Generating Data With the SAS® Dataset

Andrew J. L. Cary
Cary Consulting Services, Newark CA

Abstract:

It is often necessary to create data to test a program or methodology. The SAS software system's data step and random number generating functions provide a powerful toolbox for generating data. This paper will demonstrate how to generate data for variety of situations using this toolbox.

Introduction:

The SAS software system's data step combined with the SORT and PLAN procedures is a powerful and useful data generation tool. Data is easily generated for simulations, for program testing, and for demonstrations. SAS has a rich set of pseudo-random number generating functions which can be used to generate data drawn from one or more populations. The combination of tools makes it possible to generate data mimicking natural population based data for many situations.

Generated data is generated using a model. Models can be 'independent' or 'dependent'. An independent model is used to generate a datum which is drawn directly from some underlying distribution. A dependent model is used to generate a datum drawn as an element in a related series of random data.

Independent data models are useful in generating single point random variables used in grouping. For example, the following code fragment will generate a treatment code with two levels:

```
TRTGRP= INT(UNIFORM(0)+0.5);
```

Independent data models are also useful to generate starting values for dependent sequences.

Dependent data models can model almost any situation. Data can be generated that are time dependent such as patient weights or stock prices. They can also generate data from related single point values such as body weight and height or gender and weight.

The following code fragment illustrates how to generate body weight and height keeping a 'reasonable' relationship between them by using the a randomly generated body-mass-index .

The model is:

$$BMI = \frac{Weight\ in\ kg}{Height\ in\ m^2}$$

Using this model the following code fragment will generate random but related heights and weights.

Example 1: Construct normally distributed Body mass index;

```
BMI=22.5 + NORMAL(0)*2.5;
* Generate a normal distributed
   height in meters;
HT=ROUND(1.8 +
         0.127*normal(0),.01);
* Construct weight from BMI and
   HT;
WT = ROUND(BMI * HT **2,. 1);
```

Time dependent models are usually constructed using fixed interval times or variable intervals. Fixed intervals are generally the result of planned data collection. Variable measurements are usually event driven.

Fixed interval measurements are very easy to model in SAS. The following code fragment

illustrates how to generate a fixed interval sample with random unrelated levels of a value:

Example 2: Fixed intervals.

```
DO MINUTE=0 TO 100 BY 1;
      X=UNIFORM(0)*10;
      OUTPUT;
END;
```

These intervals do not have to be evenly spaced. The spacing can be spaced unevenly using the list form of a DO statement.

Example 3: Unequally spaced Fixed Intervals

```
DO MINUTE=0, 30, 60, 120, 240;
      X=UNIFORM(0)*10;
      OUTPUT;
END;
```

The intervals spacing may itself be effected by random measurement error. This is easily modeled by adding a random amount to the interval value.
Time dependent models can also be a function

Example 4: Intervals with random measurement

```
DO BASE=0, 30, 60, 120, 240;
      MINUTE=BASE+
            INT(UNIFORM(0)*5)
      X=UNIFORM(0)*10;
      OUTPUT;
END;
```

of the time interval itself. This is easily done by building the random variable using a time based function. Any kind of function may be used. The example below uses a linear function.

Example 5: Time dependent relationship

```
DO BASE=0, 30, 60, 120, 240;
      MINUTE=BASE+
         INT(UNIFORM(0)*5)
      X= 0.25 *
         MINUTE+NORMAL(0)*10;
      OUTPUT;
END;
```

Data can also be modeled to allow for variations with in a single entity over time. This is useful when the variable of interest varies significantly between entities. An example of this kind of data is human body weight over time.

