DATA/VARIABLE REDUCTION BY PRINCIPAL COMPONENTS, BATTERY REDUCTION AND VARIABLE CLUSTERING

Ralph B. D'Agostino, Boston University
Kimberly A. Dukes, New England Medical Center
Joseph Massaro, Boston University and Quintiles, Inc.
Zhini Zhang, Boston University

1. INTRODUCTION

Often a researcher is in the position where he/she has n variables $X_1, \ldots, X_n$ and desires to reduce the number for later analyses. The techniques of principal components, variable clustering and battery reduction are statistical techniques appropriate for this reduction. Among the options available are the following:

1) Generate m composite scores

$$Y_j = w_1X_1 + \ldots + w_nX_n \quad \text{for } j = 1, \ldots, m \qquad (1)$$

where $m<n$ and w_i, $i=1, 2, \ldots, n$, are selected to "explain" the maximum possible variance. This is **Principal Components.**

2) Select m non-overlapping subsets of variables from the n and create m composite scores. Basically this replaces the m composite scores of (1) by setting some of the weights w_i equal to zero. For example, if $n=5$ and $m=2$, possible composite scores would be:

$$\begin{aligned} C_1 &= a_1X_1 + a_2X_2 \qquad\qquad + a_5X_5 \\ C_2 &= \qquad\qquad a_3X_3 + a_4X_4 \end{aligned} \qquad (2)$$

Often in (2) the a_i are set equal to unity. Ideally these m composites "explain " almost as much variance as do the m principal components. This technique is called Variable Clustering.

3) Select m variables from the n that reproduce as much as possible the variance of the original n. This is **Battery Reduction.**

In the following we review briefly the concepts underlying these techniques. Also we have developed MACROS for each of them. In the following we present these MACROS and explain how to implement them. All the material discussed below, excluding the MACROS, is presented in detail in the book **FACTOR ANALYSIS: AN APPLIED APPROACH** by E. E. Cureton and R. B. D'Agostino (Erlbaum publishers, 1983), Chapters 12 and 14.

1.1 Principal Components

Assume we have n variables, $X_1, \ldots, X_n$, all of which are standardized and we desire to generate m composite scores of the form shown in (1) where Y_1 will have the largest possible variance (λ_1) subject to the restriction $w_1^2 + \ldots + w_n^2 = 1$ (that is, the weights are normalized), Y_2 will be uncorrelated with Y_1 and have the next largest possible variance (λ_2 with $\lambda_1>\lambda_2$), etc. until we obtain m such uncorrelated composite scores all of which have weights normalized and variances

$$\lambda_1 > \ldots > \lambda_m. \qquad (3)$$

The percentage of the variance of the original n variables explained by the m composite scores is

$$100^*(\lambda_1 + \ldots + \lambda_m)/n. \qquad (4)$$

The m composite scores in this context are called component scores. The SAS Procedure PRINCOMP can be used to perform the Principal Components. The SAS Procedure FACTOR can also be employed. Both procedures standardize the variables before employing Principal Components.

The first step in Principal Components is to determine m, the number of components to retain. One popular rule is to let m equal the number of components with variances λ_i greater than one. There are other possibilities (see Cureton and D'Agostino, 1983, Chapter 12). The second step is to obtain components that are interpretable. The usual procedure for this step is to produce the Initial Component matrix A sometimes referred to as Initial Factor F and rotate it. A (or F) is an n by m matrix containing the correlations of the original n variables to the m components (or m factors). The objective of the rotation is to produce weights in the composite scores which are large on a small number of the original variables and close to zero on the other variables.

1.2 Example of Principal Components

In the Framingham Heart Study a 10 question depression scale was administered where the responses were NO or YES to the following (corresponding variable name is enclosed in parentheses):

1) I felt everything I did was an effort (EFFORT)
2) My sleep was restless (RESTLESS)
3) I felt depressed (DEPRESS)
4) I was happy (HAPPY)
5) I felt lonely (LONELY)
6) People were unfriendly (UNFRIEND)
7) I enjoyed life (ENJOYLIF)
8) I felt sad (FELTSAD)
9) I felt that people disliked me (DISLIKED)
10) I could not get going (GETGOING)

A YES was scored as 1 and NO as 0 except for questions 3 and 7 where this scoring was reversed so that a score of 1 would indicate depression for all questions.

There were three principal components with variances greater than unity. They were 3.357, 1.290 and 1.022 for a percentage variance explained equal to 56.69%. The initial component matrix A (also called the initial loading matrix and also symbolized by F), the rotated component matrix and the weights w_i for the composite scores of (1) are below.

The Rotation matrix is the Promax Reference matrix. It is an oblique rotation. See Cureton and D'Agostino (1983) Chapters 6, 8 and 9 for details.

