Volume 14 Number 1 1986 Nucleic Acids Research

The current status and portability of our sequence handling software

Rodger Staden

Laboratory of Molecular Biology, Medical Research Council Centre, University Medical School, Hills Road, Cambridge CB2 2QH, UK

Received 8 July 1985

ABSTRACT

I describe the current status of our sequence analysis software. The package contains a comprehensive suite of programs for managing large shotgun sequencing projects, a program containing 61 functions for analysing single sequences and a program for comparing pairs of sequences for similarity. The programs that have been described before have been improved by the addition of new functions and by being made very much easier to use. The major interactive programs have 125 pages of online help available from within them. Several new programs are described including screen editing of aligned gel readings for shotgun sequencing projects; a method to highlight errors in aligned gel readings, new methods for searching for putative signals in sequences. We use the programs on a VAX computer but the whole package has been rewritten to make it easy to transport it to other machines. I believe the programs will now run on any machine with a FORTRAN77 compiler and sufficient memory. We are currently putting the programs onto an IBM PC XT/AT and another micro running under UNIX.

INTRODUCTION

The package of programs I describe here has been constantly extended and enhanced since the first simple routines were published in this journal in 1977 (1). It has been developed mostly in response to the needs of several very active sequencing groups here at the Laboratory of Molecular Biology. For example, the shotgun data handling programs (2,3,4) have been used for two of the largest sequencing projects so far undertaken, the complete sequences of bacteriophage Lambda (5) and Epstein Barr Virus (6). As well as providing a thorough test for data handling programs and an incentive to achieve efficiency, such sequencing projects always produce new computing problems.

One important class of problem that they have highlighted is that of trying to understand the function of new sequences when very little else is known about their properties. One method for understanding function is to

The package of programs described is available for £150 (non-profit) or $8000 (commercial).

compare new sequences against all the sequences contained in sequence libraries such as GenBank (7) NBRF (8) and EMBL (9), and programs for this purpose have been described by others (10,11). Another way of trying to understand the function of new sequences is to analyse the content of each section of the sequence to see if it has the properties of protein coding sequences (12), or contains signal sequences such as promoters or ribosome binding sites (13). These types of problem are addressed by the methods contained in the program ANALYSEQ (14). This program also has facilities for the more frequently performed searches like those for restriction enzyme binding sites, and routines for listing and translating sequences and counting codons etc. Programs for comparing sequences by the so called 'dot matrix' method abound. Our version DIAGON (15) is used interactively from a simple graphics terminal and contains a fast and sensitive comparison routine first described by McLachlan (16). For proteins it uses a score matrix MDM78 (17) based on accepted point substitutions in known protein families.

The value of a program to users depends on a number of things: it might be the only one that does a particular job, or it might be fastest, it might be easiest to use; it might be the only program of its type that runs with a particular operating system or machine. Generally it is a combination of all of these things and recently I have been working on the last two aspects, making the programs very much more easy to use, and have reorganised and rewritten them to make them portable.

The paper gives a summary of the current capabilities of the programs in the package with detail about new programs or functions, describes the improvements in ease of use, and outlines the conversion to portability. Some information about the simple changes required to get the programs running on different machines is also given.

THE SHOTGUN DATA HANDLING SYSTEM

In the design of our shotgun data handling package emphasis has been placed on assuring the accuracy of the final sequence and minimising the amount of time users need to spend checking their data and running the programs. The accuracy of each gel reading depends on the sequencers ability to interpret his autoradiographs and the number of errors he makes in transferring his data into the computer. In shotgun sequencing projects the sequence will be determined many times and the accuracy of the final sequence depends on both the individual gel readings and on the number of

times the sequence has been determined on each strand of the DNA. Our shotgun data handling package (the 'DB system' (2,3,4)) has several components: use of digitizers for data entry, automatic screening-out of cloning vector sequences and ligated fragments, automatic comparison and alignment of new sequences with previous data, and interactive checking and editing of sequences. Below I outline its use.

