Paper 347-2012

Selecting Unrestricted and Simple Random With Replacement Samples Using Base SAS® and PROC SURVEYSELECT

David. D. Chapman, Consultant, Alexandria, VA

ABSTRACT

This paper reviews different techniques for selecting unrestricted and simple random without replacement samples using BASE/SAS data step code, using procedures such as PROC SORT and RANK, and using the SURVEYSELECT procedure in SAS/STAT. An extremely brief review of random sampling basics is given that discusses the UNIFORM random number function and the use of the special SAS functions INT, CEIL, and FLOOR to select random samples in the data step. Data step code for selecting unrestricted random samples is given and explained. Data step code for selecting Bernoulli samples and two different approaches to selecting a simple random sample without replacement are presented and discussed. Simpler ways to select simple random samples without replacement using PROC SORT and PROC RANK are illustrated and examples given. An alternative to selecting a random sample in the data step is to use PROC SURVEYSELECT in SAS/STAT. The syntax for the SURVEYSELECT procedure is given and statements needed for selecting unrestricted and simple random samples are explained. The meaning and use of different options available using PROC SURVEYSELECT are examined. PROC SURVEYSELECT code is given to select unrestricted and simple random samples without replacement. References to published papers on how to use SAS to select random samples are given.

KEYWORD: Sample Selection, PROC SURVEYSELECT, Random Sampling.

INTRODUCTION

This paper discusses the use of SAS to select samples using both BASE/SAS® and PROC SURVEYSELECT in SAS/STAT®. Only simple random samples with and without replacement are considered here. Bernoulli sampling is considered a special case of simple random sampling without replacement. Most, if not all, the material in this paper has been presented before. This is a review paper and I have attempted to give credit to the original authors. If I missed someone, I apologize. The purpose of this paper is to present in one place the basics of sampling using BASE/SAS and to provide an introduction to PROC SURVEYSELECT. Sequential and systematic sampling are not discussed; neither are stratified sampling, cluster sampling, multi-stage sampling; nor are any probability proportional to size (pps) sample designs. The emphasis here is only on sample selection; the only estimator used is a linear estimator where weights are based on the reciprocal of the probability of selection. The most complete papers on the subject of sample selection using BASE/SAS is Boudreaux and Cranford (1996, 1995). Course notes for "SAS PROGRAMMING II: Manipulating Data with the Data Step" and "SAS PROGRAMMING III: Advanced Programming Techniques" contain detailed examples of basic data step methods to select random samples. SAS/STAT documentation of PROC SURVEYSELECT gives an excellent description of how to use it with the basic sampling methods.

This paper gives a brief review of probability sampling that explains the difference between Simple Random Sampling With Replacement (SAS/STAT refers to this as Unrestricted Random Sampling (URS)), Simple Random Sampling Without Replacement, and Bernoulli Sampling. How to select each of these three types of samples in the data step is discussed. The use of procedures such as PROC SORT and PROC RANK to selected Simple Random Samples without replacement are illustrated as well as how to make used of a data sets's index to select an simple random samples without replacement. Finally SAS/STAT's new PROC SURVEYSELECT is discussed. The syntax of PROC SURVEYSELECT is explained. There is a detailed discussion of the syntax statements and their options related to selecting unrestricted and simple random samples without replacement. SAS code to select these different types of samples are given and is accompanied by a discussion of the different options available and how they can be used. Finally, there is a brief discussion and the use of SAS to validate a random sample.

SAMPLING – THE BASICS

A basic objective of sampling is to make judgements or statements about a large universe based on a portion or a sample of that universe. The universe from which the sample is selected and about which we want to make decisions is called the population or sample frame. Here the frame will always be a SAS data set; in general, it can be many other things. An artificial frame used with all the examples is given in the appendix. Constants defined for the population are called population parameters. Examples of population parameters are the sum or mean of variables in the population(e.g. Number of people in the US or percent of people in the labor force that are unemployed). Often decisions are based directly on the value of a population parameter. A sample is a part – often a very small part – of the population. Summary information calculated from the sample are called the sample statistics. The statistical characteristic of the sample is determine by how the sample selected. When the sample is a random sample, the statistical characteristics of the sample is known. We usually know the expected value of sample statistics in terms of the true population parameters. More importantly, because the sample is a probability sample, it is possible to estimate how close the sample estimate is to the true value and to guarantee there are no systematic selection biases.

How we determine the sample is the basis of survey sampling and sample design. The portion of the universe an individual record represents is represented in the record or sampling weight. This record weight at least initially is equal to reciprocal of the probability a record is selected into the sample. Changes are often made to record weights to compensate for problems in data collection or so the sample will expand to certain totals. These record or sample weights are used by SAS to process data. SAS, with many procedures, does this through the WEIGHT and FREQ statements.

There are many different types of sample designs. A distinction often made in sample designs is between equal probability and unequal probability samples. In equal probability samples, every record on the frame has an equal chance of being included into the sample and therefore an equal weight. Equal probability sample designs mentioned in SAS documentation include Simple Random Sampling With

Replacement (a.k.a. Unrestricted Random Sampling), Simple Random Sampling Without Replacement, Bernoulli Sampling, Systematic Sampling, and Sequential Sampling. The first three have the characteristic that any two records have an equal chance of being in a sample together. In the last two, this is not true. Only the first three methods will be discussed here. In unequal probability designs different records have a different chances of being selected into the sample. Often a record will have a measure of size that is used to determine its probability of selection. Because of this, these designs are called pps – probability proportional to size – samples. When a characteristic of interest is related to the measure of the size, pps samples will often give more accurate estimates of population parameters. Unequal probability samples can more difficult to select, more difficult to use in making population estimates, and usually more difficult to make valid estimates of precision. Unrestricted and simple random without replacement samples have a fixed sample size. Bernoulli samples have a random sample size. With bernoulli and simple random without replacement samples, a frame record may appear in the sample only once. With unrestricted random samples, a record may appear in the sample multiple times.

