Paper 079-2010

Data Simulation for Piloting Clinical Trials Display Programs

Brandon Welch, Rho® Inc., Chapel Hill, NC
Rita Slater, Rho® Inc., Chapel Hill, NC

ABSTRACT

In a fast-paced clinical trials computing environment, programmers often need to create tables, listings, and figures within a limited time span. In these cases, available working data sets are paramount to insure quality deliveries. The SAS® system offers many tools for simulating data which allow users to pilot their display programs prior to receiving real working data sets. This article explores the use of SAS® functions and macro processing to generate artificial data using gamma and binomial random variates. The SAS® macro presented in this article uses these functions to simulate patient-level and adverse event (AE) data sets. The results of these simulations are then used to populate abbreviated tables and listings. By having artificial data similar in structure to what display specifications provide, statistical programmers can verify the accuracy of their code prior to receiving real data. While this article targets a clinical computing audience, the techniques are applicable to a broad range of computing scenarios.

INTRODUCTION

Statistical programmers at a contract research organization (CRO) receive specifications to produce particular displays. These displays are in the form of tables, listings, or figures (TLF). Tables offer a comprehensive summary of data and typically include both descriptive and inferential statistics; listings give the complete list of values that compliment a particular table. Figures provide a visual representation of data in the form of charts and/or graphs and often display descriptive and inferential statistical values. Although this article concentrates on table and listing production using random number generating functions, the techniques presented are easily extended to generate figures

An example of a table:

Table 1
Demographic Characteristics
ITT Population

CHARACTERISTIC	TREATA (n=XX)	PLACEBO (n=XX)	Total (n=XX)
Sex			
Male	XX (XX.XX)	XX (XX.XX)	XX (XX.XX)
Female	XX (XX.XX)	XX (XX.XX)	XX (XX.XX)
Age			
N	XX	XX	XX
Mean	XX.XX	XX.XX	XX.XX
Standard Deviation	XX.XX	XX.XX	XX.XX
Median	XX.XX	XX.XX	XX.XX
Min, Max	XX XX, XX. XX	XX XX, XX. XX	XX XX, XX. XX
Race			
White	XX (XX.XX)	XX (XX.XX)	XX (XX.XX)
Asian	XX (XX.XX)	XX (XX.XX)	XX (XX.XX)
African Descent	XX (XX.XX)	XX (XX.XX)	XX (XX.XX)
Other	XX (XX.XX)	XX (XX.XX)	XX (XX.XX)

Table 1. Demographic Table Shell

This example summarizes demographic data for a particular population (Intent-To-Treat in this table). The corresponding demographic listing for the same population appears as follows:

Listing 1
Demographic Characteristics
ITT Population

Site	Treatment	Subject	Sex	Age (years)	Race
XXXX	A	XXXXXXXX	XXXX	XX	XXXX

Listing 2. Demographic Listing Shell

Statistical programmers need sound working data for the production of many types of TLFs. In an ideal situation, programmers have all data sets at their disposal prior to production. Unfortunately, this is not always the case. The SAS® system provides many tools for generating test data for piloting display programs before the actual data sets are ready for use.

DATA SET TYPES

This article illustrates the simulation of two data set types: patient-level and adverse event (AE). It is noteworthy to mention that the word "simulation" is used literally. This is different from statistical simulation (i.e., Monte Carlo methods) in which multiple loops or iterations produce estimates. In this article, statistical simulations do not generate the test data.

Patient-level data sets are the simplest since they are organized as one record per patient. The demographic table example (***Table 1.***) uses a patient-level data set to acquire the displayed descriptive statistical values. AE data sets are slightly more complicated because they are usually organized as patient-by-event. Patients and their corresponding adverse events experienced during their trial comprise these data sets. It is common to produce tables that tabulate the frequency of these events (or number of patients per event) for a particular population.

VARIABLE TYPES

The SAS® system divides variables into two types: numeric and character. In this article, we further categorize these variables by scale of measurement – continuous and categorical.

Statisticians and programmers typically associate numeric variables with continuous data. Laboratory and/or efficacy data sets are popular sources of these variable types. Analyzing the change from baseline for a particular lab test or performing a two-sample t-test comparing the mean BMI values across treatment groups are just two examples of analyses performed on this data type.

