

Paper SD-02

Random Effects Simulation for Sample Size Calculations Using SAS ®

Matthew Psioda, Department of Biostatistics,
The University of North Carolina at Chapel Hill, Chapel Hill, NC

ABSTRACT

Sample size calculations are a critical step in the planning of any experiment. In all but the simplest of experimental designs, closed-form equations are not readily available, and statisticians are required to use simulations to estimate an appropriate sample size for the experiment. Specifically, when multiple explanatory variables are thought to be predictive of the response or when missing data is likely to occur, simulation is a valuable approach for sample size calculation.

In this paper we consider a simulation study for a continuous response measured at a set of fixed time points. We consider a randomized study comparing an experimental treatment plus standard of care (ET+SOC) to the standard of care (SOC). We assume that, based on data from a previous study, the response trajectory for the SOC subjects varies with respect to gender and that the response curves are reasonably well modeled by a quadratic polynomial in time.

Using PROC IML, we simulate data from a linear mixed effects model including a gender main effect, linear and quadratic time, and treatment by time interactions. For this scenario, the primary (null) hypothesis is that there is no treatment effect. Model fitting is performed using PROC MIXED. Our technique for simulation is easily generalizable and efficient.

INTRODUCTION

Power simulations can be broken down into a handful of steps. The first step is to describe, to the extent possible, the underlying distribution from which the data are thought to arise. Most often this involves making assumptions about the moments of that distribution based on empirical results from published literature, studies that have already been conducted which share characteristics with the study being planned, and information from subject matter experts.

For our discussion, we assume that data are available from a previous study, which included subjects receiving the SOC over a period of 5 weeks. Using those data we are able to obtain estimates for the nuisance mean model parameters, variance-covariance matrix of the random effects, and error variance. For the purposes of our discussion, we treat these estimates as the true values of the corresponding parameters. However, in practice one might consider repeating the simulation process under various perturbations of the nuisance parameters to assess the impact of minor variations on power.

The second step is to generate a large number of samples from the assumed true distribution using various sample sizes that are thought to be adequate to achieve the desired power. The true distribution will have moments based on the known values of the nuisance parameters and also specified values for the parameters of interest. In our example, the parameters of interest are the treatment by linear time and treatment by quadratic time interactions. A treatment effect exists precisely when at least one of these parameters is non-zero. In other words, the mean response curves are not coincident precisely when at least one of the parameters of interest is non-zero.

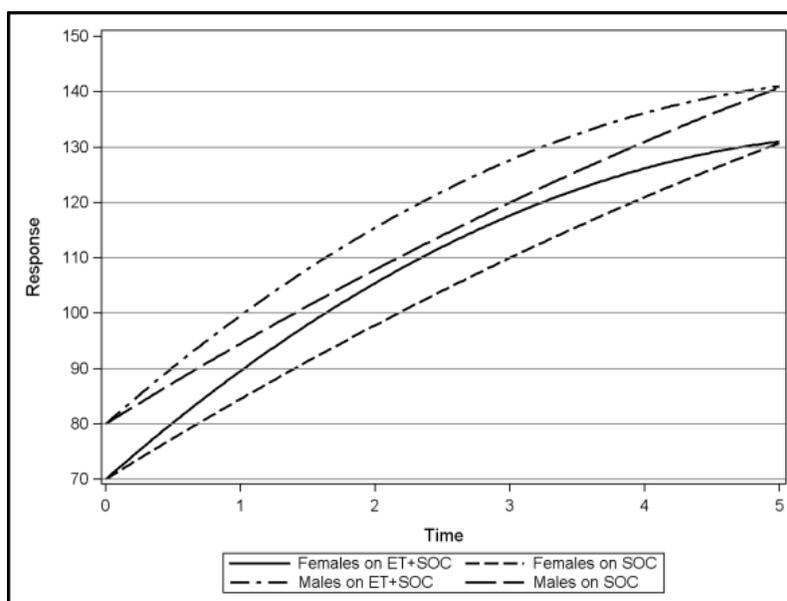
The third step is to fit the assumed model to the samples that have been generated. For each model fit, we perform a hypothesis test and determine if sufficient evidence exists to reject the null hypothesis for that sample. Once all sample data sets have been processed, we can use the testing results to estimate the power of the test. For preliminary simulations where the approximate sample size is not well known, we consider a wide range of sample sizes and use smoothing splines to get an approximation of the power function. Once we have a reasonable idea of what sample size is appropriate, we generate a large number of samples of that sample size and repeat the testing process to confirm the choice.

