PharmaSUG 2016 - Paper SP09

Simulation of Data using the SAS System, Tools for Learning and Experimentation

Kevin R. Viel, Ph.D., inVentiv Health Clinical, Atlanta, GA

"**LET**, perhaps the most powerful word in all of mathematics!"

Professor Bevan K. Youse
Emory University
Circa 1992

ABSTRACT

Statistical models, like many fields of mathematics, rely upon assumptions (postulates). The successful use of these tools conventionally involves examination of corresponding statistics that inform the statistician whether violations of model assumptions might have occurred. Simulation, enabled by ever increasing computational power, is making experimentation a mainstay of statistics and mathematics. Further, the ability to simulate data should be required of every student of statistics, much like how the fluency of matrix "language" and the ability to code the likelihood should be requisite skills. The goal of this paper is to introduce simulations using the SAS System® and to provide the technical (programming) and statistical basis to examine the use of models in time to event data with special consideration of recent and important reports of inhibitors in Hemophilia A patients.

INTRODUCTION

Two avenues to understanding statistics include the mathematics and the application; respectively, the theory and the demonstration in practice. Both required the art of imagination. Not infrequently, the statistician gains insight into one by exploring the other. Yet, it might be the case that application often involves "real" data from studies that may or may not be close to the requirements (assumptions), whereas the ability to simulate test data, and the related ability to write the likelihood directly, often teaches the statistician and student alike.

The foundation of simulation is the random number generator. Not uncommonly, a statistician is asked how to determine whether certain data were generated by a given distribution. Prominent among the answers is that one cannot. To demonstrate both the answer and imagination in mathematics, consider the archetypical example, the toss of a fair coin. With a large number of tosses, the proportion of heads will tend towards the probability of heads being p = 0.50. Actually, given enough tosses, one could accurately report the probability as p = 0.50000, or even more significant digits with more tosses. Imagine, however, what happens with an infinite number of tosses. One aspect of an infinite number of tosses is that a series of (HHH…H_∞) occurs. What if, in our finite human experience, or at least during working hours, we find ourselves observing a series of heads that is "never ending"? Can we conclude that the coin is not fair? Not exactly, but we can be "certain". With imagination, though boggling, this situation must occur within a series of infinite tosses. In fact, this situation must occur an infinite number of times in an infinite number of tosses. The same is true for (TTT…T_∞) and (HTHT…$\{HT\}_\infty$) and any sequence (combination) that one can imagine. Such is the peril of infinity or imagination, if one can tell difference. One way to determine if data were generated by a given distribution is to examine the mechanism of generation (the code).

The SAS System® provides excellent functions and call routines to generate data from a given distribution. The heart of the generation of these data is the random number generation (RNG), which technically is pseudo-random number generation. The focus of this paper is the use of these functions; the mathematics and theory are out of scope. The goal of this paper is to introduce some basic simulations and to analyze the resulting data as an investigation or exploration of both the process of simulation and the examination of the statistical models. We will also address how simulation might support an investigation into whether grouping of time-to-event data for analyses might or might not be appropriate.

SEED

The initial starting point from which the RNG functions in the SAS system generate a stream of pseudo-random numbers is called the **seed**. The seed ranges in value from 0 to 2^{31}-1 (2,147,483,647). Importantly, if the seed value is 0, then the computer clock initializes the stream. If the programmer wishes to replicate the stream, then he or she should use a positive value for the seed and record it. Apparently, the maximum value of this range is the maximum number of numbers that can be generated before the sequence begins to repeat. Presumably, some functions may have shorter periods. While this may seem to be a large number, the online documentation warns that modern computers can exhaust the sequence in minutes in "typically" simulations studies. The more recent RAND() function has a reported period of $2^{19937} - 1$, which is approximately 10^{6000} and much larger than 10^{307}, the largest value that can be represented in eight bytes on most computers that run SAS.

