Determining Power using Simulation

This program shows how you might obtain an empirical estimate of power for a coefficient in a multiple regression analysis using simulation.

Consider a researcher who wants to determine the effect of independent variable *X* on dependent variable *Y*, controlling for covariates C1 and C2. A literature search indicates that the predictor variables (*X*, *C1*, and *C2*) have the following correlation matrix: $\begin{bmatrix} 1 & .2 & .2 \\ .2 & 1 & .2 \\ .2 & .2 & 1 \end{bmatrix}$. The literature indicates that the variances of the predictors are $\begin{bmatrix} 5 \\ 3 \\ 7 \end{bmatrix}$. The means of the predictors are $\begin{bmatrix} 20 \\ 5 \\ 10 \end{bmatrix}$. The regression coefficients are expected to be $\begin{bmatrix} 1 \\ 2 \\ .5 \end{bmatrix}$ and the residual variance is expected to be 3.7. The researcher plans to collect a sample of size *n*=30, and wants to know her power to detect a significant effect for the partial regression coefficient predicting *Y* from *X* at alpha equals .05.

1. Specify the number of replications to be used for the simulation and the random number seed.

```
proc iml;
   nRep=1000;
   call randseed(112358);
```

2. Enter the study parameters (sample size, mean vector, correlation matrix, variance vector, beta coefficients, and residual variance).

```
   n=30;

   mean={20,
         5,
         10};

   corr={1 .2 .2,
         .2 1 .2,
         .2 .2 1};

   var={5,
        3,
        7};

   beta={ 1,
          2,
         .5};

   resvar=14;
```

3. Before simulating the predictor variables, you need to calculate the covariance matrix for the predictor variables from the variance vector and correlation matrix. Create a matrix called **sddiag**, which is a diagonal matrix containing the standard deviations of the variables on the diagonal. Next, calculate the covariance matrix by pre-multiplying and post-multiplying the correlation matrix by **sddiag**.

```
   sddiag=diag(sqrt(var));
   cov=sddiag * corr * sddiag;
```

4. The distribution of the predictor variables is determined by the mean vector and covariance matrix that you entered earlier. Notice that the number of observations that you want to draw is specified as **(n * nrep)**. Rather than using a loop to draw 30 observations at a time for each of 1000 iterations, you draw all 30000 observations in a single call of the RANDNORMAL function. This helps improve the efficiency of the program.

```
   x=randNormal(n*nRep,mean,cov);
```

5. The vector of predicted scores for the dependent variable, **ypred**, is calculated as the matrix of predictor variables post-multiplied by the vector of beta coefficients. The error values are drawn from a univariate normal distribution centered at zero with a spread equal to the square root of **resvar** (the residual standard deviation). The dependent variable is calculated as the sum of the predicted scores and the errors.

```
   yPred=x * beta;
   error=randfun(n*nrep,"Normal",0,sqrt(resvar));
   y=yPred + error;
```

6. Matrix **x** and vector **y** contain the simulated values for the independent and dependent variables, respectively, across all iterations. They do not indicate which observations correspond to which iterations. The column variable indicating iteration needs to contain the number 1 listed 30 times followed by the number 2 listed 30 times, and so on. This vector could easily be created in a loop, as shown in the commented code above. Alternatively, you can improve the efficiency by removing the loop and using a combination of REPEAT and COLVEC statements. The statement `temp=repeat((1:nrep)`,1,n);` creates an **n** x **nrep** (in this case, 30 x 1000) matrix named **temp**. Each column contains the numbers 1 through **nrep**. The statement `iteration=colvec(temp);` converts each row to a column vector and then stacks the column vectors. The final result is **iteration**, a 30000 x 1 column vector.

```
   temp=repeat((1:nrep)`,1,n);
   iteration=colvec(temp);
```

7. Combine the iteration numbers, the predictor variable values, and the dependent variable values into a single matrix for output. Then output the matrix to a data set called **temp** and store all of the matrices for later use.

```
   sampleData=iteration || x || y;

   create temp from sampleData [colname={iteration x c1 c2 y}];
   append from sampleData;
   close temp;

   store;
quit;
```

8. The GLM procedure can be used to perform regression analyses. The DATA command specifies our newly created **temp** data set as the data set for analysis. The NOPRINT option suppresses the printing of results. OUTSTAT= outputs the results from GLM to a data set called **regResults**. The BY statement requests that a separate analysis be performed for each value of **iteration**. The MODEL statement specifies the regression model as predicting **y** from **x**, **c1**, and **c2**.

```
proc glm data=temp noprint outstat=regResults;
   by iteration;
   model y=x c1 c2;
run;quit;
```

9. The **regResults** data set contains more information than you want. It has Type I and Type III tests of all three regression parameters for each iteration. Read the data set into IML using a WHERE statement to read in only observations corresponding to Type III tests of the regression coefficient for **x**. Read in only the **prob** variable, which provides the *p*-value for the test of the regression coefficient.

```
proc iml;
   use regResults;
     read all var {prob}
       where(_SOURCE_='X' & _TYPE_='SS3') into prob;
   close regResults;
```

10. Finally, calculate the proportion of elements in **prob** that are less than .05 and output the result.

```
   significant=prob < .05;
   power=significant[:,];
   print power;
quit;
```

power
0.806

You see that the empirical estimate for the power to detect a significant partial regression coefficient for *X* is .806.

Performing this power analysis in the POWER procedure would yield an analytical estimate of power at .799. The empirical estimate of power would converge to this number for larger sample sizes (for example, one million iterations).