Categorical variables are quantified with either numeric or character values. For example, a GENDER variable may take on the numeric values of 1 and 2 (where 1 = MALE and 2 = FEMALE) or the character values of M and F (M = MALE and F = FEMALE). Categorical variables are further divided as dichotomous and polytomous. Dichotomous variables take on two possible values (GENDER for instance) and polytomous have more than two values (ETHNICITY for example). See flow chart for clarity:

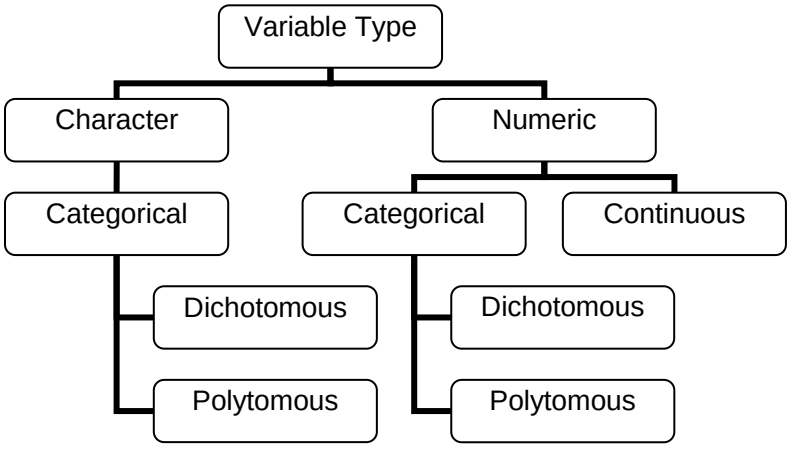

Chart 1. Variable Type and Scale

RANDOM NUMBER GENERATION

The methods presented in this paper use RANBIN and RANGAM SAS® functions.

CATEGORICAL VARIABLES

Binomial random variates via the RANBIN function create all categorical variables in this article. The RANBIN function requires a seed, *n*, and *p* value – *RANBIN(seed, n, p)*. For illustration, the following code creates a numeric dichotomous treatment variable (1 = TREATA and 2 = PLACEBO) for five patients:

```
DATA sim;
  do patient = 1 to 5;
    treat = RANBIN(12345,1,0.5) + 1;
    output;
  end;
RUN;
```

OUTPUT

PATIENT	TREAT
1	1
2	2
3	2
4	1
5	1

Note that the user arbitrarily chooses *p* and seed values. Using the same seed value insures that the program generates the same random numbers upon running. A popular substitute for the seed value is DATE() -- the current SAS® date value. However, the use of this date function could result in different random numbers on subsequent days.

By using *n = 1*, RANBIN considers only one Bernoulli trial (values 0 or 1) at each iteration of the data step. In this example, adding unity to the function value insures the program does not generate zero values. While coding treatment variables as 0/1 is acceptable for statistical modeling interpretations, treatment variables typically do not take on binary values. If the user desires a polytomous treatment variable (e.g., three treatment groups), the following code applies:

```
DATA sim;
  do patient = 1 to 5;
    treat = RANBIN(12345,2,0.5) + 1; *NOTE n > 1;
    output;
  end;
RUN;
```

OUTPUT

PATIENT	TREAT
1	2
2	2
3	3
4	2
5	1

Code for creating a character polytomous treatment variable follows:

```
DATA sim;
     alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
     do patient = 1 to 5;
       rand  = RANBIN(12345,2,0.5) + 1; *NOTE n > 1;
       treat = SUBSTR(alpha, rand, 1);
       output;
     end;
   RUN;
```

OUTPUT

PATIENT	TREAT
1	B
2	B
3	C
4	B
5	A

CONTINUOUS VARIABLES

The RANGAM function generates gamma random variates. Note that this function requires both a seed and a shape parameter – *RANGAM(seed, k)*. This article uses the gamma distribution because it always yields positive values. This facilitates the creation of variables such as AGE or BMI – both commonly found in demographic tables. Depending on the shape parameter's value, the program may not yield realistic values. Alternatively, other continuous distributions (*F-distribution* , for example) are interchangeable in this context.

