Lab 2

Psychology 319 (GCM)

Instructions. Work through the lab, saving the output as you go. If you work in Microsoft Word, you can easily copy any graph to Word via the clipboard. Numerical output may also be copied easily by highlighting, moving it to the clipboard, then copying into Word. However, you should format R output in TrueType Courier New font so that it is *monospaced*. Output from this lab is to be handed in by Monday, February 8. Your output file should be named `LAST_FIRST_LAB2.DOC`, where `LAST` is your last name, and `FIRST` is your first name. Any additional files should have the same naming scheme, except the file extension should be correct. You may add any description text you wish after `LAB2` in the file name.

Preamble. Today's lab involves the use of R's simulation capabilities to explore certain aspects of multilevel models and longitudinal data.

1 Introduction

One of the nice things about R is the ease with which it can be used to do Monte Carlo simulations. We shall employ some R library routines I have created. These can be downloaded, along with the documentation, from the *R Code and Support Materials* section of the course website. After downloading the R file to your working directory (remember to right-click, add the .R extension, and save it with the "all files" attribute), you can add the commands in my library to your session with the command

```
> source("Steiger R Library Functions.R")
```

Several of these functions are highly specialized for use in multivariate analysis. However, two functions will be very useful for us:

1. `MultivariateNormalSample` creates a simulated sample from a multivariate normal distribution with desired mean vector and covariance matrix.

2. `MakeExactData` creates a simulated sample with sample mean and covariance matrix exactly equal to what is requested.

You should consult the documentation file to see how to call these functions, but there are also some examples below.

We use `MultivariateNormalSample` to analyze how statistics perform in their assumed natural environment, that is, when a particular population situation holds and statistics are used on a sample to estimate the nature of the population. We use `MakeExactData` for two distinct purposes:

1. To create data whose means and covariances (or correlations) match some published results or examples that do not include the raw data, so that we can replicate the analyses that were published.

2. To investigate the truth of general statements about the relationships between quantities that are solely a function of variances, covariances, and means.

Consider a simple example of the second usage. Suppose you didn't know much about statistics and someone told you that the correlation between X and Y always satisfies the rule

$$r_{x,y} = \frac{s_{x,y}}{s_x s_y} \tag{1}$$

If you had computational routines for computing covariances, correlations, and standard deviations, you could, in a few minutes, create thousands of examples of real data with a given $r_{x,y}$ and quickly verify that the relationship is never falsified. While lack of falsification is not proof of truth, it can be very suggestive.

What many people fail to realize is that the same principle can be extended beyond sample statistics to population statistics, because any sample of size n can be considered to be a discrete multivariate population where every vector of results has equal probability. So *if* a theoretical result is rendered in terms of only means, variances, covariances, and correlations, we can examine it using essentially the same technique.

Let me illustrate this with a very basic example. Suppose I set the random seed to 12345 and generate a set of 3 observations with a sample correlation of exactly .50. Notice how I use the function `CompleteSymmetricMatrix` to produce the input covariance matrix. This isn't necessary with a 2×2 covariance matrix, but saves a substantial amount of time if you have to enter a larger matrix. You simply enter the lower triangle, and the function produces the complete matrix.

```
> set.seed(12345)
> means <- c(0,0)
> covariance.matrix <- CompleteSymmetricMatrix(c(1,.5,1))
> data <- MakeExactData(means,covariance.matrix,3)
> data
            [,1]        [,2]
[1,]  1.01823865  0.03753419
[2,] -0.03753419  0.98070446
[3,] -0.98070446 -1.01823865
> cov(data)
     [,1] [,2]
[1,]  1.0  0.5
[2,]  0.5  1.0
> column.means <- apply(data,2,mean)
> round(column.means,10)
[1] 0 0
```

As you can see, the data have, within an acceptable level of rounding error, precisely the statistical sample values that were requested. If we look at the 3 vectors of scores in the object `data`, we realize that if population covariances were defined in terms of a denominator of $n - 1$, we could consider the scores to be the 3 equally likely outcomes in a discrete bivariate distribution with means of $0, 0$ and a covariance matrix having variances of 1 and covariances of .5. Remember, though, that the population covariances are computed with n in the denominator. We can quickly create a function to compute them.

```
> pop.cov <- function(data){
+ n <- length(data[,1])
+ return(((n-1)/n)*cov(data))
+ }
> pop.cov(data)
          [,1]      [,2]
[1,] 0.6666667 0.3333333
[2,] 0.3333333 0.6666667
```

In other words, although the above data have sample variances of 1 and a sample covariance of .5, if they were to be considered the entire population, then they would have variances of $2/3$ and covariances of $1/3$.

