

Paper 168-31

## Getting Your Random Sample in Proc SQL

Richard Severino, Convergence CT, Honolulu, HI

### ABSTRACT

Proc SQL can be used to get a random sample from a large dataset with relative ease. A more common method of getting a random sample from a large dataset requires using the data step along with some programming or using the SURVEYSELECT procedure which became available in SAS/STAT beginning with SAS Version 8 ®. It is relatively easy to get a simple random sample using only the SQL procedure.

### INTRODUCTION

There are times when a user may desire or need to work with a subset of a very large dataset and therefore may need to obtain a random sample of the larger dataset. The SURVEYSELECT procedure, introduced in SAS/STAT with SAS Version 8, is a procedure which provides a variety of methods for selecting simple random samples or random samples based on more complex multistage sample designs. Prior to the introduction of Proc SURVEYSELECT, selecting a random sample from a large dataset required programming using the DATA step. Another way to obtain a data subset which is a random sample of a large dataset is to use Proc SQL. While this method requires some basic knowledge of Proc SQL, it does not require any data step programming.

The purpose of this poster/paper is to demonstrate the relative ease with which the user can obtain a simple random sample from a large dataset using only Proc SQL.

### RANDOM SAMPLES

Ideally, random samples should be representative of the “population” from which they are randomly selected or drawn. For our purposes, the large dataset that we wish to sample from is the “population” and the dataset records are the observations.

To obtain a simple random sample, we must select records from the population in a random manor such that each record has an equal chance of being selected, i.e. the last record should have the same chance of being selected as the first record or any of the records in between. For example: if we select a random sample from a dataset consisting of a company’s employee data in which 70% of the employees are female, then our random sample from this employee file should have approximately 70% females in it and every employee in the complete dataset should have had an equal chance of being selected.

### SAMPLE SIZE

The size of a random sample can be a predetermined fixed number or it can be a percentage of the total number of observations or records in the large dataset. So, we can pull a random sample of 100,000 observations from the population dataset, assuming there are more than 100,000 observations to begin with, or we can pull a random sample consisting of 10% of all the observations in the large dataset. Methods of using Proc SQL to draw random samples for both scenarios will be presented.

### THE “POPULATION” DATA

To demonstrate how to use Proc SQL to obtain a random sample from a large dataset, we will use a dataset that has one million records. The data in this ‘population’ dataset will have specific and known distributions so that we can show that the random samples we select are in fact representative of the population dataset.

The data we will use for our large “population” dataset is a fictitious dataset created for illustrative purposes only. The dataset name is *POP\_DATASET* and the data consists of the age and gender of one million people as well as two variables representing some characteristic or property of each record/person. The dataset has 5 variables:

Record\_no: this is a unique record/person identifier  
 Norm01\_rv: A random variable with a Standard Normal distribution ( $\mu = 0$ ,  $\sigma = 1$ )  
 Uni01\_rv: A random variable Uniformly distributed on the range (0,1)  
 Sex: the gender of the person  
 Age: the age in years of the person

The distribution of the variable sex is 30 percent ‘Male’ and 70% ‘Female’.

Age is normally distributed with a mean of 40 years and a standard deviation of 4 years for the females in the dataset. For males, the age is normally distributed with a mean of 50 years and a standard deviation of 4 years.

Norm01\_rv is a normally distributed random variable with mean of zero and standard deviation of one, i.e.  $N(0,1)$  and Uni01\_rv is a normally distributed random variable which is uniformly distributed on the range  $(0,1)$ , i.e.  $Uniform(0,1)$ .