	INITIAL MATRIX A			ROTATION MATRIX		
EFFORT	.60	.15	.41	.07	.60	.06
RESTLESS	.39	.07	.55	-.08	.64	-.12
DEPRESS	.77	-.13	-.10	.62	.13	.06
HAPPY	.70	-.23	-.06	.61	.12	-.06
LONELY	.64	-.23	-.21	.65	-.03	.00
UNFRIEND	.35	.67	-.33	.04	-.06	.80
ENJOYLIF	.52	-.27	-.27	.63	-.13	-.03
FELTSAD	.71	-.22	-.20	.69	.00	.01
DISLIKED	.34	.72	-.22	-.06	.04	.79
GETGOING	.58	.20	.47	.01	.66	.07

	WEIGHTS w_i		
EFFORT	.03	.42	.04
RESTLESS	-.04	.45	-.08
DEPRESS	.27	.08	.05
HAPPY	.26	.07	-.04
LONELY	.28	-.03	.00
UNFRIEND	.02	-.04	.59
ENJOYLIF	.28	-.10	-.02
FELTSAD	.30	-.01	.01
DISLIKED	-.02	.03	.58
GETGOING	-.01	.46	.05

1.3 Variable Cluster Analysis

As we see from the above even after the rotation all the variables tend to be included in all component scores. Variable Cluster Analysis techniques attempt to group variables in non-overlapping sets or equivalently set some of the weights in the composite scores equal to zero so that each variable is in at most one composite score. As an **INTUITIVE CLUSTER ANALYSIS** an examination of the rotation matrix or the matrix of weights would suggest the DEPRESS, HAPPY, LONELY, ENJOYLIF and FELTSAD form one cluster, EFFORT, RESTLESS and GETGOING a second and UNFRIEND and DISLIKED a third.

In addition to setting some weights equal to zero, often the variables in a cluster are given equal weights. For the above example the first cluster composite score would then be

C_1 = DEPRESS + HAPPY + LONELY + ENJOYLIF + FELTSAD.

Note the variables are standardized. If they are of the same scale and have approximately the same variance the sum of the original unstandardized variables is usually the preferred cluster composite score.

Later we present two MACROS for cluster analyses, a graphical analyses and a formal procedure employing measures of the closeness of variables to each other.

1.4 Battery Reduction

The above techniques of Principal Components and Cluster Analysis retain all the original variables and generate composite scores. At times it is desirable to reduce the number of variables (for example, if we have a questionnaire and are concerned about the burden a long questionnaire places upon a subject). This is Battery Reduction. There are numerous ways of accomplishing this. One method employs the application of Gram-Schmidt orthogonal rotations to the initial component matrix A (or F) of principal components. Basically it takes the matrix A and finds the variable with the largest variance shared by the other variables (this is called the communality). A Gram-Schmidt rotation is then performed so that the first component is made identical to this variable. Next out of the remaining variables the one with the largest shared variance is found and a Gram-Schmidt rotation is performed so that the second component is identical to this variable. This is continued until m variables are identified. An important aspect of this battery reduction is to see how much of the original variance of all n variables is explained by the retained m variables. This is simply the sum of squares of all the loadings in the final transformed matrix. Later we give a MACRO for performing this battery reduction.

2. GRAPHICAL CLUSTER ANALYSIS

The graphical cluster analysis discussed on pp 348-357 of Cureton and D'Agostino (1983) is a pictorial method to show which variables in a study may be clustered together due to the sharing of similar information. Two macros have been developed to perform this, GRAFCLUS and GGRAFCLS. Both are essentially the same but the latter takes advantage of the graphics available in SAS/GRAPH. SAS version 6 or higher is needed to run these macros. Each macro:

1) Allows the user to run a factor analysis specifying the same options available in SAS PROC FACTOR (e.g., mineigen criterion, priors criterion, rotation method, etc.);
2) Outputs the unrotated factor matrix F (keeping as many factors as the user desires). Each row of **F** corresponds to one of the original variables in the study;
3) Row normalizes **F** to create F_n;
4) Post-multiplies F_n with L_t to create L_n, where L_t is the Landahl transformation matrix of dimension *nfact* by *nfact*, where *nfact* is the number of factors kept in the original factor matrix;
5) Plots each row of L_n versus column (or factor) number as on page 355 of Cureton and D'Agostino (1983);
6) Finds the matrix product $F_n \times F_n'$ containing the cosines of the angles between each test vector;
7) Gives the user the opportunity to plot the axes of the factor space (after row normalizing them and post-multiplying them by L_t of above);
8) Gives the user the opportunity to plot subsets of variables on different graphs to facilitate identification of clusters.

Each row of L_n is considered the *profile* of a variable. Variables with a profile following a similar path when plotted can be considered to be in the same cluster. Also, if the cosine of the angle between two test vectors is approximately 1, the test vectors may also be considered to be in the same cluster.

2.1 Graphical Cluster Analysis Macro Call

To call the macro, the user inputs the following parameters. The variables do <u>not</u> need to be standardized first.