Figure 1 shows the results of two nearly identical data steps that generate data from the standard normal distribution using the RANNOR() function. As stated above the values of X are equal; they use the same seed. The values of Y and Z also match, but the values of the seeds are different. Understanding this behavior is essential; the value of the seed in the first function executed determines the stream. So the streams in the right and left panels are the same, regardless of subsequent values of the seed.

```
data _null_ ;
  x = rannor( 1 ) ;
  put x= ;
  y = rannor( 1 ) ;
  put y= ;
  z = rannor( 1 ) ;
  put z= ;
run ;

x=1.8048229506
y=-0.079915021
z=0.396576855
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
data _null_ ;
  x = rannor( 1 ) ;
  put x= ;
  y = rannor( 2 ) ;
  put y= ;
  z = rannor( 3 ) ;
  put z= ;
run ;

x=1.8048229506
y=-0.079915021
z=0.396576855
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Figure 1. Seeds and subsequent calls. The seed in the first function executed determines the stream of pseudo-random numbers.

A distinction exists between SAS code and the macro facility with regard to seeds. Each invocation of a data step "resets" the stream for a given seed in SAS code. However, the macro facility continues the stream and only closing and re-opening the SAS System will reset the stream in the macro facility. **Figure 2** demonstrates this distinction.

```
data _null_ ;
  x = rannor( 1 ) ;
  put x= ;
run ;

x=1.8048229506
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


data _null_ ;
  x = rannor( 1 ) ;
  put x= ;
run ;

x=1.8048229506
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
%put %sysfunc(rannor(1)) ;
1.80482295064194
%put %sysfunc(rannor(1)) ;
-0.07991502090275
```

Figure 2. Resetting the Stream. The behavior of resetting the stream differs between the SAS data step and macro facility.

To this point, we have discussed functions, but CALL ROUTINES are available for the corresponding functions. An important distinction is the ability to control the value of the seed, which must be initialized before the first call. The value of the seed is updated by the call routine, but it can also been explicitly assigned. **Figure 3** demonstrates the flexibility of the seed in call routines. Notably, assigning the seed to the same value results in the same pseudo-random number and changes it affects the pseudo-random number; the sequence of the seed is not "anchored" as it is for the corresponding function.

```
data _null_ ;
  s1 = 1 ;
  s2 = 1 ;
  call rannor( s1 , x1 ) ;
  call rannor( s2 , x2 ) ;
  put x1= / x2= ;
  put s1= / s2= // ;

  s1 = 1 ;
  s2 = 2 ;
  call rannor( s1 , x1 ) ;
  call rannor( s2 , x2 ) ;
  put x1= / x2= ;
  put s1= / s2= ;
run ;

x1=1.8048229506
x2=1.8048229506
s1=2083249653
s2=2083249653


x1=1.8048229506
x2=1.3118295938
s1=2083249653
s2=2019015659
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Figure 3. Call Routines and Control of Seeds. The value of seeds can be explicitly controlled and the same results will result from the same seed.

The default seed stream of a call routine and its corresponding function appear to be the same. That is, one can see what the next seed in a stream is by viewing the updated seed value after a call. The program in **Figure 4** demonstrates this informally, but the log is not provided.

```
data _null_ ;
  s1 = 1 ;

  do _n_ = 1 to 5 ;
    x1 = rannor( s1 ) ;
    call rannor( s1 , x2 ) ;
    put x1= @20 x2= @40 s1= ;

    call execute( "data _null_ ;" ) ;
    call execute( cat( "  s1 = "
                     , strip( put( s1 , 32. ))
                     , " ;"
                     )
                ) ;
    call execute( "  x1 = rannor( s1 ) ; "  ) ;
    call execute( "  put x1 = ; "  ) ;
    call execute( "run ; "  ) ;
  end ;
run ;
```

Figure 4. The Seed Stream. The sequence of seeds revealed by the call routine that corresponds to the function.

SIMULATING A COIN TOSS

The first simulated trial is the well-known coin toss, with a twist. The probability of a head was set at p = 0.3 and the number of tosses was set at 10. Since the expected number of heads is 3, one might prefer exact statistics. The code in **Figure 5** provides the exact 95% confidence interval (CI) for the trial the results in three heads out of 10 tosses using the Clopper-Pearson derivation. If the number of heads is set to 0 and the number of tosses is set to 25, then the 95% CI is (0.00, 0.1372), which agrees with the results reported by Agresti on Page 18 of "Categorical Data Analysis"[1]. In our simulated case, the 95% CI for the probability is (0.0667, 0.6525) and for the number of heads is (0.667, 6.525). The FREQ procedure code demonstrates how to obtain the data using ODS OUTPUT statements, which are best positioned or "contained" within the procedure to which they apply, and the ZEROS option to the WEIGHT statements, which is essential to obtain the results for 0/25. The code in the right panel simulates

