Behavior Research Methods & Instrumentation
1981, Vol. 13 (5), 657-666

COMPUTER TECHNOLOGY

APL functions for interactive data analysis: Principal components analysis

SELBY EVANS, JERRY D. NEIDEFFER, and FRED H. GAGE
Texas Christian University, Fort Worth, Texas 76129

APL functions to support principal components analysis are presented: a general-purpose function to obtain eigen values and eigen vectors, a more specialized function to convert these into the results commonly given by principal components analysis, and a user interface function that accesses filed data, offers flexibility in data selection, and produces labeled output. A brief introduction to the logic and uses of principal components analysis is included. General-purpose support functions to simplify file use and to increase the range of options available to the user are also provided.

Principal components analysis is a valuable analytic tool for the exploratory analysis of multivariate data. It provides an efficient description of the dimensionality of data with multiple dependent variables. From this description, the investigator may be able to identify one or a few groups of variables that represent most of the variance in the entire set. This information may be used to reduce the number of variables under investigation by selecting or creating one variable to represent each group.

Principal components analysis also provides evidence on whether variables that purport to measure the same underlying phenomenon actually do so. If they do, analysis of the set should show all the variables correlating strongly with the first principal component. If two independent phenomena are represented, the second may be identified in variables correlated with the second principal component.

The value of principal components analysis as an exploratory tool is enhanced by use in the interactive mode. With data sets typical of laboratory research, the results are usually available in less than 3 min (on a moderately loaded Sigma-9). The user can execute several analyses, use them as a basis for decisions about subsequent analyses, and complete the subsequent analyses in a 1-h work session.

The availability of principal components analyses in an interactive mode is also useful for instruction or self-education. Students and other prospective users may acquaint themselves with the typical results of principal components analysis of their data by exploring the consequences of variations in the analyses: for example, deletion of variables, analysis of subsets of subjects, and replication of the analysis on randomly selected subsets.

Programming of principal components analysis is facilitated by the powerful matrix operations offered by APL. It can be further simplified by support functions that allow the program to focus on the central objective. In this paper several support functions are presented that serve that purpose.

For exploratory data analysis, the user needs to be able to move freely from one function to another in analyzing the same data set. This facility can be provided by adopting suitable conventions for data structures or files and designing support functions compatible with these conventions. The conventions we use are as follows: (1) Data are stored in files, each file component carrying a data table as a two-dimensional array. (2) Columns represent variables. (3) Rows represent subjects, or the equivalent dimension of the data over which summation is to be done in forming means, correlations, and the like. (4) Tables represent collections of subjects likely to be analyzed as a unit: experimental groups or types of subjects. (5) Data subsets are distinguished by integers in one or more columns, Column 2 being used for the most important subset. (6) File conventions require that file components be numbered by consecutive integers starting with 1.

Data files conforming to these conventions can be prepared by forming tables with TABLE (Evans, Gage, & Neideffer, 1980) and appending them to the file. The data file can be labeled with the LABEL option in GRAPHICS (Evans, Neideffer, & Gage, 1980). (A report now in preparation will provide functions for entering a set of raw data, correcting it, organizing it into a file containing one or more tables, and labeling the file.)

SUPPORT FUNCTIONS

Two groups of support functions are described in this section. APL codes for the functions are presented in Figure 1. The first two, PARAM and SRD, are designed to support data analysis functions such as the one given in this report. The others are general-purpose file func-

0005-7878/81/050657-10$01.25/0

```
0   PARAM
1   DIMS←25 50⋄CTR←0.24⋄NVL←DAFL←SETE←0⍴SETC←2
2   OPFILE←'TEMPO'⋄OWD←80⋄TYPE←1
    ∇
0   R←SRD F
1   F←,(R←FREAD 1,F)[;SETC]∈SETE←,SETE
2   SRN←' N=',,'I5'ΔFMT 1↑⍴R←(F∨0=⍴SETE)⌿R
    ∇
0   I←FKEEP X;E
1   FDROP X[1],I←X[2]-(FLAST X[1])
    ∇
0   L←FEXIST N
1   L←∨/(0 11↑FLIB⍳29)∧.=11↑N
    ∇
0   L←F FALTIE N
1   →ND×⍳0≠⍴L←(∨/F∈' .')/'FSTIE'
2   (0≠⍴L←(~FEXIST F)/'FCREATE')/'CREATING ',F
3   ND:∈'F ',(L,(0=⍴L)/'FTIE'),' N'
4   L←FLAST N
5   ⍝MONADIC ∈ IS THE EXECUTE FUNCTION IN SIGMA-9 APL
    ∇
0   L←F FATIE N;P;Q;Y;R
1   FUNTIE(N∈FNUMS)/N
2   →0×⍳0=⍴F←(~Q∧1⌽Q←F='/')/F,L←⍳0
3   BT:P←(¯1+Q←F⍳'/')↑F
4   L←L,P FALTIE 1↑N
5   N←1↓N
6   →BT×⍳2≤⍴F←Q↓F
    ∇
```

Figure 1. APL code for support functions described in the text. The function FLAST is described by Evans, Neideffer, and Gage (1980). The other functions are available in most systems that have an APL file capability. The names of these functions vary slightly across systems.

tions[1] that support any file handling with APL. They take care of routine operations with less user attention and less opportunity for error than is typical of the standard file functions. They are designed to be used with the file convention (Convention 6) given in the introduction. If components are routinely appended and dropped only by FKEEP (see below), files will adhere to this convention.

