Appendix B
Generating Multivariate Ordinal Variables

Contents

B.1 Overview of Generating Multivariate Ordinal Variates

This supplemental material describes the SAS/IML functions that are used in Section 9.3 of *Simulating Data with SAS* (Wicklin 2013) to simulate data from correlated ordinal random variables. The functions are based on the "mean mapping method" of Kaiser, Träger, and Leisch (2011), which builds on the work of Demirtas (2006).

B.2 Working with Multivariate Ordinal Random Variables in SAS/IML Software

Suppose that you want to simulate data from $d > 2$ ordinal random variables, $X_1, X_2, \ldots, X_d$, where the jth variable has N_j values. Without loss of generality, assume the values are $1, 2, \ldots, N_j$. Each random variable is defined by its probability mass function (PMF): $P(X_j = i) = P_{ij}$ for $i = 1, 2, \ldots, N_j$. It is convenient to store all of the PMFs as columns of a probability matrix that has d columns and $N = \max_j N_j$ rows. You can use missing values to "pad" the columns that have fewer than N values.

For example, the following matrix `P` contains the PMFs for three ordinal random variables. The first variable can take on two values; the PMF is $\{0.25, 0.75\}$. The third variable can take on four values; the PMF is $\{0.20, 0.15, 0.25, 0.40\}$.

```
proc iml;
    /* P1   P2    P3  */
P = {0.25  0.50  0.20 ,
     0.75  0.20  0.15 ,
       .   0.30  0.25 ,
       .     .   0.40 };
```

In order to simulate ordinal variables, it is useful to define helper functions that take a probability matrix and return information about the probability mass functions, such as the expected value and the variance. The following statements define four helper functions:

```
/* OrdN: number of values for each variable */
start OrdN(P);
   return( countn(P, "col") );
finish;

/* OrdMean: Expected value for each variable is Sigma_i (i*p[i])    */
start OrdMean(P);
   x = T(1:nrow(P));                  /* values of ordinal vars      */
   return( (x#P)[+,] );               /* expected values E(X)        */
finish;

/* OrdVar: variance for each variable */
start OrdVar(P);
   d = ncol(P);   m = OrdMean(P);
   x = T(1:nrow(P));                  /* values of ordinal vars      */
   var = j(1, d, 0);
   do i = 1 to d;
      var[i] = sum( (x - m[i])##2 # P[,i] );    /* defn of variance */
   end;
   return( var );
finish;

/* OrdCDF: Given PMF, compute CDF = cusum(PDF) */
start OrdCDF(P);
   cdf = j(nrow(P), ncol(P));         /* cumulative probabilities    */
   do i = 1 to ncol(P);
      cdf[,i] = cusum(P[,i]);
   end;
   return( choose(P=., ., cdf) );     /* missing vals for short cols */
finish;
```

These functions compute properties of the ordinal random variables that are defined by the columns of a probability matrix:

- The ORDN function returns the number of categories for each random variable. The result is a row vector.
- The ORDMEAN function returns the expected values of each random variable. The result is a row vector.
- The ORDVAR function returns the variance of each random variable. The result is a row vector.

- The ORDCDF function returns the cumulative probabilities that are associated with each random variable. The result is a matrix that has the same dimensions as the input probability matrix.

The following statements demonstrate how to use the functions. The results are shown in Figure B.1.

```
/* test the functions for the ordinal PMF */
Descriptive = OrdN(P) // OrdMean(P) // OrdVar(P);
varNames = "X1":"X3";
print Descriptive[r={"N" "Mean" "Var"} c=varNames];

cdf = OrdCDF(P);
print cdf;
```

Figure B.1 Parameters and CDF for Ordinal Variables

Descriptive	X1	X2	X3
N	2	3	4
Mean	1.75	1.8	2.85
Var	0.1875	0.76	1.3275

cdf		
0.25	0.5	0.2
1	0.7	0.35
.	1	0.6
.	.	1

Figure B.1 shows information about each ordinal random variable. The first table shows the number of values, the expected value, and the variance of each ordinal distribution. The second table shows the CDF for each distribution. Notice that missing values are present in the **cdf** matrix in exactly the same positions as in the **P** matrix.