USING THE DATA STEP TO SELECT A RANDOM SAMPLE

One way to select a random sample is to use BASE/SAS® and the data step. This gives control over how you select the sample. Three types of random samples often used are: bernoulli sampling, simple random sampling with replacement, and simple random sampling without replacement. As explained above, to select a random sample in the data step you need to do several things:
(1) determine the number of records in the frame,
(2) generate a random number,
(3) associate the random number with a record of the data file,
(4) find and write the random record to a sample file, and
(5) repeat the process the required number of times in a method appropriate for the sample design.

FINDING THE FRAME OR POPULATION SIZE

The frame size is needed to convert a random number to an integer that is associated with a random record on the frame data set. It can be done by hard coding the number of records in data step code. However, what is often done is to use the “NOBS = *variable*” option of the SET statement. This option creates a temporary variable whose value is usually the total number of records in the data set(s) associated with the SET statement. When there are more the one data set, the variable equals the number of records in all data sets. The number of records includes those marked for deletion. This is illustrated below.

```
SET FRAME NOBS=FRAME_SIZE;
```

The variable FRAME_SIZE contains the number of records in the data set FRAME. At compile time the data step reads the descriptor portion of the data set specified in the SET statement. This means you can refer to the variable before you get to the SET statement.

GENERATING RANDOM NUMBERS

The validity of a random sample depends on generating valid random numbers. Random numbers are usually created by a pseudo random number generator. This pseudo random number generator creates a sequence of numbers between 0 and 1 using an arithmetic process to create numbers. These numbers satisfy statistical tests for random number sequences but they can be repeated by starting again at the same point. The tests check that every number is equally likely both individually and in combination with other numbers.

In BASE/SAS, the pseudo random number generator is contained in the RANUNI and UNIFORM functions. These is the same function known by two different names. It is a *pseudo* random number generator because after creating many random numbers the sequence will repeat itself. For most SAS programs, this is of no consequence. Since RANUNI and UNIFORM are the same function, only the UNIFORM function will be used here. The generation of random numbers is controlled by the SEED which is a number that starts the random number generator. When the same seed number is used, the same numbers will be generated. This is useful in testing and simulation studies. When the seed is zero or negative, the current value of the system clock is used to generate random numbers. The code below generates a data set named random that contains 20 random numbers. Every time the program is run it generate the same numbers.

```
Data random;
do i = 1 to 20;
  seed=521 ;
  random = uniform (seed);
  IF I<4 THEN DO;
    PUT "RECORD = "I 'RANDOM NUMBER = ' RANDOM;
    END;
    output;
end;
run;
```

```
RECORD = 1 RANDOM NUMBER = 0.1858340233
RECORD = 2 RANDOM NUMBER = 0.8420559796
RECORD = 3 RANDOM NUMBER = 0.4651327014
NOTE: The data set WORK.RANDOM has 20 observations and 3 variables.
```

A random number is a number between zero and one; but, it can not take the value of either zero or one.

ASSOCIATING A RANDOM NUMBER WITH A RANDOM INTEGER (SPECIAL SAS FUNCTIONS: CEIL, INT, FLOOR)

Random numbers generated using the UNIFORM function are used together with special SAS functions to identify records. This is done by associating a random number with an integer from one to the number of records in the data set. The code and table below show how the function is used and the relationship between the three functions.

```
DATA FUNCTION;
DO I=1 TO 15;
   X=7-I-.3;
   INT=INT(X);
   CEIL=CEIL(X);
   FLOOR=FLOOR(X);
   OUTPUT;
END;
RUN;
```

Three different and related functions used are: INT(.), CEIL( ), and FLOOR(.). Each of these functions converts a real number to an integer in different ways. INT(.) takes the integer part of a number (e.g. INT(7.7) = 7); CEIL( ) takes the next largest integer for a number (e.g. CEIL(7.7) = 8); and FLOOR( ) takes the next lowest integer of a number (e.g. FLOOR(7.7) = 7). The value of the INT() is the same as either the CEIL or FLOOR function depending on whether a number is positive or negative.

A PROC PRINT of this file is given below.

```
Obs  FLOOR    X    CEIL  INT
  1    5     5.7     6   5
  2    4     4.7     5   4
  3    3     3.7     4   3
  4    2     2.7     3   2
  5    1     1.7     2   1
  6    0     0.7     1   0
  7   -1    -0.3     0   0
  8   -2    -1.3    -1  -1
  9   -3    -2.3    -2  -2
 10   -4    -3.3    -3  -3
 11   -5    -4.3    -4  -4
 12   -6    -5.3    -5  -5
 13   -7    -6.3    -6  -6
 14   -8    -7.3    -7  -7
 15   -9    -8.3    -8  -8
```

FINDING AND RETRIEVING A RANDOM RECORD

The CEIL and other functions are used to identify a random record number. The “POINT=*variable* “ option on the SET statement is used to find and write the record to a sample data set. The “POINT=*variable”* option to the SET statement specifies a temporary variable whose numeric value determines which observation is read. “POINT=*variable*” causes the SET statement to use random (direct) access to read a SAS data set. The “POINT=*variable* “ option works with both compressed and uncompressed data set in version 8; however, in version 6, it does not work with compressed data sets. It also does not work with some data views and sequential data sets.