Code for creating a continuous AGE variable via the RANGAM function with arbitrarily chosen seed and shape parameter values:

```
DATA sim;
  do patient = 1 to 5;
    age = RANGAM(12345,10);
    output;
  end;
RUN;
```

OUTPUT

PATIENT	AGE
1	8.7889
2	14.4122
3	7.1036
4	5.6725
5	12.2429

MACRO

The following macro uses the RANBIN and RANGAM functions to create artificial data sets for piloting display programs. The examples above only illustrate the primary syntax that comprises the macro.

MACRO CALL

```
%DataSIM(ID, Size, TrtN, TrtNVAL, TrtC, TrtCVAL, Missing, Site);
```

This macro requires all arguments except &MISSING and &SITE. The &MISSING parameter allows the user to randomly generate missing numeric or character values in their data set, whereas the &SITE parameter represents an optional site or location variable. Of the required parameters, &ID denotes the subject or patient identification number. The RANUNI function generates these values, which are then converted to character. The &SIZE parameter represents the number of records desired in the data set. &TRTN and &TRTNVAL represent the numeric treatment variable name and numeric treatment variable values, respectively. Finally, &TRTC and &TRTCVAL denote the character treatment variable name and character treatment variable values, respectively. A space separates all values for &TRTCVAL (e.g. TREATA PLACEBO)

Prior to calling the macro, %LET defines a string for each variable in the data set:

```
%let VarX = DType | DName | AnlyVar | VType | VScale | NestVar | NestType | NestVal;
```

The macro parses each definition (delimited by the pipe "|") and uses each sub-string to create the test data set. Definitions of each sub-string are as follows:

Sub-string	Definition	Valid Values
`DType`	Data set type	PT – patient level AE – adverse event CM – concomitant medication COM - comments ECG - electrocardiogram LB - laboratory MDHX – medical history PK - pharmacokinetic PD - pharmacodynamic VITL – vital signs VIS - visit level MISC - miscellaneous
`DName`	Data set name	User defined
`AnlyVar`	Variable name	User defined
`VType`	Variable type	N – numeric C – character
`VScale`	Variable scale	D - dichotomous PV – polytomous value PL(*value1,..,valuen*) – polytomous list CONT - continuous DATE – date value TIME – time value
`NestVar`	Nesting variable	User defined
`NestType`	Nesting variable type	C – character value N – numeric value NL – numeric list CL – character list
`NestVal`	Nesting variable value(s)	User defined. If list specified in `NestType`, then input (*value1,..,valuen*)

Table 2. Sub-string Definitions

When defining a patient-level data set (DType = PT), sub-strings required include DType, DName, AnlyVar, VType and VScale. Multi-level data sets (i.e., visit-level or AE-level) require all sub-strings.

Note that users may define lists for the polytomous analysis and nesting variables. For example, using PL(*WHITE,..,ASIAN*) creates a polytomous RACE variable. Users may employ PV if the number of the levels required for the display is unknown. For example, PV(3) yields the values A, B and C (or 1,2, and 3 for numeric variables) in the resulting data set. The logic is the same for nesting variables (via the C and N values).

EXAMPLES

EXAMPLE 1: DEMOGRAPHIC LISTING AND TABLE

The demographic listing (***Listing 1***) for ten ITT patients requires six variables in the body. SITE, SUBJECT and TREATMENT are defined within the macro call. In this example we assume that SEX and RACE are character variables (C) and AGE is numeric and continuous (CONT). We further classify SEX as dichotomous (D) with values MALE and FEMALE, and RACE as polytomous (PL) with values WHITE, ASIAN, AFRICAN DESCENT, and OTHER. By defining an additional ITT population variable as numeric (N) and dichotomous (D), this scenario is more realistic.

SYNTAX

```
%let Var1 = PT | DEMO | ITT | N | D;
%let Var2 = PT | DEMO | RACE| C | PL(White,Asian,African Descent,Other);
%let Var3 = PT | DEMO | SEX | C | PL(Male,Female);
%let Var4 = PT | DEMO | AGE | N | CONT;

%DataSIM(ID      = patient,
         Size    = 10,
         TrtN    = trtn,
         TrtNVAL = 2,
         TrtC    = trtc,
         TrtCVAL = TREATA PLACEBO,
         Missing = Yes,
         Site    = site
         );
```