Gels are simultaneously interpreted and their sequence read into the computer using a program that utilizes a sonic digitizer (4). This program (GELIN) has a new lane following algorithm and has recently been implemented on a micro computer (18). Use of a micro allows us to work anywhere in the laboratory and batches of data can now be stored on floppy disks and later transferred to the larger machine for processing.

The next stages in the processing are automatic in that they require no user intervention (3). Gel readings are treated in batches as passed on from GELIN. The readings may contain only the sequences of the vectors used in their preparation, or may include a restriction site indicating a religation event or the circularisation site used prior to sonication (19). To check for these problems the data is passed through programs SCREENV and SCREENR and only good data is passed on to the next stage.

The data for a sequencing project is stored in a project database. When a new gel reading is put into the database it is aligned with all the previous data that it overlaps. This process was once performed in two parts. overlaps were found by one program (DBCOMP) and users would align overlapping sequences interactively as they added them to their project database using the program DBUTIL. Automation of these two parts was first described in 1982 (3). A program DBAUTO takes the data passed on from the screening programs, calculates a consensus for the current state of the project database, and then compares each new gel reading in turn with the consensus. It then adds each new gel reading to the database, automatically aligning it correctly with all the other gel readings overlapped. After each sequence is added to the database the consensus is updated ready for the next comparison. On the VAX it takes about 1 second to compare and align a 250 base gel reading with a consensus of 10,000 bases. The screening programs run at similar speed.

DBAUTO achieves alignments by inserting padding characters in the shorter of the two sequences. These padding characters can be left in the sequences until such time as sufficient information is available to decide which sequences are correct. Because the data is sequenced so many times it

```
                    10        20        30        40        50
-6  HINW.010  GCGACGGTCTCGGCACAAAGCCGCTGCGGCGCACCTACCCTTCTCTTATA
              GCGACGGTCTCGGCACAAAGCCGCTGCGGCGCACCTACCCTTCTCTTATA

                    60        70        80        90       100
-6  HINW.010  CACAAGCGAGCGAGTGGGGCACGGTGACGTGGTCACGCGCGGACACGTC
-3  HINW.007                                          GGCACA*GTC
              CACAAGCGAGCGAGTGGGGCACGGTGACGTGGTCACGCCG-G-ACACGTC

                   110       120       130       140       150
-6  HINW.010  GATTAGGAGACGAACTGGGGCG3CGCC*GCTGCTGTGGCAGCGACCGTCG
-3  HINW.007  GATTAG4AGACGAACTGGGGCGACGCCCG*TGCTGTGGCAGCGACCGTCG
-5  HINW.009                                    GGCAGCGACCGTCG
17  HINW.999                                      AGCGACCGTCG
              GATTAGGAGACGAACTGGGGCGACGCCCGCTGCTGTGGCAGCGACCGTCG

                   160       170       180       190       200
-6  HINW.010  TCT*GAGCAGTGTGGGCGCTG*CGGGCTCGGAGGGCATGAAGTAGAGC*
-3  HINW.007  TCT*GAGCAGTGTGGGCGCTGC*CGGGCTCGGAGGGCATGAAGTAGAGC*
-5  HINW.009  TCT*GAGCAGTGTGGGCG*T*G*CGGGCTCGGAGGGCATGAAGTAGAGC*
17  HINW.999  TCTCGAGCAGTGTGGGCGCTG**CGGGCTCGGAGGGCATGAAGTAGAGCG
12  HINW.017                                          GTAGAGC*
              TCTCGAGCAGTGTGGGCGCTG-CCGGGCTCGGAGGGCATGAAGTAGAGCG
```

Figure 1. A display of aligned gel readings produced by the program DBUTIL.

soon becomes apparent which gels need checking and users need not repeatedly check each film. The program DBUTIL has as one of its functions the ability to display all of the gel readings covering each section of the sequence. The display is now as seen in figure 1. which shows the left end of a contig (3) from position 1 to 200. Overlapping this region are gels named HINW.010, HINW.007, etc, and numbered 6,3,5,17 and 12; 6, 3 and 5 are in reverse orientation to their originals (denoted by a minus sign). There are a few uncertainty codes (3) and a few padding characters (shown by * symbols) in the gel readings, but the consensus (shown below each 50 base page width) has a definite assignment for almost every position. With such a display it is very easy to assess the accuracy of the sequence, since the user can see all the original base and uncertainty code assignments, the relative strandedness, and the number of times each base has been sequenced.