DESIGN OF THE EXAMPLE STUDY

In order to set up a simulation, we must first fully understand the design of the study being planned. For our purposes we will work with a simple randomized, two arm study comparing the standard of care (SOC) to the standard of care plus a new experimental add-on treatment (ET+SOC). We will refer to the latter group as ET for the remainder of the paper. Adults having the illness in question (defined as baseline response values of approximately 70 for females and 80 for males) will be randomized to one of the two treatment groups. The randomization will be stratified by gender so that the arms will have approximate balance with respect to gender. Response levels vary with gender but the effect of treatment is believed to be independent of gender.

Past research has shown that the SOC is effective at treating the illness. That is, after about 5 weeks of treatment with SOC, response values return to normal (approximately 130 for females and 140 for males). The researchers hypothesize that the ET treated subjects will have a quicker rate of improvement during the early stages of treatment leading to a shorter period of severe illness. For this particular illness, symptom severity is directly related to response levels so the hypothesized effect of ET is desirable. The following figure presents the hypothesized response trajectories for the study population.

Figure 1: Hypothesized Mean Response Trajectories



Such differences in the curves would be clinically meaningful. The primary goal for the study is to determine if over the five week period, the mean response curves differ for the ET group as compared to SOC. For our simulation, we will formulate this as a two-sided statistical test of whether or not the response curves are coincident. Measurements of the response will be collected prior to randomization (baseline) and at each week over the five week period.

MODEL SPECIFICATION

We begin with a formal statement of the model from which samples will be generated. The general linear mixed model, where i indexes the i^{th} subject, is given by

$$Y_i = X_i\beta + Z_i b_i + e_i,$$

where

- i. Y_i is the $n \times 1$ response data vector,
- ii. X_i is the $n \times p$ fixed effects design matrix with corresponding $p \times 1$ parameter vector β ,
- iii. Z_i is the $n \times k$ random effects design matrix with corresponding $k \times 1$ random effects vector b_i , and
- iv. e_i is the $n \times 1$ vector of random errors.

In this example, there are $n = 6$ measurement occasions, $p = 6$ fixed effects, and $k = 3$ random effects. Regarding the random components, we make the standard distributional assumptions.

- i. We assume that $\mathbf{b}_i \sim N_k(\mathbf{0}, \mathbf{D})$. In general, \mathbf{D} has no assumed structure. For our simulation, we assume the variance-covariance parameters are known (based on information from the previous study) and that the distribution does not vary with gender or treatment. For our simulation, the assumed matrix is provided below.

$$\mathbf{D} = \begin{bmatrix} 68.70 & -2.82 & -1.90 \\ -2.82 & 23.87 & -3.68 \\ -1.90 & -3.68 & 0.90 \end{bmatrix}$$

The three components of the random effects vector are the random intercept, random linear term, and random quadratic term, denoted as b_0 , b_1 , and b_2 , respectively.

- ii. We assume that $\mathbf{e}_i \sim N_n(\mathbf{0}, \sigma^2 \mathbf{I})$. That is, we assume that the random errors within a subject are independent so that all correlation between repeated measurements on a subject arises due to the random effects. We further assume \mathbf{b}_i is independent of \mathbf{e}_i . For our simulation, we will take $\sigma^2 = 169.20$.