The code below demonstrates how direct access works. A data set is created defining a variable RECORD_ID containing the record number. A data step assigns a random integer to a variable named SELECTED and uses the “POINT=*variable*” option to write the selected record to a data set. The variables SELECTED and RECORD_ID are compared to show we got the correct record.

```
 DATA RECORD_FILE;
  DO I =1 TO 20;
    RECORD_NUMBER=I;
    OUTPUT;
  END;
  RUN;
NOTE: The data set WORK.RECORD_FILE has 20    observations and 2 variables.
  DATA GET_RECORD;
  SEED = 987;
  RANDOM_RECORD=CEIL( UNIFORM(SEED)*FILE_SIZE);
  PUT "RANDOM RECORD   = " RANDOM_RECORD;
  SET RECORD_FILE POINT=RANDOM_RECORD
NOBS=FILE_SIZE;
  PUT "SELECTED RECORD = "RECORD_NUMBER;
  OUTPUT;
  STOP;
  RUN;

RANDOM RECORD   = 12
SELECTED RECORD = 12
NOTE: The data set WORK.GET_RECORD has 1 observations and 2 variables.
```

How we choose the random record number is determined by the sample design to be used to select the sample.

THREE BASIC SAMPLE DESIGNS

Three different sample designs are used to select samples in the data step: bernoulli sampling, unrestricted random sampling, and simple random sampling without replacement. These three different sample designs are similar in that in each design each record has an equal chance of selection, and equal chance of being in the sample with the other records.

There are a large number of excellent books on sampling. The best non mathematical book on basic sampling is by Stuart(1976) titled “Basic ideas of scientific sampling”. Two good intermediate texts are Cochran’s “Sampling Techniques” (Cochran,1977), and Sukhatme’s (1970) titled “Sampling Theory of Surveys with Applications.”

BERNOULLI SAMPLING

Bernoulli sampling produces a sample with a random sample size. With simple random sampling with and without replacement, the number of samples is fixed and known before the sample is selected. With bernoulli sampling, only the expected number of samples is known; the sample size is a binomial random variable. Bernoulli sampling only requires the data set be read once. The sample can be selected either by reading each record in the data set or by using a random access method. A sample is selected by creating or reading a random number for each record in the data set. If the random number is less than or equal to the probability of selection, the record is selected into the sample. If the random number is greater than the probability of selection, it is not selected. The SAS code to select a bernoulli sample by reading every record in the data set is given below.

```
%let sample_size=5000;
DATA FRAME_FINAL BERNOULLI;
  SamplingProbability=&SAMPLE_SIZE/FRAME_SIZE; ❶
  SET  FRAME NOBS=FRAME_SIZE;
  SELECTED=0;
  SamplingWeight=0;
  SEED=12345;
  IF UNIFORM(SEED) =< SamplingProbability
     THEN
       DO;  ❷
         SamplingWeight=1/SamplingProbability; ❸
         SELECTED=1;
         OUTPUT BERNOULLI;
       END;
  OUTPUT FRAME_FINAL;
RUN;
```

Several things to notice in the SAS code above.

❶ Determines the probability a record is selected.
❷ Checks to see if the random number generated for the record (UNIFORM(seed)) is less than or equal to the SamplingProbability.
❸ Calculates the SamplingWeight by dividing one by the probability of selection.

Bernoulli Sampling is simple; but, it gives a random sample size that some find unacceptable. Also, the sample when expanded has as estimated universe size different from the true universe size. This, however, can be adjusted for using a ratio estimator if needed.

UNRESTRICTED RANDOM SAMPLING (SIMPLE RANDOM SAMPLING WITH REPLACEMENT)

In unrestricted or simple random sampling with replacement a fixed number of records is selected. Each time a record is selected, any record on the frame can be selected into the sample. This means that a sample record can be selected into the sample multiple times. The code to select a simple random sample with replacement is given below. This code is patterned after the code in Boudreaux (1995) and example code in the course notes for “SAS PROGRAMMING III: Advanced Programming Techniques” (page 131, SAS Institute, 2000a) .

```
%let sample_size=5000;
DATA  SRS_WR  ❶   ;
SEED=123466;
SamplingProbability=&SAMPLE_SIZE/FRAME_SIZE;❺
SamplingWeight=1/SamplingProbability;  ❻
DO samp_cnt=1 TO &sample_size;  ❷
   SELECT=CEIL( UNIFORM(SEED) * FRAME_SIZE ); ❸
   SET FRAME POINT=SELECT NOBS=FRAME_SIZE;  ❹
   output;
end;
stop;
run;
NOTE: The data set WORK.SRS_WR has 5000 observations and 12 variables.
```

Things to notice about the code.

❶ The sample data set with one record for each selection and may contain duplicate records.
❷ This DO loop executes once for each sample record.
❸ SELECT is a random integer from 1 to the number of records in FRAME data set.
❹ This statement goes to the record number specified by SELECT.
❺ Calculates the probability of a record being selected.
❻ Calculates the sampling weight of a record

The code above selects a sample that can contain duplicate records. The code below removes the duplicate records, and writes a file of unique records containing a variable with a count of the number of duplicate records associated with each unique record.

```
PROC SORT data=srs_wr ;
by id_num;run;

NOTE: There were 5000 observations read from the data set WORK.SRS_WR.
NOTE: The data set WORK.SRS_WR has 5000 observations and 12 variables.

DATA srs_unique;  ❶
SET  srs_wr;  ❷
RETAIN HITS ;
by id_num;  ❸
if first.id_num=1 then Hits=0;  ❹
Hits+1;  ❺
if last.id_num=1 then
  do;  ❻
      NumberHits=HITS;  ❼
      output;  ❽
  end;
run;
NOTE: There were 5000 observations read from the data set WORK.SRS_WR.
NOTE: The data set WORK.SRS_UNIQUE has 4734 observations and 13 variables.
```

Things to notice about the code.