1000 trials of ten coin tosses, which follow a binomial distribution. The RANBIN() function derives the variate from the random binomial.

```
%let heads  =  3 ;
%let tosses = 10 ;

data heads ;
  heads  = 1  ;
  freq   = &heads. ;
  output ;
  heads  = 0 ;
  freq   = &tosses. - &heads. ;
  output ;
run ;

ods listing close ;

proc freq data  = heads
          order = data
          ;
  ods output BinomialCLs = bcls
                              ( keep   = lowercl
                                         uppercl
                              )
             ;
  tables heads
       / binomial
         ( exact )
         ;
  weight freq / zeros ;
run ;

ods listing ;
```

```
%let seed = 23 ;
%let p    = %sysevalf( &heads. / &tosses. ) ;

data bin_&seed. ;
  retain seed &seed.
         p    &p.
         ;

  do sim = 1 to 1000 ;
      heads = ranbin( Seed
                    , &tosses.
                    , p
                    ) ;
      output ;
  end ;

  stop ;

run ;
```

Figure 5. Simulating Coin Toss Trials.

EXLORING CONFIDENCE INTERVALS WITH SIMULATIONS

Figure 6 shows the histograms of two simulations using the code in the right panel of Figure 5 with Seed = 23 (left panel) and Seed = 500 (right panel). This figure provides empirical evidence that the confidence interval actually contains 95% of the results, that is the 95% confidence bounds appear to contain at least 95% of the number of heads resulting from the trials. Actually, for Seed = 23 and Seed = 500, the results are 96.0% and 95.9%, respectively.

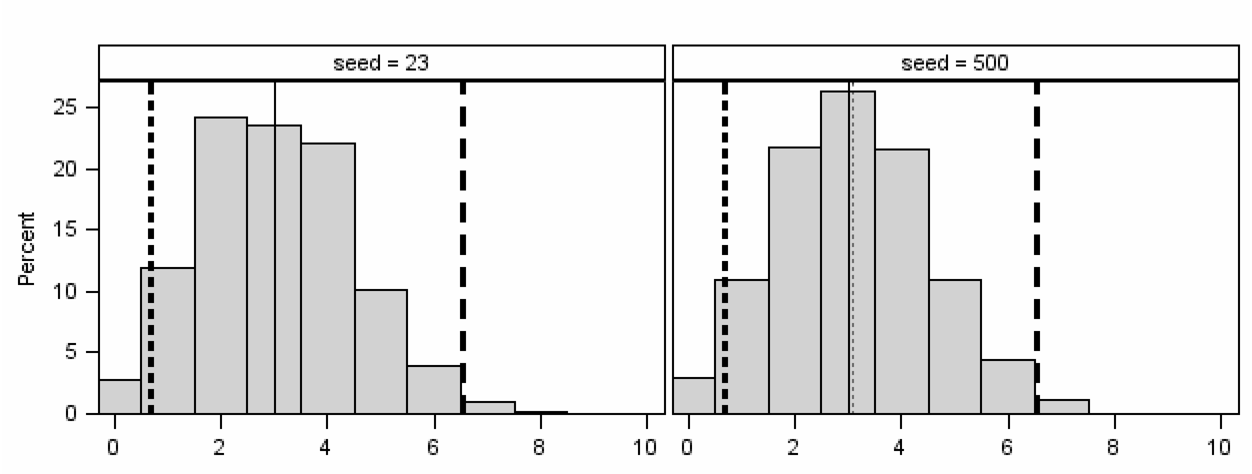

Figure 6. Histograms of 1000 Simulated Coin Toss Trials. Both trials have p = 0.3 and 1,000 simulated trials of ten tosses. The solid line is the expected number of heads (3.0), the dotted line is the mean number of observed heads (3.026 and 3.087 for Seed = 23 and Seed = 500, respectively). The bold dashed and medium dashed lines are the lower and upper bounds, respectively, of the 95% CI for the number of heads expected from a single trial of 10 coin tosses with p = 0.3, (0.667, 6.525).

