DECUS
PROGRAM LIBRARY

DECUS NO.	8-557
TITLE	CLUSTR CLUSTER ANALYSIS PROGRAM
AUTHOR	Marjorie H. Kleinman
COMPANY	Center for Community Research New York, New York
DATE	June 19, 1972
SOURCE LANGUAGE	FORTRAN

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.

DECUS Program Library Write-up DECUS NO. 8-557

CLUSTR Cluster Analysis Program

This program is based on hierarchical grouping, as described in Fortran Programming for the Behavioral Sciences, by Donald J. Veldman, and is adapted from the program provided in that book.

The program works as follows: First all data are read in and stored in a 2-dimensional array on magnetic tape. If optioned, all scores are standardized to z-score form. Also, an option is provided for transposing the data matrix to permit grouping of variables instead of subjects. Next, all profiles are converted into a matrix of error potentials for all pairings of these profiles. The next section of the program locates the combination of objects involving the least increase in error. Each time through this part of the program, one group (of subjects or variables) is combined with one other group, until all have been absorbed into 1 of 2 groups.

Please note that each time, those groups are combined which show the most similar profile, between which there exists the least distance. Therefore, variables (or subjects) which show a high negative correlation will not be grouped together.

MINIMUM HARDWARE

PDP-8/I with 8K of core, with teletype, printer, and 1 magnetic tape drive or disk.

OTHER PROGRAMS NEEDED

PS/8 Operating System with Fortran Compiler

NOTE:

Intermediate date are stored either on disk or tape. The actual device is ASSIGNed at execution time.

This program consists of 4 chained segments, named CLUSTR, TRANSP, MATRX1, and MATRX2, respectively. Each must be compiled, loaded, and saved separately on the system device. Specify I and O options to the loader for each.
To allow ro execution-time formatting of input data, load CLUSTR into core and change the contents of location 14671 from 4725 to 0400, before saving on the system device. For an explanation of execution-time format statements, see the accompanying write-up.

How to run CLUSTR Cluster Analysis Program:

1) .ASSIGN (device) IN
2) .R CLUSTR (Starting Address is 14161)
3) Mount tape on unit 1 if it is device IN. It must contain the input file, if input is from tape. Oherwise, tape with zeroed directory is preferable.
4) Type in number of variables. Maximum is 80 if variables are to be clustered; otherwise, maximum is 200.
5) Type in number of subjects. Maximum is 80 if subjects are to be clustered; otherwise, maximum is 200.
6) Type in ISTAN, the option for standardization of data. (1 means yes, 0 means no)
7) Type in ITRAN, transposition option. If variables are to be clustered, type 1. Otherwise, type 0.
8) Type in a format statement. It may be 72 characters in length. Input is read in F format (no identifying information is read in).
9) Type in a heading to appear on output. It may be 72 characters in length.
10) Type input device (1,2,3, or 4). If input is magnetic tape, the file must reside on unit 1 with the file name 'CLUSIN.DA'.

Note: The 'Read' statement is at location 14664. If the card reader fails, STOP, Load Address 14664, and press START. Cards should continue to read in.

Execution-time Format Statements

Execution-time format statements are not permitted by the compiler. However, by changing one location in the program after it is in core, it is possible to read in a format statement (from the teletype only), and use it for reading the input. Set up the program (or subprogram) as follows:

```
      DIMENSION FMT (12)
C     This is the array where the format statement will
C     be stored.
      READ (1,15) FMT
15    FORMAT (12A6)
C     A format statement as long as 72 characters can
C     be read in here.
      .
      .
      .
20    READ (IN,25) DATA1,DATA2
C     Input data is read in from device IN
25    FORMAT (F3.0)
C     This is a dummy format that will not be used.
      .
      .
      .
      END
```

Be sure to obtain an assembly listing of the program, and also a map at execution time, so that the precise location to be changed can be determined.

Load the program and all subroutines, and begin execution. When the program pauses or waits to accept data, the appropriate location can be changed.

Now look at the assembly listing. Determine the actual address of the first word of array FMT. (See page C-4 of 8K SABR Assembler manual or PS/8 manual for computing actual addresses.) Now look through the listing for the Fortran statement where the reading of the data is done, in this case, statement 20. You will notice that in the assembled code will appear the following:

```
4Ø33  CALL  2, READ     (call to subroutine READ
                         with 2 arguments)
62Ø1  ARG   (n          (n is the device number)
DDDD                     (DDDD is the address where
                         the device number is stored.)
62Ø1  ARG   /fa         (fa is the format number, in
                         this case, 25)
FFFF                     (FFFF is the address of the
                         beginning of the format
                         statement.)
```

In our example, FFFF is the address of the first word of format statement 25. This is the location to be changed, so that it contains the address of the first word of array FMT. Therefore, compute the address where FFFF appears and change

its contents accordingly. (As a check on your computations, first compute the actual value FFFF, then load the address where FFFF appears, and see if the contents agree with your computation of FFFF. Then reload the address and deposit the address for the first word of FMT.)

Once this one location has been changed, save the core image on tape.

Altering Format Statements Already in Core

Format statements are stored 2 characters per word in packed, 6-bit ASCII code. Thus, the first half of the first word of a format statement is always a 50, the stripped ASCII code for open parenthesis. F2.0 takes 2 locations in storage, and appears as 0662 for F2, and 5660 for .Ø. Therefore, one can change all or any part of a format statement in core by determining its exact location and substituting the codes for the desired format. For any format statement that is likely to be changed in this manner, it is advisable to leave plenty of extra spaces when writing the program.