We use a reference cell coding scheme so that the parameter vector $\boldsymbol{\beta}$ has the following form (values used in simulation are in parenthesis):

- i. β_0 is the intercept which will correspond to the females at baseline (70.00),
- ii. β_1 is the increment in the intercept for males (10.00),
- iii. β_2 is the linear time effect (15.10),
- iv. β_3 is the quadratic time effect (-0.59),
- v. β_4 is the increment in the linear time effect for ET subjects (6.30), and
- vi. β_5 is the increment in the quadratic time effect for ET subjects (-1.25).

It is important to have a firm grasp of the dimensions for each component of the model. This is especially the case for simulations since we must generate the data for each random and fixed component of the model according to the dimensions specified and combine them appropriately to obtain the response data.

CODING THE SIMULATION

In theory our task is quite simple. We must generate independent samples of a given size according to the model stated in the previous section, fit models to each sample, and determine the proportion of samples that lead to rejection of the null hypothesis. There are a variety of ways in which this process could be implemented. One might consider generating data for each subject in an iterative process and append the results together to form a complete sample. Conversely, one could generate an entire sample at once. Since we often need to generate a large number of data sets of a given sample size, a third option would be to generate all such data sets at one time. What is the best approach to implement? The answer depends on the size of the data sets being generated and the available system resources being used to execute the simulation. We present a SAS macro that provides the flexibility to scale the amount of data that is generated and analyzed at once. After discussing the SAS macro, we provide some general advice based on empirical results from simulations under a variety scenarios.

We break our discussion of the SAS macro into three parts. First we introduce the set of macro parameters that are used within the simulation macro and discuss the function of each. Next, we discuss the data generation component of the macro. We end with a discussion of the section of the macro that fits the models and estimates the power function.

MACRO PARAMETERS

For the sake of simplicity, we prefer to define the set of relevant macro variables just prior to invocation of the SAS macro with %LET statements. Just as easily one pass the values as parameters to the macro. The SAS code provided below creates the macro variables used for an example simulation. Throughout the paper, we will use the convention of presenting the SAS code with identifiers for key sections followed by a discussion of the code.

```

%let TimePointVec = {0,1,2,3,4,5};
%let Beta = {70.0, 10.0, 15.10, -0.59, 6.3, -1.25};
%let SigmaSQ = 169.2;
%let CovRandom = {68.70 -2.82 -1.90,
                  -2.82 23.87 -3.68,
                  -1.90 -3.68 0.90};

%let SampSizeList = 88|92|96|100|104;
%let Iterations = 2500;
%let NumIterPer = 250;
    
```

1. The first four macro variables define matrix, vector, and scalar components needed for the simulation process. In PROC IML, both matrices and vectors are defined using curly braces and a comma is used to indicate the end of a row. Hence, when referenced within PROC IML, "TimePointVec" will resolve to a 6×1 vector of measurement times, "Beta" will resolve to a 6×1 vector of mean model parameters, "SigmaSQ" will resolve to the scalar error variance, and "CovRandom" will resolve to a 3×3 matrix of covariance parameters for the random effects.
2. The second set of macro variables control aspects of the simulation process. The "SampSizeList" macro variable contains the set of sample sizes that will be investigated in the simulation. A vertical pipe is used as a delimiter for each distinct sample size. The "Iterations" macro variable controls the number of data sets of each sample size that are generated and analyzed during the simulation process. The "NumIterPer" macro variable determines the number of samples that are generated during each step of the simulation process. Based on the values defined in this example, in the first iteration 250 samples of size 88 are generated in a single data set. This data set is then by-processed using PROC MIXED and the results of the 250 hypothesis tests are stored. Next, 250 more data sets of size 88 are generated and so on until all 2500 data sets of each sample size have been generated and analyzed. When the value of "Iterations" equals the value of "NumIterPer", one (possibly very large) data set per sample size is generated and analyzed. The value of "Iterations" must be evenly divisible by the value of "NumIterPer".

