An algorithm to generate variates with desired intercorrelation matrix

SK Mishra
Dept. of Economics
NEHU, Shillong, India

1. Introduction

Sometimes, especially in simulation (e.g. Monte Carlo experiments), it is required to generate a matrix of sample variates, $X(n,m)$, that characterizes a desired intercorrelation matrix, $R(m,m)$, where n stands for the number of observations (or sample size) and $m < n$ is the number of variates (or variables). If each column (variate) of $X(n,m)$ has zero mean and unit standard deviation then the intercorrelation matrix $R(m,m) = n^{-1}X'X$. Being the quadratic form (see Theil, 1971, pp. 22-29), the intercorrelation matrix, R, is necessarily a positive (semidefinite) matrix and all the successive principal minors of R are non-negative (see Takayama, 1974, pp. 118-121, pp. 383-385). All of the eigenvalues of R are non-negative. Each element $r_{ij} \in R$ is the cosine of angle θ_{ij} between the vectors x_i *and* x_j. In practice, however, no x_i is a linear combination of x_j ; $i \neq j$; $j = 1,2,...,m$ (that is, $x_i = \sum_{j=1;\, j\neq i}^{m} x_j a_j$ is *not true* for any non-null real vector $a = (a_1 \;\; a_2 \;\; ... \;\; a_m)'$). In case, one requires such a vector, it can be obtained by a linear combination such as $x_i = \sum_{j=1;\, j\neq i}^{m} x_j a_j$. This case being very specific and trivial (and so set apart in practice), one requires to generate R, which is a positive definite matrix with all its successive principal minors being positive.

It may be noted that arbitrary *real* symmetric matrices, say Q, that have elements $q_{ii} = 1 \;\forall\; i = 1,2,...,m$ and $-1 \le q_{ij} \le 1 \;\forall\; i, j = 1,2,...,m$; $i \neq j$ are *not* the genuine intercorrelation matrices, R, obtainable from some *real* X although they may appear to be so. For example, the following three matrices appear to be genuine intercorrelation matrices while they are not.

$$Q_1 = \begin{bmatrix} 1.00 & 0.70 & 0.00 \\ 0.70 & 1.00 & 0.80 \\ 0.00 & 0.80 & 1.00 \end{bmatrix};\; Q_2 = \begin{bmatrix} 1.00 & 0.90 & 0.10 \\ 0.90 & 1.00 & 0.80 \\ 0.10 & 0.80 & 1.00 \end{bmatrix};\; Q_3 = \begin{bmatrix} 1.00 & 0.60 & 0.13 \\ 0.60 & 1.00 & 0.90 \\ 0.13 & 0.90 & 1.00 \end{bmatrix}$$

$\text{Det}(Q_1) = -0.13$, $\text{Det}(Q_2) = -0.316$ and $\text{Det}(Q_3) = -0.0465$. One of the eigenvalues of each matrix is negative. Several such examples may be generated.

The objective of this note is to provide an algorithm that generates $X(n,m)$ with a desired intercorrelation matrix, $R(m,m)$. Each column of $X(n,m)$, has (practically) zero mean and

unit standard deviation with a sufficient degree of accuracy (say, correct up to 8 places after decimal or so). Since the variables are generated on a computer, some approximation error (due to rounding off) is inevitable.

2. The Algorithm

The algorithm runs in the following steps:

1. Generate $Y(n,m)$ from a random number generator that yields $Y \sim U(0,1)$.
2. Standardize Y such that its each column has zero mean and unit standard deviation. Call this standardized Y by a new name, say Y^*.
3. Compute intercorrelation matrix S from Y^*.
4. Compute all eigenvalues (D) and the associated eigenvectors (V) of S. Here D is a diagonal matrix and V is an orthogonal matrix. Moreover, each column of V has a unit length (Euclidean norm).
5. Compute $Z = (Y^*)V$. Now $Z(n,m)$ is column-wise orthogonal.
6. Standardize $Z(n,m)$ such that each one of its columns has zero mean and unit standard deviation. This $Z(n,m)$ will be used at step 10.
7. Choose an intercorrelation matrix, $R(m,m)$. This is the intercorrelation matrix that is induced into Z. In choosing R one must be cautious to see that it should not violate the properties of an intercorrelation matrix described earlier. None of its eigenvalues should be negative. This is done in the next step.
8. Compute all eigenvalues (say, L) of R and the associated vectors (say E). If any of the eigenvalues are negative, change the R matrix since no intercorrelation matrix, by necessity, can have negative eigenvalues (if X is real). In that case, go to step 7.
9. Standardize E to obtain W such that each of its column has a squared (Euclidean) norm equal to the eigenvalue associated with it. Let $k_j = \{(\sum_{i=1}^{m} e_{ij}^2) / L_j\}^{1/2}$ then $w_{ij} = e_{ij} / k_j \;\; ; \;\; i, j = 1, 2, ..., m.$ This guarantees that $\sum_{i=1}^{m} w_{ij}^2 = L_j \;\; \forall \;\; j = 1, 2, ..., m.$
10. Compute $X = ZW$.
11. Standardize X (such that each of its column has zero mean and unit standard deviation) if so required.