To direct attention to regions of disagreement between gel readings I have written a new program called HIGH. It is used for highlighting differences between individual gel readings and their consensus. HIGH produces output that looks like that seen in Figure 1. but which marks only those bases that are different from the consensus; all bases that are identical to the consensus are set to . characters. For example, if the contig shown in Figure 1. were passed through HIGH the output would look like that shown in Figure 2.

Such a display is particularly important when checking long (say 10,000 base) contigs and as can be seen makes it very obvious where the

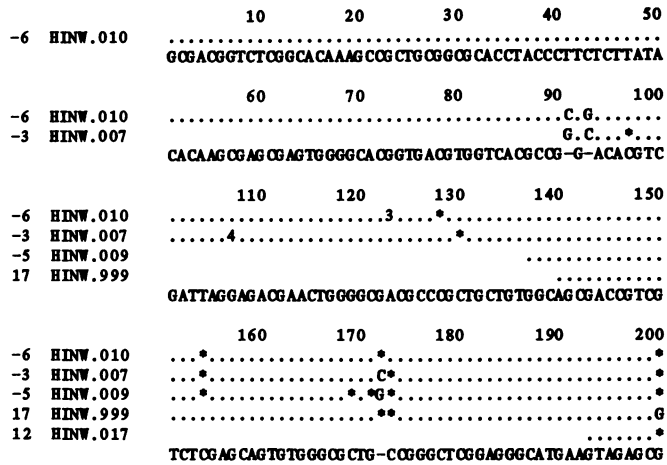

Figure 2. A display of aligned gel reading differences produced by the program HIGH.

disagreements lie. Users can then check the relevant films and edit the gel readings in the database.

DBUTIL has always had interactive editors for this purpose but recently I have made it possible for users to use screen editors on the aligned sequences. This means that they can use word-processing editors on the sequences as they appear in figure 1. As is emphasised later we believe it is important that the programs are portable, and the decision was taken not to write a special editor but instead to make it possible to use the local editors available on the particular system on which the data is stored. For example VAX users can operate on their aligned sequences with the screen editor EDT. A new function has been added to DBUTIL to send the DISPLAY output (as in Figure 1.) to a disk file where it can be edited by any local screen editor, and then a further routine is used to put the data back into the database.

The data in the database is not stored as it appears in Figure 1. but is held as a series of linked lists (2), and so putting the edited data back into the database is not a trivial task. Three new programs are used to make the screen editing possible. PREDIT (pre-edit) prepares the display output for editing; POSTED (post-edit) checks the edited data for any errors made by users and then creates a temporary database for the edited data, finally DBMERGE (database merge) puts the edited data back into the original database. Each program requires only two inputs, namely an input file name

and an output file name. We have found the combination of HIGH and screen editing greatly speeds up and simplifies the process of cleaning-up and editing sequences.

The shotgun handling programs have also been improved by the addition of checks on user input and by including an online help function in DBUTIL, the major interactive program. Help can be selected for each option and appears one page at a time. A page is equal to the number of lines on the screen, and the user can move to the next page by hitting carriage return, or he can select help on a different option by typing 1. Currently 41 pages of help are available.

THE ANALYSIS OF SINGLE SEQUENCES

Almost all of our analysis of single sequences is performed using the program ANALYSEQ (14). The program currently contains 61 different functions ranging from simple listing and translation routines to unique statistical tests for locating protein coding regions and signal sequences. A simplified list of these options is shown below.