1) **DATASET** - The data set name on which the factor analysis is to be run (this can be any type of data set on which a factor analysis can be run by PROC FACTOR).
2) **VARLIST** - The variables on which the factor analysis will be run. Separate each variable name by a space. The maximum number of variables to be plotted can not exceed 62. If the axes are to be plotted, the number of variables can not exceed 62 minus the number of factors;
3) **OPTIONS** - The same options the user would specify when using PROC FACTOR (e.g., mineigen criterion, rotation method). The number of factors kept can not exceed 99. *A rotation method must be specified if the axes are to be plotted*;
4) **PRIORS** - Prior communality estimates if user specified PRIORS = INPUT in the **OPTIONS** parameter. This is not needed if no prior communality estimates are to be input by the user;
5) **PLOTAXES** - Specify <u>YES</u> or <u>yes</u> if it is desired to see a plot of the axes with the variables, otherwise specify <u>NO</u> or <u>no</u>. The default is <u>yes;</u>
6) **SUBSETS** - The subsets of variables which the user desires to sees plotted. Each subset is plotted on a separate graph and there is no limit to the number of subsets. In the macro call, the list of variables within each subset is separated by a space; the subsets are separated by an asterisk (*).

For both macros the SAS option DQUOTE must be set so that all double quotes in the macro are recognized as such.

GRAFCLUS employs PROC PLOT to plot the profiles of each variable. It works best if the user has the SAS option OVP (overprinting) set so that the symbols in the plot do not get hidden. The user must then draw by hand for each variable the lines connecting the symbols. GGRAFCLS employs PROC GPLOT to plot the profiles of each variable so the user needs to draw no lines by hand.

2.2 Graphical Cluster Analysis Macro

```
options ovp dquote;

%macro grafclus(dataset=,
                varlist=,
                options=,
                priors=,
                plotaxes=yes,
                subsets=);
  proc factor data=&dataset &options
    outstat=temp;
    var &varlist;
    priors &priors;
  run;
  data fprime lambda;
    set temp;
  if _type_='UNROTATE' or _type_='unrotate'
  then do;
    nfact+1;
    if nfact>99 then do;
      put '***ERROR: NUMBER OF FACTORS
           EXCEEDS 99***';
      abort;
    end;
    else output fprime;
  end;
  else if _type_='TRANSFOR' or
         _type_='transfor'
  then output lambda;
  drop _type_ nfact;
run;
data fprime;
  set fprime;
  call symput('nfact',_n_);
run;
proc transpose data=fprime(drop=_name_)
  out=ff prefix=factor name=vars;
run;
proc iml;
  start;
  use ff;
  read all into f (|rowname=vars
                 colname=factors|);
  close ff;
  nfact=ncol(f);
  nvars=nrow(f);
  nfactm1=nfact-1;
  use lambda;
  read all into lambda;
  close lambda;
  do ii=1 to nvars;
    f(|ii,|)=f(|ii,|)*inv(sqrt(ssq(f(|ii,|))));
  end;
  cosines=f*f`;
  maxcos=j(nvars,1,0);
  do k=1 to nvars;
    cosines(|k,k|)=0.0;
    maxcos(|k,1|)=max(cosines(|,k|));
    cosines(|k,k|)=1.0;
  end;
  coswith=j(nvars,1,'00000000');
  do k=1 to nvars;
    do ii=1 to nvars;
      if maxcos(|k,1|)=cosines(|ii,k|) then
      coswith(|k,1|)=vars(|ii,1|);
    end;
  end;
  x=i(nfactm1);
  zero=j(1,nfactm1,0);
  x=x//zero;
  left=j(nfact,1,sqrt(inv(nfact)));
  x=left||x;
  call gsorth(landahl,dontneed,lindep,x);
  landahl=landahl`;
  lambda=lambda`;
  lambda=shape(lambda,nfact,nfact);
  tprime=inv(lambda);
  do i=1 to nfact;
    tprime(|i,|)=tprime(|i,|)*
               inv(sqrt(ssq(tprime(|i,|))));
  end;
  f=f//tprime;
  f=f*landahl;
  origvars=vars`;
  axes=factors`;
  %do ii=1 %to &nfact;
    axes(|&ii,1|)=concat("axes","&ii");
  %end;
  vars=vars//axes;
  print 'LANDAHL TRANSFORMATION
         MATRIX'
  landahl (|format=8.5 rowname=factors
           colname=factors|);
  print 'MATRIX FROM WHICH CLUSTER
         GRAPH WILL BE PLOTTED'
  f (|format=8.5 rowname=vars
     colname=factors|);
  print 'COSINES OF ANGLES BETWEEN
         TEST VECTORS'
  cosines (|format=8.5 rowname=vars
           colname=vars|);
  create f from f (|rowname=vars
                  colname=factors|);
  append from f (|rowname=vars|);
  create maxcos from maxcos (|rowname=vars|);
  append from maxcos (|rowname=vars|);
  create coswith from coswith;
  append from coswith;

  create origvars from origvars;
  append from origvars;
  finish;
run;
data maxcos;
  merge maxcos(rename=(col1=cosine))
       coswith(rename=(col1=with));
  call symput('nvars',_n_);
run;
proc print data=maxcos noobs split='@';
  title 'MAXIMUM COSINE FOR EACH TEST
         VECTOR';
  var vars cosine with;
  label vars='@variable'
       cosine='maximum@cosine'
       with='@with';
run;
data origvars;
  set origvars;
  %do i=1 %to &nvars;
    call symput("var&i",col&i);
  %end;
run;
```

```
proc transpose data=f out=f;
  id vars;
run;
data f(drop=_name_);
  length factor $3;
  set f;
  factor=left(substr(_name_,7));
run;
%let nplots=%eval(&nvars+&nfact);
%let symbol =
     abcdefghijklmnopqrstuvwxyzABCDEFGH
     IJKLMNOPQRSTUVWXYZ01234567890;
%do i=1 %to &nplots;
  %let symb&i=%substr(&symbol,&i,1);
%end;
proc plot data=f;
  title 'GRAPHICAL CLUSTER ANALYSIS:
        PLOT OF PROFILES';
  plot
  %do i=1 %to &nvars;
     &&var&i*factor="&&symb&i"
  %end;
  %if &plotaxes=yes or &plotaxes=YES %then
  %do j=1 %to &nfact;
     %let k=%eval(&nvars+&j);
     axes&j*factor="&&symb&k"
  %end;/overlay;
  %if "&subsets" ne "" %then %do;
      %let i=1;
      %let sub=%scan(&subsets,&i,*);
      %do %while ("&sub" ne "");
         plot
         %let j=1;
         %let plotvar=%scan(&sub,&j);
         %do %while ("&plotvar" ne "");
            &plotvar*factor="&&symb&j"
            %let j=%eval(&j+1);
            %let plotvar=%scan(&sub,&j);
         %end;
         %if &plotaxes=yes %then
         %do j=1 %to &nfact;
           %let k=%eval(&nvars+&j);
           axes&j*factor="&&symb&k"
         %end;/overlay;
         %let i=%eval(&i+1);
         %let sub=%scan(&subsets,&i,*);
      %end;
  %end;
run;
%mend grafclus;
```

