Paper SD-04

Using Predetermined Factor Structures to Simulate a Variety of Data Conditions

Kevin Coughlin, Edison State College, Fort Myers, FL
Jeffrey Kromrey, University of South Florida, Tampa, FL
Susan Hibbard, Edison State College, Fort Myers, FL

ABSTRACT

This paper presents a method through which data sets of varying characteristics can be simulated based on predetermined, uncorrelated factor structures. As demonstrated in a series of studies, this Monte Carlo method yields factor structures that are clear and simple. The process begins with the application of conceptual and actual factor loadings to the creation of correlation matrices; samples are then generated based on these correlation matrices. The method for generating correlation matrices allows the researcher to manipulate the number of observed variables, the communality among variables, and the number of common factors. The process for simulating samples of observations provides additional options for specifying sample size and level of measurement. This paper includes an example of the process for generating a correlation matrix and a distribution of simulated observations. This paper is intended for researchers that are interested in factor analytic designs and are familiar with PROC IML.

Keywords: COMMUNALITY, CORRELATION MATRICES, DICHOTOMY, FACTOR ANALYSIS, MULTIVARIATE NORMALITY, OVERDETERMINATION, PROC IML, SAMPLE SIZE

BACKGROUND

Latent variables or factors are not directly observable; however, as a central assertion in factor analytic theory, much of the variation in the phenomena that researchers witness and measure is attributable to these underlying traits (Bartholomew, 1984; Cureton & D'Agostino, 1983; Stevens, 2002; Tucker & MacCallum, 1997). Moreover, factor analytic theory asserts that these hypothetical, internal attributes are more "fundamental" than the surface attributes which we observe (Tucker & MacCallum, 1997, p. 2). This paper presents a tool that can facilitate exploration of exploratory factor analysis as a research method.

Although exploratory factor analysis is useful in "both measurement and substantive research contexts" (Henson & Roberts, 2006, p. 396), it has been subjected to persistent criticism. Many of these objections are based on the subjectivity of the decisions that researchers must make when conducting their factor analyses (Henson & Roberts, 2006). Attempts to incorporate non-normal data types into factor analyses represent another potential area of criticism (Yuan, Marshall, & Bentler, 2002). Guiding principles are well established for contexts that include continuous variables that exhibit multivariate normality. However, "no firm guidelines have as yet emerged concerning situations in which qualitative and quantitative variables are mixed together" (Krzanowski, 1983, p. 235).

The program presented in this paper allows researchers to simulate data with known factor structures. The program provides researchers with the opportunity to manipulate a variety of research contexts including the number of observed variables, communality, and sample size. Through specifying the proportion of observed variables that are dichotomized, researchers can also simulate varying levels of violations to assumption of multivariate normality.

SOURCES AND PREVIOUS METHODOLOGICAL RESEARCH

Tucker, Koopman, and Linn (1969) presented a method for simulating correlation matrices that correspond to pre-specified common factor structures. Their method yields matrices that are more similar to real data correlation matrices than those obtained directly from a structural model.

In the Tucker, Koopman, and Linn (1969) method, the population correlation matrix, **R**, is generated based on major, minor, and unique factors,

$$R = A_1 A_1' + A_2 A_2' + A_3 A_3'$$

where $\mathbf{A}_1$ is the p x k matrix of input factor loadings for the major factors, $\mathbf{A}_2$ is the matrix of input factor loadings for the minor factors, and $\mathbf{A}_3$ is the p x p diagonal matrix of input factor loadings for the unique factors. If the contribution

of the minor factors ($\mathbf{A}_2$) is set to zero, the data generation model will exactly match a factor analytic model with *k* common factors.

The process for creating $\mathbf{A}_1$ starts with the creation of a matrix of conceptual input factor loadings, $\tilde{\mathbf{A}}_1$. To create $\tilde{\mathbf{A}}_1$, the loading on a randomly selected factor, *j* = 1 to *k*, is set to a value randomly chosen between 0 and *k* - 1 (i.e., for a 3-factor model the loading $\tilde{a}_{1j}$ could be 0, 1, or 2). Next the loading on a randomly selected factor from those remaining is set to a value randomly chosen between 0 and $k - 1 - \tilde{a}_{1j}$. This process continues until a conceptual input factor loading has been chosen for each factor, and ensures the sum of the loadings across the factors is *k*-1. This process is then repeated for each of the *p* variables.

