Volume 10 Number 1 1982 Nucleic Acids Research

A convenient and adaptable package of DNA sequence analysis programs for microcomputers

James Pustell and Fotis C.Kafatos

Department of Cellular and Developmental Biology, The Biological Laboratories, Harvard University, 16 Divinity Avenue, Cambridge, MA 02138, USA

Received 16 October 1981

ABSTRACT

We describe a package of DNA data handling and analysis programs designed for microcomputers. The package is convenient for immediate use by persons with little or no computer experience, and has been optimized by trial in our group for a year. By typing a single command, the user enters a system which asks questions or gives instructions in English. The system will enter, alter, and manage sequence files or a restriction enzyme library. It generates the reverse complement, translates, calculates codon usage, finds restriction sites, finds homologies with various degrees of mismatch, and graphs amino acid composition or base frequencies. A number of options for data handling and printing can be used to produce figures for publication. The package will be available in ANSI Standard FORTRAN for use with virtually any FORTRAN compiler.

INTRODUCTION

A number of excellent computer programs to manage and analyze nucleic acid sequence data have been written in recent years (1-7). The most widely used programs, however, are designed for large mainframe computers. This is necessary for some of the more elaborate analyses, but not for the usual manipulations frequently needed by a modern sequencing operation. Putting programs for routine analysis on a mini- or microcomputer obviates the expense and inconvenience of frequently accessing a large computer, which may not even be available to some users. By encouraging immediate analysis of data it also can lead to more timely detection of errors. We have generated a convenient, highly interactive package of programs designed for users with no computer experience, used it for a year on our small (48K) microcomputer, and optimized it according to the experience of more than 15 users. Although somewhat less flexible and interactive than our package, a very nice set of minicomputer programs to meet similar needs has been prepared by R. Staden (2-5). However, the use of nonstandard FORTRAN, fixed logical unit numbers, and a sprinkling of PDP-11 system calls

has made it surprisingly difficult to adapt the Staden package to some other systems. To facilitate dissemination, we are converting our programs to entirely standard FORTRAN, using variables for logical unit numbers, and encapsulating all disk I/O (which is not at all standardized) into a very few, well defined subroutines. Our package also meets additional needs, including plotting of base compositions and amino acid sequences in a manner that highlights internal patterns; it has flexible formats which facilitate preparation of figures for publication directly from the microcomputer. The data files can also be transferred to mainframe computers and are immediately compatible with Queen-Korn (6) or SEQ (7) programs. Programs and documentation are available as hard copy, on floppy disk or tape, and can be sent at 300 or 1200 BAUD over telephone lines via modem.

INTERACTION

Routine programs should be usable by people with a variety of backgrounds, including those who know nothing about computers and may even be intimidated by them. Even experienced users may not understand a too brief instruction or may forget if they use a program only sporadically. For these reasons we have made considerable effort to aid interaction and to shift as much error detection as possible to the computer.

The programs are extensively documented with examples of the types of outputs and explanations of the questions the computer will ask during an actual run. By typing a single command, the user is presented with the "menu" shown in Figure 1. The options fall into two broad categories, data management and data analysis (see below). After you choose a program, the computer lists your options and possible responses at each point, and when asking for parameters it provides brief explanations or reminders whenever the question may not be obvious. If a program involves entering a large number of parameters (e.g. an oligonucleotide for a homology search, or a particular string of amino acids for graphing), the computer asks at the end of each run if you wish to reuse the same parameters on another sequence rather than re-enter them. Replies are checked by the program whenever possible; for example the user is warned if a file name is too long, if a parameter is not consistent with other parameters entered, if a character is not a base, etc. Other features specific to particular programs are explained below.

```
DNA Sequence Analysis Programs
J.Pustell--1981

All files to be analyzed must be on the disk in drive B

      0) Stop all programs
      1) Print the sequence directory
      2) Print a sequence file
      3) Translate a sequence
      4) Plot the base compostition of a sequence
      5) Plot the AA composition of a sequence
      6) Enter a new sequence
      7) Search a sequence for restriction sites
      8) Generate reversed complement of a sequence
      9) Search a sequence for homology with a subsequence
     10) Correct the sequence directory
     11) Correct a sequence file
     12) Print the restriction enzyme directory
     13) Enter or alter data in restriction enzyme directory
     14) Delete a sequence from the disk

Choose a program by entering a number and hitting 'RETURN'
```

Figure 1 "Menu" of choices presented by computer upon entering program system.