If SAS/GRAPH is available, use the macro **GGRAFCLS** which is essentially the same as above except the shaded region is replaced with:

```
data anno(keep=x y xsys function ysys text position
          style);
  length position xsys ysys $ 1 text $ 15 style $ 4
         function $ 8;
  set f;
  function='label';
  xsys='1'; style='none'; ysys='2'; position='b';
  if _n_=1 then do;
    x=4;
    %do i=1 %to &nvars;
       y=&&var&i;
       text="&&var&i";
       output;
    %end;
  end;
  else
  if _n_=&nfact then do;
    x=96;
    %do i=1 %to &nvars;
       y=&&var&i;
       text="&&var&i";
       output;
    %end;
  end;
run;
proc gplot data=f;
  axis1 label=none;
  title 'GRAPHICAL CLUSTER ANALYSIS:
        PLOT OF PROFILES';
  plot
  %do i=1 %to &nvars;
     &&var&i*factor
  %end;
  %if &plotaxes=yes %then %do j=1 %to
     &nfact;
     axes&j*factor
  %end;/vaxis=axis1 overlay annotate=anno;
  %do i=1 %to &nplots;
   symbol&i l=1 i=join;
  %end;
run;
quit;
%if "&subsets" ne "" %then %do;
  %let i=1;
  %let sub=%scan(&subsets,&i,*);
  %do %while ("&sub" ne "");
  data subanno(drop=nobs);
    set anno;
    %let j=1;
    %let subvar&j=%scan(&sub,&j);
    if
    %do %while ("&&subvar&j" ne "");
       compress(text)=upcase("&&subvar&j")
       %let j=%eval(&j+1);
       %let subvar&j=%scan(&sub,&j);
       %if "&&subvar&j"="" %then %str(;);
       %else %str(or);
    %end;
  run;
  %let nsubvar=%eval(&j-1);
proc gplot data=f;
```

```
    axis1 label=none;
    title 'GRAPHICAL CLUSTER ANALYSIS:
        PLOT OF SUBSET PROFILES';
    plot
    %do j=1 %to &nsubvar;
      &&subvar&j*factor
    %end;
    %if &plotaxes=yes %then %do j=1 %to
&nfact;
      axes&j*factor
    %end;/vaxis=axis1 overlay annotate=subanno;
    %do j=1 %to &nplots;
     symbol&i l=1 i=join;
    %end;
    run;
  quit;
  %let i=%eval(&i+1);
  %let sub=%scan(&subsets,&i,*);
  %end;
%end;
%mend ggrafcls;
```

2.3 Example of GGRAFCLS

If the above depression data is named DEPRESS and the user would like an analysis keeping 3 factors with promax rotation, the macro call is:

```
%grafclus(dataset=depress,
        varlist=effort restless depress happy lonely
              unfriend enjoylif feltsad disliked
              getgoing,
        options=nfactors=3 method=prinit
              rotate=promax maxiter=100,
        plotaxes=no,
        subsets=depress happy lonely enjoylif
              feltsad*effort restless getgoing*
              unfriend disliked);
```

Since "plotaxes"=no the user would not like a plot of the axes. Note by the "subsets" parameter that the user would like, in addition to the plot of all profiles on one graph, a plot of profiles for DEPRESS, HAPPY, LONELY , ENJOYLIF and FELTSAD on one graph, EFFORT, RESTLESS and GETGOING on another, and UNFRIEND and DISLIKED on a third. For the output, excluding the matrix of cosines, see Fig. A at the end of this report.