So that certain computations can be vectorized, it is also useful to introduce the EXPAND2DGRID function, which takes two vectors and returns a two-column matrix that contains all pairwise combinations of elements in the two vectors:

```
/* Function that returns ordered pairs on a uniform grid of points.
   Return value is an (Nx*Ny x 2) matrix */
start Expand2DGrid( _x, _y );
   x  = colvec(_x); y  = colvec(_y);
   Nx = nrow(x);    Ny = nrow(y);
   x = repeat(x, Ny);
   y = shape( repeat(y, 1, Nx), 0, 1 );
   return ( x || y );
finish;
```

If you have a vector of values $x = (x_1, x_2, \ldots, x_{N_x})$, and a second vector of values $y = (y_1, y_2, \ldots, y_{N_y})$, then the EXPAND2DGRID function returns the following ordered pairs:

$$\begin{aligned}&(x_1, y_1), \ldots, (x_{N_x}, y_1),\\&(x_1, y_2), \ldots, (x_{N_x}, y_2), \ldots,\\&(x_1, y_{N_y}), \ldots, (x_{N_x}, y_{N_y}).\end{aligned}$$

For example, the following statements call the EXPAND2DGRID function on a small example with $N_x = 2$ and $N_y = 3$. The results are shown in Figure B.2.

```
/* test the function */
g = Expand2DGrid({0 0.524}, {-0.84 -0.38 0.25});
print g;
```

Figure B.2 Expanding Two Vectors into a Grid of Values

g	
0	-0.84
0.524	-0.84
0	-0.38
0.524	-0.38
0	0.25
0.524	0.25

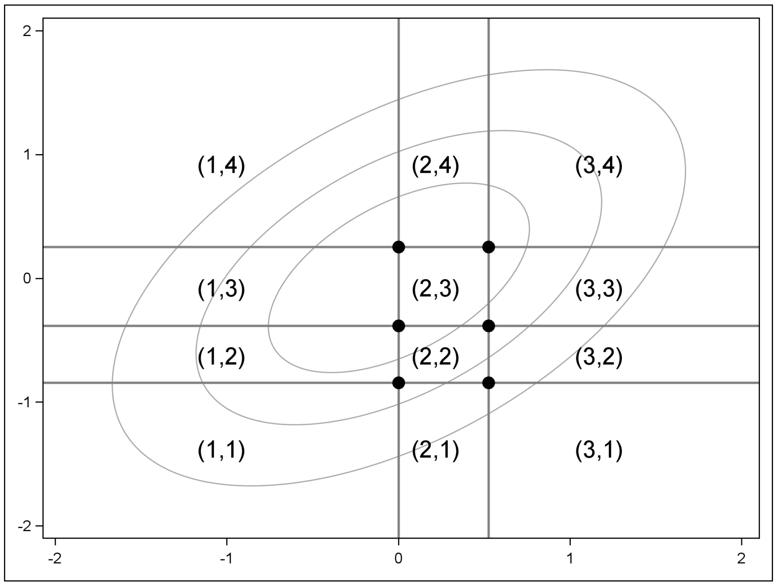
B.3 Overview of the Mean Mapping Method

The paper by Kaiser, Träger, and Leisch (2011) contains two algorithms. The algorithm presented here is called the "mean mapping method." This algorithm generalizes the Emrich-Piedmonte method (Emrich and Piedmonte 1991), which is described in Section 9.2 of Wicklin (2013). Given marginal PMFs for the ordinal variables and a target correlation matrix that describes their pairwise correlations, the mean mapping algorithm is as follows:

1. (Optional) Check that it is feasible to solve the problem. This appendix skips this step.
2. For each pair of ordinal variables, X_i and X_j, solve a complicated equation that involves the cumulative distribution function for the standard bivariate normal distribution. The solution to the equation gives a pairwise correlation, ρ_{ij}. Let R be the matrix with $R_{ij} = \rho_{ij}$.
3. Generate multivariate normal variates $(Y_1, Y_2, \ldots, Y_k) \sim \text{MVN}(0, R)$.
4. Use quantiles of the univariate standard normal distribution to convert the normal variates (Y_j) into the ordinal variates (X_j).

The idea of the algorithm is shown in Figure B.3. You generate multivariate normal data according to some correlation matrix, R. You can then use the probability of each marginal category to discretize the data. In the figure, which depicts the geometry for two ordinal random variables, normal bivariate points with coordinates in the lower left corner are assigned the value 1 for both ordinal variables. Points in the upper left corner are assigned 1 for the first ordinal variable and 4 for the second variable. Points near the origin are assigned to 1 or 2 for the first variable and 3 for the second variable. Other points are assigned values in a similar manner.