A point that can be lost when considering the confidence intervals of continuous, symmetric distributions such as the normal distribution is that the upper bound of the confidence interval is not the value above which the cumulative probability is $\alpha/2$. Rather it is the mean (and standard deviation) of the distribution for which the mean of interest is the lower bound of its 100*(1-α)% confidence interval. For a symmetric, continuous distribution these values coincide. For a discrete distribution, such as the binomial, this is more complex and the standard deviation, and, thus, the values of the bounds of the confidence interval, depends on the mean. Consider the upper bound of the Clopper-Pearson exact confidence interval for p = 0.3 in a trial of 10 ten tosses, p = 0.6525. In a simulation of 1,000 trials of 10 tosses with p = 0.6525, the number trials with the number of heads less than 3.0 (p = 0.3 for 10 tosses), should be less than 2.5% (α/2). Simulations allow the student (this author included in that class) to explore and experiment. **Table 1** presents the proportion of trials for which the number of heads was greater than 3.0 for Seeds 23 and 500.

Table 1. Empirical coverage of binomial distributions.

Number of tosses	Percent not covering 3.0	
	Seed = 23	Seed = 500
10	0.003	0.006
30	0.010	0.013
50	0.009	0.017
100	0.012	0.015
1000	0.026	0.027
10000	0.026	0.027

Another way to think of this is if more than 2.5% of the trials have the number of heads less than 3.0, then the evidence suggest that the probability might not 0.6525, i.e. we reject that hypothesis (and consider a lower probability). One issue is that the binomial distribution is discrete. Table 1 shows an "improvement" in the percent of trials not including the product of p = 0.3 and the number of tosses as the percent approach the Clopper-Pearson value of 0.05/2. One issue is that the RANBIN() function does not adhere to the binomial distribution with increasing n or p, but rather the normal approximation takes effect. The reader is encouraged to study the documentation since the detail and scope cannot be covered in this paper.

SIMULATION: LOGISTIC REGRESSION

The logical extension to the simulation of binomial distribution might be logistic regression. The student of regression should know the logit function, which is the log odds, that is logit(p) = log[p / (1 – p)]. To be brief, the logistic regression models are general linear models (GLMs) that have a binomial random component with a link function that is the logit of the probability. **Figure 7** presents code that simulates logistic regression data in which two factors affect the probability.

```
Data _null_ ;
  Call Sympute( "Beta_1"
              , "Log( 2 )"
              ) ;
  Call Sympute( "Beta_2"
              , "Log( 1.2 )"
              ) ;
Run ;

Data Log_Sim
        ( Keep   = Sim
                   ID
                   Trt
                   Factor_1
                   Disease
        )
     ;

  Seed = 10 ;

  Beta_0 = 0   ;
  Beta_1 = &Beta_1. ;
  Beta_2 = &Beta_2. ;

  Do Sim = 1 to 1000 ;
```

```
ODS Listing Close ;

Proc Logistic Data       = Log_Sim
              Descending
              ;
  ODS Output OddsRatios = OR ;
  Model Disease =
        Trt
        Factor_1
      / ExpB
        ;
  By Sim ;
Run ;

ODS Listing ;

Proc Datasets library = WORK
              NoList
              ;

  Modify OR ;
    Attrib _all_ Label = " " ;
Quit ;

Data OR ;
```

```
    ID = . ;
    Do Trt = 0 to 1 ;
      Do _n_ = 1 to 50 ;
        ID + 1 ;

        Call Ranbin( Seed
                   , 2
                   , 0.5
                   , Factor_1
                   ) ;

        Eta_Prob = Beta_0
                 + Beta_1 * Trt
                 + Beta_2 * Factor_1
                   ;

        p_0 = Exp( Eta_Prob )
            / ( 1 + Exp( Eta_Prob ))
              ;

        If 0 < p_0 < 1
        Then Call Ranbin( Seed
                        , 1
                        , p_0
                        , Disease
                        ) ;
         Else Disease = p_0 ;

        Output ;
      End ;
    End ;
  End ;

Run ;
```

```
  Set OR ;
  If Effect = "Trt"
  Then True = LowerCL <= Exp( &Beta_1. ) <= UpperCL ;
   Else If Effect = "Factor_1"
   Then True = LowerCL <= Exp( &Beta_2. ) <= UpperCL ;
Run ;

Proc Means Data = OR
           n
           Mean
           StdDev
           Median
           Min
           Max
           ;
  Class Effect ;
  Var OddsRatioEst
      LowerCL
      UpperCL
      True
      ;
Run ;
```

Figure 7. Simulation of Logistic Regression Data.