DATA MANAGEMENT

Nucleic acid sequences can be entered as RNA or DNA, and are stored in duplicate files. In the FORTRAN version the files use 60 character lines and terminate with a numeral, so as to be immediately compatible with either Queen-Korn (6) or SEQ (7) mainframe programs. However the package can use data in any format (see Compatibility). A separate file contains the sequence directory, including date of data entry, name of the person making each entry, and descriptive information which may be valuable in a lab where many people may need access to the sequences.

The data entry program asks for and records in the directory the necessary information, asks for and files the sequence while checking it for impermissible characters, and prints it at the end for proofreading. If errors are found (or if one wishes to alter the sequence at a later date), a correction program offers the options of 1) adding to the 5' end, 2) adding to the 3' end, 3) deleting a segment, 4) inserting a segment, or 5) replacing a segment. A note is automatically added to the directory that the sequence was altered and by whom; we find this a useful record-keeping provision, but in the FORTRAN version it can be disabled if inappropriate. Direct access to the directory file is also possible, permitting modification of descriptions or

renaming of sequences (in which case the sequence files themselves are also renamed automatically).

Appropriate commands permit printing of the entire directory, or printing of a sequence or a part thereof in a standard format (60 characters per line with every tenth character numbered). Another command will produce and store the reverse complement of a file. The greater capacity of the FORTRAN programs permits any sequence to be analyzed in both strands without requiring a separate file.

Restriction enzyme information is included in the data base. The relevant program provides a complete library, including name, recognition sequence, schizoisomers and any unusual features of each enzyme currently known. As more information becomes available, this library can be easily updated by the user through an interactive program.

DATA ANALYSIS

The restriction enzyme program will search any desired sequence for all the restriction enzymes in the library. For each positive enzyme, the recognition sequence, any schizoisomers or comments, and all cleavage sites within the sequence are listed. The enzymes which do not cleave the sequence can also be listed if desired. If the sequence contains any unknown bases an appropriate warning is issued as part of the output.

The conceptual translation program is flexible, to facilitate both analysis and the production of figures. Any segment of a sequence can be selected; segments to be printed and to be translated can be selected independently, permitting optional display of untranslated regions. Translation may be performed in any desired frame (or all three) and may begin at a specified base (e.g. for handling sequences with introns), or with the first in-phase ATG (the origin of translation is listed explicitly in the output). Translation can be stopped at the first termination codon encountered, or proceed to the end of the desired segment. Numbering of the sequence can be done from any point (e.g. from the first ATG encountered). A table of codon usage is also prepared. It only reflects the region translated, allowing various domains to be analyzed separately. From this table the amino acid composition of the encoded polypeptide can be obtained by simple addition. An example of the translation output is presented in Figure 2.