CONTROLLING THE ITERATIVE PROCESS

As described above, the data set generation process can occur in a single step but most likely will be an iterative process. The need to iteratively repeat execution of SAS code can be done with a DO loop. One can use a DO loop within PROC IML or a macro %DO loop to execute any SAS code. Since we need to alternate generating data using PROC IML and fitting models to the data using PROC MIXED, we elect to use the macro %DO loop. The reader should note that it is possible to call PROC MIXED (and other SAS procedures) within PROC IML using SUBMIT blocks. We have not considered the efficiency of the proposed approach compared to the use of SUBMIT blocks. We now present the looping code which surrounds the body of the simulation code.

```

%macro Simulation;

%let OverallIteration = 1;
%let NumSizes = %eval(%sysfunc(count(&SampSizeList.,|))+1);

%do k = 1 %to &NumSizes.;
  %let SampSize = %scan(&SampSizeList.,&k,|);

  %do Iter = 1 %to %eval(&Iterations. / &NumIterPer.);

    [BODY OF THE SIMULATION CODE]

  %let OverallIteration = %eval(&OverallIteration.+1);
%end;
%end;

%mend Simulation;
    
```

1. The "OverallIteration" macro variable counts the number of times the iterative process has executed. It is assigned a value of 1 prior to the first iteration and is incremented by one using the %EVAL() macro function after the body of the simulation code. Note that since SAS macro variables are text strings, the use of %EVAL() is required in order for the SAS macro compiler to evaluate the arithmetic expression. Without it, after the first iteration the "OverallIteration" macro variable would have the value 1+1.
2. The "NumSizes" macro variable uses the %EVAL() and %SYSFUNC() macro functions to count the number of sample sizes that are stored in the "SampSizeList" macro variable. The %SYSFUNC macro function is arguably the most powerful macro function. It gives one the ability to use data step functions outside of the data step on macro variables and other text expressions. Note that although standard use of the COUNT() function requires a data step variable name or quoted string as the first and second arguments, this is not appropriate when used with the %SYSFUNC() macro function.
3. The outer %DO loop iterates the simulation process across the list of desired sample sizes.
4. The "SampSize" macro variable is created based on the current value of the index macro variable associated with the outer %DO loop. We use the %SCAN() macro function to choose the appropriate term (commonly called a word) from the "SampSizeList" macro variable value. Note here that we need not rely on the %SYSFUNC() macro function along with the SCAN data step function since a scanning function already exists in the SAS macro language.
5. The inner %DO loop iterates the simulation process for each sample size the appropriate number of times based on the quotient of "Iterations" and "NumIterPer". Again we use the %EVAL() macro function to evaluate the arithmetic expression. As the %EVAL() macro function only performs integer arithmetic, it is imperative that the quotient is integral.

SIMULATING SAMPLE DATA SETS

Simulating data is nothing more than an exercise in plugging numbers into a formula. In this case, the formula is the mixed model specified above. If we were content to simulate data one subject at a time and add to the sample data set in a looping process, the coding would be a straight forward combination of DO loops and matrix operations. However, the computational inefficiency of such an approach greatly undermines its simplicity. In the discussion of efficiency that we provide at the end of the paper, we will quantify just how bad this approach really is. It turns out that if we are willing to accept just a bit of coding complexity, we can realize substantial efficiency gains. For now, we present a very effective approach that generates multiple complete samples at one time. We need to create a SAS data set having the number of observations equal to [sample size × number of time points × number of complete samples to be generated at once] that contains the variables needed to fit the model in PROC MIXED. We now present the code in several sections.

```

PROC IML;
  Beta                = &Beta.;
  Cov_Random_Effects = &CovRandom.;
  Cov_Errors          = &SigmaSQ.;

  TimeVec             = &TimePointVec.;
  TimePoints          = nrow(TimeVec);
  NumIterPer          = &NumIterPer.;
  SampSize            = &SampSize.;

  Mean_Errors         = 0;
  Mean_Random_Effects = j(1,nrow(Cov_Random_Effects),0);

  /* Generate Random Components */
  Random_Effects = shape(
    repeat(
      randnormal(
        SampSize*NumIterPer,
        Mean_Random_Effects,
        Cov_Random_Effects
      ),
      1,
      TimePoints
    ),
    SampSize*TimePoints*NumIterPer,
    nrow(Cov_Random_Effects)
  );

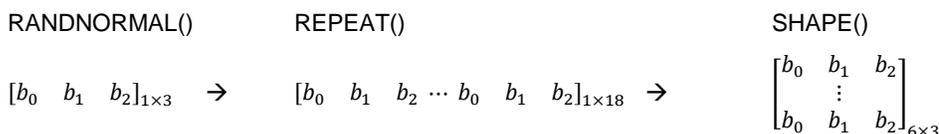
  Random_Errors = randnormal(SampSize*NumIterPer*TimePoints,Mean_Errors,Cov_Errors);

