DECUS
PROGRAM LIBRARY

DECUS NO.	8-558
TITLE	CORREL CORRELATION PROGRAM and PCOMP-VARMX FACTOR 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.

CORREL Correlation Program

ABSTRACT

This program will compute Pearson product moment correlations on a matrix as large as 80 by 80 variables. Missing data are permitted. The program first computes means and standard deviations for all variables, and if there is any missing data, the appropriate means are substituted before correlations are computed. Output from this program can be used as input to PCOMP-VARIMAX factor analysis program (see accompanying write-up).

MINIMUM HARDWARE

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

OTHER PROGRAMS NEEDED

PS/8 Operating System with Fortran Compiler

RESTRICTIONS

Maximum no. of variables is 80. Maximum no. of subjects is 2048.

NOTE:

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

The program consists of 4 chained segments named CORREL, CALC, ECALC, and OUTC, 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 for execution-time formatting of input data, load CORREL and change location 3300 from 3334 to 2544, and then SAVE on the system device. For an explanation of execution time formatting, see the accompanying write-up.

The program works as follows: first the data are read in from the specified input device and stored on magnetic tape (unit 0) or disk. If input is from magnetic tape, the file must reside on unit 1 with the file name 'RAWDAT.DA'. Means and standard deviations are computed and printed out, along with the number of subjects for each variable. Next the program checks for any missing data, and substitutes the appropriates means. Next the program computes as much of the correlation matrix as possible, due to space limitations in core, and prints this out. Then the data are read in again from tape or disk, and the next portion of the matrix is computed and printed out, and so on, until all correlations have been computed. Correlations on 40 variables will require 2 passes of the data; on 80 variables, 8 passes are needed.

If so optioned, the program will also store the correlation matrix on tape, and will automatically call PCOMP-VARMX factor analysis program.

How to run CORREL correlation program:

. ASSIGN (device) IN

.R CORREL

After being loaded into core, the program will pause with a 1 in the accumulator. Ready the input data. Be sure to mount a scratch tape on DTA1 if that unit has been assigned as device IN or if a factor analysis is wanted. If input is <u>from</u> magnetic tape, the file must be on unit 0 with the file name 'RAWDAT.DA'. Press continue. The program will request a format statement. All data is read in F format (no identifying information is read in). Next appears a request to tape in a heading for the job. Both the heading and format may be as long as 72 characters. After this, the program will request no. of cases, no. of variables, the input device (1,2,3, or 4) and Factor Analysis option (1 or 0). Then the program will begin to read in the data. No further operator intervention should be required.

PCOMP-VARMX Factor Analysis Program

ABSTRACT

This program uses the principal components method of extracting roots and vectors, and then performs varimax rotation on the factor loading matrix. Input is in the form of a square correlation matrix, and can be read from any input device. Output from CORREL correlation program may be used directly as input.

MINIMUM HARDWARE

PDP-8/I with teletype, printer, 1 DF32 disk and 1 magnetic tape drive.

OTHER PROGRAMS NEEDED

PS/8 operating system with Fortran compiler

RESTRICTIONS

Maximum no. of variables is 80. Maximum no. of roots extracted is 12.

NOTE:

Intermediate data are stored on 2 devices, disk and tape. It is also possible to run this program if there are 2 tape drives and no disk by using the ASSIGN feature of PS/8.

This program consists of 5 chained segments named FAMATR, PCOMP, EIG1, EIG2, and VARMX, respectively. Each must be compiled, loaded, and saved separately on the system device. Specify I and O options to the loader for each.

PCOMP-Varmx Factor Analysis Program

This program is designed for maximum flexibility. It can either be called automatically by the correlation program, CORREL, or run independently. The Varimax portion can be called automatically or run independently, as well. Since a sizeable factor analysis on a small machine can take a long time, it is especially useful to be able to run such a job in 3 separate segments at 3 separate times.(If the program is called by CORREL, there is a special routine, FAMATR, which squares the triangular output matrix and stores it in proper form for PCOMP-VARMX.) Since a job can be run in separate segments, maximum use of the disk is possible. It can be assigned as a device at the execution time of any or all segments, depending on the size of the files involved. Also, its contents can be erased between segments once they are no longer of use.

The principal components portion of the program will extract up to 12 roots, one at a time. The maximum number of iterations for extracting each root is set by the operator, depending on the degree of accuracy required (maximum is 999). After the extraction of each root, a print-out of all factor loadings plus communalities appears. When a root with a value of less than 1.00 is found, the extraction process ceases. At this point VARMX will automatically be called. However, if fewer roots are wanted, one can return control to the monitor whenever sufficient roots have been extracted, and then call VARMX.All

factor loadings and communalities will have been saved on device OUT.

The Varimax routine will go through as many cycles of rotations as requested by the user. (A cycle is defined as one complete pass through all possible combinations of pairs of factor vectors.) After each cycle, the resulting normalized factor loadings are printed out. After all cycles are completed, the final unnormalized factor matrix, with communalities, appears.

How to run PCOMP-VARMX Factor Analysis Program:

1) ASSIGN (device) IN
   ASSIGN (device) OUT
2) .R PCOMP

Device IN contains the square correlation matrix, and device OUT contains the factor loading matrix. Since the file on device IN is read and re-written most frequently, it is preferable to assign a disk as device IN, if it is available and if there is sufficient room on it for this file. If only 1 DF32 disk is available, the maximum correlation matrix that can be stored on it is 49 by 49.

When PCOMP has been called, it will request input device and no. of variables. Type each on a separate line. Input must be a squared correlation matrix with the format (80A6) if input device is 4, and must reside on DTA1 with the file name 'CORMAT.DA'. If the input device is not 4, the format must be (12F6.4). Also, type in the maximum no. of iterations for extracting each root. The maximum is 999.

After the extraction of each root, the program will print out all factor loadings up to that point. The program will continue extracting roots until it has found one with a value less than 1.00, or until it has extracted 12 roots. VARMX, the varimax rotation routine, will automatically be called at this point. (However, if fewer roots are wanted, type CTRL C at the appropriate point, and control will return to the monitor. Then call VARMX.) Program VARMX will request the following information to be typed in, each on a separate line: Input device (usually 4), no. of

variables, no. of factors that have been extracted, and the no. of cycles of rotations desired. Usually, 4 cycles is sufficient. (A cycle consists of all possible pairs of factor vectors being rotated.) After each cycle the program will print out the normalized factor loadings. When all cycles are completed, the final loadings and communalities will be printed out.

How to run CORREL, then PCOMP-VARMX

1) .ASSIGN (device) IN

   .ASSIGN (device) OUT

2) .R CORREL (F. A. option is 1)

When the correlation matrix is completed, FAMATR, the matrix squaring program, will be called automatically, and will request that the no. of variables be typed in. When the correlation matrix has been squared, the program will automatically call PCOMP. Continue as described above.

Please note that if CORREL and PCOMP-VARMX are run together as one job, Device IN will contain the input file to CORREL as well as the square correlation matrix input to PCOMP-VARMX. If there are space limitations, the assignment of this device IN can be changed before calling FAMATR. Also, the input file to CORREL can be erased by calling PIP before calling FAMATR.

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.