It would be a routine matter to adjust the data so that the population covariances and means were as desired. Actually, the function `MakeExactData` will do this for you, if you add the statement `use.population = TRUE` to the input parameters. Below, we see that, using this input parameter, we can create a "statistical population" with exactly the "population means" and "population covariance structure" that we want.

```
> data2 <- MakeExactData(means,covariance.matrix,3,use.population=TRUE)
> pop.cov(data2)
     [,1] [,2]
[1,]  1.0  0.5
[2,]  0.5  1.0
> apply(data2,2,mean)
[1] 1.849468e-17 0.000000e+00
```

2 Difference Scores and their Reliability

Last week, you derived some results on the reliability of difference scores, and we'll begin by using R to produce data that demonstrate some of the theoretical results. Then we'll move on to produce more general multilevel data. Let's generate some artificial data that occur at two time periods. We can generate the data via a classical true score model, and have the distinct advantage of knowing the true scores as well as the observed scores.

There are several versions of the true score model that we could work with. Let's work with the following simplified model, given in the last homework assignment.

1. $X_i = T_i + E_i, \quad i = 1, 2$
2. T_1 and T_2 correlate ρ_{12}
3. The T's the E's are uncorrelated
4. The T's have variances of ρ_x, the E's have variances $1 - \rho_x$, so the X's have variances of 1
5. The T's and E's are assumed to have means of zero
6. The T's and E's are generated from unit variance variables ξ and ϵ via the formulas $T_i = \rho_x^{1/2}\xi_i$ and $E_i = (1 - \rho_x)^{1/2}\epsilon_i$

I asked you on the homework assignment to come up with a bunch of formulas. One formula I gave you to prove is

$$\rho_{x_1,x_2} = \rho_x \rho_{12} \tag{2}$$

If you look carefully at the assumptions I listed above, you can see that, if we place the variables ξ_1, ξ_2, ϵ_1, and ϵ_2 in a vector, it would have a variance-covariance matrix given by

$$\begin{bmatrix} 1 & \rho_{12} & 0 & 0 \\ \rho_{12} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

So, creating the T, E, and X variables is quite straightforward. Let's write some R functions to do that. Read the comments carefully and make sure you understand how this routine works.

```
> create.X1X2.data <- function(rho12,rhox,n,use.population=FALSE)
+ {
+ means <- c(0,0,0,0)
+ covariance.matrix <- CompleteSymmetricMatrix(c(1,rho12,1,0,0,1,0,0,0,1))
+ data <- MakeExactData(means,covariance.matrix,
+ n,use.population)    ## Create data for xi's and epsilons
+ xi1 <- data[,1]      ## Grab variables
+ xi2 <- data[,2]      ## from appropriate columns
+ epsilon1 <- data[,3] ## of the
+ epsilon2 <- data[,4] ## data matrix
+ T1 <- sqrt(rhox)*xi1 ## Then create
+ T2 <- sqrt(rhox)*xi2 ## T and E variables
+ E1 <- sqrt(1-rhox)*epsilon1
+ E2 <- sqrt(1-rhox)*epsilon2
+ X1 <- T1 + E1        ## Next create X1 and X2
+ X2 <- T2 + E2        ## Put vars in a data frame & return
+ return(data.frame(X1,X2,T1,T2,E1,E2))
+ }
```

Suppose we now run this routine and create some data.

```
> data2 <- create.X1X2.data(0.50,0.80,100,use.population=TRUE)
> cor(data2)
```

```
              X1            X2            T1            T2            E1
X1  1.000000e+00  4.000000e-01  8.944272e-01  4.472136e-01  4.472136e-01
X2  4.000000e-01  1.000000e+00  4.472136e-01  8.944272e-01 -9.990715e-16
T1  8.944272e-01  4.472136e-01  1.000000e+00  5.000000e-01 -4.789514e-17
T2  4.472136e-01  8.944272e-01  5.000000e-01  1.000000e+00 -1.067073e-16
E1  4.472136e-01 -9.990715e-16 -4.789514e-17 -1.067073e-16  1.000000e+00
E2 -8.514050e-16  4.472136e-01  5.966110e-17 -1.948826e-16 -2.012898e-15
              E2
X1 -8.514050e-16
X2  4.472136e-01
T1  5.966110e-17
T2 -1.948826e-16
E1 -2.012898e-15
E2  1.000000e+00
```

From the correlation matrix, we can see that the correlation between X_1 and X_2 is indeed equal to $0.50 \times 0.80 = 0.40$. So, for this one randomly created data set, the rule holds. Let's try it again.

```
> data2 <- create.X1X2.data(0.40,0.90,100,use.population=TRUE)
> round(cor(data2),3)
      X1    X2    T1    T2    E1    E2
X1 1.000 0.360 0.949 0.379 0.316 0.000
X2 0.360 1.000 0.379 0.949 0.000 0.316
T1 0.949 0.379 1.000 0.400 0.000 0.000
T2 0.379 0.949 0.400 1.000 0.000 0.000
E1 0.316 0.000 0.000 0.000 1.000 0.000
E2 0.000 0.316 0.000 0.000 0.000 1.000
```

This time, we used input values of .40 and .90, and again observe the expected result, i.e., the correlation between X_1 and X_2 is indeed equal to $0.40 \times 0.90 = 0.36$.