The matrix of input factor loadings, $\mathbf{A}_1$, is then created from the matrix of conceptual input factor loadings, $\tilde{\mathbf{A}}_1$, through a series of three steps: (1) normal deviates are added to introduce error, (2) a skewing function is used to limit negative factor loadings, and (3) the matrix is scaled to ensure desired levels of communality. The diagonal matrix, $\mathbf{A}_3$, for the unique factors is also scaled to ensure the desired levels of communality.

An important aspect of this simulation approach is that an infinite number of population correlation matrices may be generated from a single specification of number of variables, number of common factors, and level of communality (analogous to the inevitability that an infinite number of factor solutions may be obtained from a single correlation matrix).

Tucker, Koopman, and Linn (1969) used this method in a Monte Carlo study comparing three methods of factor extraction. Subsequently, MacCallum, Widaman, Zhang, and Hong (1999) used the method to simulate data for a study of sample size requirements for factor recovery, controlling the number of variables in the correlation matrices, the number of common factors, and the level of communality. In a follow-up study, MacCallum, Widaman, Preacher, and Hong (2001) replicated their research with matrices for which the common factor model did not hold exactly in the population (i.e., by including large numbers of minor common factors in the simulation). In addition, Hogarty, Hines, Kromrey, Ferron, and Mumford (2005) used the method in their investigation of sample size requirements for factor recovery. This research extended the range of conditions that were examined by MacCallum et al. (1999, 2001) and included a broader range of criteria for evaluating the congruence of the sample factor solutions with the known population factor structure. Finally, Coughlin (2013) used this simulation method to investigate factor extraction methods applied to correlation matrices obtained from mixtures of continuous and binary variables.

SIMULATION STRATEGY

The simulation program contains two phases. The first phase includes the derivation of population correlation matrices from conceptual and actual factor input loadings. Through the second phase, population correlations are used to simulate samples of data. As the flow chart in figure 1 demonstrates, researchers can manipulate different research characteristics at each phase of the simulation strategy.

GENERATING POPULATION CORRELATION MATRICES

As described by Tucker, Koopman, and Linn (1969), the simulation procedures include a “mathematical, probabilistic model” and presume the existence of major, minor, and unique factors. The major factors represent the “influences on observed scores of individuals for the phenomena which the experimenter wishes to study” (Tucker, Koopman, & Linn, 1969, p. 424); minor factors exert systematic influence on the value of observations but are not within the experimenters’ control, and unique factors represent error. Major factors are identified by a subscript value of one; minor factors are given a subscript value of two, and a subscript of three indicates a unique factor. The number of each type of factor is designated by M_s (Tucker, Koopman, & Linn, 1969).

The generation of correlation matrices begins with a matrix $A_s$of “actual input factor loadings” (Tucker, Koopman, & Linn, 1969, p. 425). Through a three-step process, these actual input loadings are derived from a matrix of conceptual input loadings, $\tilde{A}$. Conceptual input factor loadings represent the researcher’s expectations concerning the “factorial composition of the variables” (Tucker, Koopman, & Linn, 1969, p. 426).

The first step in the development of conceptual input loadings involved the creation of “relative conceptual input loadings” for each variable. For a three-factor domain, the loadings conform to the following guidelines:

1. A zero, one, or two is chosen at random and is assigned to the first factor.
2. The sum of the loadings for any one variable is limited to two; this limit implies that if the first loading is two, then other two must be zero; if the first loading is one, then the other two have an equal probability of being a zero or a one.
3. The loading of the third factor is chosen so that the sum of all three will be two (Tucker, Koopman, & Linn, 1969).