3. FORMAL CLUSTER ANALYSIS (CYCLE HUNT AND CHANGE)

Cycle Hunt and Cycle Change discussed on pp 360-363 of Cureton and D'Agostino (1983) is a formal technique to find which variables may be clustered. To begin, calculate the cosine matrix K = $F_n{}^*F_n'$, where F_n is defined as above with the diagonal elements being replaced by the communalities. Create the *reduced cosine matrix* K_r from K by deleting any row and column that corresponds to a variable with all cosines below a specified *acceptance level* (e.g., .950). Such a variable becomes an *intrinsic outlier*, a variable not "close enough" to any other variable to be a member of a cluster.

Cycle Hunt:

At each state, one or more of the following can be performed:
1) Combine two variables, neither of which is in a cluster, to form a cluster of two.
2) Add to an existing cluster a variable not previously in any cluster.
3) Combine two clusters to form a larger cluster.

The operation that is performed and the variables/clusters on which it is performed are the ones that leave the new lowest within-cluster cosine highest. When neither operation (1) nor (2) above can be applied without leading to a lowest cosine below the acceptance level, Cycle Hunt terminates. However, the clusters are usually less than optimal at this point mainly due to the fact that once a variable enters a cluster, it remains there. Cycle Change helps remedy this fact.

Cycle Change:

1) For each variable, including any outliers (variables not in a cluster) resulting from Cycle Hunt, find its lowest cosine with any variable in each cluster and its cosine with each variable not in any cluster.
   a) If the highest of these is with a variable in its own cluster, leave it there.
   b) If the highest is with a variable in another cluster, reassign it to that cluster.
   c) If the highest is with a forced outlier, combine it with that outlier to form a new cluster.
   d) If the variable is a forced outlier, perform (b) or (c) only if the resulting cosine is above the acceptance level.

   When all variables have been considered and if at least one change has been made, return to the first variable and repeat. Terminate when all variables are considered but no change has been made.
2) Combine the pair of clusters whose lowest within-cluster cosine will be highest and still above the acceptance level. No further

combining of clusters is performed.

3) Repeat (1) and (2) until at (2) no clusters can be combined without producing a within-cluster cosine below the acceptance level. The cluster analysis is now complete.

3.1 Cycle Hunt and Change Macro Call

Two macros, MCORR and CLUSTER, were developed to achieve Cycle Hunt and Cycle Change. Both macros must be run while in PROC IML. MCORR calculates the variable correlation matrix (and calls it MCORR) using as a parameter DATASET, the SAS data set containing only the variables for which the correlation matrix is to be calculated. Any record where there is at least one missing data point must be deleted from the data set before MCORR is called. The variables do not need to be standardized first. CLUSTER performs the Cycle Hunt and Cycle Change using as parameters:

1) MCORR - The correlation matrix calculated from the macro MCORR.
2) LEVEL - The acceptance level cosines must be above.
3) NFACT - The number of components the user desires to keep.