read in a new sequence	change current active region of sequence
list a sequence	translate a sequence (1-6 phases)
reverse and complement	search for restriction enzymes
count codons	count dinucleotides
count amino acids	count bases
store codon table on disk	store amino acid sequence on disk
search for repeats	search for hairpin loops
search for inverted repeats	search for Z DNA
display a text file	draw a map
calculate codon pressure	direct output to disk files
search for local similarity	search for local complementarity
compare a short sequence	
write out active sequence to a disk file	

plot stop codons
find longest open reading frames
search for E. coli promoters
search for E. coli ribosome binding sites
search for polyA sites

search for eukaryotic ribosome binding sites
search for splice junctions
general signal search

Search for protein coding regions using the following methods:
Staden and McLachlan codon usage
Staden positional base preferences
Staden uneven positional base frequencies
McLachlan, Staden and Boswell codon improbability
Fickett Testcode for sliding windows
Shepherd RNY preference for sliding windows
search for tRNA genes using the method of Staden

plot base composition
plot negentropy of composition
plot local deviations in base, dinucleotide and codon composition

clear screen
draw a ruler
redefine margins
clear text
use cross-hairs

Some of these problems require graphical representation of results and need to be interactive, and so the program is used from a graphics terminal (a vdu on which lines and points can be drawn as well as characters). The reason for using graphical output is that sequences are getting too long for us to be able to grasp their organisation by looking at listings. Furthermore, in the deduction of properties, we are mostly dealing with probabilities and need to apply several different methods because no single one gives a conclusive result. Graphical output allows us to superimpose the results from different types of search enabling us to judge their consistency. Thus we can easily check whether the most probable reading frame is free of stop codons and preceded by a promoter and a ribosome binding site.

The program can currently handle data stored in three formats: EMBL, GenBank and the format used by our shotgun programs. We use EMBL format for storing our own finished sequences and the program can interpret EMBL feature tables for several purposes. It can automatically translate only the coding regions of sequences, or produce codon tables using only the

coding sequences, and it can draw maps of the sequence marking features automatically extracted from the EMBL format files.

Recent general improvements to the program include error checking for all user input, giving default values for most parameters, making the program very hard to crash, allowing users to escape from any part of the program, and the inclusion of online help. Currently 62 pages of online help are available from within the program.

A method for calculating the randomness of codon usage (20) is one of the new forms of analysis available in the program. This allows us to measure the extent to which codon choices are constrained independent of the amino acid composition of the resulting protein sequence, and hence permits comparisons between genes. An example of use might be to compare levels of protein production with the degree of constraint on codon choices to see if they were correlated (20).

One important form of analysis offered by the program is the ability to search for poorly defined signal sequences such as promoters (13). The basis of the searches performed by the program is a set of weight matrices that define the particular signals. The weights are derived by aligning all known occurrences of each signal sequence and calculating a frequency table for each of the 4 bases at each position in the signal sequence. Each of these known signal sequences is then analysed using the weight matrix. The range of scores obtained can be used as expected values when the matrix is applied to new sequences. Weight matrices for E. coli promoters, E. coli ribosome binding sites (21), intron/exon junctions, and eukaryotic ribosome binding sites are effectively built into the program. Clearly it would be inconvenient if users had to wait for a programmer to add the necessary code for each new class of special sequence as it was discovered. Also users need the ability to test putative new signal sequences of their own against other sequences. In order to facilitate this two extra programs (CUTOUT and GETFRQ) have been written, and a new function called 'the general signal search' has been added to ANALYSEQ. CUTOUT simply allows users to extract and align any number of putative signal sequences from within other sequences. They are stored aligned in a file that is operated on by GETFRQ (get frequencies) to derive the values needed by the general signal search of ANALYSEQ. ANALYSEQ needs to know several different things: it needs to know the length of the signal, the weights for each base at each position of the signal, expected maximum and minimum scores from applying the weights, and the position on the screen to draw the resulting plots. GETFRQ