PARAM

PARAM specifies initial values for parameters used by functions in this report and others in this series. It would be used to initialize a library copy of a work space or to reset parameters in an active work space after they have been changed by the user. Here, PARAM also serves to identify and give standard values for parameters and their effects in one place.

DIMS controls the shape of the array used in plotting. The standard value produces 25 rows and 50 columns. CTR determines the minimum magnitude of correlations to be displayed. SETE carries integers to select subsets of the data, as described in connection with SRD below. Its standard or default value is the empty vector, which turns off set selection. SETC specifies the column to be used in selecting elements given in SETE; Column 2 is selected by the standard setting. OPFILE carries the name of the file to receive output. Other variables specified in PARAM are included because they are used by functions to be presented in future reports.

R ← SRD F

SRD provides for the selection of subsets of the rows of a data table (assumed to represent subjects). From the user's standpoint, the selection is accomplished by specifying the global variable SETE to contain one or more positive integers designating the subset. To support this selection, data tables must have at least one column with integers representing set membership. For example, integers in Column 2 of the data table might identify experimental or demographic groups. In order to analyze a particular group, the user would assign to SETE the value of the integer identifying that group. To analyze several groups as a single subset, the user would assign to SETE a list of integers, each identifying one of the groups.

The column to be used for subset selection can be determined by specifying the global variable SETC to contain the number of the column to be used. The standard setting provided by PARAM causes set selection based on Column 2.

SRD is useful when several data-analytic functions are to be included in the same system. It avoids redundant code, with consequent saving of programmer effort and available work space. From the user's standpoint, it

insures that all functions follow the same conventions. Indeed, one setting of SETE and SETC in a work space controls all functions executed in that work space, provided they all use SRD.

From the programmer's standpoint, SRD replaces the standard file read function, FREAD. The right argument, F, carries only the number of the component to be read, not the file tie number, because SRD is written under the assumption that the primary data file is tied to the number 1. SRD delivers in R the indicated data table. If set selection is operating, any rows that do not belong in the set are deleted before the result is delivered.

SRD also leaves a global variable, SRN, which is a character vector indicating the number of rows retained. If appropriate, this vector may be included in the analytic output to document the number of cases. Adequate documentation of the analytic procedures also requires identifying the set column and the SETE values used to select the subset, as illustrated in the user interface program described later in this report. Subsequent reports will present support functions that routinely provide this documentation.

L ← FEXIST N

FEXIST determines whether a file with the name given in the right argument, N, exists in the user's account. The result, L, is a 1 if the file exists and a 0 otherwise.

The code for this function is provided to define the function in terms of the standard file functions and to insure that it can be produced on any system with standard file functions. The method is not particularly efficient, primarily because of the time required for FLIB to get a list of files in the user's library. Note that the first 11 columns are dropped from the result of FLIB and that a take operation is used to adjust N to have 11 elements, with trailing blanks as needed. These steps conform to Sigma-9 APL characteristics in which the first 11 columns of the result of FLIB carry the account number and the remaining 11 columns carry the file name. Other systems may differ somewhat in the location and length of the file names. The operations in FEXIST would have to be adjusted accordingly.

Some systems have a FEXIST function already available. Sigma-9 users will find one in the work space FILAIDS in the APL1 account. Users whose systems do not provide FEXIST may prefer to write their own in terms of primitive file functions. For example, on some systems, the standard FTIE will generate an error message that can be used to determine that the file does not exist.

C ← F FALTIE N

FALTIE is a general-purpose file-tying function that relieves the user and the programmer of attention to several frequently occurring details. It ties the file named in F to the number given in N. If the file does not exist, FALTIE creates it and reports that action. If the file name refers to an account number, FALTIE makes a shared (read-only) tie, as would normally be required to access a file in another account. FALTIE delivers in C the number of the last component in the file. In conjunction with conventions described earlier, this number indicates the number of components in the file and is often needed to support iteration over the file or selection of components at the end of the file.

In Sigma-9 APL, the monadic epsilon causes the text vector in its right argument to be executed as an APL expression. FALTIE forms the appropriate text vector on the basis of logical tests and then executes it (Line 3). Some other systems (such as APLSV and those patterned after it) use a different symbol for the execute function, which would have to be substituted for the epsilon. A few older systems (such as APL/360) may not offer the execute function. APL programmers will be able to determine whether the function is available and which symbol is used by consulting their local system manual. If the function is not available, programmers should have no difficulty rewriting the function to use logical tests to branch around unneeded operations, just as FALTIE eliminates them with a logical compress. FALTIE recognizes a period or a blank space as indicating the presence of an account number in F. These are commonly used separators between file names and account numbers. If some other separator were used on a particular system, Line 1 of FALTIE would need to be modified to include that separator in the test.