```

The diagram illustrates the flow of data from macro variables to the simulation code. A box labeled (1) points to the assignment of macro variables. A box labeled (2) points to the generation of random components. A box labeled (3) points to the generation of random errors.

1. Based on the user defined macro variables, we instantiate several scalars, vectors, and matrices that are needed in the simulation. In several places, we use the NROW() function to return the number of rows contained in a vector or matrix. We also make use of the J() function to create a matrix which stores the means of the random effects. Although the mean of a multivariate normal distribution is generally represented as a column vector, we generate a 1×6 matrix (or row vector) to store this data. This is a requirement for the RANDNORMAL() function which we use to generate multivariate normal data.
2. To generate the random effects, we nest the SHAPE(), REPEAT(), and RANDNORMAL() functions.
 - i. Based on the value of the first argument, the RANDNORMAL() function will generate the specified number of observations from the multivariate normal distribution defined by the second and third arguments. The observations are stored as rows in the output matrix and the dimension of the multivariate normal distribution defines the number of columns in the output matrix.
 - ii. The REPEAT() function is used to replicate a matrix or vector a specified number of times either horizontally, vertically, or both.
 - iii. The SHAPE() function is used to reshape the contents of a given matrix or vector to create a new matrix or vector of specified dimensions.

In the application presented above, we generate a number of random effects observations based on the product of sample size ("SampSize") and the number of complete data sets to be generated at once ("NumIterPer"). This will generate the set of random effects needed for all subjects included in the 250 complete data sets generated at each step. For each subject in each complete data set, we calculate six responses as a function of the random quantities, the design matrices, and the fixed effects. The random effects associated with a given subject will necessarily figure into the calculation for each of the six responses. Hence, we need to replicate each set of random effects across six rows. To achieve this, we repeat the random effects matrix horizontally based on the number of time points planned for the study and then shape the resulting matrix into one that contains the desired number of observations. The following diagram depicts the process for a single subject's random effects.



We use the fact that the SHAPE() function traverses the input matrix from left to right starting with the first row and then for each subsequent row in order to fill out the output matrix. By replicating the random effects six times per person, we ensure that the random effects matrix contains six copies of each subject's random effects.

3. Since the random errors arise from a univariate normal distribution, we are able to use the RANDNORMAL() function to generate all the necessary random errors for the set of complete samples being created. The result of this call will be a vector of dimension [sample size \times number of time points \times number of complete samples to be generated at once].

Now that we have generated the random components necessary to create the response data, we need to generate the design matrices. To do this we need to create variables for gender, treatment, time, and treatment by time interaction. Since this data is fixed (i.e. not random), we can simply create the data vectors directly. The following code presents one possible method.

```

/* Design Matrix Components */
Intercept = J(SampSize*NumIterPer*TimePoints,1,1);
Factor    = repeat(
  J(SampSize/2*TimePoints,1,0)//
  J(SampSize/2*TimePoints,1,1)
  ,NumIterPer
  ,1
);
Time      = repeat(TimeVec,SampSize*NumIterPer,1);
TimeSq    = Time#Time;
Tx        = repeat(
  J(SampSize/4*TimePoints,1,0)//
  J(SampSize/4*TimePoints,1,1)//
  J(SampSize/4*TimePoints,1,0)//
  J(SampSize/4*TimePoints,1,1)
  ,NumIterPer
  ,1
);
TxByTime  = Tx#Time;
TxByTimeSQ = Tx#TimeSQ;

XDesign = Intercept || Factor || Time || TimeSq || TxByTime || TxByTimeSQ;