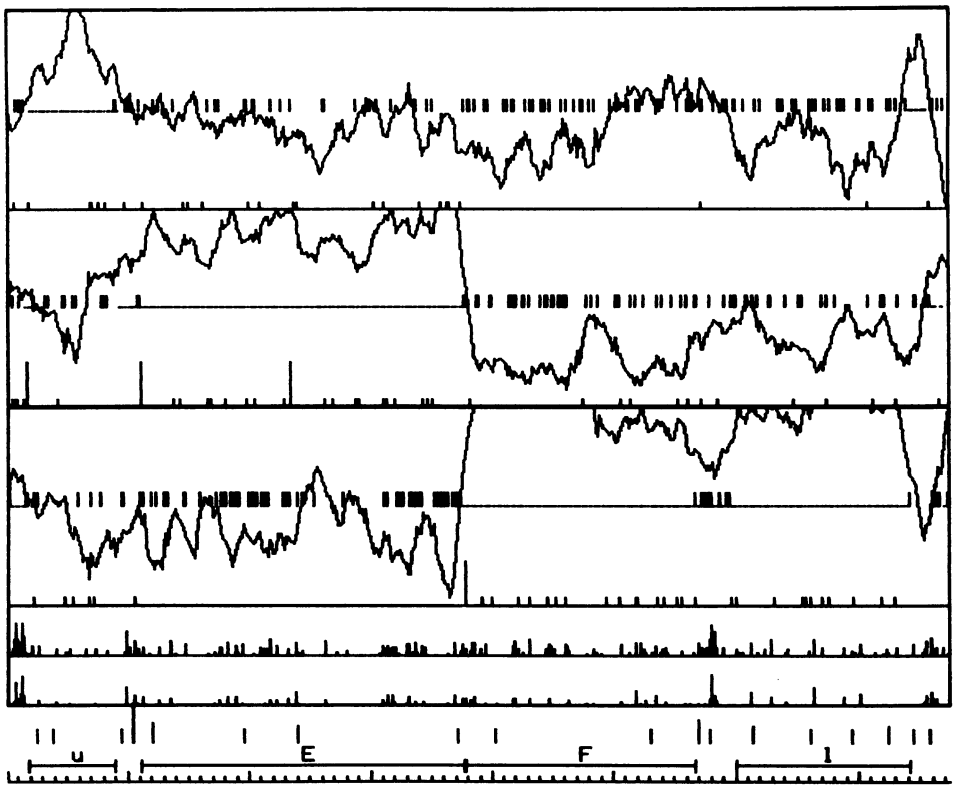

Figure 3. An analysis of the pyruvate dehydrogenase complex of E. coli using several functions of the program ANALYSEQ.

calculates all these values and writes them to a special file used by the general signal search function of ANALYSEQ. When using this function in ANALYSEQ the user only has to type the appropriate file name to apply the corresponding weight matrix to any sequence. We anticipate building up large numbers of such files, both from our own work and from published collections of sequences.

An example showing the graphical output produced by some of the functions of ANALYSEQ is shown in Figure 3. For all the plots the sequence is represented by the x axis of the screen and the results of analysis are plotted in the vertical direction. The figure shows an analysis of the pyruvate dehydrogenase complex of E. coli containing the lpd gene, aceF and aceE, and an unidentified gene (22,23). At the base of the figure is a scale marked in 100's of bases; above this is a map showing the positions of

the genes. Above, the figure is divided into horizontal strips: the top three show plots for the probability that the sequence is coding for a protein in each of the three possible reading frames. This is shown by the continuous jagged line and represents the result of applying the 'positional base preferences method' (12) to the sequence. The positions of stop codons are also marked by the short vertical lines halfway up each of the three strips, and at the base of each strip start codons are marked in the same way. If we take the predictions for aceE as an example we can see that in reading frame two there is a strong probability of coding, and that there are no stop codons in that frame. Also shown in these three strips are predictions for the beginnings of genes (13) and these are marked by the longer vertical lines starting at the base of each strip. Again taking aceE as an example we see there are two gene starts predicted, one of which is in the correct position. Just above the map of the genes are several short vertical lines marking the positions of potential hairpin loops (there is one of 19 consecutive basepairs at position 1021 and one followed by a run of T residues at position 5678). Immediately above the hairpin loop results are two narrow strips containing searches for E. coli promoters (13) (the strongest sites are 5' of the unidentified reading frame, 5' of the coding region for aceE, and 5' of the coding region for lpd). Performing all these analyses and producing this figure exactly as is shown here took only a few minutes using only ANALYSEQ.