Figure B.3 Generate Correlated Ordinal Values from Bivariate Normal Data



Because of the discretization process, the correlation between the ordinal variables will usually be different from the correlation of the multivariate normal distribution from which the variables were generated. The purpose of Step 2 is to find the correlation matrix R that will result in the desired correlations between the ordinal variables.

B.4 Computing the Normal Quantiles of the Ordinal Probabilities

Several steps of the algorithm use univariate normal quantiles. If an ordinal variable has the PMF $\{0.5, 0.2, 0.3\}$, then the associated CDF is $\{0.5, 0.7, 1\}$. The corresponding normal quantiles are $\{0, 0.524, \infty\}$, since, for example, $\Phi^{-1}(0.7) = 0.524$, where Φ is the standard normal distribution function. These quantiles are used in Step 2 (solve a complicated equation) and also in the final step to convert normal variates into ordinal variates.

The quantiles cut the real line into disjoint intervals $I_1 = (-\infty, 0)$, $I_2 = [0, 0.542)$, and $I_3 = (0.542, \infty)$, which are shown as vertical bands in Figure B.3. Normal variates for this coordinate are converted into ordinal values based on these intervals. For example, the normal variates $\{0.12, 2.21, 1.65, -1.43\}$ are converted to the ordinal values $\{2, 3, 3, 1\}$ because the first value is in the interval I_2, the second value is in I_3, and so forth.

The following function computes the normal quantiles that are associated with the ordinal probabilities. The results are shown in Figure B.4.

```
/* OrdQuant: Compute normal quantiles for CDF(P) */
start OrdQuant(P);
   N = OrdN(P);
   CDF = OrdCDF(P);
   /* QUANTILE function in SAS/IML 12.1 does not accept 1 as parameter */
   /* Replace 1 with missing value to prevent error */
   CDF = choose(CDF > 1 - 2e-6, ., CDF);
   quant = quantile( "Normal", cdf );
   do j = 1 to ncol(P);      /* set upper quantile to .I = infinity */
      quant[N[j],j] = .I;    /* .I has special meaning to BIN func  */
   end;
   return( quant );
finish;

/* test the function */
quant = OrdQuant(P);
print quant;
```

Figure B.4 Normal Quantiles for Ordinal CDFs

quant		
-0.67449	0	-0.841621
I	0.5244005	-0.38532
.	I	0.2533471
.	.	I

Notice that the regions shown in Figure B.3 correspond to the normal quantiles defined by the second and third column of the example matrix, **`P`**.

B.5 Solving for the Intermediate Correlations

In Kaiser, Träger, and Leisch (2011), the algorithm that solves for the intermediate correlations is not efficient. A more efficient alternative is presented here. In the following function, a target value (called h^* in Kaiser, Träger, and Leisch (2011, p. 18)) is supplied to the function, along with the PMFs of X_i and X_j. The function uses a root-finding method to solve for the bivariate normal correlation such that a certain sum equals the target value. In the example shown in Figure B.3, the sum is over the probabilities defined by the six points that are located at the intersection of the vertical and horizontal lines. Notice also from Figure B.2 that the EXPAND2DGRID function can be used to generate the intersection points given the values of the vertical and horizontal lines. The following function finds the correlation given the target value:

```
/* OrdFindRoot: Use bisection to find the MV normal correlation that
   produces a specified MV ordinal correlation. */
start OrdFindRoot(P1, P2,  target);
   N1 = countn(P1);   N2 = countn(P2);
   q1 = OrdQuant(P1); q2 = OrdQuant(P2);
   v1 = q1[1:N1-1];   v2 = q2[1:N2-1];
   g = Expand2DGrid(v1, v2);
   /* find value of rho so that sum(probbnrm(g[,1], g[,2], rho))=target */
   /* Bisection: find root on bracketing interval [a,b] */
   a = -1; b = 1;                 /* look for correlation in [-1,1] */
   dx = 1e-8; dy = 1e-5;
   do i = 1 to 100;               /* iterate until convergence      */
      c = (a+b)/2;
      Fc = sum( probbnrm(g[,1], g[,2], c) ) - target;
      if (abs(Fc) < dy) | (b-a)/2 < dx then
         return(c);
      Fa = sum( probbnrm(g[,1], g[,2], a) ) - target;
      if Fa#Fc > 0 then a = c;
      else b = c;
   end;
   return (.);                    /* no convergence                 */
finish;
```