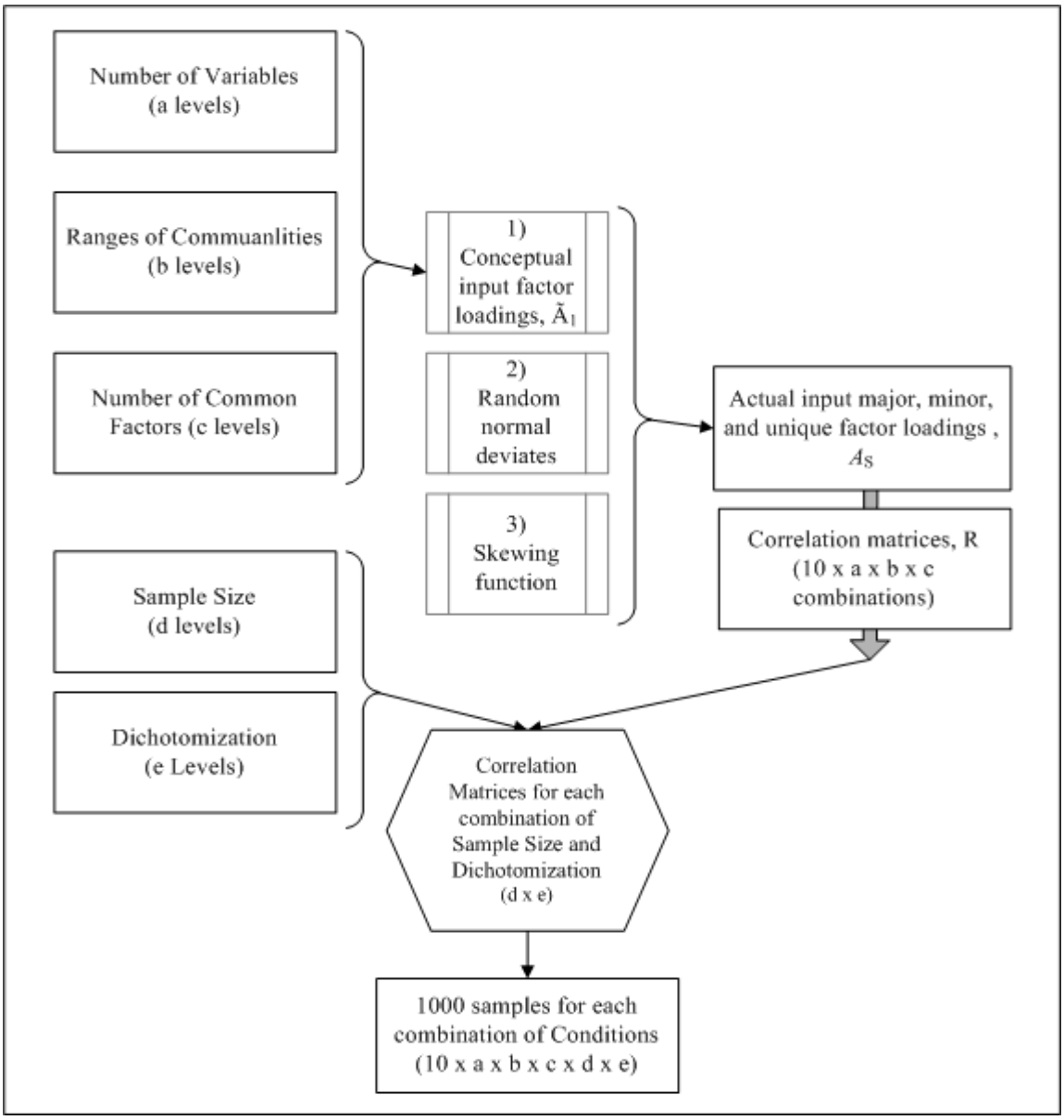

Figure 1. Flow chart summarizing simulation strategy

In the first step towards creating actual input factor loadings, the conceptual input factor loadings are combined with random normal deviates; these deviates represent the natural "discrepancies" that occur in the construction of instruments (Tucker, Koopman, & Linn, 1969, p. 428). The output of this step, $(y_1)_{jm_1}$, is defined by:

$$(y_1)_{jm_1} = (\tilde{a}_1)_{jm_1} c_{m1} + d_{1j} x_{1m_1} (1 - c_{m1}^2)^{1/2}$$

Where:

1. $(\tilde{a}_1)_{jm_1}$ is the entry in row j and column m_1 of matrix $\tilde{A}_1$
2. x_{jm_1} is a random, normal deviate ($\mu = 0,\ \sigma = 1$)
3. c_{m_1} is a constant for each factor m_1; the possible values range from zero to one; the constants represent the "general control an experimenter has on the loading of actual variables on the factors" (Tucker, Koopman, & Linn, 1969, p. 429)

4. d_{1j} is a constant for each variable j; this constant normalizes each row of x_{1m_1} to a unit length vector; it is defined as: $d_{1j} = \left(\sum_{m_1} x_{jm_1}^2\right)^{-1/2}$ (Tucker, Koopman, & Linn, 1969, p. 429)