Some users may prefer to write FALTIE in terms of the primitive file functions; if this is done, FTIE, FSTIE, and FCREATE can be eliminated from the file functions in the work space.

L ← F FATIE N

FATIE serves as a general-purpose file-tying function that augments the capabilities of FALTIE with provisions for automatically untying a set of files and for making multiple ties at one time. With FATIE, virtually all file-tying operations are handled with a single line of code. FATIE first unties all the files tied to the numbers listed in the right argument, N. It then ties the file names listed in F (separated by a slash) to the numbers given in N. Names and numbers are taken in the same order. The result, in L, is the number of the last component in each file, in order matching the names. Programmers may prefer to combine FALTIE and FATIE into a single function. The functions are given separately here to simplify programming and testing.

A useful convention in programming with FATIE is to give the integers from 1 to 9 (iota 9) as the right argument when a function first calls FATIE. With this convention, the first nine integers are reserved for use by functions and are always untied before the function starts to make its own ties. By using tie numbers greater than nine, users can establish ties that are protected from interference by functions.

D ← FKEEP F

FKEEP deletes components from a file with more convenience and safety than is offered by the standard file functions. The right argument, F, is a two-element vector carrying, in order, the file tie number and the number of components to be kept. Components with numbers greater than the second value in F are dropped. The file can be emptied by indicating that zero components are to be kept.

FKEEP makes simple the most frequently needed operations in dropping file components: clearing a file completely and dropping the most recently appended components. Since the function will only drop from the high-numbered end of the file, it avoids the risk posed by more general functions that the user will inadvertently drop components from the wrong end of the file. FKEEP relies upon and helps to enforce the file convention described in the introduction.

PRINCIPAL COMPONENTS ANALYSIS

In this section, we provide a brief introduction to the logic and terminology of principal components analysis. The discussion is intended to help the programmer understand what the functions do and to give the prospective user an intuitive picture of what the analysis is designed to accomplish. Users who want further background should refer to texts in multivariate statistics, such as Green (1978) or Tatsuoka (1971). The presentation here may help prospective users to determine what principal components analysis has to offer in their research.

Consider a set of data representing two intercorrelated variables. If these are expressed as standardized variables and plotted, the result is an ellipse centered at the origin and slanted with respect to the original axes. An ellipse has its own, self-defined frame of reference, its major and minor axes. We could lay down transparent graph paper with the axes over the axes of the ellipse. If we read off the locations of the data points in reference to the new axes, we would have a new pair of variables describing the data points. These variables might be called "factor" scores or component variates. (The term "variate" is used here to maintain a distinction between the original variables and those more abstract variates that are derived as composites of the original variables.)

Clearly, these variates, the result of a rigid rotation of the axes, would contain the same information as did the original variables. In fact, we could recover the values on the original variables by reversing the rotation.

The component variates would express the original information in a different, possibly more useful, form. The two variates would be essentially uncorrelated. Since the information in the original variables is fully conserved (can be recovered by inverse rotation), one might reasonably ask what happened to the correlation. Considering the ellipse in terms of its own axes shows that the correlation has been converted into a discrepancy between the variances on the major and minor axes. The greater the correlation is, the greater the discrepancy. Two correlated variables of equal variance are replaced by two uncorrelated variates of unequal variance. The variance common to the two original variables has been collected or aggregated into one variate associated with the major axis of the ellipse.

In this illustration, the new variances could be computed in the usual way. A principal components analysis would also find the variances, but in a different way. The variances would be the eigen values of the correlation matrix.

If the variance on the minor axis were relatively small, we might decide to discard that variance and describe the data points entirely in terms of the locations on the major axis, that is, in terms of the first principal component. In that case, we could not completely reconstruct the original data. The retained component would, however, reconstruct the approximate location of the data points on each original axis. The reconstruction would locate each data point within a small region of uncertainty, analogous to a confidence interval.

Since much of the variance on the minor axis would probably be error variance, a researcher might be pleased to discard it in exchange for a reduction in the number of dimensions used to describe the data. Discarding data in this way is quite comparable to replacing repeated observations with their average. Indeed, averaging such data could be regarded as creating the first principal component by assuming equal weights (and equal variances) for all variables.

The logic for the two-dimensional case can be extended to higher dimensionality. In a three-dimensional case, the resulting ellipsoid would look something like a loaf of French bread. The appropriate axes would again be intuitively obvious. But without three-dimensional graph paper, we would find it difficult to read off the coordinates of the points. Fortunately, if we know the angles of rotation, the new coordinates can be computed directly from the old ones.

Each new coordinate is found by multiplying the three original coordinates by appropriate weights (trigonometric functions of the angles of rotation) and adding the result. Thus, for determining the location of any point on the major axis of the ellipsoid, there is a vector, a set of three weights. Multiplying the point's original coordinates by the weights and summing gives the location, or projection, of the point on the major axis. The operation is conveniently represented in vector notation; it is the inner product of that point's original coordinates and the weights.