❶ The output data set contains only unique records and will usually have fewer records than the input data set.
❷ The input data set is the sample set with duplicate records.
❸ The input data set is sorted by a record identification so that duplicate will records will be together on the file. Using a "BY" statement will allow the identification of the first and last duplicate record.
❹ Identifies the first record with a unique identifier and set the HITS variable to zero.
❺ Counts the number of duplicate records.
❻ Identifies the last record with a unique identifier.
❼ When it's the last record with an identifier, the variable NumberHits is set equal to the number of records with that identifier.
❽ The last record with a unique identifier is written to the output file (SRS_WR_UNIQUE)

The NumberHits variable can be used with a FREQ statement to process and analyze this sample. Notice how it is used in the example in the Validating the Sample section.

SIMPLE RANDOM SAMPLING WITHOUT REPLACEMENT

In simple random sampling without replacement, a fixed number of records is selected each time. A record can be selected into the sample only once; therefore, the sample contains unique records. Only records not selected earlier can be selected into the sample. Simple random sampling without replacement is one of the most common methods of random sampling. For the same sample size, it will always be statistically more efficient than simple random sampling with replacement. There are many ways to select simple random samples without replacement. Two methods – the moving constant and array – are given below.

Moving Constant Method

The moving constant method selects a sample in at most one pass through the data set. It can be less because the method goes through the frame sequentially and stops when the fixed sample size is obtained. The code is patterned after the code in Boudreaux (1995) and example programs in the course notes for
"SAS PROGRAMMING II: Manipulating Data in the Data Step" (page 362, SAS Institute2000b).

```
%Let sample_size=5000;
DATA  SRS_WTR_moving_constant  ;  ❶
SEED=123;
pointer=0;  ❷
SamplingProbability =&SAMPLE_SIZE/FRAME_SIZE;
SamplingWeight = Frame_Size/&Sample_SIZE;
Sample_Size = &SAMPLE_SIZE;  ❸
DO WHILE (Sample_Size > 0) ;  ❹
    Pointer=Pointer + 1;  ❺
    Random_Number = UNIFORM(SEED);  ❻
    Moving_Constant=
    Sample_Size /(FRAME_SIZE -(POINTER-1)); ❼
    IF RANDOM_NUMBER =< MOVING_CONSTANT THEN
       DO;  ❽
         SET FRAME  POINT=POINTER
                 NOBS = FRAME_SIZE;  ❾
          OUTPUT;
          SAMPLE_SIZE = SAMPLE_SIZE -1 ;  ❿
       END;
END;
STOP;
RUN;
```

```
NOTE: The data set WORK.SRS_WTR_MOVING_CONSTANT has 5000 observations and 13 variables.
```

Things to notices about this code.

- ❶ This is the output sample data set.
- ❷ This is a pointer that starts at zero and is incremented by one until all samples have been selected.
- ❸ Creates a variable called sample size that contains the number of records to be selected. See ❹❼❿
- ❹ The do loop will continue as long as samples need to be selected.
- ❺ The is a counter that indicates the next record to read from the FRAME data set. It starts at 1 and is incremented by 1 until all samples are selected.
- ❻ Identifies a random number for each record considered for selection on the FRAME.
- ❼ Calculates a moving constant which is the proportion of the frame remaining to be included in the sample.
- ❽ The random number between 0 and 1 is compare to the Moving_Constant When the random number is less than or equal to the Moving_Constant a record is selected.
- ❾ Goes to the record identified as selected.
- ❿ In ❸ the variable sample_size is set to the number of samples needed. It is decreased by one when a sample is identified. In ❹ it is checked to see if more samples are needed.

Array Method

A second and more elegant way to select a simple random sample without replacement is to use an array to identify all records selected and then to repeatedly select records at random. New records are entered into the array; records already in the array are discarded and another record selected. The code is pattern after the code in Boudreaux(1995).

```
%let sample_size=5000;
DATA  SRS_WTR_array  ;
SEED=123;
COUNT=0;  ❶
SamplingProbability=&SAMPLE_SIZE/FRAME_SIZE;  ❷
SamplingWeight = Frame_Size/&Sample_Size;  ❸
ARRAY SEL_OBS{ &SAMPLE_SIZE } _temporary_ ;  ❹
DO K=1 TO &SAMPLE_SIZE;  ❺
   ReDo:  SELECT=
          CEIL(UNIFORM(SEED)*FRAME_SIZE);  ❻
   DO SAMPL_CNT=1 TO COUNT;  ❼
          IF SEL_OBS(SAMPL_CNT)=
                         SELECT then GOTO ReDo; ❼
   END;
   SET FRAME POINT=SELECT NOBS=FRAME_SIZE; ❽
        COUNT=COUNT+1;  ❾
   SEL_OBS(COUNT)=SELECT;  ❾
   OUTPUT;
END;
  STOP;
  RUN;
NOTE: The data set WORK.SRS_WTR has 5000 observations and 14 variables.
```

Things to notice about the code.

- ❶ Count is a variable that contains the number of sample records that have been selected at this point in processing.
- ❷ Calculates the probability of a record be selected into the sample
- ❸ The sample weight associated with each sample record is the reciprocal of the SamplingProbability.
- ❹ This creates the array to hold the identifier of the records selected into the sample.
- ❺ Do loop that has one loop for each sample record. You don't get out of the loop until you select a record.
- ❻ This codes identifies a random record to pick. If the record has already been selected, the data step returns to this line. See ❼.
- ❼ The DO loop compares all records selected to the record just selected SELECT. If there is a match– the record has already been selected – the data step returns to ❻ and a new number is picked.
- ❽ It is a record number is not already select, the value of SELECT is used to get the record selected.
- ❾ After a record is selected, the variable COUNT is increment to identify the sample record number and where in the array the record is to be entered.