COMPARISON OF PAIRS OF SEQUENCES

The program that we use for comparison of pairs of sequences is called DIAGON (15) and it is operated from a simple graphics terminal. Its functions include: two comparison algorithms that represent their results graphically (to produce 'dot-matrices'), a comparison routine to calculate and list the observed scores, a routine to calculate the expected scores using the McLachlan 'double matching probability' (16), routines to align and display the sequences, routines to alter the size and position of the diagrams and to draw scales around the plots, and a help function that gives 23 pages of online help information. The program has also been improved to trap all user errors and to offer defaults for most of the parameters. The ability to control the position at which the plots appear on the screen allows several comparisons to be presented on the screen at once (see Figure 4.).

Here I digress slightly to comment on the sensitivity of the different

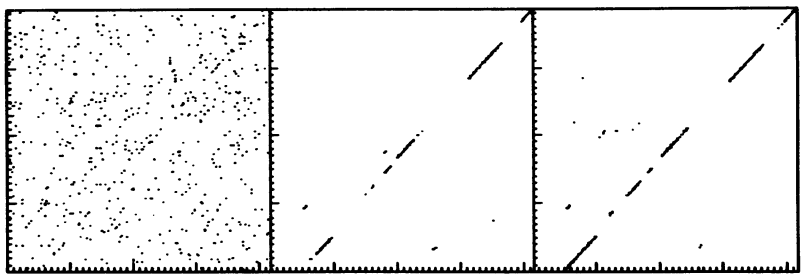

Figure 4a **Figure 4b** **Figure 4c**

Figure 4. A comparison of "dot matrix" methods using the program DIAGON to compare the alpha-1-antitrypsin and ovalbumin protein sequences.

comparison algorithms, using as an example a comparison of Alpha-1-antitrypsin (24) and Ovalbumin (25). The most simple-minded algorithms look for runs of identical sequence elements. The result of applying such an algorithm to our test sequences is shown in Figure 4a, where all runs of at least 2 identical amino acids are marked by a dot. As can be seen any similarity is hard to spot. A dramatic improvement can be achieved by looking diagonally left and right from each point in the matrix, counting the number of identical sequence elements found within a given distance, and then marking all points where some minimum number of identities are found. For our test sequences, if we look diagonally over a distance of 21 amino acids, and draw a dot at each point where at least 7 identical amino acids are found within the 21 positions, we get the result shown in Figure 4b. The similarity is now obvious. A further improvement can be gained if, instead of looking for identities, we take into account the similarities of amino acids, and using a score matrix that gives similarity values for all pairs of amino acids, we sum the scores found along each diagonal. By applying this method and the relatedness odds matrix MDM78 (17) to our test sequences we get the result shown in Figure 4c. Each of these methods is available in DIAGON but obviously the latter, first described by McLachlan (16), is used most often. The test sequences are 418 and 385 amino acids in length and each plot took less than 6 seconds on a VAX 11/780. The whole figure was produced as shown in a few minutes using only DIAGON.

PORTABILITY

One of the reasons for the proliferation of sequence analysis software packages is that most of the available programs are not portable. I believe that, in its present form, the package described in this paper can now be easily got running on any machine with sufficient memory (see below) and a FORTRAN77 compiler. In order to achieve this I have recently reorganised and rewritten all of the programs. I have adhered completely to the FORTRAN77 standard for programs not using graphics, and for graphics to the Tektronix Plot4010 standard.

FORTRAN77 is one of the most completely defined languages but still contains some unspecified and therefore system specific features. This includes such things as OPEN statements, logical unit numbers, allowed sizes of numeric variables and lengths of string variables. As long as one is aware of these points it is possible to write portable code. Problems about permissible sizes of numeric variables and maximum string lengths can be avoided by careful use of double precision variables for numerically sensitive calculations and by sticking to short string variables. All system-specific code can be isolated into a few special subroutines that are called by all programs. When it is known how to set the values and names in these few routines, all the programs should run without further changes.