Such a vector of weights could be determined in various ways. It could be arbitrarily formed by setting each weight to 1/3. In that case, the three scores for each subject would be replaced by their average. The result would not be the first principal component, of

course, although it might be a fair approximation with strongly intercorrelated variables.

So far, we have discussed geometric rotations of standard variables plotted in space. We have also described an equivalent algebraic rotation applied to the original coordinates, that is, to the normalized data matrix. The geometric discussion although intuitively simpler, has run out of dimensions. The algebraic rotation can be extended to as many dimensions (variables) as are offered by the data. We need only to introduce a new procedure, more general than French bread, for finding the vectors of weights to accomplish the rotation. Technically, these are the eigen vectors of normalized data matrix.

Finding the eigen vectors is most easily done with the symmetric correlation matrix (the variance-covariance matrix for the normalized data). Fortunately, its eigen vectors are the same as those of the normalized data matrix. Its eigen values, although different from those of the data matrix, represent the variances on the component axes.

The main task of principal components analysis is to find the eigen values and eigen vectors of the correlation matrix. This information is then combined to produce results such as the correlations of the original variables with the component variates, the proportion of variance accounted for by each component, and perhaps the new variates themselves. In the final output, the components are ordered so that the component accounting for the most variance is first, the component accounting for the second largest amount of variance is second, and so forth. The components account for distinct, additive portions of the variance in the original data; the variates representing the components are uncorrelated.

The most commonly reported output of a principal components analysis is the table of correlations between the original variables and the component variates. These correlations are sometimes called "loadings," but nothing in the present context suggests a need for calling correlations by another name. Inspection of these correlations is useful in determining what variables are best represented by each component.

Principal components analysis is often treated in association with factor analysis. Actually, it represents a different model, and its use in factor analysis is open to question (Lee & Comrey, 1979). We use principal components analysis as a data reduction technique and associate it with discriminant function analysis, multiple regression, and canonical correlation.

When principal components analysis is used as a putative factor-analytic technique, the retained components are usually subjected to secondary rotation by a program such as varimax (Kaiser, 1958). The rationale is that the results of principal components analysis are difficult to interpret, whereas rotation produces more interpretable results. When principal components analysis is used to explore and describe the dimensionality of a data set, the objectives are quite different from those of factor analysis. The characteristics of the data set and the kind of interpretation required may be quite different from the experience of factor analysis. The need for a secondary rotation from the principal components solution has to be evaluated in the context of these differences.

Rotating from a principal components solution poses problems that one might prefer to avoid. Not only is the choice of rotation procedure open to dispute, but so, also, is the number of components to rotate. (See the discussions by Green, 1978, and by Lee & Comrey, 1979.) Two researchers using the same data and the same analytic procedure could produce substantially different results by differing in their criteria for how many variates to retain. The unrotated principal components solution is insensitive to such a choice, because the deletion of components does not affect those retained. Since the comparison of principal components solutions results poses fewer uncertainties, it is probably good practice, even if secondary rotation is deemed necessary, to publish the principal components solution as well.

We have a varimax function in our system and have found that secondary rotation is sometimes useful in working with questionnaire data. With laboratory data, however, we find that the results of a principal components analysis are usually able to support such interpretation as may be needed in that research domain. In interpreting a principal components analysis, users should keep in mind that the results are determined by the variables that went into the analysis. A different mix of variables could substantially change the result.

Although principal components analysis has not been widely used with laboratory data, there are studies to illustrate its application and results. Gage (1978) and Gage and Lieberman (1978) used it to identify empirical variables that would be good candidates for indexes of social dominance. These studies started by selecting sets of empirical variables commonly used in the literature as measures of social dominance. The variables were observed concurrently in social dominance paradigms. The resulting data were subjected to a principal components analysis. In both studies, the analyses identified several empirical variables that correlated strongly with the first principal component and thus could be treated as measuring the same underlying phenomenon. Both studies included replications indicating that the selection of variables on the basis of the principal components analysis could be replicated.

The analyses also identified commonly used measures of dominance that did not correlate well with the first principal component. Such measures could not safely be treated as measuring the phenomenon measured by the first principal component; indeed, such measures could lead to quite different conclusions.

A methodological paper on the psychophysics of image quality (Evans & Attaya, 1978) illustrates the use of principal components analysis to reduce the dimensionality of both physical and subjective variables.

Other studies illustrate the application of principal components analysis in the definition of aggressive behavior (Crabtree & Moyer, 1973), in the analysis of operant data (Gage et al., 1979), and in data representing the consequences and treatment of brain damage (Gage et al., 1980).

In laboratory research with good control over extraneous sources of variance, the typical correlations between related measures should be much greater than in field research. When the research is planned on the basis of good prior knowledge about the relationships between measures and laboratory manipulations, the strength and interpretability of the principal components analysis is likely to be enhanced. We find that it is not unusual to obtain correlations of .75 or greater in laboratory studies of constructs for which related variables have been suggested from previous research.