The array method is a bit more sophisticate than the moving constant method. It should also be faster because of the use of the random access feature of the SET statement.

USING PROC SORT, PROC RANK, AND INDEXES TO SELECT SIMPLE RANDOM WITHOUT REPLACEMENT SAMPLES

An alternative to selecting the sample in the data step is to use a procedure such as PROC SORT or PROC RANK to select the sample, or to use an indexed data set. A requirement of all three methods is that each record has a random variable associated with it. The validity of selecting a simple random sample without replacement using one of these three methods is that the FRAME data set is sorted by a random variable on the data set and that taking the records in a random order is equivalent to selecting a random number. The first n records in a randomly ordered population is a simple random sample without replacement of size n. When a data set has an independent uniform random number associated with each record, then PROC SORT, PROC RANK, can be used to put the data set into a random order. Indexing a data set on a random number also creates a random order. The three techniques are discuss below.

These methods can use either an ordinary data step or a data view. Both ways are illustrated below. PROC SORT creates a new data set ordered by the specified variable. PROC SORT changes the physical order of the data set. PROC RANK is similar to PROC SORT;but, it assigns a unique number called the rank to one or more variables on a data set where the record with the smallest value of a variable has rank one and the record with the largest value of the variable has rank N. The order of the records in the data set is not physically changed.

PROC SORT

PROC SORT changes the physical order of a data set. Two ways to use PROC SORT to select simple random samples without replacement are give below.

Method One
```
/* LONG VERSION */;
%let Sample_Size = 5000;
proc sort data=frame OUT=FRAME_SORT;
by random;run;

data sample_LONG_SORT;
set frame_SORT (obs=&SAMPLE_SIZE) NOBS=FRAME_SIZE;
SamplingWeight = FRAME_SIZE/&SAMPLE_SIZE;    *
run;
```

The code above is conceptually simple; but requires sorting the entire data set. An abbreviated version can save time if the FRAME data set has either a lot of records or variables. This resembles using a tagsort or index.

Method Two
```
/*SHORTER VERSION */;
PROC SORT  data=frame(keep=id_num random)
           out=frames;  ❶
by random;
run;

PROC SORT data =frames(obs=&SAMPLE_SIZE);  ❷
by id_num;
run;

DATA _NULL_;
SET FRAME NOBS=F_SIZE;
IF _n_=1 THEN CALL SYMPUT('FRAME_SIZE',F_SIZE);❸
STOP;
RUN;
DATA sample_SHORT_SORT;  ❹
merge frame
       frames(in=in_sample );
       by id_num;
       if in_sample;
       SamplingWeight=&FRame_size/&sample_size;
       run;
```

Things to notice in the code.

❶ The first PROC SORT keeps only a record identifier and a random number to order the FRAME.
❷ The second PROC SORT keeps only the first n records and then sorts the sample records by their ID number for matching.
❸ The Data _NULL_ statement is used to capture the FRAME size.
❹ The final data step is used to obtain complete information for the sample records from the FRAME data set and to add the sampling weight to sample file.

The use of PROC SORT is conceptually simple. It also can consume a lot of resources since you must add a random number to each record of the data set, sort the data set by that random number, and then sort the data set again to extract the first n records

PROC RANK
An alternative to using PROC SORT or an indexed data set is to use PROC RANK. For numeric variables, the rank of a variable is its order. Ranking a set of numbers is equivalent to sorting them. For example, five random 2 digit integers and their ranks are give below. Ordering a data set by the rank of a random number is the same as sorting the data in random order.

There are many ways to determine a rank. PROC RANK reads the data set , sorts specified variables that you identify, and places the order or rank of the variable into either the original variable or a new variable. In order to determine the first n records in a randomly ordered data set, we can use PROC RANK to determine the rank of the random variable and then keep only records with a rank of less then the sample size.

Original Order			Rank Order		
_N_	RANDOM INTEGER	RANK	_N_	RANDOM INTEGER	RANK
1	34	3	4	3	1
2	67	4	3	21	2
3	21	2	1	34	3
4	03	1	2	67	4
5	78	5	5	78	5

One problem is how PROC RANK deals with ties. When two records have identically the same value for a variable, PROC RANK considers this a tie. Tied records are assigned the same value for a rank. When dealing with random number, it is unlikely there will be a tie; however, to be safe you should specify “TIES=LOW.” This will assign the lowest rank to all tied values and the number of records will be greater than the sample size. The code below illustrates the use of PROC RANK to select a simple random sample without replacement. It also illustrates how a data view can be used to add a random variable to a SAS data set.

```
data frameV/view=frameV;   ❶
set  frame (drop=random);
seed=12345;
random=uniform(seed);
run;

DATA _NULL_;  ❷
SET FRAME NOBS=F_SIZE;
IF _n_=1 THEN CALL SYMPUT('FRAME_SIZE',F_SIZE);
STOP;
RUN;

proc rank data=frameV  TIES=LOW
  out=sample(where=(SELECTED <= &sample_size));❸
var random;
ranks SELECTED;
run;

DATA SAMPLE_RANK;  ❹
SET  SAMPLE(obs=&sample_size);
     SamplingWeight=&frame_size/&sample_size;
run;
```

Things to notice in the code.

❶ The use of a data view effectively adds a random variable to the data set without actually changing the data set.
❷ The DATA _NULL_ statement is used to identify the number of records in the FRAME.
❸ The PROC RANK statement does multiple things. The variable SELECTED specifies the random order to FRAME data set. The where statement writes records until then number of records equals at the least the sample size.
❹ This adds the SamplingWeight to the sample data set.