The dataset *POP\_DATASET* was generated using the following code:

```

/* ***** *
| create a large data set where the distributions of the variables are known. |
| - create 2 groups such that 30% of the observations are in one group      |
|           and 70% of the observations are in the other group              |
| - create two variables whose distributions do not vary by group          |
| - create one variable whose distribution varies by group                  |
* ***** */

libname ps168 "d:\sug\sug2005\data" ;

data ps168.pop_dataset;

format rec_no 7.0  age 4.0  sex $6. ;

label rec_no = "Record Number / ID"
      age = "Age in Years"
      sex = "Gender"
      norm01_rv = "RV N(0,1)"
      uni01_rv = "RV U(0,1)" ;

do i=1 to 1000000 ; /* one million records */

    rec_no = i ; /* unique record identifier */

    norm01_rv = rannor(2736); /* Random Variable/Number: Normal(0,1) */
    uni01_rv = ranuni(3627); /* Random Variable/Number: Uniform(0,1) */

    if i le 300000 then do ; /* first 30% will be in "Male" group */
        sex = "Male" ;
        age = ROUND( 4*RANNOR(36) + 50 ) ; /* Normal(50,4) */
    end;

    else do ; /* the rest (70%) will be in 'Female' group */
        sex = "Female" ;
        age = ROUND( 4*RANNOR(36) + 40 ) ; /* Normal(40,4) */
    end;

    output;

end;

drop i;

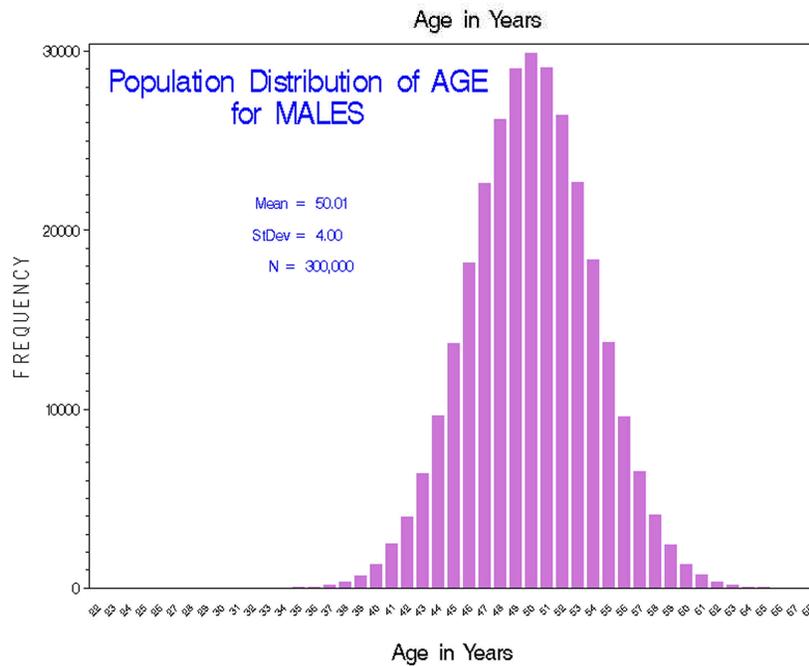
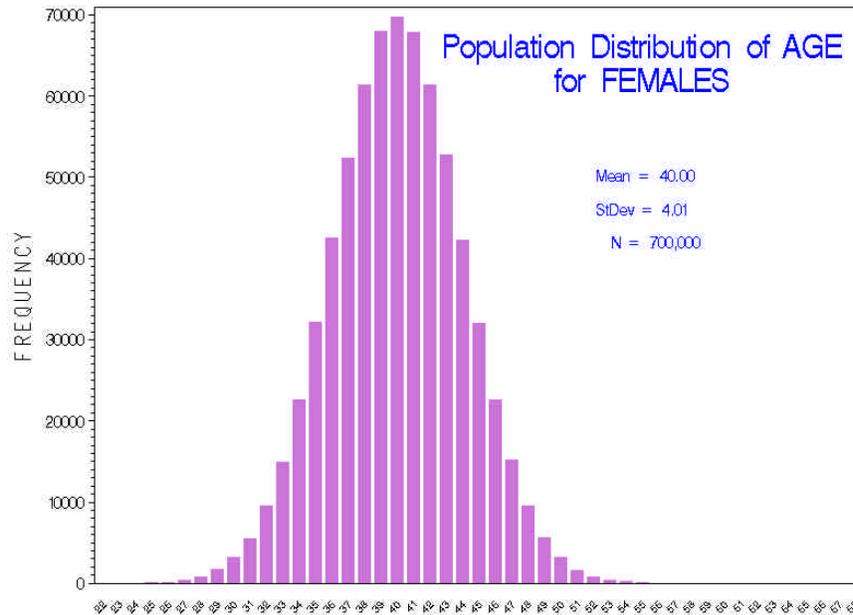
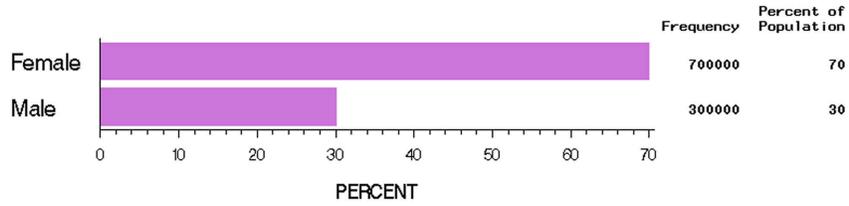
run;

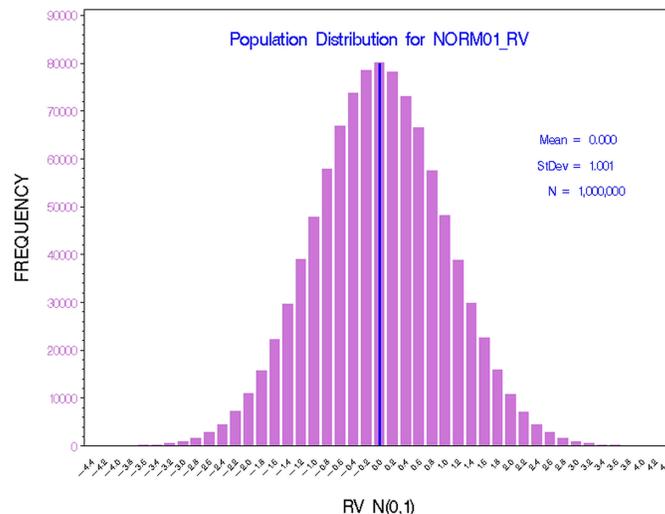
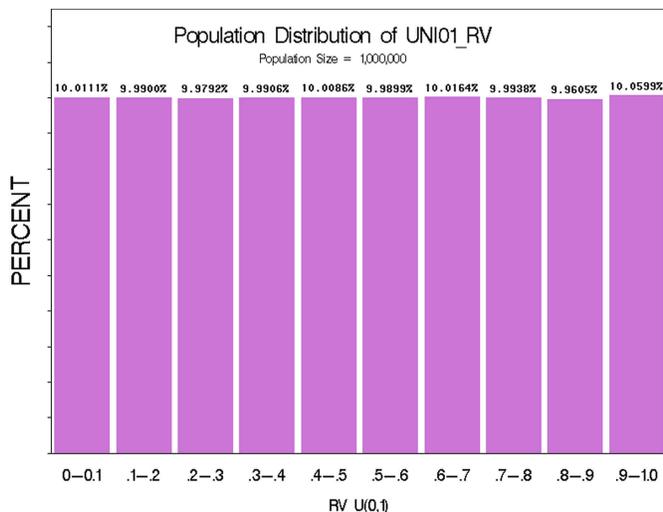
```

The following figures show the distributions of the data in the population dataset POP\_DATASET.