The second step in this translation process includes a skewing function that reduces negativity in factor loadings. This function yields coefficients, $(z_1)_{jm_1}$, according to the following equality:

$$(z_1)_{jm_1} = \frac{(1+k)}{(2+k)} \frac{(y_1)_{jm_1}\left[(y_1)_{jm_1} + \left|(y_1)_{jm_1}\right| + k\right]}{\left[\left|(y_1)_{jm_1}\right| + k\right]}$$

In this expression, k is a parameter that can range from zero to infinity. Each vector of $(z_1)_{jm_1}$ coefficients is reduced to unit length by the following:

$$(a_1^*)_{jm_1} = g_{1j}(z_1)_{im_1}$$

Where:

$$g_{1j} = \left[\sum_{m_1} (z_1)_{jm_1}^2\right]^{-1/2}$$

The third step in this process includes scaling the matrix "to ensure desired levels of communality" (Hogarty et al., 2005, p. 207).

The matrix of actual input factor loadings, A_s is a $J \times M_s$ matrix that contains a row for each variable J and a column for each major, minor, and unique factor (Tucker, Koopman, & Linn, 1969, p. 425). For each matrix A_s, a matrix A_s^* can be defined by adjusting the rows of A_s to unit length vectors. P is a square, symmetric matrix of order J; P is positive and semi-definite; it is defined by:

$$P_s = A_s^* A_s^{*'}$$

$Diag(P_s) = I$.

The simulated correlation matrix is given by:

$$R = B_1 P_1 B_1 + B_2 P_2 B_2 + B_3 P_3 B_3$$

B_s are diagonal matrices that include b_{1i}, b_{2i}, and b_{3i} as entries. These entries are real, positive numbers that have the following property:

$$b_{1i}^2 + b_{2i}^2 + b_{3i}^2 = 1$$

These considerations imply the following equalities:

$$r_{ii} = 1$$
$$Diag\,(R) = I$$

Matrix A_s is now defined as:

$$A_s = B_s A_s^*$$

The correlation matrix is given by:

$$R = A_1 A_1^{'} + A_2 A_2^{'} + A_3 A_3^{'} = (A_1, A_2, A_3)(A_1, A_2, A_3)'$$

The supermatrix (A_1, A_2, A_3) contains the matrices A_1, A_2, and A_3 as horizontal sections (Tucker, Koopman, & Linn. 1969).

The coefficients in the B_s matrices "regulate" the amount of variability in the variables that is related to the major, minor, and unique factors. The B_1^2 matrix contains communalities, and the B_3^2 contain values for uniqueness. When B_2 matrix is zero, the "simulation model" equals the "formal model" (Tucker, Koopman, & Linn, 1969, p. 426).

This simulation program included the formal model as the simulation model. The B_2 matrix is set to zero, and by implication, the input factor loadings for minor factors were zero. This forced the "data generation model" to match "a factor analytic model" with the number of common factors equal to the levels specified for each combination of research contexts that was examined in this study.

SAMPLE GENERATION

With few modifications, the simulation strategy included in this study is derived from Hogarty, Hines, Kromrey, Ferron, and Mumford's (2005) investigation of the relationship between sample size and factor solutions. This strategy allows the researcher to control for the number of observed variables, levels of communality, and sample size. The current program provides researchers with the opportunity to vary the percentage of observed variables that are categorical or dichotomous.

APPLICATION EXAMPLE

To illustrate program output, a population correlation matrix and an example data set were simulated using SAS 9.3©. This example includes eight observed variables, two factors, and 100 observations (*n = 100*). The communality level was set at the high condition, and 25% of the observed variables were dichotomized. Table 1 presents the communalities, the conceptual input factor loadings, and the actual input factor loadings.

Table 1
Communalities, Conceptual Input Factor Loadings, and Actual Input Factor Loadings by Eight Observed Variables

		Conceptual Input Factor Loadings ($\tilde{A}_1$)		Actual Input Factor Loadings (A_1)	
Variable	Communalities	Factor 1	Factor 2	Factor 1	Factor 2
1	.836	1	0	0.836	-0.007
2	.836	1	0	0.834	-0.067
3	.836	0	1	0.558	0.623
4	.774	0	1	0.495	0.596
5	.774	1	0	0.769	0.093
6	.836	0	1	-0.115	0.829
7	.774	0	1	0.585	0.507
8	.894	0	1	-0.197	0.872

This illustrative example included five population correlation matrices. However, the number of population correlation matrices simulated is a design characteristic that can be specified by the researcher. Table 2 contains one of these correlation matrices.