OUTPUT DATA SET

PATIENT	SITE	TRTN	TRTC	ITT	RACE	SEX	AGE
36292445	0506	1	TREATA	2	African Descent	Male	8.3295
18382375	0511	2	PLACEBO	1	African Descent	Female	11.4813
70725354	0488	1	TREATA	1	Asian	Male	7.169
15276748	0479	2	PLACEBO	2	African Descent	Male	9.4228
39253603	0515	1	TREATA	1	White		5.8565
28021324	0490	2	PLACEBO	1	Asian	Female	15.8379
61897077	0507	1	TREATA	2	White	Female	5.6643
96894348	0481	1	TREATA	2	White	Female	8.1226
83181805	0498	1	TREATA	2	Asian	Male	16.5037
41884027	0499	1	TREATA	1	African Descent	Male	18.1457

OUTPUT LISTING

```
                                  Listing 1
                          Demographic Characteristics
                               ITT Population

Site   Treatment  Subject     Sex     Age
                                     (years)     Race
_____________________________________________________________

0488    TREATA    70725354    Male    7          Asian

0490    PLACEBO   28021324   Female   16         Asian

0499    TREATA    41884027    Male    18         African Descent

0511    PLACEBO   18382375   Female   11         African Descent

0515    TREATA    39253603            6          White
```

Note that only five patients are randomly assigned to the ITT population.

OUTPUT TABLE

Table 1
Demographic Characteristics
ITT Population

CHARACTERISTIC	TREATA (n=3)	PLACEBO (n=2)	Total (n=5)
Sex			
Male	2 (66.7)	0	2 (40.0)
Female	0	2(100.0)	2 (40.0)
Age (years)			
N	3	2	5
Mean	10.39	13.66	11.70
Standard Deviation	6.75	3.08	5.32
Median	7.17	13.66	11.48
Min, Max	5.86 ,18.15	11.48 ,15.84	5.86 ,18.15
Race			
White	1 (33.3)	0	1 (20.0)
Asian	1 (33.3)	1 (50.0)	2 (40.0)
African Descent	1 (33.3)	1 (50.0)	2 (40.0)
Other	0	0	0

EXAMPLE 2: AE LISTING AND TABLE

Now consider generating an AE listing (***Listing 2***) and table (***Table 3***) for the same ITT patients.

Listing 2
Adverse Events
ITT Population

Center	Subject	Treatment	AE system organ class/ Preferred term/	Start date/ End date
XXXX	XXXXXXXX	X	XXXXX/XXXXX	MMDDYYYY/MMDDYYYY

Listing 2. AE Listing Shell

Table 2
Adverse Events
ITT Population

System Organ Class / Preferred Term	TREATA (N=XX)	PLACEBO (N=XX)	Total (N=XX)
Any AE	XX (XX.X)	XX (XX.X)	XX (XX.XX)
System Organ Class1	XX (XX.X)	XX (XX.X)	XX (XX.XX)
Preferred Term1	XX (XX.X)	XX (XX.X)	XX (XX.XX)
System Organ Class2	XX (XX.X)	XX (XX.X)	XX (XX.XX)
Preferred Term2	XX (XX.X)	XX (XX.X)	XX (XX.XX)
.	.	.	.
.	.	.	.
.	.	.	.
System Organ Classn	XX (XX.X)	XX (XX.X)	XX (XX.XX)
Preferred Termn	XX (XX.X)	XX (XX.X)	XX (XX.XX)

Table 3. AE Table Shell

In total, the body requires seven variables. As before, the macro call designates SITE, SUBJECT and TREATMENT. In practice, programmers merge the ITT and treatment variables onto the AE data set prior to producing the listing and table. However, the user could define the ITT variable in the `VarX` definitions for

the AE data set. System organ class (AESOC) and preferred term (AEPTERM) are nesting variables within each patient. Therefore, the program requires only two `VarX` definitions for the AE start and end dates. We designate AESOC as character (C) with five levels and AEPTERM as character (C) with ten levels. We also designate the AE start (AESTRT) and end (AESTOP) dates as both numeric (N) and date (DATE). Note the macro call is identical to the one used to generate the demographic data set. For brevity, we only display ten records.