The mean (median) of the Odds Ratio (OR) for FACTOR_1 and TRT is 1.25 (1.19) and 2.24 (2.01), respectively. The 95% CI for these ORs correctly contained the true OR in 93.6% and 95.2% of the simulations for FACTOR_1 (OR = 2.0) and TRT (OR = 1. 2), respectively. Increasing the number of patients per treatment increases the precisions of these estimates. The reader should now be able to experiment with sample size to explore its effects on accuracy and precision. Further, the code can be adopted to include more covariates and/or to include continuous covariates. The detail and scope cannot be covered in this paper.

SIMULATION: ZERO-INFLATED POISSON MODEL

The Poisson distribution is another discrete distribution that can model events such as soldiers being kicked in the head by horses or the number of hospitalizations of patients in a catchment area. Notably, distributions may be mixed; one might suspect that the number of hospitalizations would be an interesting distribution. Most people do not need to be hospitalized, but others might, unfortunately, may be hospitalized numerous times. Attempting to model such as population might be problematic. In this case, we might say that the "excess" number of 0 hospitalizations is inflated. We thus have a description of a zero-inflated model. Another such model is the zero-inflated negative binomial. The author leaves it to the reader to conclude why the zero-inflated binomial is, basically, a contradiction. **Figure 8** presents the code for a zero-inflated Poisson (ZIP) model, that is, given the probability, the frequency of the zero variate exceeds the expectation based on a Poisson distribution.

```
Proc Format ;
  Value ru
     0.00 - <0.25 = 0
     0.25 - <0.50 = 1
     0.50 - <0.75 = 2
     0.75 -  1.00 = 3
     ;
Run ;

/* ZI = Zero Inflation */
%Let Beta_ZI_0 = 0.2 ;
%Let Beta_ZI_1 = 0.3 ;
```

```
ODS Listing Close ;

Proc NLMixed Data = ZIP_Sim ;

  ODS Output FitStatistics      = FS
             ParameterEstimates = PE
             Contrasts          = C
             ConvergenceStatus  = CS
             ;

  Parms /* parameters for Bernoulli */
        Beta_ZI_0 = 0
```

```
%Let Beta_ZI_2 = 0.0 ;

/* P = Poisson */
%Let Beta_P_0 = 0.1 ;
%Let Beta_P_1 = 0.0 ;
%Let Beta_P_2 = 1.0 ;

Data ZIP_Sim
        ( Keep   = Sim
                   ID
                   Trt
                   Con_Factor
                   H
                   ZI
                   p_0
                   Lambda
        )
     ;

  Seed = 10 ;

  /* parameters for 0 inflation prob */
  Beta_ZI_0 = &Beta_ZI_0. ;
  Beta_ZI_1 = &Beta_ZI_1. ;
  Beta_ZI_2 = &Beta_ZI_2. ;

  /* parameters for poisson mean */
  Beta_P_0 = &Beta_P_0. ;
  Beta_P_1 = &Beta_P_1. ;
  Beta_P_2 = &Beta_P_2. ;

  Do Sim = 1 To 1000 ;
    Do ID = 1 To 1000 ;

      /* Treatment */
      Trt = Mod( ID , 2 ) ;

      /* "Continuous" factor: 0 - 3 */
      Con_Factor = Input( Put( Ranuni( Seed )
                             , ru.
                             )
                         , 8.
                         ) ;

      /* Prob of 0 inflation as a
         function of Trt and Con_Factor */
      Eta_Prob = Beta_ZI_0
               + Beta_ZI_1 * Trt
               + Beta_ZI_2 * Con_Factor
                 ;

      p_0 = Exp( Eta_Prob )
          / ( 1 + Exp( Eta_Prob ))
            ;

      If 0 < p_0 < 1
      Then Call Ranbin( Seed
                      , 1
                      , p_0
                      , ZI
                      ) ;
       Else ZI = p_0 ;

      If ZI = 1 Then H = 0 ;
       Else
         Do ;
            /* Poisson mean as a function of
               Trt and Con_Factor */
            Eta_Lambda = Beta_P_0
                       + Beta_P_1 * Trt
                       + Beta_P_2 * Con_Factor
```

```
        Beta_ZI_1 = 0
        Beta_ZI_2 = 0
        /* parameters for poisson mean */
        Beta_P_0 =  0
        Beta_P_1 =  0
        Beta_P_2 =  0
        ;

  Eta_Prob = Beta_ZI_0
           + Beta_ZI_1 * Trt
           + Beta_ZI_2 * Con_Factor
             ;

  p_0 = Exp( Eta_Prob )
      / ( 1 + Exp( Eta_Prob ))
        ;

  Eta_Lambda = Beta_P_0
             + Beta_P_1 * Trt
             + Beta_P_2 * Con_Factor
               ;

  Lambda = Exp( Eta_Lambda ) ;

  /* Loglikelihood */
  If H = 0
  Then Loglikelihood = Log( p_0
                          + ( 1 - p_0 )
                          * Exp( -Lambda )
                          ) ;
   Else Loglikelihood = Log( 1 - p_0 )
                      + H
                      * Log( Lambda )
                      - Lambda
                      - LGamma( H + 1 )
                        ;

  Model H ~ General( Loglikelihood ) ;

  /* Test whether the CI for the estimated
     parameters covers their true parameter
     values
  */
  /* Parameters for Bernoulli */
  Contrast "Beta_ZI_0 = &Beta_ZI_0."
           Beta_ZI_0 - &Beta_ZI_0.
           ;
  Contrast "Beta_ZI_1 = &Beta_ZI_1."
           Beta_ZI_1 - &Beta_ZI_1.
           ;
  /* Parameters for Poisson */
  Contrast "Beta_P_0 = &Beta_P_0."
           Beta_P_0  - &Beta_P_0.
           ;
  Contrast "Beta_P_2 = &Beta_P_2."
           Beta_P_2  - &Beta_P_2.
           ;

  By Sim ;

Run ;

ODS Output Close ;
ODS Listing ;

Title1 "Beta_ZI_0 = &Beta_ZI_0.
        Beta_ZI_1 = &Beta_ZI_1.
        Beta_ZI_2 = &Beta_ZI_2." ;
Title2 "Beta_P_0  = &Beta_P_0.
        Beta_P_1  = &Beta_P_1.
        Beta_P_2  = &Beta_P_2." ;
```

```
                         ;

            Lambda = Exp( Eta_Lambda ) ;

            /* Simulation the number of
               H as a Poisson
               with mean Lambda */
            Call RanPoi( Seed
                       , Lambda
                       , H
                       ) ;
         End ;

      Output ;
    End ;
  End ;

Run ;
```

```
Proc Means Data = PE ;
  Class Parameter ;
  Var Estimate ;
Run ;

Data C ;
  Set C ;
  True = ProbF > 0.05 ;
Run ;

Proc Means Data = C
           N
           Mean
           ;
  Class Label ;
  Var True ;
Run ;
```