Population Gender Distribution

Population Size = 1,000,000





The variables in any random sample selected from dataset POP\_DATASET should have approximately the same distribution as they have in the population dataset.

### SELECTING A PERCENTAGE OF THE RECORDS FROM THE LARGER DATASET

To randomly select a certain percentage of a large dataset using only Proc SQL, we will make use of the WHERE clause and a random number generating function such as RANUNI().

#### THE WHERE CLAUSE

In Proc SQL, WHERE is used to subset the data, i.e. to select records that meet criteria specified by WHERE. WHERE operates on the rows of the dataset from which we are selecting data. For example, if we wanted to select all the records for females from dataset POP\_DATASET, we would run the following code:

```
/* example of WHERE used in Proc SQL */

proc sql outobs=10;
select * from ps168.pop_dataset WHERE sex="Female" ;
quit;
```

The value of sex is compared to "Female" for each record in *ps168.POP\_DATASET*. If the value of sex is "Female" then the record is selected, otherwise it is not. Note that the OUTOBS option has been used and is set to 10 so that only the first 10 records that meet the criteria where sex="Female" are sent to the output window.

How do we use the WHERE clause to randomly select a percentage of the records in *POP\_DATASET*? That is where the function RANUNI(*seed*) comes in.

#### RANUNI: RANDOM NUMBERS FROM THE UNIFORM(0,1) DISTRIBUTION

RANUNI(*seed*) is a random number generating function which returns a random number that is generated from the uniform distribution on the interval (0,1) . The value *Seed* is used to generate the first random number in the series of random numbers generated by the function. The value of *seed* is optional and must be less than 2,147,483,647, or  $2^{31}-1$ . If *seed* is set to some value less than or equal to zero, or if it is omitted, then time of day is used as the seed value.

RANUNI will generate random values between 0 and 1 such that the percent of numbers generated falling in an interval (a,b) where  $0 \leq a < b \leq 1$  is equal to  $(b-a) \times 100$  percent. If, for example, 1000 random numbers are generated using the function RANUNI, then approximately 10% of the numbers will be between 0 and .1, approximately 10% of the numbers will be between .34 and .44 and approximately 40% of the numbers will be between .25 and .65. We

will use these characteristics of RANUNI and the Uniform(0,1) distribution to select a percentage of the records or observations from a large dataset.

#### RANDOMLY SELECTING 10% OF A LARGE DATASET

We are to select 10% of the records randomly from the *POP\_DATASET* dataset. The following Proc SQL code will create a table called *SAMPLE\_10P* consisting of approximately 10% of the records randomly selected from dataset *POP\_DATASET*:

```
/* select a 10% sample */
proc sql;
create table ps168.sample_10p as
select A.*
  from ps168.pop_dataset as A
 where RANUNI(4321) between .45 and .55 ;
quit;
```

Notice that the seed has been set to 4321.

Recall that the WHERE clause operates on the rows of the table being selected from. Here, records are being selected from dataset *POP\_DATASET*, and therefore the where clause will be evaluated for each record of this dataset. Each time a record from *POP\_DATASET* is being considered or tested for selection, the RANUNI function generates a random number between 0 and 1. If this random number is between .45 and .55, then the record is selected, otherwise the record is not selected. Based on the discussion above, we know that approximately 10% of all numbers generated by RANUNI will be between .45 and .55 which will lead to the WHERE clause to be true approximately 10% of the time as Proc SQL processes records from *POP\_DATASET*. The resulting dataset *SAMPLE\_10P* will therefore have approximately 10% of the records from dataset *POP\_DATASET*.

After running the preceding PROC SQL code, the following is printed in the LOG:

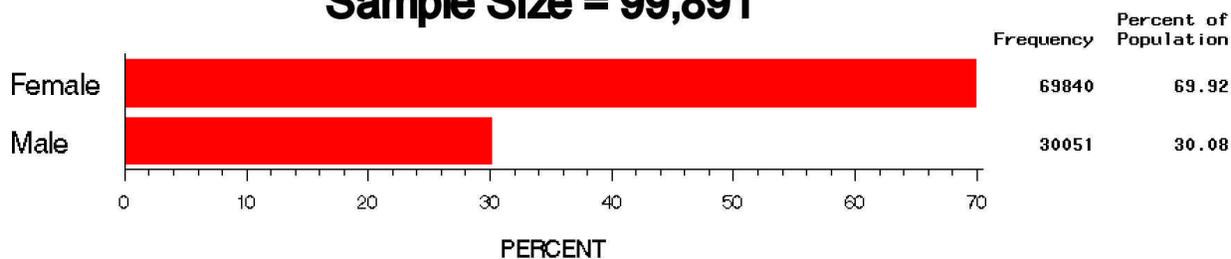
NOTE: Table PS168.SAMPLE\_10P created, with 99891 rows and 4 columns.