PROC RANK is similar to using PROC SORT.

USING AN INDEXED DATA SET

An alternative to using either PROC SORT or PROC RANK is to use an indexed data set where the index is built using a random number. Using an indexed data set is similar to using a sorted data set. When an index variable is used with a “BY” statement, the data set is processed in index order. When the index is a uniform random number, the data step reads and processes the records in random order. The code below illustrates how this can be used to select a simple random sample without replacement.

```
%let Sample_Size = 5000;
proc datasets library=work;  ❶
modify frame;_
    index create random;
    quit;
    run;
data sample_index;
set frame(obs=&Sample_Size) ;  ❸
by random;  ❷
    run;
```

Things to notice in the code.

❶ PROC DATASETS creates an index for the data set frame in the library work. The index is based on the single variable random in the data set.
❷ When a "BY" statement is used with an indexed data set, the records are processed in index order.
❸ Restricts the number of records read to the sample size.

PROC SURVEYSELECT

PROC SURVEYSELECT and companion procedures PROC SURVEYMEANS and PROC SURVEYREG are three relatively new procedures in SAS/STAT oriented to survey sampling. Only PROC SURVEYSELECT is discussed here. PROC SURVEYSELECT is specifically written to select a wide variety of random samples and takes into consideration many of the special things that you need to do when selecting a random sample. Its real strength probably lies in its ability to select unequal probability samples. The table below lists the sample designs that can be selected with PROC SURVEYSELECT.

PROC SURVEYSELECT Sample Selection Methods Available	
Equal Probability	Unequal Probability
Unrestricted Random Sampling	PPS Sampling With Replacement
Simple Random Sampling Without Replacement	PPS Sampling Without Replacement
Systematic Sampling	PPS Systematic Sampling
Sequential Sampling	PPS Sequential Sampling
	Brewer's PPS Method
	Murthy's PPS Method
	Sampford's PPS Method

Only unrestricted and simple random sampling without replacement sampling are discussed here. PROC SURVEYSELECT does not do bernoulli sampling.

The basic syntax of the PROC SURVEYSELECT statements are

```
   PROC SURVEYSELECT options;
   STRATA   variable(s);
   CONTROL variable(s);
   SIZE variable ;
   ID variable;
   RUN;
```

This is illustrated by the code below that selects a sample of 1,000 records and writes then to a SAS data set named WORK.SAMPLE. The sample is selected from the SAS data set WORK.US using a stratified PPS systematic sample design where the strata are defined by the data set variable STATE and the frame is sorted by the variable POPULATION. The measure of size used to select the PPS sample is the data set variable HOUSEHOLDS.

```
PROC SURVEYSELECT
     DATA = WORK.US
     OUT = WORK.SAMPLE
     METHOD = PPS_SYS
     SAMPSIZE = 1000;
   STRATA  STATE;
   CONTROL POPULATION ;
   SIZE HOUSHOLDS ;
   RUN;
```

The STRATA, CONTROL, and SIZE statements are not used to select unrestricted or simple random without replacement samples. The STRATA statement is used to identify variables needed to define strata when samples are selected from different groups. The CONTROL statement is used to specify the order of frame units prior to selection for designs such as systematic and sequential sampling. The CONTROL statement has no meaning in the context of unrestricted or simple random sampling. The SIZE statement is used to specify a data set variable to be used as a measure of size when an unequal probability sample design is used. The only statement that has meaning here is the ID statement. The ID statement can, but does not have, to be used.

The PROC SURVEYSELECT ID statement determines what data set variables will be written to the output sample data set. Using the ID statement is the equivalent of using a KEEP statement with the input data set. The KEEP statement is likely to be more efficient particularly if the frame data set has a large number of variables. The following two statements give the same result.

PROC SURVEYSELECT	
ID Statement	Keep Statement on Data =
PROC SURVEYSELECT;	PROC SURVEYSELECT;
DATA=FRAME	DATA=FRAME
METHOD=URS	(KEEP= ID_NUM N1 N2)
OUT=SAMPLE_URS	METHOD=URS
SAMPSIZE=5000;	OUT=SAMPLE_URS
ID ID_NUM N1 N2 ; *	SAMPSIZE=5000;
RUN;	RUN;

For equal probability sample designs, only options on the PROC SURVEYSELECT statement control how the sample is selected. The following options can be used with the PROC SURVEYSELECT.

PROC SURVEYSELECT options;			
DATA=	SAMPSIZE=	NOPRINT	CERTSIZE=
OUT=	SAMPRATE=	OUTHITS	MINSIZE=
METHOD=	NMIN =	OUTSIZE	MAXSIZE=
REP=	NMAX =	STATS	SORT =
OUTSORT=	SEED=	JTPROBS	

The following options are not used with unrestricted and simple random sampling without replacement: JTPROBS, CERTSIZE=, MINSIZE=, MAXSIZE=, SORT=, OUTSIZE=, and OUTSORT=. These options are used with the selection of unequal probability samples or samples that require the FRAME to be sorted in some particular order.

The following are general purpose options used with most sample designs.

Options	Description
METHOD=	Specifies the sampling method to use (e.g. URS, SRS, SYS, SEQ)
DATA=	Specifies the input data set or frame.
OUT=	Specifies the sample data set.
SEED=	Default is SEED=0. It specifies a seed for random numbers used to select sample records.
REP=	Default is REP=1. It specifies the number of samples selected and written to the output data file.
STATS	Optional; Writes the SamplingProbability and SamplingWeight and other sample design information to the output data set
NOPRINT	Optional; Suppresses the printing of Display Output.
OUTSIZE	Optional; Writes the universe total and sample total to the output data set.
SAMPSIZE=	Specifies the number of sample records to select. Used instead of SAMPRATE=
SAMPRATE=	Specifies the sampling rate. Used instead of SAMPSIZE=

SAMPLE DATA SETS FROM PROC SURVEYSELECT

PROC SURVEYSELECT creates a file of sample records selected where the contents of the file can be controlled. The ID statement can control what information from the FRAME data set is written to the sample file. Several options control whether additional information is written to the sample data set. The actually information written depends on the method or sample design used.