3. A FORTRAN Computer Program

We provide here the *source codes* of the computer program that implements the algorithm given above. The main program invokes three *subroutine* and one *function* subprograms. Some procedures in the computer program (the one that computes eigenvalues and eigenvectors, in particular) have been adapted from Krishnamurthy and Sen (1976), pp. 242-247. These source codes may easily be translated into any other computer language such as Pascal, C^{++} or even BASIC, if needed. Some languages may not have a provision to perform double precision arithmetic. In that case, single precision arithmetic may be used. The results would be sufficiently accurate for the desired purpose. In its present FORTRAN codes, the program may be compiled by any suitable FORTRAN compiler.

4. Inputs to the Computer Program

When this program is run, it asks for the following parameters (and inputs). Although they have been sufficiently explained in the program queries, they are explained here.

1. Have you stored the intercorrelation matrix, etc. Before running this program, it is required that the intercorrelation matrix R is already stored in some file in text mode. This can be done by some text editor such as EDIT.COM (a DOS program of MICROSOFT). The name of this file is, say ***inputfile***
2. What are N and M ? Here N is the no. of observations in X(n,m) to be generated and m is the number of variables. Presently, in the codes given here, maximum N is 100 (=NL) and the maximum M is 10 (=ML). These parameters can be increased. Accordingly, dimensions in the program may be changed before compilation.
3. Feed non-zero scalar, etc : Feed 1 or any other non-zero number.
4. Seed to generate random number: With this seed the uniformly distributed random numbers lying between (0, 1) = U(n,m) are generated. This number should lie between –32767 and 32767, zero excluded. This is a suitable number for most personal computers.
5. File in which correlation matrix is stored : When X(n,m) is generated, R is induced into it. The user should feed this R matrix before running the program. The file name is fed against this query. The file name should be in single quotes '***inputfile***' .
6. Output file in which the generated X(n,m) characterizing intercorrelation matrix R will be stored : the output file name in single quotes ' ***outputfile***' is fed.

On termination the program stores the results X(n,m) in the ***outputfile***. It also stores the computed R matrix there, which may be different from the desired matrix R only slightly (may be at the 9^{th} or the 10^{th} place onwards after decimal).

5. Fields of Application

In Monte Carlo experiments that evaluate performance of competing estimators of regression coefficients (or evaluates the efficacy of a method of estimation of parameters) under severe multicollinearity conditions, we require to generate X(n,m) that are highly multicollinear across the variables. The author (see Mishra, 2004) generated highly multicollinear X(n,m) variables to test the performance of Maximum Entropy Leuven estimators vis-à-vis the OLS estimator of β in the model $y = X\beta + u$. To generate X(n,m), a slightly different algorithm (than the one presented here) was used. Filzmoser and Croux (2002) generated highly multicollinear $Z = [X_1 \mid X_2]$ by first generating $X_1 \sim N(0, \Sigma)$ and then obtaining $X_2 = X_1 + \Delta$ where $\Delta \sim N(0,\ 0.001)$. This procedure yielded $Z = [X_1 \mid X_2]$ highly correlated across X_1 *and* X_2. In his paper Paris (2001) dealt with multicollinear regressors but he did not explain how multicollinearity was induced into X(n,m). He (see Paris, 2001, p. 4) wrote: "X was drawn from a uniform U[-1.7, 2.0] … each component of the disturbance vector u was drawn from a normal distribution N[1, 5]."