Figure 8. Simulation of a Zero-Inflated Poisson (ZIP) Model.

For the model specified in Figure 8, the simulation results agree well with the true values. **Table** 2 presents the results of this simulation. For the parameters tested in the CONTRAST statements, the coverage was acceptable, that is approximately 95%. Less than 5% of the simulations resulted in rejection of hypotheses that the differences between the estimated values and the parameter (true) values were different from zero.

Table 2. Results of the Zero-Inflated Poisson Simulation

Parameter	True Value	Mean	Coverage (%)
P_0	0.1	0.098	94.6
P_1	0.0	-0.002	N/A
P_2	1.0	1.00	94.0
ZI_0	0.2	0.192	95.4
ZI_1	0.3	0.310	95.8
ZI_2	0.0	0.000	N/A

This simulation provides that opportunity to emphasize a few points. The reader who ran the code (and the other code) should have notice that it was not instantaneous, despite being a modest run of only 1,000 simulations. For efficiency, it is best to use one procedure (PROC) and analyze the separate simulations using a BY statement. Secondly, the author will leave it as an exercise for the reader to reduce this model from a ZIP to a conventional Poisson model. It may be informative, however, to analyze both data sets using both models. In addition, obtaining measure of model fit like the Akaike Information Criteria (AIC) or the Bayesian Information Criteria (BIC) should be a useful demonstration. This point could be true of any of the models; consider, for instance, the effects of categorizing age into age groups. Categorization without motivation from the substantive field is STRONGLY discouraged; continuous (quantitative) variable contain more information and inclusion in a model versus the categorized version of them should result in a better model with a better fit. The author is particular found of cubic splines, if the substantive knowledge suggests that they are appropriate. Consider, example, the use of a quadratic formation of age for adults (18-75 year old). The model imposes this relationship, so that the quadratic nature forces the same results from 18-27 years as for 66-75 years. Depending on the outcome, this may not be accurate. Simulating the various relationships should help the statistician better under the models.

