.g. base count, length, molecular weight)
[ORI]	Origin (e.g. chromosome location)

The question `interleukin AND receptor NOT human [de kw]` would tell **IRx** only to list documents which match interleukin and receptor and **not** the word human restricted to the DE and KW fields. The intention of this question was to search only for non-human-interleukin-receptors.

Wild-Card Matching

Wild-card matching allows a term to be expanded to a group of related words. This is normally used to code for a group of related words or to allow for unsure spelling. The following wild-card characters can be used:

# - match any single character
$ - match zero or one character
* - match zero or more characters

When regular expressions are used, the user is presented a list of the words matched by the expressions. The user may select all the words or individual words. The selected words are included in the generated question. For the sake of efficiency, wild-cards should **not** appear at the beginning of a word.

Here is an example of wild-card-matching, showing the matched words belonging to `interl*n`:

```
===============================================================================
   M(Mark/unMark word for retrieval), RETURN(continue processing), U(Up),
   D(Down), F(Forward screen), Q(enter new Question), ??(Help), E(Exit IRX)
-------------------------------------------------------------------------------

  4 words match interl*n
-------------------------------------------------------------------------------
 ==>   interleucin
       interleuken
       interleukin
       interlukin





===============================================================================
```

As you can see, wild-card matching is a good tool to avoid loosing information because of misspelling. You can either mark all words to retrieve all documents or one to proove the information. Here is the document corresponding to **`interlukin`** and now you can easily prove whether this document belongs to "interleukin":

```
==============================================================================
DOCUMENT SELECTION                EMBL Nucleotide Sequence Database (31, 6/92)
R(Read current document), D(move pointer Down), U(move pointer Up)
F(move Forward a screen), Q(enter a Question), W(Write), ??(Help), E(Exit IRX)
-----------------------------------------------------------------------------

            1 document contain one or more words from your question.
-----------------------------------------------------------------------------
 ==>  1: [Weight=-86, 1 word; interlukin]
[IRXID] embl:MMIL25RR
[DE] Mouse interlukin 2 gene 5'-regulatory region





==============================================================================
```

Proximity Searching

If a series of query terms are enclosed in double quotes ("), IRx retrieves the document only if all those terms appear in the same field of that document. For example, "`interleukin receptor human`", means retrieve documents with the words `interleukin`, `receptor` and `human` appearing all in the same field. This is the closest approximation in IRx at the present for searching for a phrase.

A proximity search expression can be embedded in a more complex expression including additional terms, Boolean operators, or field restrictions. In such cases, it is treated as if it were a single word.

Proximity searching is also useful if you want to search for terms containing non-alphanumeric characters. IRX simply ignores search characters, so entrying expression like c-myc would make IRX search for C OR myc finding almost every sequence in the database. With "c-myc", only entries are shown containing both c AND myc in the above field.

Grouping

IRx permits complex expressions to be built by grouping simpler expressions using parentheses. The expressions within the parentheses are evaluated first and the results used to evaluate the enclosing expressions. For example, the query `((interleukin interferon) AND receptor)[DE]` searches for either `interleukin` or `interferon`. The result of this search is AND'ed with `receptor` and the whole expression is then subject restricted to the `DE` field.

Stemming

IRx uses a technique called stemming in an attempt to deal with sets of related words which differ in ending. This allows the query term "tree", to match "trees" and "treed" as well. Stemming is an imperfect tool. It increases the number of relevant documents retrieved, but unfortunately also increases the number of irrelevant ones as well. In addition, stemmers often lump together unrelated terms which happen to have similar spellings.

Stemming can be disabled on a given term by suffixing the term with an '@'. Thus, "tree@" will match "tree", but not "trees" or "treed". In addition, words with wild-card characters are not stemmed.

GENERAL-PURPOSE COMMANDS

When you are in the QUESTION INPUT screen, you may invoke the COMMAND screen by pressing ESCAPE.

```
================================================================================
COMMAND:                           EMBL Nucleotide Sequence Database (31, 6/92)
      V(examine Vocabulary), Q(enter a Question), H(question History)
     ??(help), R(Return to reading last set of documents), E(exit IRX)
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------








================================================================================
```

This menu contains a set of general-purpose commands. These commands provide several functions like:

V(examine Vocabulary) eases access of specific data, see below.
H(question History) supports question formulation, see below.
A(Alternate database) changes the database, is a "hidden" command, see below.