Sometimes two variables Y and Z are each *cointegrated* with another variable X, but Y and Z do not appear to be cointegrated with each other, although, intuitively, one would expect that they should be cointegrated with each other and the transitivity property would be exhibited. By carrying out a Monte Carlo simulation, Ferré (2004) showed that even though

the two variables were in fact cointegrated, the test for cointegration was not able to pick this up due to the interplay of the error terms of the relationships between the variables. By using the algorithm presented here, several such examples may be generated for experiments and further investigation. We present here two intercorrelation matrices which can be used (as inputs to the program given here) to generate X(n,m) that would show intransitivity of cointegration.

Table 1. Intercorrelation Matrix of X Showing Intransitivity of Cointegration

Variables	X_1	X_2	X_3	X_4	X_5
X_1	1.00	0.61	0.52	0.58	0.00
X_2	0.61	1.00	0.62	0.65	0.50
X_3	0.52	0.62	1.00	0.64	0.61
X_4	0.58	0.65	0.64	1.00	0.76
X_5	0.00	0.50	0.61	0.76	1.00

In the matrix above, $r(x_1, x_5)$ is zero while other elements are large enough to exhibit cointegration. If this matrix is used to generate X(n,m) for n howsoever large (say 500 or so), we will obtain an example to show a lack of transitivity relation in cointegration. Another intercorrelation matrix with elements : $r_{11} = r_{22} = r_{33} = 1.00$, $r_{12} = r_{21} = 0.60$, $r_{13} = r_{31} = 0.00$, $r_{23} = r_{32} = 0.55$ will produce a similar instance. Many such examples may be generated.

Experiments that directly or indirectly use multivariate analysis methods (such as Principal components analysis, Factor analysis or Cluster analysis; see Kendall and Stuart, 1968) as a *procedure* may require X(n,m) with a desired R matrix. In such experiments our algorithm may be useful.