SYNTAX:

```
%let Var1 = AE | ADAE | AESTRT | N | DATE | AESOC AEPTERM | C C | 5 10;
%let Var2 = AE | ADAE | AESTOP | N | DATE | AESOC AEPTERM | C C | 5 10;
```

OUTPUT DATA SET (TEN RANDOMLY SELECTED RECORDS)

PATIENT	SITE	TRTN	TRTC	AESOC	AEPTERM	AESTRT	AESTOP
36292445	506	1	TREATA	ABC	EDCBA	6092	6096
18382375	511	2	PLACEBO	AB	DCBA	6088	6103
70725354	488	1	TREATA	AB	GFEDCBA	6114	6106
15276748	479	2	PLACEBO	ABC	GFEDCBA	6102	6106
96894348	481	1	TREATA	ABCD	CBA	6089	6102
96894348	481	1	TREATA	ABC	GFEDCBA	6086	6098
96894348	481	1	TREATA	ABC	DCBA	6086	6098
96894348	481	1	TREATA	ABC	FEDCBA	6086	6098
96894348	481	1	TREATA	ABC	HGFEDCBA	6119	6091
83181805	498	1	TREATA	ABC	FEDCBA	6088	6093

OUTPUT LISTING (TEN RANDOMLY SELECTED RECORDS)

Listing 2
Adverse Events
ITT Population

Site	Treatment	Subject	AE system organ class/ Preferred term	Start date/ End date
0488	TREATA	70725354	ABC/IHGFEDCBA	04OCT1976/08SEP1976
0490	PLACEBO	28021324	ABCDE/CBA	11SEP1976/26SEP1976
			ABC/EDCBA	/29SEP1976
0499	TREATA	41884027	ABC/IHGFEDCBA	04OCT1976/
			ABC/GFEDCBA	04OCT1976/
			ABCDE/GFEDCBA	24SEP1976/
			ABCDE/GFEDCBA	24SEP1976/
			ABC/HGFEDCBA	26SEP1976/12SEP1976
0511	PLACEBO	18382375	ABC/GFEDCBA	15SEP1976/24SEP1976
			AB/EDCBA	01SEP1976/16SEP1976

OUTPUT TABLE (FIRST TEN RECORDS)

Table 2
Adverse Events
ITT Population

System Organ Class Preferred Term	TREATA (N=3)	PLACEBO (N=2)	Total (N=5)
Any AE	3(100.0)	2(100.0)	5(100.0)
ABC	3(100.0)	2(100.0)	5(100.0)
EDCBA	3(100.0)	2(100.0)	5(100.0)
FEDCBA	3(100.0)	2(100.0)	5(100.0)
GFEDCBA	3(100.0)	2(100.0)	5(100.0)

System Organ Class Preferred Term	TREATA (N=3)	PLACEBO (N=2)	Total (N=5)
HGFEDCBA	3(100.0)	2(100.0)	5(100.0)
CBA	2 (66.7)	2(100.0)	4 (80.0)
DCBA	3(100.0)	1 (50.0)	4 (80.0)
IHGFEDCBA	2 (66.7)	1 (50.0)	3 (60.0)
BA	1 (33.3)	0	1 (20.0)

CONCLUSION

Although most of the `%DataSIM` macro syntax is devoted to parameter checking and parsing the definitions strings (not shown in this article), it is possible to generate patient-level and AE-level data sets with relative ease. The user can then employ these data to generate any type of display – including figures, although not demonstrated in this article.

We may easily extend these examples to more complicated data sets. For example, creating chemistry and hematology laboratory test values at two visits, one may use the following:

```
%let Var1 = LB | LAB | LABVL | N | CONT | TYPE TEST VIS | CL CL CL | (CHEM,HEME)
(BUN,CREAT) (Base,Follow);
```

As this paper illustrates, a programmer can create an entire library of data with one two random-number-generating functions. With the flexibility provided by macro processing, programmers may adapt their code to fit any data generation need.

REFERENCES

SAS Institute Inc., SAS® 9.1.3 Help and Documentation, Cary, NC: SAS Institute Inc., 2000-2004.

ACKNOWLEDGMENTS

Steve Noga
John Ingersoll
Eva J. Welch

CONTACT INFORMATION

Brandon Welch
Rho®, Inc.
6330 Quadrangle Dr., Ste. 500
Chapel Hill, NC 27517
Phone: 919-595-6339
Fax: 919-408-0999
Email: Brandon_Welch@rhoworld.com
Web: www.rhoworld.com