3.2 Cycle Hunt and Change Macro

```
%macro mcorr(dataset);
  start corr;
    n=nrow(x);
    sum=x[+,];
    xpx=t(x)*x-t(sum)*sum/n;
    s=diag(1/sqrt(vecdiag(xpx)));
    mcorr=s*xpx*s;
    print "Correlation Matrix",,mcorr[rowname=vars
       colname=vars];
  finish corr;
  use &dataset;
  read all into x(|colname=vars|);
  call corr;
%mend mcorr;

%macro cluster(cosine,level,nfact);
  start case1;
    noc=noc+1;
    iclus[noc,1]=is1;
    iclus[noc,2]=is2;
    nvc[noc]=2;
    isw[is1]=1;
    isw[is2]=1;
  finish case1;

  start case2;
    nvc[is2]=nvc[is2]+1;
    iclus[is2,nvc[is2]]=is1;
    isw[is1]=1;
  finish case2;

  start case3;
   iclus[is1,nvc[is1]+1:nvc[is1]+nvc[is2]]
      =iclus[is2,1:nvc[is2]];
    nvc[is1]=nvc[is1]+nvc[is2];
    call collap(is2);
  finish case3;

  start collap(is) global(noc,nvc,nov,iclus);
    noc=noc-1;
    nvc[is:nov-1]=nvc[is+1:nov];
    iclus[is:nov-1,]=iclus[is+1:nov,];
  finish collap;

  al=&level;
  call eigen(mm,ee,&cosine);
  dd=diag(mm);
  aa=ee*sqrt(dd);
  temp=aa(|,1:&nfact|);
  nvars=nrow(temp);
  do ii=1 to nvars;
    temp(|ii,|)
     =temp(|ii,|)*inv(sqrt(ssq(temp(|ii,|))));
  end;
  vind=temp*temp`;
  print 'Cosine matrix' vind(|format=8.5
       rowname=vars colname=vars|);
  vind=vind-diag(vind);
  vind=vind#((vind>0)-0.5)#2;
  nov=nrow(vind);
  iclus=j(nov,nov,0);
  isw=j(nov,1,0);
  nvc=j(nov,1,0);
  noc=0;

/*---------Cycle Hunt Start-------------*/
  do;
  L100:  hli=0;
  do i=1 to nov-1;
    if isw[i]=0 then
    do;
      do j=i+1 to nov;
        if isw[j]=0 then
        do;
          if vind[i,j]>hli then
          do;
          hli=vind[i,j];
            ito=1;
            is1=i;
            is2=j;
          end;
        end;
      end;
    end;
  end;
  if noc^=0 then
```

```
    do i=1 to nov;
      if isw[i]=0 then
      do j=1 to noc;
        dum=min(vind[i,iclus[j,1:nvc[j]]]);
        if dum>hli then
        do;
          hli=dum;
          ito=2;
          is1=i;
          is2=j;
        end;
      end;
    end;
  if hli<al then goto L300;
  if noc<=1 then goto L200;
  do i=1 to noc-1;
    do j=i+1 to noc;
      dum=min(vind[iclus[i,1:nvc[i]],iclus[j,1:nvc[j]]]);
      if dum>hli then
      do;
        hli=dum;
        ito=3;
        is1=i;
        is2=j;
      end;
    end;
  end;
  L200:  if ito=1 then call case1;
         if ito=2 then call case2;
         if ito=3 then call case3;
  goto L100;
  L300:  end;
  print "cluster after cycle hunt",,iclus;

/*-------Cycle Change Start--------*/

  do;
  L600:  if noc=0 then goto L800;
  ichg=0;
  do i=1 to nov;
    ito=0;
    hli=0;
    do j=1 to noc;
      dum=1;
      ism=0;
      do jd=1 to nvc[j];
        if i=iclus[j,jd] then ism=1;
        else if vind[i,iclus[j,jd]]<dum then
           dum=vind[i,iclus[j,jd]];
      end;
      if dum>hli then
      do;
        hli=dum;
        if ism=1 then ito=1;
        else ito=2;
        is1=j;
      end;
    end;
    do j=1 to nov;
      if (i^=j)&(isw[j]^=1)&(vind[i,j]>hli) then
      do;
        hli=vind[i,j];
        ito=3;
        is1=j;
      end;
    end;
    if hli<al then goto L700;
    if ito>1 then ichg=1;
    if ito=1 then goto L700;
    if ito=2 then
    do;
      if isw[i]=1 then
      do;
        j1=loc(iclus=i);
        j=int((j1-1)/nov)+1;
        j1=mod(j1-1,nov)+1;
        nvc[j]=nvc[j]-1;
        iclus[j,j1:nov-1]=iclus[j,j1+1:nov];
        if nvc[j]=0 then
        do;
          if j<is1 then is1=is1-1;
          call collap(j);
        end;
      end;
      else isw[i]=1;
      nvc[is1]=nvc[is1]+1;
      iclus[is1,nvc[is1]]=i;
    end;
    if ito=3 then
    do;
      if isw[i]=1 then
      do;
        j1=loc(iclus=i);
        j=int((j1-1)/nov)+1;
        j1=mod(j1-1,nov)+1;
        iclus[j,j1:nov-1]=iclus[j,j1+1:nov];
        nvc[j]=nvc[j]-1;
        if nvc[j]=0 then call collap(j);
      end;
      else isw[i]=1;
      noc=noc+1;
      iclus[noc,1]=i;
      iclus[noc,2]=is1;
      nvc[noc]=2;
      isw[is1]=1;
    end;
  L700:  end;
  if ichg=1 then goto L600;

  hli=0;
  do i=1 to noc-1;
    nvci=nvc[i];
    do j=i+1 to noc;
      nvcj=nvc[j];
      dum=min(vind[iclus[i,1:nvci],iclus[j,1:nvcj]]);
      if dum>hli then
      do;
       hli=dum;
       is1=i;
       is2=j;
      end;
    end;
  end;
  if hli>al then
  do;

iclus[is1,nvc[is1]+1:nvc[is1]+nvc[is2]]=iclus[is2,1:n
vc[is2]];
    nvc[is1]=nvc[is1]+nvc[is2];
    call collap(is2);
    goto L600;
  end;
  L800: print "final cluster",,iclus;
  end;
%mend cluster;
```

3.3 Example of MCORR and CLUSTER

To run MCORR and CLUSTER on the depression data, all records with at least one missing value must first be deleted:

```
data depress(keep=effort restless depress happy
   lonely unfriend enjoylif feltsad disliked getgoing);
  set depress;
  if n(effort, restless, depress, happy, lonely,
      unfriend,enjoylif,feltsad,disliked,getgoing)=10;
run;
```

If the user desires to retain the three factors the call would be:

```
proc iml;
  %mcorr(depress);
  %cluster(mcorr,0.90,3);
```

The output, excluding the correlation and cosine matrices, is:

```
Cluster After Cycle Hunt:
5  8  7  0  0  0  0  0  0  0
1 10  2  0  0  0  0  0  0  0
6  9  0  0  0  0  0  0  0  0
3  4  0  0  0  0  0  0  0  0
Final Cluster:
5  7  3  4  8  0  0  0  0  0
1 10  2  0  0  0  0  0  0  0
6  9  0  0  0  0  0  0  0  0
```

The numbers represent the variable which lies in that numbered position in the data set. For the DEPRESS data, the positions are the same order in which the variables are listed on the second page of this report (so for example, 1 corresponds to EFFORT, 2 to RESTLESS, etc.). The variables on the same row under "Final Cluster" above form a cluster. So DEPRESS, HAPPY, LONELY, ENJOYLIF and FELTSAD from one cluster, EFFORT, RESTLESS and GETGOING a second and UNFRIEND and DISLIKED a third.