```

The above SAS code uses several functions that have been previously discussed, and matrix operators to create the fixed effects design matrix which we call “XDesign”. The # operator performs element wise multiplication on conforming matrices. The // and || operators concatenate conforming matrices vertically and horizontally, respectively.

1. The “Factor” and “Tx” vectors are created for the complete data set by first creating a column vector of 0’s and 1’s (representing females and males, or SOC and ET) for one sample and then replicating the vector vertically the number of times necessary based on the number of complete samples being generated at once.
2. The “Time” vector is just a simple vertical replication of the “TimePointVec” vector that the user passes as a macro variable parameter.
3. The “TimeSq”, “TxByTime”, and “TxByTimeSQ” vectors are created by element wise multiplication of other previously created vectors.
4. The “XDesign” matrix is a horizontal concatenation of the vectors described above.

The purpose of creating the fixed effect design matrix is to facilitate matrix multiplication of the design matrix and the corresponding fixed effects, which is necessary for generating the simulated responses. One might also be inclined to create a random effects design matrix. However, since the random effects vary by subject (in contrast with the fixed effects), this approach will not work when data for more than one subject is generated at a time. Instead, the following SAS code can be used.

```

Response = XDesign*Beta
          + Intercept#Random_Effects[,1]
          + Time#Random_Effects[,2]
          + TimeSq#Random_Effects[,3]
          + Random_Errors;

```

We use the bracket notation to select a particular column (and all rows) of the random effects matrix and perform element-wise multiplication of the selected column with the corresponding column of the random effects design matrix. This approach ensures that each subject’s random effects are incorporated in the response calculation.

This next step is to generate a SAS data set that contains the simulated data and any ancillary variables necessary for modeling with PROC MIXED. The following SAS code can be used for this purpose.

```

Data      = Subject || Response || Factor || Tx || Time || TimeSq || Iter;

Cname = {"Subject" "Response" "Factor" "Tx" "Time" "TimeSq" "Iteration"};
create SimulatedData
  from Data[c=Cname];
  append from Data;
close SimulatedData;

```

We create the data set in a two-step process. First, we concatenate the relevant vectors into a large matrix named “Data”. Then we write out a SAS data set with the CREATE and APPEND statements. We must also use the CLOSE

statement so that the data set, named “SimulatedData”, is available for use. In order to control the names of the variables in the SAS data set that is created, we can define a character matrix containing the desired names as elements and then use the C= option on the create statement to request those elements be used to define the data set variable names. The SAS code for the “Subject” and “Iteration” vectors is omitted for brevity. The “Subject” vector provides unique identification for each subject in the sample and the “Iteration” vector provides unique identification for each complete sample in the larger data set. Both vectors can be created in a way similar to the design matrix vectors shown above.

MODEL FITTING

Each time a complete data set is created, we need to use PROC MIXED to fit a model to each sample contained in the data set and store the results of each model fit. Unless we generate all samples in a single step, we will need to loop through the data generation and model fitting process while keep tracking of the results of all model fitting. The following SAS code can be used.

```

proc mixed data = SimulatedData method=REML;
  by Iteration;
  class Subject;
  model Response = Factor Time TimeSQ Tx*Time Tx*TimeSQ / ddfm=KR solution;
  repeated / subject=Subject;
  random Intercept Time TimeSQ / subject=Subject type=UN;
  contrast "Any Treatment Effect" Tx*Time 1, Tx*TimeSQ 1;
  ods output Contrasts = Contrasts;
run;
quit;

data Results;
  set %if &OverallIteration. ^= 1 %then %do; Results %end; Contrasts(in=b);
  if b then do;
    OverallIteration = &OverallIteration.;
    SampSize = &SampSize.;
    if . < ProbF <= 0.05 then Reject = 1;
    else if ProbF > . then Reject = 0;
  end;
run;

proc SQL noprint;
  drop table Contrasts;
quit;