SIMULATION: TIME TO EVENT

Many students have simulated linear and perhaps logistic regression data. Simulating a zero-inflated model is the next plateau, especially if the statistician then can write the log-likelihood equations and experiment with initial parameter values. Though relatively simple compared to a model with mixed distribution, simulating survival data is a challenge many statisticians might not encounter unless they seek the challenge. A slight complicate is simulating censored data. **Figure 9** presents the code that simulates time to event (survival analysis) data based on the Weibull distribution. For each patient two random variates were simulated. One of which was the time to event and the second was the time to censoring. If the latter occurred before than former, then patient contributed censored data.

```
%Let Beta1  = Log( 1 ) ;
%Let Lambda = 20 ;

Data Sim_TTE
        ( Drop      = Linear_Predict
                      Time_Censor
        )
     ;

  Beta1  = &Beta1. ;
  Lambda = &Lambda. ;

  Seed   = 1 ;

  Do Sim = 1 To 100 ;
    Do Trt = 1 To 2 ;
      Do _n_ = 1 To 200 ;

        ID + 1 ;

        Linear_Predict = Exp( -Beta1 * Trt ) ;
        Time = Rand( "WEIBULL"
                   , Seed
                   , Lambda
                   * Linear_Predict
                   ) ;

        Time_Censor = Rand( "WEIBULL"
                          , Seed
                          , Lambda
                          * Linear_Predict
                          * 3.00
                          ) ;

        If Time_Censor < Time
        Then
          Do ;
             Censored = 1 ;
             Time     = Time_Censor ;
          End ;
         Else Censored = 0 ;

        Output ;

      End ;
    End ;
  End ;
Run ;
```

```
ODS Listing Close ;

Proc PHReg Data = Sim_TTE ;
  ODS Output ParameterEstimates = PE
             CensoredSummary    = CS
             ;
  Model Time * Censored( 1 )
      = Trt
      / RiskLimits = Wald
        ;
  By Sim ;
Run ;

Proc Lifetest Data = Sim_TTE ;
  ODS Output Quartiles = Q ;
  Time Time * Censored( 1 ) ;
  Strata Trt ;
  By Sim ;
Run ;

ODS Output close ;
ODS Listing ;

Title1 "Beta1 = &Beta1. Lambda = &Lambda." ;
Proc Means Data = Q
                   ( Where = ( Percent = 50 ))
           n
           Mean
           MaxDec = 2
           ;
  Class Trt ;
  Var Estimate
      LowerLimit
      UpperLimit
      ;
Run ;

Data PE ;
  Set PE ;
  If Parameter = "Trt"
  Then
    Do ;
       True = HRLowerCL <= Exp( &Beta1. ) <=
HRUpperCL ;
    End ;
Run ;

Proc Means Data = PE ;
  Class Parameter ;
  Var HazardRatio
      True
      ;
Run ;

Proc Means Data = CS ;
  Var PctCens ;
Run ;
```

Figure 9. Simulation of Time to Event (Survival) Data.

The median survival for patients with TRT = 1 was 14.04 (months) and for patients with TRT = 2 was 13.97. This was by design, the true parameter (Beta1) was Log(1) so that exp(Beta1) = 1. In fact, the mean hazard ratio (HR) from 100 simulations was 1.01 and the 95% CI for the estimated HR contained the true value in 96% of the simulations.

The reader can experiment with several factors, such as the ratio of patients to controls, how unbalanced the censoring might be before, for a given effect, the results start to be biased. The mean proportion of subjects censored in this simulation was 25.5%.