Problem 1. Another formula I gave you is

$$\rho_D = \frac{\rho_x - \rho_{x_1,x_2}}{1 - \rho_{x_1,x_2}} \tag{4}$$

where ρ_D is the reliability of D, ρ_x is the reliability of X_1 and of X_2, and ρ_{x_1,x_2} is the correlation between the observed scores X_1 and X_2.

Following the approach I used above, create two data sets that corroborate the formula for the reliability of the difference score in the population.

Demonstrate that the relationship shown in the formula for ρ_D holds in your data.

3 Multilevel Data

Creating simulated multilevel data with various characteristics is also possible with R. Again, the advantage we have is that we can actually know the population parameters that yielded the data. Let's choose a simple two-level model like the one in the lecture notes. At level 1, we have

$$Y_{ij} = \beta_{0j} + \beta_{1j} X_{ij} + r_{ij} \tag{5}$$

The r_{ij} are regression residuals and are assumed to be independently and identically normally distributed with a mean of 0 and a variance of σ^2.

At level 2, we have

$$\beta_{0j} = \gamma_{00} + u_{0j} \tag{6}$$
$$\beta_{1j} = \gamma_{10} + u_{1j} \tag{7}$$

The β_0s and β_1s are allowed to correlate, so they have a covariance matrix $\boldsymbol{T}$ that is not necessarily diagonal. Their distribution is bivariate normal, as is the joint distribution of the u_{0j} and u_{1j}. Another way of writing the above assumptions is that (β_{0j}, β_{1j}) is a vector with a bivariate normal distribution having a mean vector of $(\gamma_{00}, \gamma_{10})$ and a covariance matrix of

$$\boldsymbol{T} = \begin{bmatrix} \tau_{11} & \tau_{21} \\ \tau_{21} & \tau_{22} \end{bmatrix} \tag{8}$$

Problem 2. Write your own routine to create the data, but also keep the actual β values as part of the output. The n_j sample sizes should vary randomly between 20 and 50 across j. There should be 50 schools. To keep life simple, assume just one classroom for each school, so there is no need to keep separate ids for the classrooms. Your routine should return a data frame that includes a school id `id2` and a subject id `id1`.

I suggest the following approach:

1. Assume that the (β_{0j}, β_{1j}) vector has a bivariate normal distribution with mean vector $(8.00, 0.70)$ and covariance matrix

$$\boldsymbol{T} = \begin{bmatrix} 20.00 & -0.72 \\ -0.72 & 0.03 \end{bmatrix} \tag{9}$$

   These values are similar to what we saw in the JSP file.

2. For each school(class), sample the β values randomly, using my multivariate normal sampling routine, and randomly sample the number of students uniformly from the interval (20,50).

3. For each student, sample a residual r_{ij} randomly from a normal distribution with a mean of 0 and a variance of 30.

Now comes the interesting part, the X_{ij}. Note that the standard approaches to multilevel modeling leave a great big blank here by assuming that the X_{ij} values are fixed, or given. In other words, they are not modeled as having arisen through a sampling process. Now, in a number of cases the X_{1j} values are, in fact, fixed. However, in this case, they clearly arise through a random sampling process, even if all the students in a given class are included in the experiment.

In other words, when we sample a school and a class within that school, we not only sample β values, we also sample a set of X values along with them. We can view these, within each school, as a "mini-population."

Without extensive information about the nature of the experiment, it may be difficult and/or presumptuous to propose a sampling model for the classrooms. However, clearly, the characteristics of the X_{ij} vary across classrooms.

In our case, let's keep things simple. Let's assume that the population mean for the Raven test is 25, and the population variance is also 25, and all variation across classrooms is essentially random. So simply sample n_j values from the normal distribution with mean 25 and standard deviation 5. After constructing the Y values, round off the X and Y values to integer form. Finally, put everything into a data frame, including X, Y, `id1`, `id2`, β_{0j}, β_{1j}, and r_{ij} values for each student.

4 Analyzing the Multilevel Data

Problem 3. Once you have your data file, analyze the multilevel model using R just as we did in class. Produce a plot of the individual school trajectories as estimated from the sample data. Compare the fixed and random effect estimates in the output from R with the numbers we used to construct the data.