FUNCTIONS FOR PRINCIPAL COMPONENTS ANALYSIS

Two functions, EIG and PRN, provide the basic results of a principal components analysis. APL code for these functions is given in Figure 2.

E ← EIG R

EIG computes the eigen values and eigen vectors of a real symmetric matrix, R. The result, E, carries the eigen values as the first row and the eigen vectors as the columns with the first row removed. In other words, Column 1 of E carries the first eigen value as its first element and the corresponding eigen vector as the remaining elements. The elements of the eigen vectors correspond to the original variables in the order represented in the correlation matrix. The eigen values are not ordered by magnitude as they emerge from EIG. When EIG is used as the basis of a principal components analysis, it is up to the calling function to arrange the eigen values (which are the variances of the components) in order of decreasing magnitude. The same rearrangement, of course, must be applied to the eigen vectors.

The algorithm for EIG is Jacobi's method, described by Hemmerly (1967), among others. It finds all of the eigen values and eigen vectors by an iterative procedure that progressively rotates the correlation matrix into a matrix containing the eigen values in the diagonal and near-zero elements elsewhere. How near the off-diagonal elements are to zero is determined by the criterion in Line 4. As given in the function, it is .00001. This is probably more stringent than necessary, considering the typical precision in correlation data.

The progressive rotation is accomplished by finding at each iteration a rotation that will reduce to zero the largest off-diagonal element of the matrix. That rotation is applied to the matrix and is accumulated as one step in a successive matrix product in the matrix Q, which is initialized as an identity matrix. (The rotation is actually applied only to the rows and columns that include the element to be reduced to zero. Since the matrix is symmetric about the diagonal, the element appears twice.) When the function completes its work, the cumulated product in Q is the matrix of eigen vectors. The rotated correlation matrix contains the eigen values in its diagonal. The function thus finds matrices that satisfy the relation given in Line 17 of EIG.

Reducing an element to zero does not permanently eliminate it. It may become large again as a result of later rotations. For some matrices (not typical correlation matrices), the process will not converge on the criterion. In that case, EIG will carry the iterations to 300, or to 1.25 times the number of distinct off-diagonal entries. It will then terminate with the notice "NOT MEETING CRITERION." Failure to meet criterion may indicate the violation of certain technical assumptions required for the analytic procedure.

These assumptions will normally be met in statistical work if two precautions are taken. First, one should avoid analyzing a set of variables in which one variable is completely predictable from some combination of the others. The most common source of this problem is the use of averages or sums of other variables in the analysis. Less commonly, the problem arises when the variables are required to have a fixed sum across each subject, as with proportions. Second, one should avoid analyzing a correlation matrix unless the number of subjects is greater than the number of variables in the analysis. Preferably, the number of subjects will be two or three times the number of variables. The best precaution of all, of course, is to recognize that results are tentative until replicated.

Since the Jacobi method finds all eigen values and eigen vectors, it requires more computer time than does the method described by Cooley and Lohnes (1971). It is simple to program, however, and computer time is not usually a scarce commodity in an APL environment. Computation times are discussed in connection with the illustration given later in this report.

Comment lines at the bottom of EIG are given to assist the programmer in determining that the function is operating correctly and to provide an indication of the technical meaning of eigen values and eigen vectors. Line 16 specifies a condition that is commonly given as the definition for eigen values and eigen vectors. Line 17 states that the eigen vectors convert the correlation matrix into the diagonal matrix of eigen values. Line 18 states that the eigen values and eigen vectors reconstruct the correlation matrix. All three lines represent mathematical equalities restated to reflect the fact that the obtained values are approximations to some level of precision rather than true equalities. Thus they say, not that the values are equal, but that the values differ in magnitude by less than some small amount.

Many APL systems have functions similar to EIG somewhere in the public libraries. For programmers who

```
 0    R←EIG M;N;Q;I;T;K;L;C;S;U;V;LIT
 1    I←ρρM←,M∘.<M←⍳N←⌈/ρQ←Qρ1,(N←1↑Q←ρR←M)ρ0
 2    LIT←300⌈⌈1.25×N×0.5×N-1
 3    ST:L←L+N×0=L←N|C←(M\C=T←⌈/C←|M/,R)⍳1
 4    →OT×⍳1E¯5>T
 5    T←-R[K←⌈C÷N;L]
 6    S←2×V←(+/(T,U←0.5×R[K;K]-R[L;L])*2)*0.5
 7    S←(T÷S×(C←((V+|U)÷S)*0.5))×1-2×U<0
 8    →OT×⍳∨/1=|2ρU←2 2ρU,S,C,U←C,-S
 9    R[V;]←U+.×R[V←K,L;]
10    R[;V]←R[;V]+.×U←⌹U
11    Q[;V]←Q[;V]+.×U
12    →ST×⍳0=ρ⎕←(LIT≤I←I+1)/'NOT MEETING CRITERION'
13    OT:I,0ρR←(1 1⌹R),[1]Q;' ITERATIONS'
14    ⍝LET M BE THE ORIGINAL MATRIX
15    ⍝LET L←(T∘.=T←⍳1↓ρR)×(ρQ)ρR[1;]
16    ⍝.0001>|(M+.×Q)-Q+.×L
17    ⍝.0001>|L-(⌹Q)+.×M+.×Q
18    ⍝.0001>|M-Q+.×L+.×⌹Q
      ∇
```