Table 2
Population Matrix of Pearson Product Moment Correlations for Eight Observed Variables

	1	2	3	4	5	6	7	8
1		.698	.462	.409	.642	-.102	.485	-.171
2			.424	.973	.635	-.151	.454	-.222
3				.648	.487	.452	.642	.434
4					.436	.437	.592	.422
5						-.011	.497	-.070
6							.353	.746
7								.327
8								

Table 3 contains simulated data for the first ten and last ten observations from a data set based on a sample size of 100. This illustrative example included five simulated data sets per population correlation matrix. As was the case with the number of population correlation matrices, the number of data sets per correlation matrix is specified by the researcher.

Table 3
Simulated Data Set Excerpt (Sample Size = 100, Eight Observed Variables, and .25 Dichotomization Level Condition)

	Observed Variables							
Observation	1	2	3	4	5	6	7	8
1	1	1	1.070	0.247	0.993	-1.568	-0.611	-1.634
2	0	0	-1.431	-1.547	-0.883	1.100	-0.470	-0.267
3	1	1	-0.027	-0.444	0.997	-1.036	0.129	-0.237
4	0	1	0.406	0.626	0.052	0.753	1.523	0.800
5	0	0	-0.270	1.190	-1.876	1.945	-0.120	0.662
6	1	1	-1.088	0.886	0.247	-0.987	-0.437	-1.222
7	1	1	1.172	0.625	1.438	-0.120	0.682	-0.722
8	1	0	-0.041	0.502	-0.467	0.649	0.037	-0.154
9	0	0	0.458	0.952	-0.806	1.010	0.181	1.704
10	0	0	-1.789	-0.938	-1.622	0.964	-1.583	-0.242
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
91	1	1	0.903	0.082	1.035	-0.905	-0.432	-0.060
92	1	1	-0.473	-1.464	-0.314	-0.981	1.061	-1.182
93	0	1	-0.313	0.220	-0.740	-1.214	-0.902	-0.900
94	1	1	1.546	2.260	1.441	1.059	2.000	1.185
95	1	1	0.701	0.601	1.401	0.815	0.402	-0.614
96	1	0	0.862	-0.017	-0.028	0.854	1.208	1.300
97	1	1	-0.527	-1.300	1.086	-0.044	-0.337	-0.374
98	0	0	-0.514	-0.655	-1.562	0.165	-0.472	0.656
99	1	1	1.262	0.340	1.853	-1.202	0.472	-1.220
100	0	0	0.589	0.380	-1.536	0.968	0.111	2.570