I have organised the programs so that only two small subroutines are used for system specific code. One contains the OPEN statements for all the programs, there are currently ten different types of OPEN used and the calling programs simply send a number to say which is required. The other subroutine assigns values to numeric variables that are used as logical unit numbers by all input and output statements. I am currently moving the programs from the VAX to a micro running under UNIX. By changing only 3 values in the two system specific subroutines I have got all the shotgun programs running on the UNIX machine (26). (The logical unit numbers for terminal input and output are 5 and 6 on the VAX but 0 and 0 for UNIX, the VAX specifies unformatted direct access record lengths in words and UNIX does so in bytes.) No code in the main programs was changed at all and the changes necessary in the two subroutines were simply found in manuals.

Program size is another factor affecting portability. As all of the programs currently keep the whole of each sequence in memory during analysis, the size of the programs depends on the length of the longest sequence they can handle. As a guide, for a maximum sequence length of 50,000 bases, the sizes of the largest programs on the VAX are: DBUTIL =

159k, DBAUTO = 364k, DIAGON = 403k, ANALYSEQ = 533k bytes. (ANALYSEQ is actually set for sequences up to 180,000 bases.) For all programs the maximum sequence length is set by a single PARAMETER statement and so is easily changed.

Use of graphics presents its own portability problems. This is because as yet, we have no device independent standard for graphics and some of the necessary operations cannot be coded in FORTRAN77, and also because each particular device can influence the way the graphics is used to present results to the user.

We run our graphics programs on a VT100 terminal that has been modified (27) to give two independent planes - one the normal scrolling text plane, and the other a fixed graphics plane. This enables us to draw a graph which will stay on the screen, and to prompt users in the text plane, these prompts and the users replies will initially overwrite the graph but will subsequently scroll up off the screen leaving the graph in position. Obviously such a facilty is very convenient and on devices with only a single plane we would need to use separate windows for text and graphics to achieve a similar effect. Although a number of manufacturers make terminals that offer two planes there is not a DEC (28) equivalent and so some users have not been able to use the programs as intended. An alternative, and one which we intend to take with the UNIX micro, is to use two terminals side-by-side: one for scrolling text and the other for graphics. On a VAX this could be done using a VT100 and a VT240 and would require one OPEN statement to be changed in the system specific subroutines.

The graphics routines for the programs have also been completely rewritten and reorganised to conform to the Plot4010 standard and, as far as is possible, FORTRAN77. They are separated into two levels: the higher level takes care of all scaling, clipping etc, and is independent of the graphics standard used, the lower level routines set up the escape sequences for the Plot4010 standard. Each set of routines forms a separate library and again those sections of code, in the lower level routines, that could cause problems on different machines have been isolated and clearly identified. They could easily be replaced by, or interfaced to, a different standard.

One further area of possible difficulty concerns the use of program GELIN. GELIN allows users to delete characters from the sequences they are reading after they have appeared on the terminal screen. This requires the program to control the cursor position and hence is terminal specific. The

FORTRAN77 version of GELIN therefore expects to be run from a VT100 compatible terminal. The version for the micro computer is written in BBC BASIC and can only be run on the BBC ACORN model B.

DISCUSSION

I have summarised the current capabilities of our sequence analysis programs. The package currently contains 24 programs, libraries containing over 300 subroutines, and consists of over 20,000 lines of FORTRAN77. Apart from the new programs and routines, of most interest to existing users will be the improvement in ease of use. I know of no way in which the programs can be crashed: all user input is checked for errors and defaults are offered for many values. The help functions within the programs currently give access to 125 pages of information describing both the purpose of each function and giving instructions on their use. Help can be selected on each individual option and will appear one page at a time. The user can page through the information by typing carriage return, or can switch to help on another option by typing the number 1. The information appears instantly. All of the textual output produced by the programs can be sent to disk files for later printing or editing. This means, for example, that tables or sequences can be annotated ready for publication.

Of interest to those who do not have access to a VAX is the fact that the programs are now easily moved to other machines. We plan to have the programs running on IBM PC XT and AT's and on UNIX machines by the time this article appears. We are also hoping to get feedback from those with other machines to see what changes to the code they found necessary. This information can then be passed on to people with similar machines.