A related program displays the results of a conceptual translation graphically, by plotting each amino acid residue (in the single-letter code) in an appropriate vertical column (Fig. 3). We have found this program a

```
                    TRANSLATED SEQUENCE OF PC18

          -70          -60          -50          -40          -30          -20
C CCA TTC AGT GCA AGC ATG TCT ACC TTC GCT TTC TTA CTT CTC TGC GCC CAG GCT TGC CTG
                      Met Ser Thr Phe Ala Phe Leu Leu Leu Cys Ala Gln Ala Cys Leu

          -10            1           10           20           30           40
ATC CAA TCT GTG TAC AGC TAT GGC TGT GGT TGC GGC TGT GGT CTA GGT GGC TAC GGC GGT
Ile Gln Ser Val Tyr Ser Tyr Gly Cys Gly Cys Gly Cys Gly Leu Gly Gly Tyr Gly Gly

          50           60           70           80           90          100
CTC GGT TAC GGC GGT CTC GGT TAC GGA GGT CTT GGT TAT GAG GGT ACT GGA GCC TGT CTT
Leu Gly Tyr Gly Gly Leu Gly Tyr Gly Gly Leu Gly Tyr Glu Gly Thr Gly Ala Cys Leu

         110          120          130          140          150          160
GGA GAG TAC GGC GGC ACT GGC ATC GGT AAT GTT GCT GTA GCC GGT GAG CTG CCG GTC GCT
Gly Glu Tyr Gly Gly Thr Gly Ile Gly Asn Val Ala Val Ala Gly Glu Leu Pro Val Ala

         170          180          190          200          210          220
GGT AAA ACC GCT GTC GGT GGA CAG GTC CCT ATC ATC GGT GCT GTC GGT TTC GGC GGT ACC
Gly Lys Thr Ala Val Gly Gly Gln Val Pro Ile Ile Gly Ala Val Gly Phe Gly Gly Thr

         230          240          250          260          270          280
GCG GGT GCT GCT GGC TGT GTC TCC ATC GCC GGC AGA TGT GGA GGA TGT GGA TGC GGA TGC
Ala Gly Ala Ala Gly Cys Val Ser Ile Ala Gly Arg Cys Gly Gly Cys Gly Cys Gly Cys

         290          300          310          320          330          340
GGT CGC GGC ATT TAC TAA CCA CCA TAA TTG TGA CTT GAT TTT CAT TCG GAC AGA ACA TTA
Gly Arg Gly Ile Tyr ---

         350          360
AAT AAT AAA TAA GTG AGC AAC


Sequence scanned beginning with base no. 2
Sequence printed beginning with base no. 1
Sequence numbered beginning with base no. 80
AAs printed beginning with first in-phase ATG after base no. 2 (at base no. 17 )
Analysis stopped at termination codon (if found)


SUMMARY OF CODON USAGE IN PC18 AS TRANSLATED ABOVE
     Total number of codons = 120
     Number of unidentified codons= 0

  TTT Phe  0   .0%    TCT Ser  2  1.6%    TAT Tyr  2  1.6%    TGT Cys  6  5.0%
  TTC Phe  3  2.5%    TCC Ser  1   .8%    TAC Tyr  6  5.0%    TGC Cys  5  4.1%
  TTA Leu  1   .8%    TCA Ser  0   .0%    TAA ---  1   --     TGA ---  0   --
  TTG Leu  0   .0%    TCG Ser  0   .0%    TAG ---  0   --     TGG Trp  0   .0%

  CTT Leu  3  2.5%    CCT Pro  1   .8%    CAT His  0   .0%    CGT Arg  0   .0%
  CTC Leu  3  2.5%    CCC Pro  0   .0%    CAC His  0   .0%    CGC Arg  1   .8%
  CTA Leu  1   .8%    CCA Pro  0   .0%    CAA Gln  1   .8%    CGA Arg  0   .0%
  CTG Leu  2  1.6%    CCG Pro  1   .8%    CAG Gln  2  1.6%    CGG Arg  0   .0%

  ATT Ile  1   .8%    ACT Thr  2  1.6%    AAT Asn  1   .8%    AGT Ser  0   .0%
  ATC Ile  5  4.1%    ACC Thr  3  2.5%    AAC Asn  0   .0%    AGC Ser  1   .8%
  ATA Ile  0   .0%    ACA Thr  0   .0%    AAA Lys  1   .8%    AGA Arg  1   .8%
  ATG Met  1   .8%    ACG Thr  0   .0%    AAG Lys  0   .0%    AGG Arg  0   .0%

  GTT Val  1   .8%    GCT Ala  8  6.6%    GAT Asp  0   .0%    GGT Gly 19 15.8%
  GTC Val  5  4.1%    GCC Ala  4  3.3%    GAC Asp  0   .0%    GGC Gly 12 10.0%
  GTA Val  1   .8%    GCA Ala  0   .0%    GAA Glu  0   .0%    GGA Gly  8  6.6%
  GTG Val  1   .8%    GCG Ala  1   .8%    GAG Glu  3  2.5%    GGG Gly  0   .0%
```