4. BATTERY REDUCTION

A SAS macro GRAMSCHM was developed using SAS/IML to perform battery reduction. It requires as input NFACT, the number of components the user would like to retain for the reduction procedure and DATASET, a SAS data set containing data for only the variables the user wishes to analyze. <u>Any record where there is at least one missing data point must be deleted from the data set before GRAMSCHM is called</u>. The variables do <u>not</u> need to be standardized first.

The output consists of the variable correlation matrix, the loading matrix A, the reduced component matrix (a submatrix of A with the number of columns equalling the number of components retained), the Gram-Schmidt rotated reduced component matrix (from which can be found the m variables to which the original n reduce) and the percent variation explained by each component.

4.1 Battery Reduction Macro

```
%macro gramschm(dataset=,nfact=);
option ls=80;
title1'GRAM-SCHMIDT ROTATION';

proc iml;
 start gramschm;
   use &dataset;
   read all into x(|colname=vars|);
/* Compute the correlation matrix */
   n=nrow(x);
   sum=x(|+,|);
   xpx=x`*x-sum`*(sum/n);
   s=diag(1/sqrt(vecdiag(xpx)));
   corr=s*xpx*s;
/* CREATES THE LOADING MATRIX A */
   call eigen(m,e,corr);
     d=diag(m);
     a=e*sqrt(d);
     nfact=&nfact;
     nvars=nrow(a);
     reda=a(|,1:nfact|);
     af=j(nvars,1,0);
/*  PERFORMS   THE   GRAM-SCHMIDT
ROTATION */
     do i=1 to nfact-1;
       if i=1 then a1=a(|,i:nfact|);
       else a1=a(|,2:colno|);
       sq=a1(|,##|);
```

```
        max=sq(|<:>,|);
        norm1=a1(|max,|);
        sq1=sq(|max,|);
        sq1=sqrt(sq1);
        normn1=norm1/sq1;
        nfact1=nfact-i;
        ident=i(nfact1);
        zero=j(1,nfact1,0);
        tempy=ident//zero;
        normn1=normn1`;
        ty=normn1||tempy;
        call gsorth(y,t,lindep,ty);
        ayp1=a1*y;
        a=ayp1;
        colno=ncol(a);
        if (i-(nfact-1) < > 0) then af=af||a(|,1|);
        else af=af||a(|,1:2|);
     end;
rota=af(|,2:nfact+1|);
per_var=rota(|##,|);
per_var=per_var/nvars;
print 'CORRELATION
MATRIX',,corr(|rowname=vars colname=vars|);
print 'EIGENVALUE',,m;
print 'REDUCED COMPONENT
MATRIX',,reda(|rowname=vars|);
print 'ROTATED REDUCED COMPONENT
MATRIX',,rota(|rowname=vars|);
print 'PERCENT VARIATION
EXPLAINED',,per_var;
finish;
run gramschm;
%mend gramschm;
```

4.2 Example of Gramschm

To run GRAMSCHM on the depression data, all records with at least one missing value must be deleted:

```
data depress(keep=effort restless depress happy
   lonely unfriend enjoylif feltsad disliked getgoing);
  set depress;
  if n(effort, restless, depress, happy, lonely,
     unfriend,enjoylif,feltsad,disliked,getgoing)=10;
run;
```

If the user desires to retain the three factors the call would be:

```
%gramschm(dataset=depress,nfact=3);
```

The relevant output is:

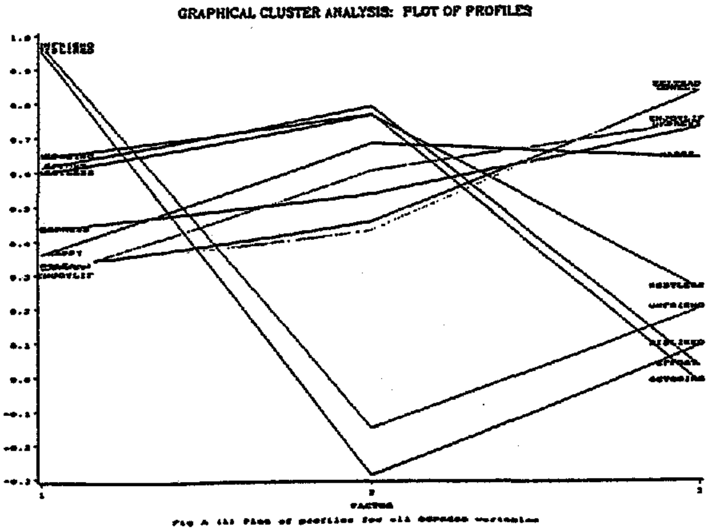
```
Rotated Reduced Component Matrix:
EFFORT        .212  .359  .614
RESTLESS     -.001  .207  .642
DEPRESS       .258  .731  .132
HAPPY         .131  .715  .120
LONELY        .167 -.688 -.035
UNFRIEND      .829  .000  .000
ENJOYLIF      .110  .625 -.139
FELTSAD       .201  .749  .000
DISLIKED     -.819 -.052  .108
GETGOING     -.217  .308  .675

Percent Variation Explained:
              .162  .274  .131
```

From the output, the three variables the data can be reduced to are UNFRIEND, FELTSAD and GETGOING (we reduce to three variables since three components are kept in the analysis). The first variable is UNFRIEND since its loadings are 0 on the last two components; the second is FELTSAD since its loading is 0 on the last component; the last is GETGOING since its loading is highest on the last component. The percent variation explained for each column is the sum of the squared loadings divided by the number of variables (10).