OUTPUT OF EIG

```
 2.76445   1.85833   0.00002   0.02189   0.35530
 0.55207   0.02270   0.31647   0.40523   0.65600
 0.08205   0.70679  ¯0.41442  ¯0.42897   0.37141
¯0.45073   0.48568   0.74893  ¯0.00552   0.00463
¯0.58679   0.07265  ¯0.39732   0.64022   0.28749
¯0.37551  ¯0.50869   0.09661  ¯0.49178   0.59080
```

```
 0    Q←S PRN R;D;W
 1    W←¯1+(D[R←⍒D←(Q←EIG R)[1;]]<1↑S←S,1)⍳1
 2    D←¯1↓1,S[2]>,1 0↓PV←D,[0.5]+\D←(W←D[R←W↑R])÷+/D
 3    PV←D/PV
 4    Q←Q×(ρQ←(1 0↓Q)[;D/R])ρ÷W*0.5
      ∇
```

Figure 2. APL code for EIG and PRN, basic functions for principal components analysis. An example of the output of EIG, which would be delivered to PRN in Line 1, is also given. See text for identification of the original data and final results associated with this example.

want to use one of these functions, the notes at the bottom of EIG may be useful to verify that the function is working correctly and that the output is being correctly interpreted. Future versions of APL may include an eigen-vector function as a primitive (Iverson, 1980), just as APL presently includes matrix inversion.

F ← T PRN R

PRN produces a principal components analysis of the correlation matrix R. T is a control parameter specifying the smallest eigen value to be retained in the solution. Optionally, T may carry a second element specifying the maximum proportion of variance to be retained in the solution. In honoring this request, PRN will take just enough eigen values to exceed the specified proportion of variance. In combining the two criteria, PRN takes the result that leads to the smaller number of components.

The result, F, is a matrix carrying the multiplicative weights to rotate the normalized data matrix and so form the principal components. The role of these weights is as described in the earlier section on principal components analysis. The rows of F correspond to the original variables in R. Each column of F contains the weights to form one principal component. The component variates can be formed by postmultiplying the normalized data matrix by F. A global variable, PV, is also formed; it carries as its first row the proportion of variance accounted for by each component; the second row carries the cumulative proportion of variance. The columns of F and PV are arranged so that the proportion of variance accounted for by successive components is in descending order.

The weights delivered by PRN are adjusted to produce standardized variates, that is, variates of unit variance. The adjustment is made in Line 4, in which the vectors are divided by the square roots of the corresponding eigen values. The adjustment simplifies subsequent use of the variates by placing them on the same standard scale. Programmers should note this adjustment, however, since they might (from the previous discussion) expect the component variances to be equal to the eigen values.

To aid in developing these functions, an example of

the output of EIG is given at the bottom of Figure 2. The input to EIG was the intercorrelation matrix from the first five variables produced by CORGEN (Evans et al., 1980). This output of EIG was delivered to PRN, which delivered its output to PRINAN (below) to produce the final results illustrated later in this report.

PRINAN

PRINAN is a user interface function to provide principal components analysis. It permits the selection of a table from a labeled file and the selection of data columns for analysis. With appropriate setting of parameters, it also permits the selection of subsets of rows in the data table. PRINAN produces as output the correlation between the original variables and the retained components. Additional rows give the proportion of variance in the correlation matrix accounted for by each component and the cumulative proportion of variance accounted for by each component and its predecessors. An additional column gives the proportion of variance in each original variable accounted for by the retained components. APL code for PRINAN is given in Figure 3.

If at least two components are retained, PRINAN produces a plot of the correlations between the original variables and the first two components. It then offers to make the variates. If the answer is yes, PRINAN makes a new file containing the data table just processed, with component variates as additional columns. (If the table is reduced by set selection, only the rows that were processed are transferred. All columns are transferred.) Finally, PRINAN labels the file as described in the report on GRAPHICS (Evans, Neideffer, & Gage, 1980), permitting the user to go immediately to graphic displays or to other functions that require labeled files. Examination of plots relating original variables to the component variates is frequently informative.

PRINAN requires a labeled file following conventions described in the introduction. Unlike GRAPHICS, PRINAN is not written to provide labeling. It merely detects the need, reports it, and terminates. Programmers can replace this part of the PRINAN code with the code used in GRAPHICS if they want to provide automatic labeling. A report now in preparation will present a function to provide routine labeling as part of the process of building a new data file.

A commonly suggested criterion for the smallest eigen value to retain is unity. This value is the most generous that could be permitted by considerations of parsimony, since it means that a component will be retained if it accounts for as much variance as is represented by one of the original variables. Evidence reviewed by Green (1978) suggests that this criterion may be too generous.