```

Diagram illustrating the SAS code for model fitting, with numbered callouts (1) through (5) pointing to specific lines of code:

- (1) `by Iteration;`
- (2) `contrast "Any Treatment Effect" Tx*Time 1, Tx*TimeSQ 1;`
- (3) `set %if &OverallIteration. ^= 1 %then %do; Results %end; Contrasts(in=b);`
- (4) `if . < ProbF <= 0.05 then Reject = 1;`
- (5) `drop table Contrasts;`

1. Using the BY statement instructs PROC MIXED to perform the model fitting separately for each level of the “Iteration” variable. Data sets must be sorted or indexed by the BY variables for this code to function properly. However, this will always be the case if the “Iteration” variable is constructed carefully.
2. To perform the desired hypothesis test, we use a contrast statement and simultaneously test whether both the linear and quadratic treatment by time interactions are zero. An ODS OUTPUT statement writes the contents of the “Contrasts” table to a SAS data set with the same name.
3. The “Results” data set is created from the “Contrasts” data set with a data step. The existing “Results” data set is included in the SET statement conditional on value of the macro variable “OverallIteration”. This condition ensures that the results are appended rather than replaced for subsequent steps in the overall process.
4. A binary data step variable named “Reject” is used to indicate whether the analysis of a given sample leads to rejection of the null hypothesis.
5. A SQL step is used to drop the existing version of the “Contrasts” data set after its contents are added to the “Results” data set.

POWER ESTIMATION

At the end of the process discussed above, the user will have a “Results” data set that includes one test result for each sample size and iteration of that sample size. There is no definitive approach for how to examine these results to estimate the power function. We offer a few suggestions. First, if little information about the appropriate sample size is known, we suggest specifying a wide range of sample sizes to investigate and a moderate number of iterations per sample size. A variety of statistical methods can be used to estimate the power associated with a given sample size. For example, binomial proportions can be computed for each sample size, logistic regression can be performed exploring some parametric function of sample size, or a penalized B-spline might be fit. The following SAS

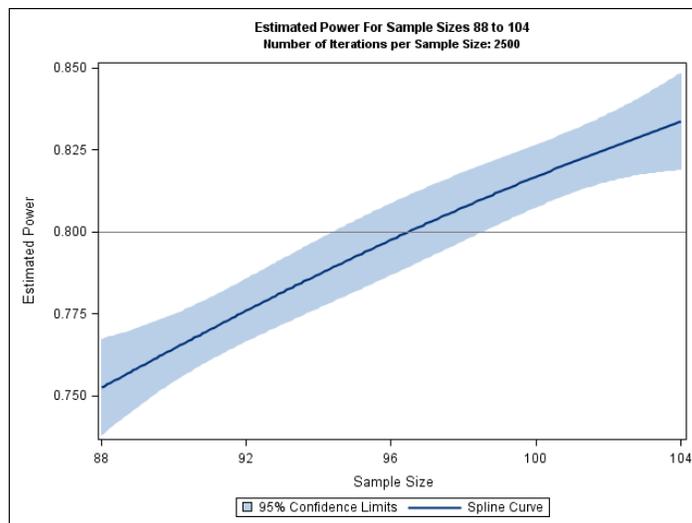
code can be used to estimate the power curve based on a penalized B-spline using the SGPLOT procedure and to estimate power via binomial proportions for each sample size being considered using the FREQ procedure.

```
proc sgplot data = Results;
  label Reject = 'Estimated Power';
  label SampSize = 'Sample Size';
  pbspline x=SampSize y=Reject /
    clm nomarkers legendlabel= 'Spline Curve' nknots=10;
  refline .80 .90 /axis=y;
  xaxis values=(%scan(&SampSizeList.,1,|) to
    %scan(&SampSizeList.,&NumSizes.,|) by 4);
run;
quit;

proc sort data = Results; by SampSize descending Reject; run;
proc freq data = Results order=data;
  by SampSize;
  table Reject / binomial;
  ods select BinomialProp;
run;
quit;
```