PROC IML CODE

The PROC IML code is presented below in its entirety. The nature of the data to be simulated is controlled by setting six scalar values at the beginning of the program. The number of variables in the data matrix is set with the scalar *p* and the number of common factors with the scalar *k*. The value of *d_frac* controls the proportion of variables that are generated as dichotomies rather than continuous variables and *commun_type* sets the communality of the variables to low, wide, or high (with the values 1, 2 and 3, respectively). Finally, *n_pops* sets the number of population correlation matrices to be generated and *replicat* controls the number of samples to simulate from each population. The sample size is manipulated within a loop in the program (see the *S_Size* scalar). As provided, samples of size 100, 200, 300, and 1000 are generated from each population correlation matrix.

```
option ls = 256 ps = max nonumber nodate nocenter;
proc printto print='c:\SESUG2013.lst';
proc iml;

p = 8;        *8;    *20;   *40;   *60;
k = 2;        *2;           *4;           *8;
d_frac = .25; *.00; *.05;  *.25;  *.50;  *.75; *.95;
Commun_type = 3;
N_pops = 5; * N of populations to generate;
replicat= 5; *1000;  * N of samples from each population;
nn1 = 100000;
means=j(1,p,0);
variance = j(1,p,1);
```

```
start Make_PopR(nvars,nfactors,commun_type,A1tilde,B1,x,x2,d,A1,R);
bp1=j(nvars,nvars,0);
bp2=uniform(bp1);
if commun_type=3 then do;
 bp3=(bp2*.2999999)+.55;
 b1square=round(diag(bp3),.1);
end;
if commun_type=1 then do;
 bp3=(bp2*.2999999)+.15;
 b1square=round(diag(bp3),.1);
end;
if commun_type=2 then do;
 bp3=(bp2*.6999999)+.15;
 b1square=round(diag(bp3),.1);
end;

B1=b1square##.5;
b3square=I(nvars)-b1square;
B3=b3square##.5;
A1tilde1=j(nvars,nfactors,0);
A1tilde2=round((uniform(A1tilde1)*(nfactors-.00000001))-.5);
A1tilde=A1tilde2;
do j=2 to nfactors;
 do i=1 to nvars;
  if j<nfactors then do;
  A1tilde[i,j]=round(((nfactors-.00000001-sum(A1tilde[i,1:j-1]))
                    *uniform(0))-.5);
  end;
  if j=nfactors then do;
  A1tilde[i,nfactors]=nfactors-sum(A1tilde[i,1:nfactors-1])-1;
  end;
 end;
end;

x=normal(A1tilde);
x2=x##2;
d=j(nvars,nfactors,0);
do j=1 to nfactors;
 do i=1 to nvars;
  d[i,j]=(sum(x2[i,1:nfactors]))##-.5;
 end;
end;

cvec=j(1,nfactors,0);
do j=1 to nfactors;
 cvec[1,j]=round((uniform(0)*.2999999)+.65,.1);
end;
c=j(nvars,1,1)*cvec;
c2=c##2;
ones=j(nvars,nfactors,1);
y=A1tilde#c + d#x#((ones-c2)##.5);
k=.2;
z=j(nvars,nfactors,0);
do j=1 to nfactors;
 do i=1 to nvars;
  z[i,j]=((1+k)*y[i,j]*(y[i,j]+abs(y[i,j])+k))/((2+k)*(abs(y[i,j])+k));
 end;
end;

z2=z##2;
g=j(nvars,nfactors,0);
do j=1 to nfactors;
 do i=1 to nvars;
  g[i,j]=(sum(z2[i,1:nfactors]))##-.5;
 end;
```

```
end;

A1star=g#z;
A1=B1*A1star;
A3star=I(nvars);
A3=B3*A3star;
R=A1*A1`+A3*A3`;

Finish;

start gendata2a(NN1,seed1,variance,bb,cc,dd,mu,r_matrix,YY,p,d_frac);
  L = eigval(r_matrix);
  neg_eigval = 0;
  do r = 1 to nrow(L);
       if L[r,1] < 0 then neg_eigval = 1;
  end;
  if neg_eigval = 0 then do; * matrix is positive definite, so use the Cholesky root
approach;
       COLS = NCOL(r_matrix);
       G = ROOT(r_matrix);
       YY=rannor(repeat(seed1,nn1,COLS));
       YY = YY*G;
       do r = 1 to NN1;
       do c = 1 to COLS;
              YY[r,c] = (-1*cc) + (bb*YY[r,c]) + (cc*YY[r,c]##2) + (dd*YY[r,c]##3);
              YY[r,c] = (YY[r,c] * SQRT(variance[1,c])) + mu[1,c];
       end;
       end;
  end;
  if neg_eigval = 1 then do; * matrix is not positive definite, so use the PCA
approach;
       COLS = NCOL(r_matrix);
       V = eigvec(r_matrix);
       do i = 1 to nrow(L);
       do j = 1 to ncol(V);
                     if L[i,1] > 0 then V[j,i] = V[j,i] # sqrt(L[i,1]);
                     if L[i,1] <= 0 then V[j,i] = V[j,i] # sqrt(.000000001);
              end;
       end;
       YY=rannor(repeat(seed1,nn1,COLS));
       YY = V*YY`;
       YY = YY`;
       do r = 1 to NN1;
       do c = 1 to COLS;
              YY[r,c] = (-1*cc) + (bb*YY[r,c]) + (cc*YY[r,c]##2) + (dd*YY[r,c]##3);