Figure 2 Translation of an A. polyphemus chorion cDNA. Numbering was begun with the amino terminus of the mature protein. Printout gives all relevant parameters.

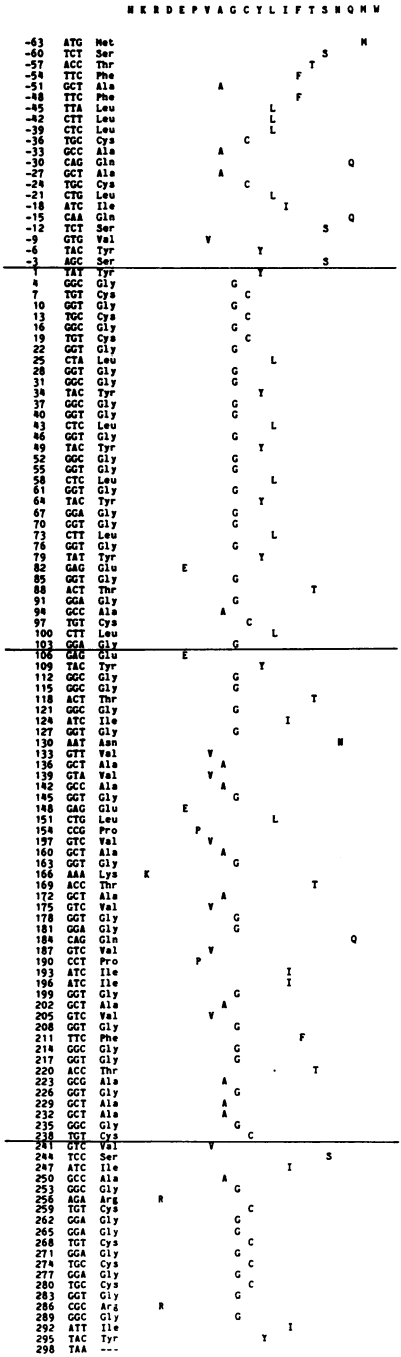

Figure 3 Amino acid plot of the same sequence as in Figure 2, using similar parameters. The plot has been divided manually by horizontal lines, delineating four sequential regions: signal peptide, amino-terminal domain (note repeats of CG and LGYGG or variants), central domain (note enrichment in V, A) and carboxy-terminal domain (note repeats of CG or variants). The computer legend for the plot is as follows:

AMINO ACID PLOT OF PC18

Sequence translated beginning with base no. 17
Bases numbered beginning with base no. 80
Stopped at termination codon (if found)

powerful means of identifying internal repeats within a sequence, or similarities between related sequences. Again, any segment of a sequence can be graphed and the numbering set relative to any base. Judicious choice of the sequential order for plotting different types of residue makes the output most informative; the appropriate order for any type of sequence can be selected by trial and error, and can be incorporated in a default string. In the FORTRAN version the default arrangement is encapsulated in a single BLOCK DATA statement and so can be easily adapted to the needs of any particular project. The order used in Figure 3 has been optimized for silkmoth chorion sequences. A different order can be selected from the console in any run (e.g. see Reference 8).