References

1. **Ferré, M** (2004). "The Johansen Test and the Transitivity Property." *Economics Bulletin*, Vol. 3 (27), pp. 1-7.
2. **Filzmoser, P and C Croux** (2002). "A Projection Algorithm for Regression with Collinearity." In K Jajuga, A Sokolowski, and HH Bock (eds), *Classification, Clustering, and Data Analysis,* Springer-Verlag, Berlin, pp. 227-234.
3. **Kendall, MG and A Stuart** (1968). *The Advanced Theory of Statistics*, Vol. 3. Charles Griffin & Co. London.
4. **Krishnamurthy, EV and SK Sen** (1976). *Computer-Based Numerical Algorithms*, Affiliated East-West Press, New Delhi.
5. **Mishra, SK** (2004). "Multicollinearity and Modular Maximum Entropy Leuven Estimator." *Social Science Research Network* at http://ssrn.com/author=353253
6. **Paris, Q** (2001). "Multicollinearity and Maximum Entropy Estimators", *Economics Bulletin*, Vol. 3 (11), pp. 1-9.
7. **Takayama, A** (1974). *Mathematical Economics*, The Dryden Press, Illinois.
8. **Theil, H** (1971). *Principles of Econometrics*, Wiley, New York.

```
C     ------------------- Main Program -------------------------------
      COMMON ML,NL,FC,FCO
      DOUBLE PRECISION X(100,10),SCALE
      CHARACTER *15 FC,FCO
      INTEGER *2 IU,IV
C     ----------------------------------------------------------
C     NL AND ML ARE THE HIGHEST PERMISSIBLE DIMENSION LIMITS TO
C     X(NL,ML) MATRICES. OTHER MATRICES HAVE COMPATIBLE DIMENSIONS
C     CHANGE THEM IF REQUIRED AND PERMISSIBLE BY MEMORY LIMITS.
      NL=100
      ML=10
C     ----------------------------------------------------------
      WRITE(*,*)'HAVE YOU STORED THE CORRELATION MATRIX, R ? IF NOT'
      WRITE(*,*)'STORE IT IN AN ASCII FILE. THEN RUN THE PROGRAM'
      WRITE(*,*)'IF NO THEN FEED ZERO (0).IF YES FEED ANY OTHER NUMBER'
      READ(*,*) NY
      IF(NY.EQ.0) THEN
      WRITE(*,*)'SO, STORE R MATRIX FIRST THEN RUN THE PROGRAM'
      STOP
      ENDIF
C     ----------------------------------------------------------
      WRITE(*,*)'WHAT ARE N AND M ? '
      WRITE(*,*)'(N = NO. OF OBSERVATIONS, M = NO. OF VARIABLES)'
      READ(*,*) N,M
      WRITE(*,*)'NON-ZERO SCALAR TO SCALE UP THE X VARIABLES ?'
      WRITE(*,*)'(NOT NECESSARY. FEED 1 OR ANY OTHER NON-ZERO NUMBER'
      READ(*,*) SCALE
      WRITE(*,*)'FEED A NON-ZERO SEED TO GENERATE RANDOM VARIABLE ?'
      WRITE(*,*)'(SEED MUST LIE BETWEEN -32767 AND 32767 AND NOT ZERO)'
      READ(*,*) IU
      WRITE(*,*)'FILE IN WHICH CORRELATION MATRIX IS STORED ?'
      WRITE(*,*)'FEED THE FILE NAME IN THE SINGLE QUOTES'
      READ(*,*) FC
      WRITE(*,*)'FILE IN WHICH OUTPUT X WILL BE STORED ?'
      WRITE(*,*)'FEED THE FILE NAME IN THE SINGLE QUOTES'
      READ(*,*) FCO
      OPEN(9,FILE=FCO,STATUS='NEW')
      CALL GENX(X,N,M,IU,IV,SCALE)
      CLOSE(9)
      END
C     ----------------------------------------------------------------
C
      SUBROUTINE GENX(X,N,M,IU,IV,SCALE)
      COMMON ML,NL,FC,FCO
      DOUBLE PRECISION A(10,10),V(10,10),W(10,10),P(10),R(10)
      DOUBLE PRECISION X(NL,ML),Z(100,10),SUML,RAND,SCALE
      DIMENSION MM(10)
      CHARACTER *15 FC,FCO
      INTEGER *2 IU,IV