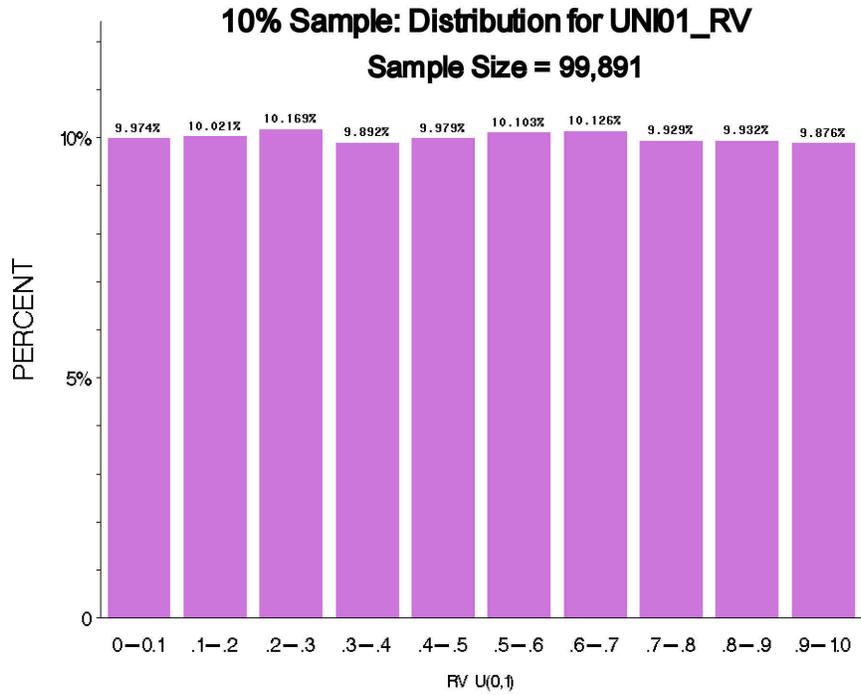
This indicates that there were 99,981 records selected, which is approximately 10% of the one million records in *POP\_DATASET*.

The following figures show the distributions of the variables in the sample dataset *SAMPLE\_10P*. The sample dataset consists of 30.08% males and 69.92% females which is reflective of the population dataset which has 30% males and 70% females. The distribution of age among males and females is also approximately the same as the distribution of age in the population dataset. In the sample dataset, *NORM01\_RV* is normally distributed with a mean and standard deviation of approximately 0 and 1 respectively just as it is in the population dataset. *UNI01\_RV* is appears to be uniformly distributed from 0 to 1 also as it is in the population data.