Fig. A: Output of GGRAFCLS

It can be seen below that DEPRESS, HAPPY, LONELY, ENJOYLIF and FELTSAD have similar profile plots as do EFFORT, RESTLESS and GETGOING. Similarly, the profiles for UNFRIEND and DISLIKED are very similar.



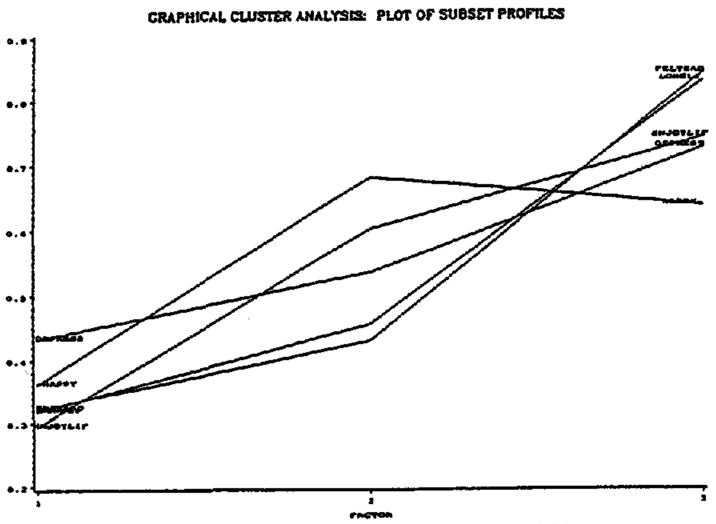


Fig A (1) Plot of profiles for depress happy lonely enjoylif feltsad

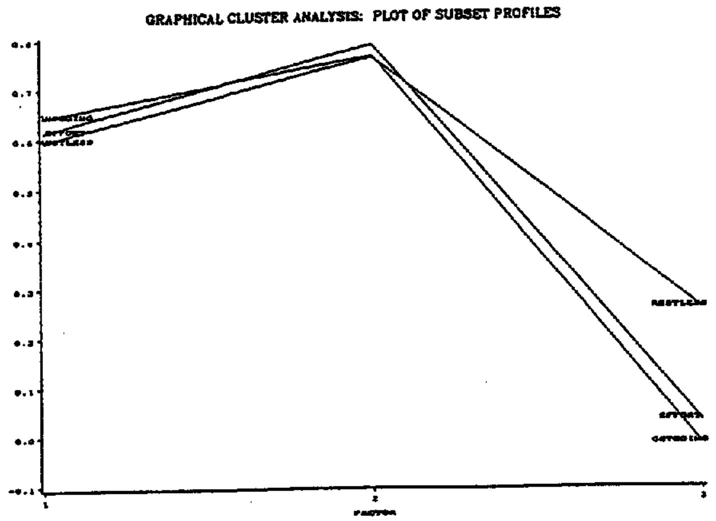


Fig A (2) Plot of profiles for effort restless getgoing

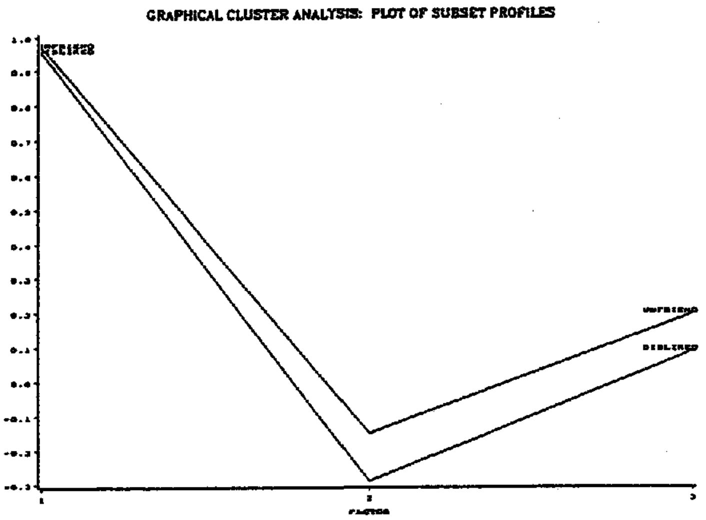


Fig A (3) Plot of profiles for unfriend disliked

Research funded by NHLBI grant R01-HL-40423-4 and AHCPR grant R01-HS-06665-3.

For a copy of these macros on diskette, please send requests to Joseph Massaro, Boston University Math Department, 111 Cummington St., Boston MA 02215. Please include a 3.5" or 5.25" diskette and a disk mailer with your request.