The option OUTSIZE causes two variables --TOTAL and SampleSize – to be written to the sample data set. TOTAL contains the number of records on the frame data set; SampleSize contains the number of records selected. When unrestricted random sampling or unequal probability sampling is used, the number of records on the sample file may be less than the value of SampleSize. The option STATS causes SamplingWeight and other method related variables to added to the sample file.

Information in the output data set is different for different sample designs. Variables may be added to the sample data set depending on the sample design selected. For unrestricted random samples and some PPS designs the variable NumberHits is added that contains the number of times a record has been selected.

DISPLAY OUTPUT FROM PROC SURVEYSELECT
PROC SURVEYSELECT displays two tables to summarize the sample selection: the sample selection method table, and the sample selection summary table. These are Output Display System (ODS) tables that can be written to SAS data sets or have their display modified with PROC TEMPLATE. The printing of these tables and the Output Display System can be suspended by using the NOPRINT option.

The display output of a PROC SURVEYSELECT statement generated for Simple Random Sampling when the sampling rate rather than a sample size is specified is given below.

```
The SURVEYSELECT Procedure

Selection Method    Simple Random Sampling
Input Data Set                       FRAME
Random Number Seed                   39756
Sampling Rate                         0.05
Minimum Sample Size                   1000
Maximum Sample Size                   2000
Sample Size                           2000
Selection Probability                 0.04
Sampling Weight                         25
Output Data Set             SAMPLE_SAMPRATE
```

SELECTING AN UNRESTRICTED RANDOM SAMPLE USING PROC SURVEYSELECT

A simple random sample with replacement can be selected using PROC SURVEYSELECT. It requires the same choices you need to make when selecting samples in the data step: How many samples to take? Do we keep duplicate records? The code below shows the PROC SURVEYSELECT options to select an unrestricted random sample with duplicate or with unique records.

PROC SURVEYSELECT METHOD=URS	
Unique Records	Duplicate Records
PROC SURVEYSELECT DATA=FRAME METHOD=URS OUT=SAMPLE_URS1 SAMPSIZE=5000; **RUN**;	**PROC SURVEYSELECT** DATA=FRAME METHOD=URS OUTHITS OUT=SAMPLE_URS2 SAMPSIZE=5000; **RUN**;

The sample SAMPLE_URS2 contains duplicate records; the sample SAMPLE_URS1 contains unique records; With URS samples, the default is for PROC SURVEYSELECT to write each unique record and the number of times it was selected in the variable NUMBERHITS. When the OUTHITS option is specified, the sample file contains duplicate records with the number of records equaling the sample size.

SELECTING A SIMPLE RANDOM SAMPLE WITHOUT REPLACEMENT USING PROC SURVEYSELECT

The example below shows how PROC SURVEYSELECT is used to select a sample by specifying the sample size. In the example, a simple random sample without replacement of 5000 records is selected from the FRAME data set.

```
PROC SURVEYSELECT
       DATA=FRAME  METHOD=SRS
       OUT=SAMPLE_SRS2
       OUTSIZE STATS
       SAMPSIZE=5000;
  ID ID_NUM N1 N2 N3 N4 ;
  RUN;

NOTE: There were 50000 observations read from the data set WORK.FRAME.
NOTE: The data set WORK.SAMPLE_SRS2 has 5000 observations and 9 variables.
```

The display output associated with this code is given below.

```
                                                        The SURVEYSELECT Procedure
   Selection Method    Simple Random Sampling
   Input Data Set                     FRAME
   Random Number Seed                 85482
   Sample Size                         5000
   Selection Probability                0.1
   Sampling Weight                       10
   Output Data Set              SAMPLE_SRS2
```

The sample is written to the SAS data set SAMPLE_SRS2.

VALIDATING THE SAMPLE

Once the sample has been selected, some survey statisticians routinely validate the sample. Validating the sample refers to comparing totals based on the expanded sample and the frame. Sometimes it may include calculating confidence limits for the sample estimate of the frame total in order to determining how close an estimate is to the true FRAME value and whether the true total is inside the estimated confidence limits. There are many ways to do this. Code that calculates the true and estimate totals and compares them is given below.

```
PROC SUMMARY DATA=FRAME NOPRINT N SUM;
VAR RECORDS RANDOM N1 N2 N3 N4;
OUTPUT OUT=F_STAT
       SUM=RECORDS RANDOM N1 N2 N3 N4;
       RUN;

PROC SUMMARY DATA=srs_wr_unique NOPRINT N SUM;
  FREQ NUMBERHITS;
  VAR RECORDS RANDOM N1 N2 N3 N4  / WEIGHT=SamplingWeight;
  OUTPUT OUT=S_STAT
    SUM=RECORDS RANDOM N1 N2 N3 N4;
    RUN;

PROC TRANSPOSE DATA=F_STAT
     OUT=FSTAT
     (RENAME=(COL1=FRAME));;RUN ;
PROC TRANSPOSE DATA=S_STAT
     OUT=SSTAT
     (RENAME=(COL1=SAMPLE));;RUN ;

PROC SORT DATA=FSTAT;BY _NAME_;RUN;
PROC SORT DATA=SSTAT;BY _NAME_;RUN;

DATA PROOF;
  MERGE FSTAT SSTAT;
  BY _NAME_;
  IF SUBSTR(_NAME_,1,1)="_" THEN DELETE;
  ABS_DIF=SAMPLE-FRAME;
  REL_DIF=ABS_DIF/FRAME;
RUN;

PROC REPORT DATA = PROOF NOWD split="*";
  TITLE1 "SIMPLE RANDOM SAMPLE - WITH REPLACEMENT";
  TITLE2 "COMPARISON EXPANDED SAMPLE AND FRAME";
  TITLE3 "SAMPLE SIZE IS &SAMPLE_SIZE";
  COLUMN _NAME_ SAMPLE FRAME
  (  DIF PCT);
  DEFINE _NAME_ / ORDER "Variable";
  DEFINE FRAME / ANALYSIS FORMAT =COMMA16.0  center  'Frame*Total';
  DEFINE SAMPLE / ANALYSIS FORMAT =COMMA16.0  center 'Expanded*Samp';
  DEFINE DIF / noprint COMPUTED FORMAT=COMMA12.0
   "ABSOLUTE";
  DEFINE PCT / COMPUTED FORMAT=PERCENT9.2
   "PCT*DIFF";
  COMPUTE DIF;
     DIF=SAMPLE.SUM - FRAME.SUM;
  ENDCOMP;
  COMPUTE PCT;
      PCT=DIF/FRAME.SUM;
  ENDCOMP;
  RUN;
```