REAL DATA: HEMOPHILIA A AND EXPOSURE DAYS

Hemophilia A (HA) patients have deficient or absent activity of coagulation Factor VIII (FVIII), a co-enzyme to Factor IX. Approximately, 1 in 10,000 males will have HA regardless of race or country of origin.[2] The average cost per year of treating HA patients in the US ranges from $50,000 to $200,000 for mild to severe patients, respectively. Treatment of HA patients consists of infusing FVIII intravenously. If any amount (dose) is injected, then the patients had an Exposure Day (ED). Patients may have an immune reaction to the infused FVIII in some cases the anti-therapy antibodies (ATA) will neutralize the infused product and be termed inhibitors. Gouw et alia[2] presented the results of concerning the time to the development of inhibitors in Hemophilia A patients treated with different types of product.

Suffice it to say that collection of data for studies is complicated; this includes monitoring the patient for the development of inhibitors. Not atypically, the protocol might obtain a blood sample for screening for inhibitors every five EDs. That could be after five days, in the event of a bleeding episode that was difficult to control or prophylaxis for a dental procedure or surgery. One might expect that EDs will be stochastic in patients not undergoing prophylaxis or immune tolerance induction (ITI). Importantly, Gouw et alia "pooled observations over all exposure days for all patients into a single sample and then used a logistic-regression model with stratification according to number of exposure days to relate the risk factors to inhibitor development." They cited D'Agostino[4] to support the statistical methodology. Recently, a study by Peyvandi et alai[5] also analyzed time to inhibitor (event) data using ED as the unit of time, i.e. pooling the data.

With the ability to simulate survival data, such as the code in Figure 9, we can approach this issue with both an understanding of mathematics underlying the statistics[6-7], which for anyone who has attempted to read the original proportional hazard model paper by Cox can attest is not for the user of statistics. However, simulations might suffice to provide an understanding, especially if simulations suggest a major limitation that happens to fit external data or expectations, like the stochastic nature of bleeding in HA patients.

CONCLUSION

Simulation is a powerful tool to improve the understanding of statistical models and can be included in attempts to verify them or to identify weaknesses. The SAS System provides multiple functions and call routines to generate (pseudo-)random numbers that are variates from various statistical distributions. This paper introduced several of such distributions with particular emphasis on the seed (stream or sequence of pseudo-random numbers). Though the models were relatively simple, this paper demonstrated a zero-inflated Poisson model. Finally, an example from real world reports in very respectable journals that peer-review the submissions before acceptance for publication demonstrated that thought process to use simulations to examine and explore the potential issues with the statistical methodology.

REFERENCES

[1] Agresti, Alan. Copyright © 2002. Categorical Data Analysis, Second Edition. Page 18. Hoboken, New Jersey: John Wiley & Son, Inc.

[2] Soucie, J. M., B. Evatt, et al. (1998). "Occurrence of hemophilia in the United States. The Hemophilia Surveillance System Project Investigators." Am J Hematol 59(4): 288-294.

[3] Gouw, S. C., J. G. van der Bom, et al. (2013). "Factor VIII products and inhibitor development in severe hemophilia A." N Engl J Med 368(3): 231-239.

[4] D'Agostino, R. B., M. L. Lee, et al. (1990). "Relation of pooled logistic regression to time dependent Cox regression analysis: the Framingham Heart Study." Stat Med 9(12): 1501-1515.

[5] Peyvandi, F., P. M. Mannucci, et al. (2015). "Source of Factor VIII Replacement (PLASMATIC OR RECOMBINANT) and Incidence of Inhibitory Alloantibodies in Previously Untreated Patients with Severe Hemophilia a: The Multicenter Randomized Sippet Study." Blood 126(23): 5-5.

[6] Singer J.D. and J.B. Willett (1993). "It's About Time: Using Discrete-Time Survival Analysis to Study Duration and the Timing of Events." Journal of Educational Statistics 18(2): 155-195.

[7] Willett J.B. and J.D. Singer (1995). "It's Déjà vu All Over Again: Using Multiple-Spell Discrete-Time Survival Analysis." Journal of Educational and Behavioral Statistics 20(1): 41-67.

CONTACT INFORMATION

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

Name: Kevin R. Viel, Ph.D.
Enterprise: inVentiv Health Clinical
Histonis, Incorporated
E-mail: kevin.viel@inventivhealth.com
kviel@histonis.org
Web: http://www.inventivhealthclinical.com/