              YY[r,c] = (YY[r,c] * SQRT(variance[1,c])) + mu[1,c];
       end;
       end;
  end;
       if d_frac > 0 then do;
              do r = 1 to nn1;
                     do c = 1 to (p*d_frac);
                     if yy[r,c] < 0 then yy[r,c] = 0;
                     else if yy[r,c] = 0 then yy[r,c] = 1;
                     else if yy[r,c] > 0 then yy[r,c] = 1;
                     end;
              end;
       end;
finish;

start gendata2b(NN2,seed1,variance,bb,cc,dd,mu,r_matrix,YY,p,d_frac);
  L = eigval(r_matrix);
  neg_eigval = 0;
  do r = 1 to nrow(L);
       if L[r,1] < 0 then neg_eigval = 1;
```

```
  end;
  if neg_eigval = 0 then do; * matrix is positive definite, so use the Cholesky root
approach;
      COLS = NCOL(r_matrix);
      G = ROOT(r_matrix);
      YY=rannor(repeat(seed1,nn2,COLS));
      YY = YY*G;
      do r = 1 to NN2;
      do c = 1 to COLS;
            YY[r,c] = (-1*cc) + (bb*YY[r,c]) + (cc*YY[r,c]##2) + (dd*YY[r,c]##3);
            YY[r,c] = (YY[r,c] * SQRT(variance[1,c])) + mu[1,c];
      end;
      end;
  end;
  if neg_eigval = 1 then do; * matrix is not positive definite, so use the PCA
approach;
      COLS = NCOL(r_matrix);
      V = eigvec(r_matrix);
      do i = 1 to nrow(L);
      do j = 1 to ncol(V);
                  if L[i,1] > 0 then V[j,i] = V[j,i] # sqrt(L[i,1]);
                  if L[i,1] <= 0 then V[j,i] = V[j,i] # sqrt(.000000001);
            end;
      end;
      YY=rannor(repeat(seed1,nn2,COLS));
      YY = V*YY`;
      YY = YY`;
      do r = 1 to NN2;
      do c = 1 to COLS;
            YY[r,c] = (-1*cc) + (bb*YY[r,c]) + (cc*YY[r,c]##2) + (dd*YY[r,c]##3);
            YY[r,c] = (YY[r,c] * SQRT(variance[1,c])) + mu[1,c];
      end;
      end;
  end;
      if d_frac > 0 then do;
            do r = 1 to nn2;
                  do c = 1 to (p*d_frac);
                  if yy[r,c] < 0 then yy[r,c] = 0;
                  else if yy[r,c] = 0 then yy[r,c] = 1;
                  else if yy[r,c] > 0 then yy[r,c] = 1;
                  end;
            end;
      end;
finish;

Do pop_num = 1 to N_pops;    * Loop for 10 populations;

run Make_PopR(p,k,commun_type,A1tilde,B1,x,x2,d,A1,R_pop);
Lambda = A1;
numr = r_pop[+,+] - p;
deno = r_pop[+,+];
ratio = numr/deno;
f2_pop = (p/(p-1))*ratio;
r2_pop = f2_pop/(1+f2_pop);
corr = r_pop;
seed1=round(1000000*rannor(0));
chg = 1;
cycle = 0;
corr_tmp = corr;
do until (chg = 0);
      run gendata2a(NN1,seed1,variance,1,0,0,means,corr_tmp,sim_data,p,d_frac);
      sim_corr = corr(sim_data);
      resid_m = sim_corr - corr;
      tot_res = sum(abs(resid_m));
      if cycle = 0 then do;
```

```
              best_corr = corr_tmp;
              best_res = tot_res;
       end;
       if cycle > 0 then do;
              if tot_res < best_res then do;
                     best_corr = corr_tmp;
                     best_res = tot_res;
              end;
       end;
       if tot_res < (.005#(((p-1)#p)/2)) then CHG = 0; * Convergence!;
       if cycle > 30 then do;
       if tot_res < (.01#(((p-1)#p)/2)) then CHG = 0; * Convergence!;
    end;
       if cycle > 200 then CHG = 0;
    if CHG = 1 then corr_tmp = corr_tmp - resid_m;  * adjust template and simulate
another large sample;
        cycle = cycle + 1;
       if CHG = 0 then do;
       end;
  end;

Do S_Size = 1 to 4;      * Loop for sample sizes;

 if S_Size = 1 then Sampsize2=100;
 if S_Size = 2 then Sampsize2=200;
 if S_Size = 3 then Sampsize2=300;
 if S_Size = 4 then Sampsize2=1000;


Do rep=1 to replicat;    * Loop for 1000 Samples;

seed1=round(1000000*ranuni(0));
nn2 = sampsize2;
corr_tmp = best_corr;

r_sing = 0;
do until (det(r_samp) > 0);
  run gendata2b(NN2,seed1,variance,1,0,0,means,corr_tmp,sim_data,p,d_frac);
  sampdat = sim_data;
  r_samp=corr(sampdat);
  if det(r_samp)<=0 then do;
       r_sing = r_sing +1;
  end;
end;

if rep = 1 then _r_sing = r_sing;
if rep > 1 then _r_sing = _r_sing + r_sing;

end; *End replications loop;

print A1tilde;
print B1;
print x;
print x2;
print d;
print A1;
print R_pop;
print sampdat;

end; *End sample size loop;
end; *End populations loop;


quit;
```