The table produced by this code is given below. This table allows us to compare how well the sample expands and how representative the sample is of the frame. Large differences often are indicators of some problem in sample selection; however, they can also be due to chance.

```
                         SIMPLE RANDOM SAMPLE - WITH REPLACEMENT
                          COMPARISON EXPANDED SAMPLE AND FRAME
                                  SAMPLE SIZE IS 5000

           Expanded          Frame          PCT
Variable     Samp            Total          DIFF
N1       249,528,549   249,964,709  (  0.17%)
N2       249,873,087   249,917,913  (  0.02%)
N3       250,373,270   249,878,071     0.20%
N4       250,253,241   250,002,038     0.10%
RANDOM        25,071        24,978     0.37%
RECORDS       50,000        50,000     0.00%
```

CONCLUSION

There are many different sample designs and a variety of methods for selecting samples from each one of them. This paper has attempted to review how others have chosen to select samples and how the new PROC SURVEYSELECT procedure chooses to do it. The code above is relatively simple and works. The problem is the world of survey sampling is not always this simple. Often you will not select a sample from every record in a data set. At the very least, some records will be excluded which will require keeping two data sets or setting a data set variable to indicate records in-scope to the survey universe. Some survey statisticians consider it a principal of good practice to keep a file of the survey frame with the sample records identified. Use of random access methods in either the data step or the use of PROC SURVEYSELECT prevents this and requires that if we want to do this we must “merge” the sample and frame back together. This may reduce some of the efficiencies of using the random access method to select the sample.

REFERENCES

Boudreaux, D. and Cranford K.(1996), “SAS Macros for Simple Random, Stratfied, and Cluster Sampling”, Observations, Fourth Quarter 1996, Cary, NC: SAS Institute Inc.

Boudreaux, D. and Cranford K.(1995), “Simple Random Sampling and Subsetting Strategies Using SAS Software”, Observations, Third Quarter 1995, Cary, NC: SAS Institute Inc.

Cochran, W.G. (1977) Sampling Techniques John Wiley and Sons, New York, NY

SAS Institute Inc. (2000a) SAS Programming III: Advanced Techniques Course Notes, Cary, NC SAS Institute Inc.

SAS Institute Inc. (2000b) SAS Programming II: Manipulating Data with the Data Step Course Notes, Cary, NC SAS Institute Inc.

SAS Institute Inc. (1999a) SAS Language Reference: Concepts, Version 8, Cary, NC, USA

SAS Institute Inc. (1999b) Chapter 63, The SURVEYSELECT Procedure, SAS/STAT User’s Guide, Version 8, Cary, NC, USA

SAS Institute Inc. (1999c) Chapter 30, The RANK Procedure, SAS Procedures Guide, Version 8, Cary, NC, USA

SAS Institute Inc. (1999d) Chapter 33, The SORT Procedure, SAS Procedures Guide, Version 8, Cary, NC, USA

Stuart, Alan (1976) Basic ideas of scientific sampling, Second Edition Griffin’s Statistical Monographs & Courses No. 4 Haffner Press, New York, NY

Sukhatme, P.V. and B.V Sukhatme (1970) Sampling Theory of Surveys with applications” 1970. Iowa State University Press, Ames, Iowa

Williams, Rick L. and James R. Chromy (1980), “SAS Sample Selection macros”, SAS Users Group International, Proceeding of the Fifth Annual Conference, SAS Institute Inc.

ACKNOWLEGEMENTS

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.

CONTACT INFORMATION

I am interested in any comments (good or bad) you have about this paper. Contact the author at:

David D. Chapman
921 N Van Dorn Street, Suite 300
Alexandria, VA 22304
Work: 703-3706708
EMAIL: chapman1943@gmail.com

APPENDIX

CREATING THE FRAME DATA SET

The code below creates the frame data set from which samples were selected in this paper.

```
Data frame(DROP=I SEED );
RECORDS=1;
DO i=1 to 50000;
  id_num=put(i,6.);                            *
  seed=123;
  random=uniform(seed);
  N1=normal(1123)*500 + 5000;
  N2=normal(2123)*500 + 5000;
  N3=normal(3123)*500 + 5000;
  N4=normal(4123)*500 + 5000;
  output;
end;
run;

NOTE: The data set WORK.FRAME has 50000 observations and 7 variables.
```