We prefer to be more conservative, using 1.3 in the function illustrated here. This issue is probably academic, however, since the other limit, 75% of the variance, usually determines the number of components. Such a limit, as noted by Green (1978), may be justified on the grounds that the remaining variance, even if reliable, may not be important enough to consider. Our choice

```
 0   PRINAN;L;HD;STR;T;ATBN;CLN;TAB;CL;SCL;N;C;P;R;PV;F;SD;VR;MX
 1   'WHAT FILE:'
 2   →0×⍳0≠ρ⎕←(' '≠1ρC←FREAD 1,L←1↑((DAFL←⍞),'/TEMPO')FATIE⍳9)/'LABEL!'
 3   TB←DAFL,FREAD 1,L-2
 4   'TABLES:  ',,ATBN←(⍕0,T←(∧/' '=C)⍳1)↓C;Δ,'TABLE NUMBER:'
 5   TB←TB,Δ,,ATBN[TAB←(1↑ρATBN)⌊1ρ⎕;]
 6   R←(1↑ρCLN←C[⍳T-1;])ρ7↑Δ
 7   'COLUMNS: ',,R,CLN;Δ,' LIST COLUMNS FOR ANALYSIS:'
 8   C←CLN[CL←⍎⍞;],[1]2 ¯8↑2 3ρ'PVRCPV'
 9   STR←(0≠ρ,SETE)/('SETS ',,'I4'ΔFMT SETE),', FROM ',,CLN[SETC;]
10   R←R+.×F←1.3 0.75 PRN R←1 MVST(SRD TAB)[;CL]
11   N[T/⍳ρT←' '=N←,'BI8'ΔFMT 0,-⍳1↓ρF]←'P'
12   P←'BF8.2'ΔFMT((R×CTR<|R),+/R*2),[1]PV,0
13   BOP'PCA',SRN,,Δ,(N,¯8↑'PVR'),[1]C,P
14   →ND×⍳2>1↓ρPV⍝R←.×⌹R RECONSTRUCTS THE CORRELATION MATRIX
15   SCL←⌹2 2ρ¯1 1
16   BOP(¯8↑'F2'),(DIMS SPLOT R[;⍳2]),¯30↑'F1'
17   ND:→0×⍳'Y'≠1↑⍞,⎕←'MAKE COMPONENT VARIATES?'
18   (⎕←'PCA',3ρDAFL)FATIE 3
19   FKEEP 3 0
20   (R,(0 MVST(R←SRD TAB)[;CL])+.×F)FAPE 3
21   (DAFL,'PCA',STR,SRN)FAPE 3
22   ((FREAD 1,L-1),(¯1+⌊⍳F),[0.5]1+⌈⍳F)FAPE 3
23   N←CLN,[1](⍕8,1↓ρF)ρ8↑N
24   (N,[1]' ',[1]ATBN[TAB;])FAPE 3
     ∇
```

Figure 3. APL code for PRINAN, a user interface function for principal components analysis. The formatting function, delta format, is probably available in the APL library of most systems. The form of the name may vary slightly from system to system.

of 75% is based on observations that suggest that the data we investigate typically contain about 60% to 75% reliable variance. Other users might prefer other proportions, depending on estimates of reliable variance in their data.

The last column of PRINAN's output gives the proportion of variance in each variable accounted for by the retained components. Since the component variates are uncorrelated, this value is simply the sum of the squared correlations with that variable. This column is useful in detecting original variables that are not well represented by the retained components.

PRINAN uses the following functions described in previous reports (Evans, Gage, & Neideffer, 1980; Evans, Neideffer, & Gage, 1980): MVST, BOP, FAPE, SPLOT. Users may wish to display the correlation matrix computed in preparation for the principal components analysis. The labeling conventions of PRINAN are the same as those of CORRELATE (Evans, Gage, & Neideffer, 1980). Thus the code for that function can be used as a guide for inserting code to prepare an output form of the correlation matrix.

The output of PRINAN is illustrated in Figure 4. The data were the first five columns of the array produced by CORGEN (Evans, Gage, & Neideffer, 1980). Only five columns were used because the deterministic process of CORGEN leads to a correlation matrix that does not meet the requirements for principal components analysis when all the columns are used.

If the analysis in Figure 4 represented a real study,

```
      PARAM
      SETE←1
      PRINAN
WHAT FILE:
PCADEMO
CREATING TEMPO
TABLES:    T1
TABLE NUMBER:
□:
      1
COLUMNS:
    ITM:      SET:       V1        V2        V3        V4        V5
      V6        V7        V8
 LIST COLUMNS FOR ANALYSIS:
□:
      3 4 5 6 7
23 ITERATIONS
1
PUSH RET


PCADEMO, SETS    1, FROM     SET:
 CTR=0.24
PCADEMO:  PCA TEST DATA

  T1
PCA N=   30
                P1       P2      PVR
      V1     0.92             0.84
      V2              0.96    0.95
      V3    ¯0.75     0.66    1.00
      V4    ¯0.98             0.96
      V5    ¯0.62    ¯0.69    0.87
     PVR     0.55     0.37
     CPV     0.55     0.92
2
PUSH RET
NO
MAKE COMPONENT VARIATES?
NO
```