Example 6: Entity related time variation

```
DATA PHYSICAL;
DO ID=1001 TO 1030 ;
   * GENERATE RANDOM BMI ;
   BMI=22.5 + NORMAL(0)*2.5;
   * Generate a normal distributed
     height in meters;
   HT=ROUND(1.8 +
         0.127*normal(0),.01);
   * Construct weight from BMI
     and HT;
   WT = ROUND(BMI * HT **2,. 1);
   * VARY WEIGHTS OVER WEEKS ;
   DO WEEK= 0 TO 8 ;
        WT=WT+INT(NORMAL(0)*2.5);
        OUTPUT;
   END;
END;
```

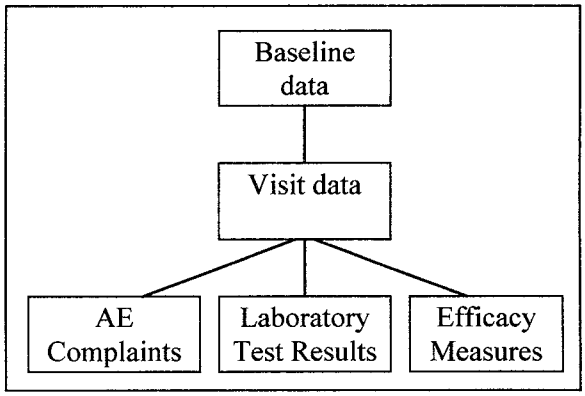
Data often resides hierarchically in several related datasets. These datasets are directly analogous to tables in a relational database.

As an example, in clinical trials data, it is very common to have information collected once about a patient stored in one table while data about that patient collected at each examination might be in another. These tables are related through a single key (which could be composed of several variables). In the clinical trials example this is usually a unique patient identifier.
This kind of data is easily generated in a SAS datastep by outputting several datasets in one datastep using several nested loops. The example below shows how this is done:

```
END; DATA DEMOG PHYSICAL;
DO ID=1001 TO 1030 ;
   * GENERATE RANDOM BMI ;
   BMI=22.5 + NORMAL(0)*2.5;
   *  Generate a normally
      distributed height in
      meters;
   HT=ROUND(1.8 +
         0.127*normal(0),.01);
   * Construct base weight from
     BMI and HT;
   WT = ROUND(BMI * HT **2,. 1);
   * OUTPUT TO DEMOG FILE ;
   OUTPUT DEMOG;
   * VARY WEIGHTS OVER WEEKS ;
   DO WEEK= 0 TO 8 ;
        WT=WT+INT(NORMAL(0)*2.5);
        OUTPUT PHYSICAL;
   END;
```

More typically in the clinical trials world, several types of data would have to be output for each patient record. Some of the records output would be output only if other conditions were true. Suppose the hierarchical structure in Figure 1 was used to collect data. For each patient baseline visit there are one or more subsequent visits which may or may not have adverse experience report, laboratory test reports and efficacy measures. This type of data is also easily generated using a datastep. The program shown in

Figure 1: Clinical data example



The source code example shows how this is done. A single datastep is used to ensure data consistency. The outermost loop is used to compute patient baseline data. A loop inside that one builds a random set of visits with the generation of efficacy and lab data. A second that loop builds a set of AE records for each patient. Dependencies between visits but within a patient are handled at the baseline level.

Conclusion

The SAS data step is eminently suited to the synthesis of data. Its rich tool set of randomization functions combined with its PL/I like programming language make it easy to generate complex data structures closely mimicking 'real' data.