Q(enter a Question)) puts you into the QUESTION INPUT screen.
R(Return to reading last set of documents).
??(help) displays available online help.
E(exit IRX) lets you return to your HUSAR-session.

Examining Vocabulary

This command permits the examination of the terms that can be searched for in the current database. If the term is present in the database, the number of its occurrences is displayed along with alphabetically-related terms. If it is not present, the location where it would occur is displayed along with alphabetically-related terms. This command is also a good tool (see **Wild Card Matching**) to avoid loosing information because of mis- or just different spelling of words. If some terms may be relevant, you can use wild-cards to match all of them within your next question.

Here is an example of using the **V**(ocabulary) command. Pressing **v** will put you into the VOCABULARY SEARCH screen, where you are requested to input a search term. Here we like to see what words are found adjacent to interleukin within the EMBL data library :

```
================================================================================
VOCABULARY SEARCH:                 EMBL Nucleotide Sequence Database (31, 6/92)
   Enter the word you wish to find in the vocabulary followed by a RETURN.
   If found, it is marked by **; otherwise, > shows where it would appear.
-------------------------------------------------------------------------------
Search term? interleukin

-------------------------------------------------------------------------------







================================================================================
```

Pressing RETURN will put you into the VOCABULARY READER screen :

```
================================================================================
VOCABULARY READER:                 EMBL Nucleotide Sequence Database (31, 6/92)
 S(Search for a word), F(Forward a screen), B(Back a screen), U(Up a line),
 D(Down a line), ??(help), ESCAPE returns to the previous screen.
-------------------------------------------------------------------------------
  'interleukin' found
 Freq.    Term
-------------------------------------------------------------------------------
     1    interkeukin
     2    interlaced
     1    interleucin
     2    interleuken
  1182 ** interleukin
     5    interleukin2
    10    interleukins
     2    interlukin
    17    intermedia

================================================================================
```

As you can see in this example list, there are some terms that probably belong to interleukin. The stemming technique will in most cases not be a sufficient tool. To be sure to retrieve all relevant data, you could use in your next question wild-card characters like `interl*n$`.

Question History

The QUESTION HISTORY screen displays a list of the questions which have already been processed. Commands are provided for examining this list and selecting a question for insertion at the current position in the edit buffer. This allows the user to derive new questions from older ones.

The following commands available from this screen.

```
D - move pointer Down           U - move pointer Up
F - Forward by screens          B - Backward by screens
T - go to Top of list           G - Go to a question by number
I - insert question and exit    RETURN - same as I.
Q - Return to question editor   ^L - repaint screen
DEL - to main menu              ESC - to previous menu
```

Changing Databases

Still in the COMMAND SCREEN, you are able to switch to a different database by typing the command **A** (= Alternate database). This command is unfortunately **not** described within IRx‘s online help!

HOW TO SELECT DOCUMENTS

After posing a question you usually retrieve a document list shown in the DOCUMENT SELECTION screen (see **EXAMPLE**). Here is the corresponding HELP FILE SELECTOR screen listing all available commands for selecting and reading of documents :

```
================================================================================
HELP FILE SELECTOR:                EMBL Nucleotide Sequence Database (31, 6/92)
           Select one of the following menu items for additional
           information.  ESCAPE returns to the previous screen.
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
        The Document Selection screen displays a list the titles of the
        retrieved documents.  Commands are provided to scan the list and
        to select documents for reading and output.  The commands avail-
        able from this screen are listed below.  Help information can be
        obtained by typing the command name.

          D - move pointer Down           U - move pointer Up
          F - move Forward by screens     B - move Backward by screens
          T - go to Top of list           G - Go to a specific document
          R - Read current document       V - examine Vocabulary
          Q - Enter a new Question      * P - Print list or documents
        * O - set program Options       * Y - ask why retrieved
          ^L - repaint screen             E - Exit from IRX
        ----------------------------------------------------------------
         ! - User Manual    DEL - to main menu   ESC - to previous menu

             * Command may be disabled by the system administrator.

================================================================================
```

HOW TO STORE INFORMATION

After selecting and reading of documents you may be interested in saving the retrieved information. Still in the DOCUMENT SELECTION screen, you can store whole documents or titles and filenames of documents using the **W**(Write) command. You may write or append the information to a file for further use, e.g. fetching sequence files.