RESULTS OF THE EXAMPLE SIMULATION

Using the model described above, we simulated data for sample sizes 88, 92, 96, 100, and 104. For each sample size, 2500 samples were generated and analyzed using the PROC MIXED code provided above. The following spline curve was produced using the SGPLOT code provided.



The plot suggests that a sample size of approximately 100 is the smallest sample size that would provide at least 80% power. We then repeated the process using 5000 newly generated samples of size 100 and obtained the following 95% confidence interval for the power (0.80, 0.83). The lower bound of 0.80 supports the selected sample size.

NOTES ON SIMULATION EFFICIENCY

Even the most elegant simulation programs can take a long period of time to execute. Often the simplest coding approach is not the best option. We presented a flexible approach which allows the user to control how many samples were generated at one time. It is logical to wonder what the most efficient number of samples to generate at once is. To explore that, we benchmarked our approach using 6 different possibilities. Benchmarking was performed using a Lenovo W520 laptop having:

- 6 GB of RAM,
- Dual 2.20 GHz processors,
- Windows 7 Enterprise Operating System, and
- 64-Bit SAS (Batch Submission).

Table 1: Benchmark Comparison of Sample Generation Methods (Sample Size = 100; Iterations = 2000)

Number of Iterations at Once	Average Elapsed Time (minutes, n=4)
1	3.353
100	1.864
250	1.857
500	1.886
1000	1.896
2000	1.909

We note that generating 250 samples at once performed the best. Most notably, simulations that generated one sample at a time performed far worse than all other simulations. It is also somewhat clear that generating one large data set containing all samples is not the most efficient approach. In practice, we recommend generating approximately 250 samples at once.

NOTES ON MISSING DATA

The simulation results presented in this paper are based on a complete data set. However, anyone who has been involved with a longitudinal study should recognize that missing data are the rule rather than the exception. Though we have omitted an investigation of the impact of missing data in our example, one would be well advised to consider the impact of missing data on power in an actual simulation. For example, males may be more likely to skip visits than females or subjects with higher baseline values might be more likely to drop out early. Each of these missing data mechanisms can be simulated easily in the data step after the complete data is written to a SAS data set and before the model fitting step.

It is important to note that when the assumed model for the response is correct (mean structure and covariance structure), inference will be valid as long as the data are missing completely at random (MCAR) or missing at random (MAR). This requires all covariates that are predictive of missingness be included in the model. Fitzmaurice, Laird, and Ware (2011) provide an accessible discussion of missing data mechanisms.

CONCLUSION

In this paper we have presented a flexible method for estimating power by simulating data for a continuous response from a mixed model using PROC IML. The approach presented can be easily generalized to the broader class of generalized linear mixed models, including longitudinal models for binary data.

REFERENCES

- Carpenter, A. (2004), *Carpenter's Complete Guide to the SAS Macro Language*, second edn, SAS Institute, Inc., Cary, NC.
- Wicklin, R. (2010), *Statistical Programming with SAS/IML Software*, SAS Institute, Inc., Cary, NC.
- Fitzmaurice, G., Laird, N., & Ware, J. (2011), *Applied Longitudinal Analysis*, second edn, John Wiley & Sons, Inc., Hoboken, NJ.

ACKNOWLEDGMENTS

I would like to thank Dr. Amy Herring and Sandy Ferber for feedback on various drafts of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Matthew Psioda
Enterprise: UNC Department of Biostatistics, Student
E-mail: psioda@live.unc.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.