Figure 4. Illustration of the use of PRINAN for principal components analysis. The parameters are first set to standard values by invoking PARAM. Then SETE is set to the value 1. This setting is made here to illustrate the use of set selection and the documentation in the output. All of the "subjects" in the illustrative data set were identified with a 1, so that set selection had no consequence for the results. The output is explained in the text. Note that the user entered NO after the second PUSH RET request. This action caused the output function, BOP, to suppress typed output of the plot described in the text. The plot was formed, however, and filed in the second component of the output file, as indicated by the number 2 just above the PUSH RET request.

the user would probably note that the five original variables can be reduced to two, with very little loss in the proportion of variance accounted for. The investigator might select Variable 4 and Variable 2 to represent the two components and proceed with further analyses using only these variables. Alternatively, the researcher might convert Variables 1 and 4 to standard scores, multiply Variable 4 by −1, and then add (or average) the two variables. This result would probably have a greater percentage of reliable variance than either of the original variables. As a third choice, the researcher might let PRINAN create the component variates.

The analysis in Figure 4, carried through to the creation of a new file with the component variates, required about 2.5 min (clock time) on a moderately loaded Sigma-9. It used .04 min of CPU time. In a 32K work space, PRINAN analyzed a data set of 240 subjects and 10 variables. This task required 1.5 min of clock time without making the new file. It required .2 min of CPU time.

The output of EIG can be validated by testing for the conditions given in Lines 16, 17, and 18, as well as by comparing its results with those in Figure 2. The output of PRINAN can be checked by comparison with the results in Figure 4. The computation of the component variates can be validated by computing the intercorrelation matrix of the original variables and the component variates. The correlations obtained in this way should be the same as those given in PRINAN's output table. The component variates should have means of zero, variances of unity, and zero intercorrelations with each other.

Applications of principal components analysis in research from a variety of behavioral data sources demonstrate its general utility. With an eigen vector function and support functions to handle file access and to provide other amenities, one can easily prepare programs for discriminant function analysis and canonical correlation. Both of these can be accomplished by a slightly more complicated eigen vector procedure. Reports on these and other analytic functions are in preparation. All of these functions use the same file conventions and permit the user to move freely from one analytic technique to another. This flexibility enhances the usefulness of all the functions.

REFERENCES

Cooley, W. W., & Lohnes, P. R. *Multivariate data analysis.* New York: Wiley, 1971.

Crabtree, J. M., & Moyer, K. E. Sex differences in fighting and defense induced in rats by shock and d-amphetamine during morphine abstinence. *Physiology & Behavior*, 1973, **11**, 337-343.

Evans, S., & Attaya, W. A. Research methods and strategies in the psychophysics of image quality. *Photographic Science and Engineering*, 1978, **22**, 92-97.

Evans, S., Gage, F. H., & Neideffer, J. D. APL programs for interactive data analysis: Data entry and correlation. *Behavior Research Methods & Instrumentation*, 1980, **12**, 372-375.

Evans, S., Neideffer, J. D., & Gage, F. H. APL functions for interactive data analysis: Graphics and labels. *Behavior Research Methods & Instrumentation*, 1980, **12**, 541-545.

Gage, F. H. A multivariate approach to the analysis of social dominance. *Behavioral Biology*, 1978, **23**, 38-51.

Gage, F. H., Armstrong, D. R., & Thompson, R. G. Behavioral kinetics: A method for deriving qualitative and quantitative changes in sensory responsiveness following septal nuclei damage. *Physiology & Behavior*, 1980, **24**, 479-484.

Gage, F. H., Evans, S. H., & Olton, D. S. Multivariate analyses of performance in a DRL paradigm. *Animal Learning & Behavior*, 1979, **7**, 323-327.

Gage, F. H., & Liberman, A. F. A multivariate analysis of social dominance in children. *Aggressive Behavior*, 1978, **4**, 219-229.

Green, P. E. *Analyzing multivariate data.* Hinsdale, Ill: Dryden Press, 1978.

Hemmerly, W. J. *Statistical computations on a digital computer.* Waltham, Mass: Blaisdell, 1967.

Iverson, K. E. Notation as a tool of thought. *Communications of the Association for Computing Machinery*, 1980, **23**, 444-465.

Kaiser, H. F. The varimax criterion for analytic rotation in factor analysis. *Psychometrica*, 1958, **23**, 187-200.

Lee, H. B., & Comrey, A. L. Distortions in a commonly used factor analytic procedure. *Multivariate Behavioral Research*, 1979, **14**, 301-321.

Tatsuoka, M. M. *Multivariate analysis: Techniques for educational and psychological research.* New York: Wiley, 1971.

NOTE

1. Sigma-9 users can get a copy of a documented work space containing these and other file functions written in terms of the primitive file function and forming a set adequate to support most file use. The set requires about 2K bytes of work space, as contrasted with about 4K for the standard set in FILES.APL1. For further information, contact the first author.

(Received for publication June 22, 1981;
revision accepted August 6, 1981.)