Here is an example, saving the first two documents retrieved with the sample question `human AND interleukin AND receptor` (see **EXAMPLE**) :

Typing the **W** command will put you into the SELECT OUTPUT TYPE screen:

```
================================================================================
SELECT OUTPUT TYPE:                EMBL Nucleotide Sequence Database (31, 6/92)
D(documents), T(titles), C(close output), ESCAPE(return to calling screen).
-------------------------------------------------------------------------------

  Write what?  (select one of the above options)
           61 documents contain one or more words from your question.
-------------------------------------------------------------------------------
 ==>  1: [Weight=-23, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2REC
[DE] Human mRNA for interleukin-2 receptor

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




================================================================================
```

Using the **D** command, IRx asks for selecting documents. Here we select the first two documents :

```
================================================================================
SELECT OUTPUT TYPE:                EMBL Nucleotide Sequence Database (31, 6/92)
D(documents), T(titles), C(close output), ESCAPE(return to calling screen).
-------------------------------------------------------------------------------

Select documents [1]: 1-2
           61 documents contain one or more words from your question.
-------------------------------------------------------------------------------
 ==>  1: [Weight=-23, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2REC
[DE] Human mRNA for interleukin-2 receptor

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




================================================================================
```

RETURN will put you into the SELECT AN OUTPUT OPTION screen, where you should confirm the output option `Append to a file`, by pressing RETURN. IRx will ask you for a filename and will create this file including the whole documents.

Using the **T** command (select output type "titles"), you will have the opportunity to choose between the following output options (you may switch from one to another by pressing the SPACE-bar) :
`Append to a file of filenames (or listfile)`, will create a file including the sequence names (= the IRX-ID fields);
`Append to a file of titles`, will create a file including the sequence names and the description of the sequences (= the DE fields).

Here we choose the `Append to file of titles` option:

```
================================================================================
SELECT AN OUTPUT OPTION:          EMBL Nucleotide Sequence Database (31, 6/92)
Press the space key until the desired output option appears, then type RETURN
             The ESCAPE key returns to the calling screen.
-------------------------------------------------------------------------------
   Output option: Append to file of titles
           61 documents contain one or more words from your question.
-------------------------------------------------------------------------------
 ==>  1: [Weight=-23, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2REC
[DE] Human mRNA for interleukin-2 receptor

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

================================================================================
```

Now IRx will ask you for a name of the output file and will then append or write the filenames, titles or documents to this file. The created output file will be stored in your current working directory.

OUTPUT

Using the **D** command and then the "`Append to a file`" option, the output file looks like this (supposed you have selected the fourth document of the example document list) :

```
[ID] HSIL2      standard; RNA; PRI; 756 BP.

[IRXID] embl:HSIL2

[AC] Accession number

     M14098;

[DT] Date

     02-APR-1988 (Rel. 15, Created)
     23-APR-1990 (Rel. 23, Last updated, Version 1)

[DE] Human T-cell interleukin-2 receptor mRNA, mature peptide region.

[KW] Keywords


////////////////////////////////////////////////////////////////////////////////
```

```
[FT] Key             Location/Qualifiers

     CDS             <1..756
                     /note="interleukin-2 receptor precursor (AA
                     at 1)"
     CDS             1..753
                     /note="interleukin-2 receptor"

[SQ] Sequence statistics

     Sequence 756 BP; 236 A; 194 C; 192 G; 134 T; 0 other;
```

Using the **T** command and then the "`Append to a file of titles`" option the output file looks like this (supposed you have selected the first three documents of the example document list) :

```
      1: [Weight=-23, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2REC
[DE] Human mRNA for interleukin-2 receptor

      2: [Weight=-23, 3 words; interleukin receptor human]
[IRXID] embl:HSIL2RB
[DE] Human interleukin-2 receptor mRNA (short form), complete cds.

      3: [Weight=-23, 4 words; interleukin receptors receptor human]
[IRXID] embl:HSIL7AA
[DE] Human interleukin-7 receptor (IL-7) mRNA, complete cds.
```

Using the **T** command and then the "`Append to a file of filenames`" option the output file looks like this (supposed you have selected the first ten documents of the example document list) :

```
embl:HSIL2REC
embl:HSIL2RB
embl:HSIL7AA
embl:HSIL2
embl:HSIL4R
embl:HSIL2RBC
embl:HSILSRAA
embl:HSIL2R2
embl:HSIL2R3
embl:HSIL2R4
```

Note, that you can use such a file of filenames (list file) as input to other HUSAR programs like Fetch. See the chapter 2, **Using Sequences** of the **User's Guide** for further explanations on using list files.