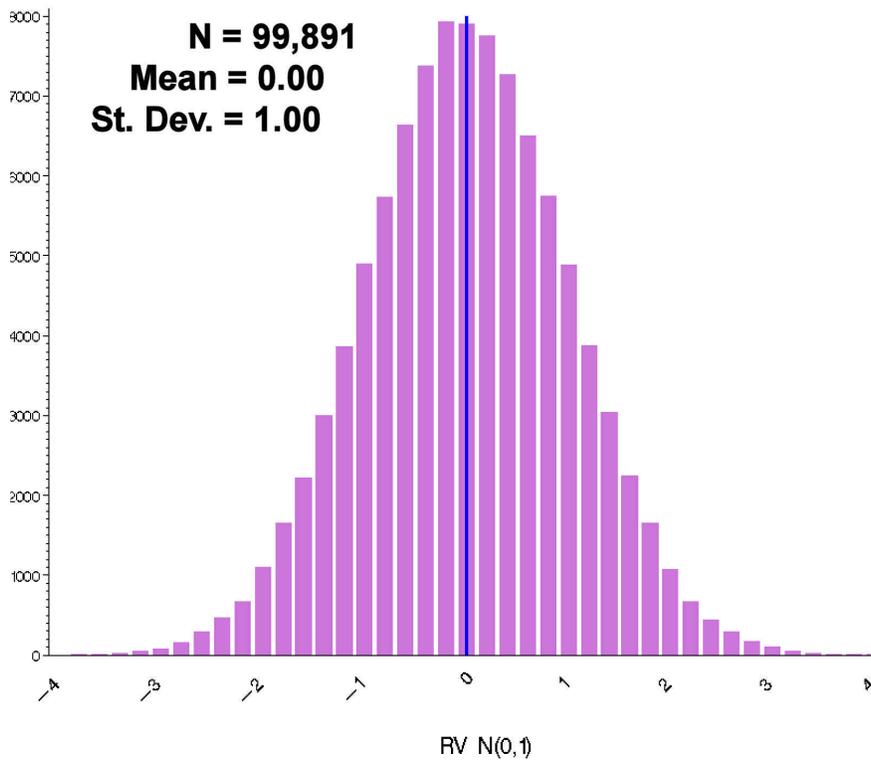
## Gender Distribution for 10% Sample

### Sample Size = 99,891

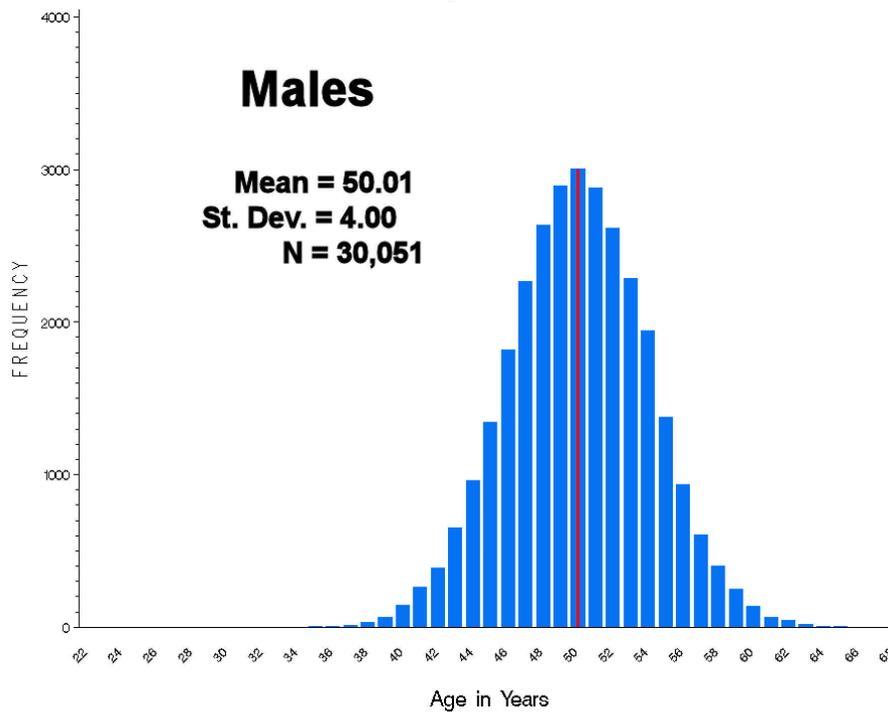
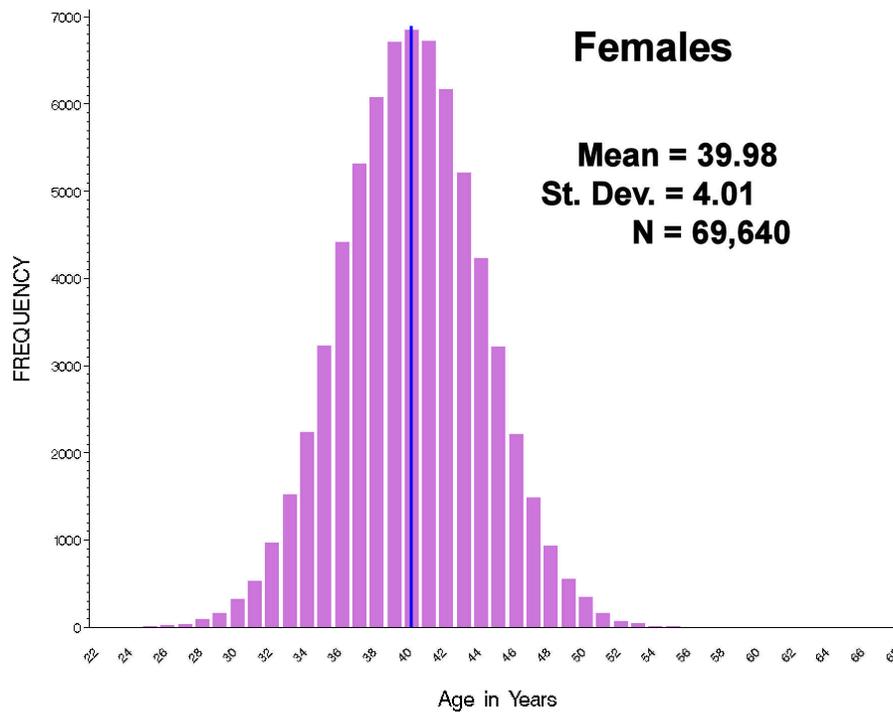




### 10% Sample: Distribution for NORM01\_RV



## 10% Sample: Age Distribution By Gender



The WHERE clause for this example could also have been written as:

where RANUNI(4321) between .65 and .75 ;  
 or  
 where RANUNI(4321) between .50 and .15 ;

Either of the preceding WHERE clauses would yield a random sample consisting of approximately 10% of the observations from *POP\_DATASET*.

## SELECTING A FIXED NUMBER OF RECORDS RANDOMLY FROM THE LARGER DATASET

To randomly select a fixed number of records from a large dataset using only Proc SQL, we will use the OUTOBS= option, the ORDER BY clause and the pseudo-random number generating function RANUNI().

### THE ORDER BY CLAUSE

In Proc SQL, ORDER BY is used to sort the data, i.e. to order the records of the query results. For example, if we wanted to select all the records from dataset *POP\_DATASET* and sort them by value of Age, we could run the following code:

```
proc sql outobs=10;
select * from ps168.pop_dataset ORDER BY age ;
quit;
```

We used OUTOBS=10 to limit the output to the first 10 sorted records which are shown below.

Record Number / ID	Age in Years	Gender	RV N(0,1)	RV U(0,1)
648299	21	Female	-0.12082	0.877108
743428	21	Female	-0.0059	0.280115
469506	22	Female	0.832678	0.093272
683493	23	Female	0.035376	0.709609
786284	23	Female	-0.35503	0.721882
796589	23	Female	-0.13531	0.453805
837711	23	Female	1.637904	0.342613
897142	23	Female	1.388519	0.284197
430417	23	Female	-0.57191	0.039375
420168	23	Female	-0.1791	0.549197

How does this help us to get a random sample of fixed size from the large dataset *POP\_DATASET*?