C     WRITE(*,*)'ENTERS GENX'
C     ----------------- PARAMETERS -------------------------------
      NN=1
      NSTD=1
C     ----------------------------------------------------------------
```

```
      DO 1 I=1,N
      DO 1 J=1,M
      X(I,J)=RAND(IU,IV)*SCALE
    1 CONTINUE
      CALL CORR(X,N,M,A,NSTD)
      CALL EIGEN(A,M,NN,V,W,P,MM)
C      WRITE(*,*)'RETURNS FROM EIGEN'
C      PAUSE
      DO 301 I=1,N
      DO 301 J=1,M
      Z(I,J)=0.0
      DO 301 K=1,M
      Z(I,J)=Z(I,J)+X(I,K)*V(K,J)
  301 CONTINUE
C     ------------------------------------------------
      CALL CORR(Z,N,M,A,NSTD)
      OPEN(8,FILE=FC)
      DO 302 I=1,M
      READ(8,*) (A(I,J),J=1,M)
  302 CONTINUE
      CLOSE(8)
      CALL EIGEN(A,M,NN,V,W,P,MM)
C      WRITE(*,*)'RETURNS FROM EIGEN'
C     ------------------------------------------------
      DO 60 I=1,M
      R(I)=A(I,I)
   60 CONTINUE
      WRITE(*,*)'EIGENVALUES = ',(R(I),I=1,M)
      DO 64 I=1,M
      IF(R(I).LE.0.0) THEN
      WRITE(*,*)'SOME OF THE EIGENVALUES ARE NOT POSITIVE'
      WRITE(*,*)'PROGRAM TERMINATED. FEED DIFFERENT R MATRIX'
      STOP
      ENDIF
   64 CONTINUE
      DO 62 J=1,M
      P(J)=0.0
      DO 61 I=1,M
      P(J)=P(J)+V(I,J)**2
   61 CONTINUE
      P(J)=DSQRT(P(J)/R(J))
   62 CONTINUE
      DO 63 J=1,M
      DO 63 I=1,M
      W(I,J)=V(I,J)/P(J)
   63 CONTINUE
      DO 304 I=1,N
      DO 304 J=1,M
      X(I,J)=0.0
      DO 304 K=1,M
      X(I,J)=X(I,J)+Z(I,K)*W(J,K)
  304 CONTINUE
      WRITE(9,*)'GENERATED X(N,M) MATRIX'
      DO 305 I=1,N
      WRITE(9,310)(X(I,J),J=1,M)
  305 CONTINUE
      WRITE(9,*)'COMPUTED INTER-CORRELATION MATRIX'
```

```
      CALL CORR(X,N,M,A,NSTD)
      DO 306 I=1,M
      WRITE(9,310)(A(I,J),J=1,M)
  306 CONTINUE
  310 FORMAT(1X,5D15.7)
      RETURN
      END
C     ---------------------------------------------------------------
C     Generates Rectangular (0,1) Random Numbers
      FUNCTION RAND(IU,IV)
      DOUBLE PRECISION RAND
      INTEGER *2 IU,IV
      IV=IU*259
      IF(IV.GE.0) GOTO 2
      IV=IV+32767+1
    2 RAND=IV
      IU=IV
      RAND=RAND*0.3051851E-04
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE EIGEN(A,N,NN,V,W,P,MM)
      DOUBLE PRECISION A(10,10),V(10,10),W(10,10),P(10)
      DOUBLE PRECISION PMAX,EPLN,TAN,SIN,COS,AI,TT,TA,TB
      DIMENSION MM(10)
C      WRITE(*,*)'ENTERS EIGEN'
C     ------------ INITIALISATION -------------------------
      DO 50 I=1,N
      DO 51 J=1,N
      V(I,J)=0.0
   51 W(I,J)=0.0
      P(I)=0.0
   50 CONTINUE
      PMAX=0
      EPLN=0
      TAN=0
      SIN=0
      COS=0
      AI=0
      TT=0
      EPLN=1.0D-310
C     -----------------------------------------------------
      IF(NN.NE.0) THEN
        DO 3 I=1,N
        DO 3 J=1,N
        V(I,J)=0.0
        IF(I.EQ.J) V(I,J)=1.0
    3   CONTINUE
      ENDIF
    2 NR=0
    5 MI=N-1
      DO 6 I=1,MI
      P(I)=0.0
      MJ=I+1
      DO 6 J=MJ,N
      IF(P(I).GT.DABS(A(I,J))) GO TO 6
        P(I)=DABS(A(I,J))
```

```
     MM(I)=J
 6 CONTINUE
 7 DO 8 I=1,MI
   IF(I.LE.1) GOTO 10
   IF(PMAX.GT.P(I)) GOTO 8
10 PMAX=P(I)
   IP=I
   JP=MM(I)
 8 CONTINUE
   IF (PMAX.LE.EPLN) THEN