The base composition program produces a similar graph, except that the horizontal axis is now base composition and the sequence is displayed as single bases rather than triplets. The composition can refer to any base or combination of bases, and is plotted as a fraction or as a percentage. The scale can be set by the program (by taking into account the maximum and minimum values encountered in that sequence) or by the user (in which case out of scale values are indicated at the edge of the graph). The composition is determined for a preset number of bases flanking each position in the sequence; this "range" is set by the user, permitting various degrees of noise suppression and signal averaging.

The homology search program does not attempt to replace the large scale homology and dyad symmetry analysis contained in the mainframe programs, but is quite adequate for the most common type of search, namely for matches with incompletely defined subsequences such as promoters, splice junctions, characteristic features of a gene family, or sequences which can become a restriction site with a single base mutation. The program compares a sequence entered from the console with any sequence chosen from the files. The user declares what the minimum acceptable homology is, and may constrain any of the bases in the subsequence to be invariant. S/he also has the option of displaying all matches above the minimum acceptable, or only the best one.

COMPATIBILITY

Software compatibility is a major problem in the implementation of outside programs. High level languages such as PASCAL and FORTRAN are attempts to deal with the problem. However, both languages are frequently distributed with "enhancements" which are designed to make the language easier and more marketable. Unfortunately, enhancements frustrate the ability to run a

program on a different system without a major rewrite. The only solution is to stick religiously to the standard version and simply put up with the less flexible language this may entail.

For this reason, we are packaging our programs in ANSI Standard FORTRAN; thus, they should be compatible with all FORTRAN compilers unless they specifically state that they do not meet ANSI Standard (and then they may still run). An unenhanced PASCAL would be another good choice but, at present, FORTRAN is much more widely distributed.

Input-Output remains a problem, however. This is a function of the machines which, like railroads in the nineteenth century (and for the same purpose), are not standardized. Disk Reads and Writes are extremely variable and will normally require any user to write two corresponding subroutines to interface with our programs. These subroutines are very simple, however: they merely use the machine-specific disk call and move the file data into or out of an array. The array itself contains the file name, the drive to select, a logical unit number (more on these below), and space for any additional signals the user may want to add, plus the sequence or other data. Thus, with the file structures outlined previously, all programs can call the same Read or Write subroutine.

FORTRAN designates I/O devices by means of a logical unit number (e.g. 1 may equal "console", 2 may equal "printer", etc). These are not standardized either. To meet this problem we have used variables (x, y, etc) to replace logical unit numbers in the body of the programs. At the beginning of the program a statement exists, which permits numbers to be assigned to the variables for subsequent use throughout the program. This can be done in a matter of minutes by most text editors, and the programs are then permanently set for that particular machine. If the user changes machines, the programs can easily be reset again. Once the user has the disk Read and Write subroutines and knows how to change logical unit values, s/he can also implement any new programs or program updates readily. We hope that additional programs, designed for microcomputers and written by various workers in this field, will adhere to the conventions used here, maximizing dissemination and minimizing duplication of programming effort.

ACKNOWLEDGEMENTS

The earliest versions of some of the programs were written by Peter Moldave. The pc18 cDNA clone displayed in Figures 2 and 3 was sequenced by C. Weldon Jones. Our work has been supported by grants from NSF and NIH.

REFERENCES

1. Gingeras, T.R. and Roberst, R.J. (1980) Science 209, 1322-1328.
2. Staden, R. (1977) Nucl. Acids Res. 4, 4037-4051.
3. Staden, R. (1979) Nucl. Acids Res. 6, 2601-2610.
4. Staden, R. (1980) Nucl. Acids Res. 8, 817-825.
5. Staden, R. (1980) Nucl. Acids Res. 8, 3673-3694.
6. Queen, C.L. and Korn, L.J. (1980) Methods Enzymol. 65, 595.
7. The "SEQ" program, SUMEX System, Stanford University.
8. Wobus, U., Bäumlein, H., Panitz, R., Serfling, E. and Kafatos, F.C. (1980) Cell 22, 127-135.