- Using OUTOBS = *n* will limit the number of records returned to exactly *n* records
- Using the function RANUNI(*seed*) in the ORDER BY clause will sort the records randomly

Because the records get sorted randomly prior to being selected for the sample, the last record in the population dataset has an equal chance of getting selected as the first record has. This is important because otherwise the last record would never have a chance of getting selected if it remained in the last “position” in the dataset.

Using these two features together will allow us to accomplish the task as will be demonstrated next.

### RANDOMLY SELECTING 100,000 RECORDS FROM A LARGE DATASET

We are to obtain a random sample of exactly 100,000 records selected from the *POP\_DATASET* dataset. What we want to do is to select 100,000 records randomly from the population dataset. To do this using Proc SQL, we must first randomize the order in which the population data is sorted, and then take the first 100,000 records from the randomly ordered records.

The following Proc SQL code will create a table called SAMPLE\_100K consisting of exactly 100,000 records randomly selected from dataset POP\_DATASET:

```
/* select exactly 100,000 */

proc sql OUTOBS=100000 ;
create table ps168.sample_100k as
select A.*
  from ps168.pop_dataset as A
 order
  by RANUNI(4537) ;
quit;
```

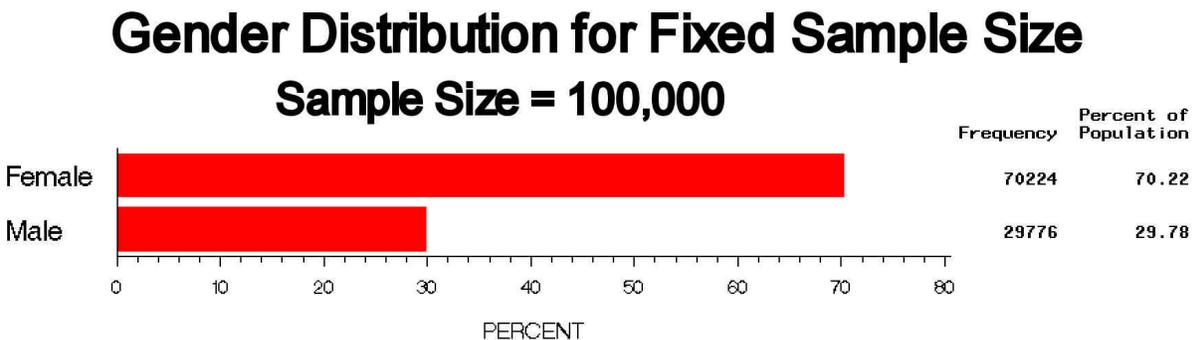
In this example, the seed for RANUNI has been set to 4537.

Because we are ordering by the value of RANUNI which is not a variable in the dataset, we see the following note in the log:

```
NOTE: The query as specified involves ordering by an item that doesn't appear in its
SELECT clause.
```

The records are sorted by the values of RANUNI, hence randomly before they are output to the resulting table. The OUTOBS= option then limits the number of records output resulting in a fixed and exact sample size.

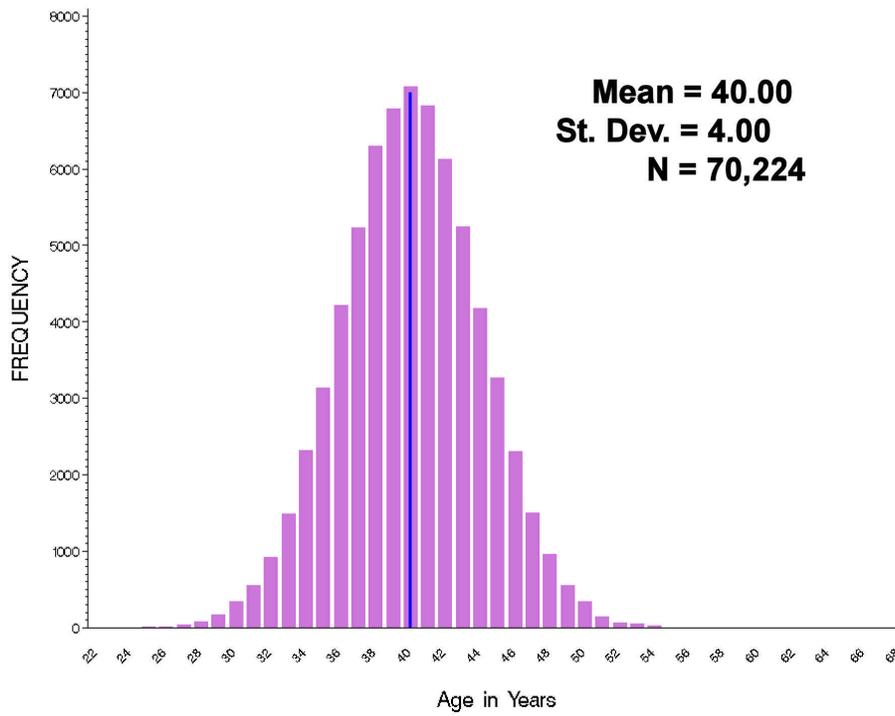
The following figures show the distributions of the variables in the sample dataset SAMPLE\_100K. The sample dataset consists of 29.78% males and 70.22% females which is reflective of the population dataset which has 30% males and 70% females.