   GO TO 12
   ENDIF
   NR=NR+1
13 TA=2.0*A(IP,JP)
   TB=(DABS(A(IP,IP)-A(JP,JP))+
  1DSQRT((A(IP,IP)-A(JP,JP))**2+4.0*A(IP,JP)**2))
      TAN=TA/TB
   IF(A(IP,IP).LT.A(JP,JP)) TAN=-TAN
14 COS=1.0/DSQRT(1.0+TAN**2)
   SIN=TAN*COS
   AI=A(IP,IP)
   A(IP,IP)=(COS**2)*(AI+TAN*(2.0*A(IP,JP)+TAN*A(JP,JP)))
   A(JP,JP)=(COS**2)*(A(JP,JP)-TAN*(2.0*A(IP,JP)-TAN*AI))
   A(IP,JP)=0.0
   IF(A(IP,IP).GE.A(JP,JP)) GO TO 15
   TT=A(IP,IP)
   A(IP,IP)=A(JP,JP)
   A(JP,JP)=TT
   IF(SIN.GE.0) GO TO 16
   TT=COS
   GO TO 17
16 TT=-COS
17 COS=DABS(SIN)
   SIN=TT
15 DO 18 I=1,MI
   IF(I-IP) 19, 18, 20
20 IF(I.EQ.JP)GO TO 18
19 IF(MM(I).EQ.IP) GO TO 21
   IF(MM(I).NE.JP) GO TO 18
21 K=MM(I)
   TT=A(I,K)
   A(I,K)=0.0
   MJ=I+1
   P(I)=0.0
   DO 22 J=MJ,N
   IF(P(I).GT.DABS(A(I,J))) GO TO 22
   P(I)=DABS(A(I,J))
   MM(I)=J
22 CONTINUE
   A(I,K)=TT
18 CONTINUE
   P(IP)=0.0
   P(JP)=0.0
   DO 23 I=1,N
   IF(I-IP) 24, 23, 25
24 TT=A(I,IP)
   A(I,IP)=COS*TT+SIN*A(I,JP)
```

```
      IF(P(I).GE.DABS(A(I,IP))) GO TO 26
      P(I)=DABS(A(I,IP))
      MM(I)=IP
   26 A(I,JP)=-SIN*TT+COS*A(I,JP)
      IF(P(I).GE.DABS(A(I,JP))) GO TO 23
   30 P(I)=DABS(A(I,JP))
      MM(I)=JP
      GO TO 23
   25 IF(I.LT.JP) GO TO 27
      IF(I.GT.JP) GO TO 28
      IF(I.EQ.JP) GO TO 23
   27 TT=A(IP,I)
      A(IP,I)=COS*TT+SIN*A(I,JP)
      IF(P(IP).GE.DABS(A(IP,I))) GO TO 29
      P(IP)=DABS(A(IP,I))
C     SEE THIS IS I not one(1)
      MM(IP)=I
   29 A(I,JP)=-TT*SIN+COS*A(I,JP)
      IF(P(I).GE.DABS(A(I,JP))) GO TO 23
      GO TO 30
   28 TT=A(IP,I)
      A(IP,I)=TT*COS+SIN*A(JP,I)
      IF(P(IP).GE.DABS(A(IP,I))) GO TO 31
      P(IP)=DABS(A(IP,I))
      MM(IP)=I
   31 A(JP,I)=-TT*SIN+COS*A(JP,I)
      IF(P(JP).GE.DABS(A(JP,I))) GO TO 23
      P(JP)=DABS(A(JP,I))
      MM(JP)=I
   23 CONTINUE
      IF(NN.EQ.0) GOTO 7
      DO 32 I=1,N
      TT=V(I,IP)
      V(I,IP)=TT*COS+SIN*V(I,JP)
      V(I,JP)=-TT*SIN+COS*V(I,JP)
   32 CONTINUE
      GO TO 7
   12 RETURN
      END
      SUBROUTINE CORR(X,N,M,A,NSTD)
      COMMON ML,NL,FC,FCO
      DOUBLE PRECISION X(NL,ML),A(10,10),AV(10),SD(10)
      CHARACTER *15 FC,FCO
C      WRITE(*,*)'ENTERS CORR'
C      DO 24 I=1,N
C      WRITE(*,10)(X(I,J),J=1,M)
C  24 CONTINUE
      NSQR=N*N
      DO 1 J=1,M
      DO 2 JJ=J,M
      A(J,JJ)=0.0
      DO 2 I=1,N
    2 A(J,JJ)=A(J,JJ)+X(I,J)*X(I,JJ)
      DO 21 JJ=J,M
      A(JJ,J)=A(J,JJ)
   21 CONTINUE
    1 CONTINUE
```

```
      DO 3 J=1,M
      AV(J)=0.0
      DO 3 I=1,N
    3 AV(J)=AV(J)+X(I,J)
      DO 4 J=1,M
      DO 5 JJ=1,M
      A(J,JJ)=(N*A(J,JJ)-AV(J)*AV(JJ))/NSQR
    5 CONTINUE
    4 CONTINUE
      DO 6 J=1,M
      AV(J)=AV(J)/N
      SD(J)=DSQRT(A(J,J))
    6 CONTINUE
      DO 7 J=1,M
      DO 7 JJ=1,M
      A(J,JJ)=A(J,JJ)/(SD(J)*SD(JJ))
    7 CONTINUE
      IF(NSTD.NE.0) THEN
      DO 8 I=1,N
      DO 8 J=1,M
      X(I,J)=(X(I,J)-AV(J))/SD(J)
    8 CONTINUE
      ENDIF
   10 FORMAT(6D12.4)
      WRITE(*,*)'CORRELATION MATRIX --------------------------'
      DO 22 I=1,M
      WRITE(*,10)(A(I,J),J=1,M)
   22 CONTINUE
      WRITE(*,*)'------------------------------------------'
C      PAUSE
      RETURN
      END
```