COMMAND-LINE SUMMARY

Command-line control is not available for this program.

RELATED PROGRAMS

StringSearch is also a program to search through sequence annotations, but should only be used with databases, that are not available for IRx. Otherwise, IRx is a much better tool for this purpose.

SEQANALREF

SEQANALREF is a bibliographic reference data bank relative to papers dealing with sequence analysis. This data bank stores the references of articles from the expanding field of mathematical and computer analysis of biomolecular sequences. SEQANALREF is compiled by Amos Bairoch (Department of Medical Biochemistry C.M.U., University of Geneva Switzerland). The majority of entries belong to one of the following categories:

- Algorithms for protein and nucleic acid sequence analysis: primary, secondary and tertiary structure analysis; pattern matching; similarity searches; alignments, etc.
- Algorithms for sequence-based phylogenetic analysis.
- Description of biopolymer data banks: nucleic acid, protein, tertiary structure, carbohydrates, etc.
- Description of software packages.
- Description of on-line services for molecular biologists.

Format of the entries in this databank

The format of this databank is a subset of that defined for the EMBL Nucleotide Sequence Database and the SWISS-PROT Protein Sequence Data bank.

The line types currently used in the data bank are:

```
ID    Reference IDentifier
RM    Reference Medline: the Medline Unique
      Identifier (UID) for that reference
RA    Reference Authors
RT    Reference Title
RL    Reference Location
KW    KeyWords
CC    Comments
AB    Abstract
```

Each ID line contains an unique identification code associated with the reference that it describe, the identication code made of 8 characters, its format is the following: `AAAAYYNN`

`AAAA` is an acronym generally made up from the first three letters and first initial of the name of the first author. When two or more authors share the same acronym, we have tried to replace for one of them the first initial by the second one, or, in some cases, to use the fourth letter of the name.
`YY` is the year of publication.
`NN` is a serial number in that year (starting with 01).

For a complete description of the format of the RA, RT, RL, KW, and CC line types please refer yourself to either the EMBL or the SWISS-PROT databases user's manuals.

Example of a reference entry:

```
ID   BAIA9101
RM   91283906
RA   Bairoch A.;
RT   "SEQANALREF: a sequence analysis bibliographic
     reference data bank.";
RL   Comput. Appl. Biosci. 7:268-268(1991).
KW   NUCLEIC ACID; PROTEIN; BIBLIOGRAPHY.
CC   Abstract.
AB   (content of abstract, omitted here)
```

Comments lines (CC) are used for the following purposes:

- To indicate if the abstract of the paper is available. Example: `CC Abstract.`
- To indicate if a publication is not in English (some of the papers cited are in French, in Japanese, in Norvegian, in Portuguese, or in Russian). Example: `CC In French.`
- To indicate if a paper is not yet published. Example: `CC In Press.`

REFERENCE LIBRARY

Note: The Reference Library Database is implemented only for experimental use.

The Reference Library Database (RLDB) is the database of the Reference Library System of the Laboratory of Genome Analysis, ICRF. It contains all the information about clone libraries, about the picking of the clones on the microtiter plates as well as the spotting onto high density hybridization filters. Probe data and positive hybridization results are also collected and appended to the database. And the administration data like addresses and clone or library requests or submissions are also part of the database. Each library is spotted onto multiple copies of high density filters and these filters are then sent out to collaborating scientists.
Thus results from one group can be related to results of another group just because they work on the same material.

RESTRICTIONS

Only databases mentioned in the SELECT ONE OF THE DATABASES screen (usually the second screen after starting IRx) are available with IRx. Never use any cursor-keys!!

CONSIDERATIONS

IRx is not a typical HUSAR program and you might miss some usual characteristics (**OPT. PARAMETERS** etc.). IRx provides instead all options interactively, including online help.

ACKNOWLEDGEMENTS

Irx was written by Randy S. Huntzinger (National Library of Medecine, NIH, Washington D.C., U.S.A.) and implemented to HUSAR by Detlef Wolf (DKFZ, Heidelberg).