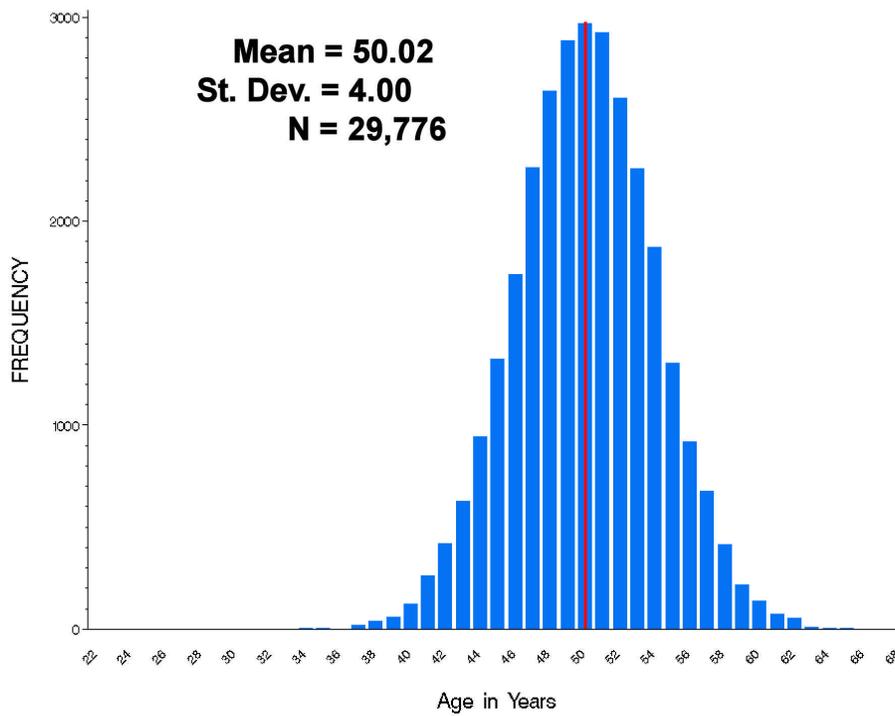
The next two figures below show that just as in the previous sample, the distribution of age among males and females is also approximately the same as the age distributions for males and females in the population dataset. The fixed size sample therefore is representative of the age distributions for males and females of the population dataset.

Fixed Sample Size: Age Distribution By Gender

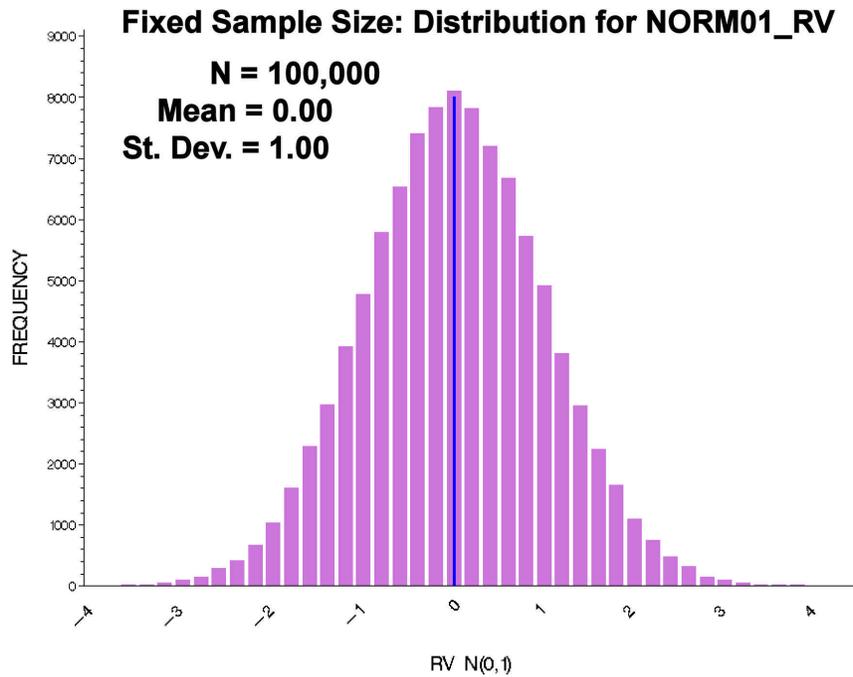
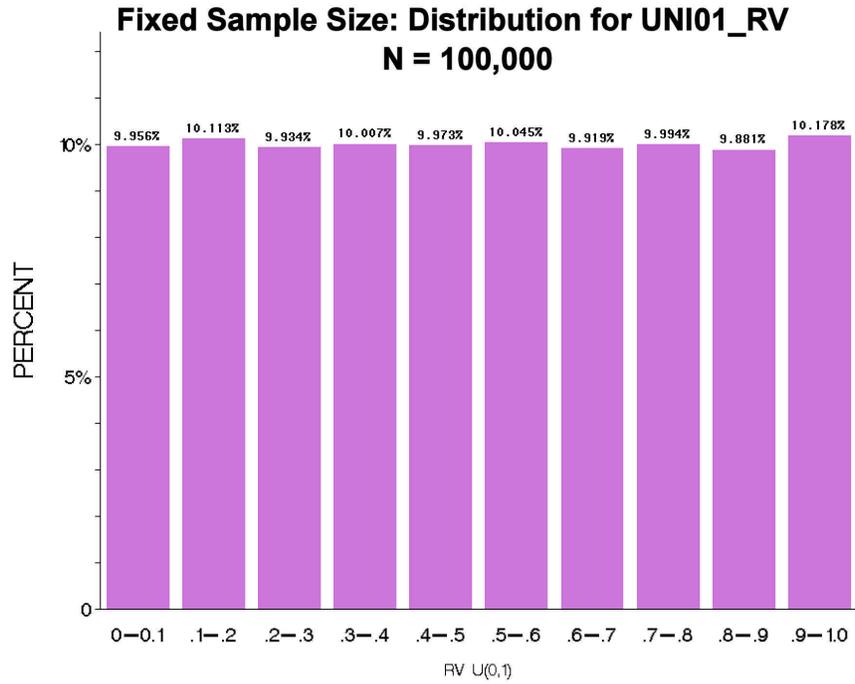
**Females**