When the programs are distributed VAX users receive a set of simple command procedures that will get all the programs up and running, those with other machines receive notes on which subroutines may need changing. Of course, all future new programs will be written in portable code.

ACKNOWLEDGEMENTS

I would like to thank Andrew McLachlan, Arthur Lesk and Sydney Brenner for critical reading of this manuscript.

REFERENCES

(1) Staden, R. Nucl. Acid Res. 4, 4037-4051 (1977)
(2) Staden, R. Nucl. Acid Res. 8, 3673-3694 (1980)
(3) Staden, R. Nucl. Acid Res. 10, 4731-4751 (1982)

(4) Staden, R. Nucl. Acid Res. 12, 499-503 (1984)
(5) Sanger F, Coulson, A R, Hong G F, Hill D F and Petersen G B, J. Mol. Biol. 162, 729-773 (1982)
(6) Baer R, Bankier A T, Biggin M D, Deininger P L, Farrell P J, Gibson T J, Hatfull G, Hudson G S, Satchwell S C, Seguin C, Tuffnell P S and Barrell B G, Nature 310, 207-211 (1984)
(7) GenBank, c/o Bolt Beranek and Newman Inc., 10 Moulton Street, Cambridge, M.A. 02238. USA
(8) Protein Identification Resource, National Biomedical Research Foundation, Georgetown University Medical Center, 3900 Reservoir Road, N.W., Washington, D.C. 20007. USA
(9) EMBL Nucleotide Sequence Data Library, European Molecular Biology Laboratory, Postfach 10 22 09, D-6900 Heidelberg. FRG
(10) Wilbur W J and Lipman D J Proc. Natl. Acad. Sci. U.S.A. 80 726-730 (1983)
(11) Lipman, D J and Pearson W R, Science 227, 1435-1441 (1985).
(12) Staden, R. Nucl. Acid Res. 12, 551-567 (1984)
(13) Staden, R. Nucl. Acid Res. 12, 505-519 (1984)
(14) Staden, R. Nucl. Acid Res. 12, 521-538 (1984)
(15) Staden, R. Nucl. Acid Res. 10, 2951-2961 (1982)
(16) McLachlan, A. D. J. Mol. Biol. 61, 409-424 (1971)
(17) Dayhoff, M. O. Atlas of Protein Sequence and Structure, National Biomedical Research Foundation, Silver Springs, Maryland (1969)
(18) Acorn BBC model B micro computer. Acorn, Cambridge, UK.
(19) Deininger,P Anal. Biochem. 129, 216-223 (1983)
(20) McLachlan A D, Staden R and Boswell D R, Nucl. Acid Res. 12, 9567-9575 (1984)
(21) Stormo, G. D., Schneider, T. D., Gold, L. and Ehrenfeucht A., Nucl. Acid Res. 10, 2997-3011 (1982)
(22) Stephens P E, Darlison M G, Lewis H M and Guest J R, Eur. J. Biochem. 133, 481-489 (1983)
(23) Stephens P E, Lewis H M, Darlison M G and Guest J R, Eur. J. Biochem. 135, 519-527 (1983)
(24) Carrell, R.W., Jeppsson, J.-O., Laurell, C.-B., Brennan, S.O., Owen, M.C., Vaughan, L., and Boswell, D.R., Nature 298, 329-334, 1982
(25) Woo S. L. C., Beattie W. G., Catterall J. F., Dugaiczyk A., Staden R., Brownlee G. G. and O'Malley B. W. Biochem. 20, 6437-6446 (1981)
(26) The UNIX machine is called a Torch Unicorn and is made by Torch computers, Aberley House, Shelford, Cambridge, U.K. The code for this 68000 based machine will be compatible with many others running under UNIX.
(27) The terminal is called a VT640 and the modification for the VT100 is made by Digital Engineering Inc., Sacramento, Calif., USA.
(28) Digital Equipment Corp., Maynard, Mass. USA.