SOURCE CODE EXAMPLE

```
DATA BASELINE (KEEP=ID GENDER HEIGHT WEIGHT)
     VISIT    (KEEP=ID DATE VISIT SYST_BP DIAST_BP)
     AES      (KEEP=ID STARTDT STOPDT AE SEVERITY)
     LABS     (KEEP=ID DATE WBC RBC)
     EFFICACY (KEEP=ID DATE EFFICACY);

     * POPULATE A TEMPORARY ARRAY OF VALUES TO PROVIDE VALUES OF AES,
       EFFICACY, AND SEVERITY VALUES ;

     ARRAY AELIST [10] $16 _TEMPORARY_
                          ('HEADACHE','NAUSEA','TINNITUS', 'VOMITING',
                           'ITCHING','ABDOMINAL PAIN','DIZZINESS',
                           'SKIN RASH','PALPITATIONS','HALLUCINATIONS');

     ARRAY EFFLIST [4] $16 _TEMPORARY_ ( 'NONE','POOR','MODERATE','GOOD');

     ARRAY SEVLIST [3] $16 _TEMPORARY_ ('MILD','MODERATE','SEVERE');

     * LOOP THROUGH PATIENT POPULATION ;

     * SET BASEDATE TO 15 JAN 1996 ;

     BASEDATE = '05JAN1996'D;


     DO ID=1001 TO 1030;

            * CREATE BASELINE DATA, PROBLEM DATA, AND
              VISIT DATA ;

            * GENERATE RANDOM GENDER, BODY MASS INDEX (BMI), HEIGHT,
              WEIGHT, AND BASE BLOOD PRESSURES BY GENDER ;

            IF UNIFORM(0) > .45 THEN DO;

                 GENDER=   'M';

                 BMI=      22.5 + NORMAL(0)*2;
                 HEIGHT=   1.82+INT(NORMAL(0)*.03);
                 WEIGHT=   ROUND(BMI*HEIGHT**2,.01);

                 SYST_BP=  100 + INT(NORMAL(0)*10);
                 DIAST_BP= SYST_BP -20 + INT(NORMAL(0)*10);
```

```
     END;

ELSE DO;

     GENDER=    'F';

     BMI=       20.0 + NORMAL(0)*1.8;
     HEIGHT=    1.62+INT(NORMAL(0)*.05);
     WEIGHT=    ROUND(BMI*HEIGHT**2,.01);

     SYST_BP=   90 + INT(NORMAL(0)*10);
     DIAST_BP=  SYST_BP -20 + INT(NORMAL(0)*10);

     END;

* OUTPUT BASELINE DATA ;

OUTPUT BASELINE;

* GENERATE BASELINE LAB VALUES ;

WBC= 2000 + INT(NORMAL(0)*100);
RBC= 2000 + INT(NORMAL(0)*100);

* GENERATE DATE OF BASELINE VISIT AND NUMBER OF VISITS;

BASEDATE=    BASEDATE+ INT(UNIFORM(0)*5);

NVISITS=     RANTBL(0,.20,.25,.25,.30);

* LOOP THROUGH VISITS FOR THIS PATIENT ;

DO  VISIT=0 TO NVISITS;

     * CALCULATE VISIT DATE;

     DATE = BASEDATE + (NVISITS* 14);

     * CALCULATE BLOOD PRESSURE AND EFFICACY AND OUTPUT ;

     SYST_BP=   SYST_BP   + INT(NORMAL(0)*6);
     DIAST_BP=  DIAST_BP  + INT(NORMAL(0)*6);

     EFFICACY = EFFLIST[RANTBL(0,.1,.1,.4,.4)];
```

```
            OUTPUT VISIT;
            OUTPUT LABS;
            OUTPUT EFFICACY;

        END;

        * CREATE A MAX OF 3 AES FOR EACH PATIENT ;

        NAES =  RANTBL(0,.20,.25,.25,.30) -1;

        * LOOP THROUGH AES ;

        DO IAE = 1 TO NAES;

            * CREATE START AND STOP DATES AND SEVERITY ;

            STARTDT=  BASEDATE + INT(UNIFORM(0)*10);
            STOPDT=   MIN(DATE, STARTDT+INT(UNIFORM(0)*20));

            SEVERITY= SEVLIST[RANTBL(0,.2,.4,.4)];

            * GET AN AE FROM THE LIST ;

            AE=AELIST[RANTBL(0,.5,.1,.05,.05,.05,.05,.05,.05,.05,.05)];

            * OUTPUT AES ;

            OUTPUT AES;

        END;

    END;
RUN;
```

Trademarks