The ORDFINDROOT function is called inside a double loop from the ORDMVCORR function, which returns the entire intermediate correlation matrix, R, as follows:

```
/* OrdMVCorr: Compute a MVN correlation matrix from the PMF and
   the target correlation matrix for the ordinal variables. */
start OrdMVCorr(P, Corr);
   d = ncol(P);
   N = OrdN(P);
   mean = OrdMean(P);
   var  = OrdVar(P);
   cdf  = OrdCDF(P);
   R = I(d);
   do i = 1 to d-1;
      sumCDFi = sum(cdf[1:N[i]-1, i]);
      do j = i+1 to d;
         sumCDFj = sum(cdf[1:N[j]-1, j]);
         hStar = Corr[i,j] * sqrt(var[i]*var[j]) + mean[i]*mean[j]
                 - N[i]*N[j] + N[i]*sumCDFj + N[j]*sumCDFi;
         R[i,j] = OrdFindRoot(P[,i], P[,j], hStar);
         R[j,i] = R[i,j];
      end;
   end;
   return(R);
finish;
```

You can test the ORDMVCORR function by calling it on a matrix that specifies the correlations between three ordinal variables. The function returns an intermediate correlation matrix that can be used to generate normal variates. The matrix is shown in Figure B.5.

```
/* test the function */
Corr = {1.0  0.4  0.3,
        0.4  1.0  0.4,
        0.3  0.4  1.0 };
R = OrdMVCorr(P, Corr);
print R;
```

Figure B.5 Intermediate Correlation Matrix

R		
1	0.6615601	0.4302979
0.6615601	1	0.5136719
0.4302979	0.5136719	1

B.6 Simulating Multivariate Ordinal Variates

The main function—and the only one that you need to call explicitly to simulate ordinal data—is the RANDMVORDINAL function, which carries out the remaining steps of the algorithm. The ORDQUANT function and the SAS/IML BIN function are used to convert the multivariate normal variates into ordinal variates.

```
/* RandMVOrdinal:
   N     Number of desired observations from MV ordinal distribution,
   P     Matrix of PMF for ordinal vars. The j_th col is the j_th PMF.
         Use missing vals if some vars have fewer values than others.
   Corr  Desired correlation matrix for ordinal variables. Not every
         matrix is a valid as the correlation of ordinal variables. */
start RandMVOrdinal(N, P, Corr);
   d = ncol(P);
   C = OrdMVCorr(P, Corr);     /* 1. compute correlation matrix, C  */
   mu = j(1, d, 0);
   X = RandNormal(N, mu, C);   /* 2. simulate X ~ MVN(0,C)          */
   N = OrdN(P);
   quant = OrdQuant(P);        /* compute normal quantiles for PMFs */
   do j = 1 to d;              /* 3. convert to ordinal             */
      X[,j] = bin(X[,j], .M // quant[1:N[j],j]);
   end;
   return(X);
finish;
```

You can test the RANDMVORDINAL function by simulating 1,000 observations from a trivariate ordinal distribution. The empirical means and correlations of these observations are shown in Figure B.6.

```
/* Test: simulate 1000 obs from MV ordinal distribution */
call randseed(1);
X = RandMVOrdinal(1000, P, Corr);
mean = mean(X);
S = corr(X);
print mean, S;
```

Figure B.6 Sample Mean and Correlation for Ordinal Data

mean		
1.767	1.855	2.894

S		
1	0.3904562	0.3138122
0.3904562	1	0.3972897
0.3138122	0.3972897	1

As shown in Figure B.6, the sample means and correlations are close to the population means and correlations, which are shown in Figure B.1 and in the program statements that precede Figure B.5.

B.7 References

Demirtas, H. (2006), "A Method for Multivariate Ordinal Data Generation Given Marginal Distributions and Correlations," *Journal of Statistical Computation and Simulation*, 76, 1017–1025.

Emrich, L. J. and Piedmonte, M. R. (1991), "A Method for Generating High-Dimensional Multivariate Binary Variables," *American Statistician*, 45, 302–304.

Kaiser, S., Träger, D., and Leisch, F. (2011), *Generating Correlated Ordinal Random Values*, Technical report, University of Munich, Department of Statistics.
URL `http://epub.ub.uni-muenchen.de/12157/`

Wicklin, R. (2013), *Simulating Data with SAS*, Cary, NC: SAS Institute Inc.