**Males**



In this fixed sample size dataset, just as in the population data, NORM01\_RV is normally distributed with a mean of zero (0) and standard deviation of approximately one (1) while UNI01\_RV appears to be uniformly distributed from zero to one (0,1).



## ANOTHER EXAMPLE

Suppose we would like to select a random sample from POP\_DATASET and that the sample is to consist of exactly 30,000 males and 30,000 females. In other words we want a sample that has an equal number of males and females rather than the 3 to 7 ratio that is in the population dataset.

To accomplish this task we will have to apply the technique used for selecting a sample of fixed size. We will first select a random sample of females, then a random sample of males and then combine the two to create our final random sample.

The following Proc SQL code will create a table called SAMPLE\_30K\_FM consisting of records for exactly 30,000 females and 30,000 males randomly selected from dataset *POP\_DATASET*:

```
/* select exactly 30,000 females and 30,000 males */

proc sql OUTOBS=30000 ;

create table work.sample_30k_females as
select A.*
  from ps168.pop_dataset as A
 where sex eq "Female"
 order
   by RANUNI(4321) ;

create table work.sample_30k_males as
select A.*
  from ps168.pop_dataset as A
 where sex eq "Male"
 order
   by RANUNI(4321) ;

RESET OUTOBS=; /* reset Proc SQL options --- don't need OUTOBS=30000 anymore */

create table ps168.sample_30k_fm as
select M.*
  from work.sample_30k_males as M

OUTER UNION CORR

select F.*
  from work.sample_30k_Females as F ;

quit;
```

Please not the use of the RESET statement. The RESET statement resets all the Proc SQL options to their default without having to stop and restart Proc SQL. Because the OUTOBS= option is used to limit the output to 30,000 records when selecting the females and then the males, we must reset that option so that our final sample will contain 60,000 records. The reader can verify that omitting this RESET statement would result in having only 30,000 records in the final table.

Since the gender specific samples are only intermediate results used to create the final sample, they are saved in the WORK rather than a permanent library.

OUTER UNION CORR is used to concatenate the male and female random samples for our final random sample of 30,000 females and 30,000 males.

The following frequency table shows that there are in fact an equal number of male and female records in our random sample.

SAMPLE\_30K\_FM: Equal Number of Females and Males

Gender				
sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Female	30000	50.00	30000	50.00
Male	30000	50.00	60000	100.00

The following table shows the sample mean, standard deviation and size for AGE and NORM01\_RV. The means and standard deviations for these variables in the sample dataset are very close to their population values.

**Descriptive statistics for SAMPLE\_30K\_FM:  
a random sample with equal number of males and females**

		N	Mean	Standard Deviation
Age	Female	30,000	39.988	4.012
	Male	30,000	49.995	4.005
Norm01_rv		60,000	0.005	1.000

The reader may wish to graph the sample data to verify the shapes of the distributions.

The next frequency table shows that the variable UNI01\_RV appears to be uniformly distributed on the range (0,1) in this sample just as it is in the population dataset. Each interval of width 0.1 contains approximately 10% of the observations.

RV U(0,1)				
uni01_rv	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0 - .1	5917	9.86	5917	9.86
.1 - .2	5978	9.96	11895	19.83
.2 - .3	6066	10.11	17961	29.94
.3 - .4	5940	9.90	23901	39.84
.4 - .5	5975	9.96	29876	49.79
.5 - .6	5923	9.87	35799	59.67
.6 - .7	6061	10.10	41860	69.77
.7 - .8	5959	9.93	47819	79.70
.8 - .9	6084	10.14	53903	89.84
.9 - 1	6097	10.16	60000	100.00

## CONCLUSION

This paper/poster has demonstrated how very easy it is to obtain simple random samples from a large dataset using PROC SQL. The sample sizes can be fixed sizes or a percentage of the number of records in the large dataset from which the sample is being selected. This is just one way to get a simple random sample using Proc SQL without requiring any data step programming or using the SURVEYSELECT procedure in SAS/STAT SAS/STAT.

**REFERENCES**

SAS<sup>®</sup> 9.1 SQL Procedure User's Guide. Cary, NC: SAS Institute Inc., 2004.

**RECOMMENDED READING**

Lafler, Kirk Paul. 2004. *PROC SQL: Beyond the Basics Using SAS<sup>®</sup>*. Cary, NC: SAS Institute Inc.

**CONTACT INFORMATION**

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

Richard Severino  
Convergence CT  
1132 Bishop Street Suite 2302  
Honolulu HI 96813

Email: [severino@hawaii.edu](mailto:severino@hawaii.edu)  
[rseverino@convergencect.com](mailto:rseverino@convergencect.com)

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.