CONCLUSIONS

The SAS/IML code provided in this paper provides a straight-forward method for simulating sample data that may arise from a given population factor structure. The two-step method of simulating population correlation matrices followed by the simulation of samples from those matrices yields samples that have the variability expected from a common factor model. As provided, the program allows easy manipulation of variable communality, number of latent factors and observed variables, sample size, and proportion of dichotomous variables. Further, the program may be readily modified to allow the simulation of non-normal observed variables, including discrete ordinal variables that are obtained from rating scale items.

The program will be useful primarily to methodologists who study techniques of exploratory and confirmatory factor analysis and other latent variable methods such as structural equation models. In addition, the program has utility for the investigation of meta-analytic techniques that synthesize sample results across a variety of empirical studies which may differ in their operationalization of latent variables. Finally, this program should be valuable for research related to psychometric methods including the investigation of measurement invariance.

REFERENCES

Bartholomew, D. J. (1984). The foundations of factor analysis. *Biometrika, 71*(2), 221-232.

Coughlin, K. B. (2013). *An Analysis of Factor Extraction Strategies: A Study of the Relative Strengths of Principal Axis, Ordinary Least Squares, and Maximum Likelihood Factor Extraction Methods in Research Contexts that Include Varying Ratios of Categorical to Continuous Variables* (Unpublished doctoral dissertation). University of South Florida, Tampa, FL.

Conway, J. M., & Huffcutt, A. I. (2003). A review and evaluation of exploratory factor analysis practices in organizational research. *Organizational Research Methods, 6*, 147- 168.

Costello, A. B., & Osborne, J. W. (2005). Recommendations for getting the most from your analysis. *Practical Assessment, Research & Evaluation, 10* (7), 1-9.

Cureton, E. E. & d'Agostino, R. B. (1983). *Factor analysis: An applied approach*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Harman, H. H. (1976). *Modern factor analysis* (3rd ed.). Chicago: University of Chicago Press.

Hesnon, R. K. & Roberts, J. K. (2006). Exploratory factor analysis in published research: Common errors and some comment on improved practice. *Educational and Psychological Measurement, 66*(3), 393-416.

Hogarty, K. Y., Hines, C. V., Kromrey, J. D., Ferron, J. M. & Mumford, K. R. (2005). The quality of factor solutions in exploratory factor analysis: The influence of sample size, commuanltiy, and overdetermination. *Educational and Psychological Measurement, 65,* 202-226.

Krzanowski, W. J. (1983). Distance between populations using mixed continuous and categorical variables. *Biometrika, 70*(1), 235-243.

MacCallum, R. C., Widaman, K. F., Zhang, S., & Hong, S. (1999). Sample size in factor analysis. *Psychological Methods, 4,* 84-99.

MacCallum, R. C., Widaman, K. F., Preacher, K. J., & Hong, S. (2001). Sample size in factor analysis: The role of model error. *Multivariate Behavioral Research*, *36*, 611-637.

Stevens, J. P. (2002). *Applied multivariate statistics for the social sciences* (4th ed.). Mahwah, NJ: Lawrence Erlbaum.

Tucker, L. R., Koopman, R. F., & Linn, R. L. (1969). Evaluation of factor analytic research procedures by means of simulated correlation matrices. *Psychometrika, 34*, 421-459.

Tucker, L. & MacCallum, R. (1997). *Exploratory factor analysis: A book manuscript*. Retrieved August 20, 2008, from http://www.unc.edu/~rcm/book/factornew.htm.

Yuan, K. H., Marshall, L. L., & Bentler, P. M. (2002). A unified approach to exploratory factor analysis with missing data, nonormal date, and in the presence of outliers. *Psychometrika, 66*(1), 95-122.

Contact Information

Kevin Coughlin, Ph.D.
Office of the Registrar & Academic Course Level Assessment
Edison State College
8099 College Parkway
Fort Myers, FL 33919
(239) 489-9027
Kcoughlin@edison.edu