

Generating Random Numbers from a Discrete User Defined Distribution

Amy Boulris, Bank of America, Charlotte, NC

ABSTRACT

With the continual increase in computational speed, the use of simulation to analyze complex statistical problems has become more and more feasible. A once purely theoretical situation can now be analyzed by using computer generated data which mimics the situation. This generated data can come from either a known distribution (for example, Standard Normal) or an empirical distribution (one based off of historical data). SAS[®] has built in several functions which will generate random numbers from a known distribution; however, in a case where the distribution is unknown, there is no direct implementation in SAS[®]. While we do not have the luxury of a built in procedure in SAS[®], the theory to generate a random number from an empirical, or user defined, distribution is statistically quite simple and can be easily applied in the form of a macro. This paper will address a practical method for performing this type of simulation in SAS[®].

INTRODUCTION

Simulation is a valuable method for analyzing currently unknown data under varying scenarios. While SAS[®] has several built in functions to simulate random numbers from a known distribution (for example, the `ranuni` function to generate numbers from a uniform[0,1] distribution), there is not a direct method for generating a random number from an empirical distribution. There is, however, simple macro code which can be applied in order to generate empirical data and achieve the simulation results desired. This paper addresses such macro code which can be applied to generating discrete random numbers based off historical data.

A real estate valuation application will be used as an example of this type of simulation. Many lenders use Automated Valuation Models (AVM's) to estimate the value of a customer's property during a real estate transaction. Because AVM's have varying performance in different geographies, it is beneficial for the lender to rank each AVM from "best" to "worst" in each area, based on analytics. When a request is made for an automated valuation, the "best" AVM is asked to return a value. However, if that model cannot value the requested property, a request will be sent to the next AVM. This will continue until a value is returned or all adequate models in that geography are exhausted. A lender's ordered list of AVM's is called a cascade. In the cascade, the "best" model is ranked #1, the second best #2, etc. The rank of the vendor which returned the valuation will be termed "rank," for the remainder of this paper.

Now suppose the lender is interested in simulating which AVM in the cascade will be valuing properties over the next year at a loan level. The lender can then use this information to evaluate the overall expected accuracy of its valuations, as well as the expected business which will be given to each AVM. Because we have historical data on the rank of the model that returns a value, we can create an empirical distribution to randomly assign a rank to future generated loans. While this application is specific to the real estate business, the same concept could be applied to several other industries, for instance, we could generate the number of cars a person is expected to own over his/her life, or the number of accounts a person has open at the bank - any application in which we are generating a discrete random number.

Data Requirements

In order to obtain meaningful output, you have to start with meaningful historical data. We will be generating data based off of the historical distribution, so if the historical distribution is skewed or biased in any form, the output of the simulation will be innately incorrect. Also, it is important to ensure that the historical dataset has enough elements to mimic the population data. As is true of most statistical concepts, a distribution based off too small of a sample size will not be representative of the true underlying distribution.

Methodology

In the valuations example, there is a dataset containing records (in our case loans) and the rank of the vendor which gave a valuation. To determine the empirical distribution, a frequency procedure (`proc freq`) is run. In this procedure, the `tables` statement will be the discrete variable to be simulated ("rank" in the real estate valuations example). The cumulative percentages need to be saved by using the `outcum` option in the `tables` statement.

```
Proc Freq Data=loans noprint;
Tables Rank / Out=Table outcum;
run;
```

The dataset resulting dataset, TABLE, looks as follows:

Obs	rank	COUNT	PERCENT	CUM_FREQ	CUM_PCT
1	1	100	2.1988	100	2.199
2	2	130	2.8584	230	5.057
3	3	558	12.2691	788	17.326
4	4	512	11.2577	1300	28.584
5	5	351	7.7177	1651	36.302
6	6	612	13.4565	2263	49.758
7	7	654	14.3799	2917	64.138
8	8	891	19.5910	3808	83.729
9	12	687	15.1055	4495	98.835
10	15	53	1.1653	4548	100.000

In order to create an empirical distribution, every value between the maximum and minimum must have an associated probability. For the numbers that were not present in the dataset, the associated probability will be zero. To program this logic into SAS®, the maximum and minimum value need to be located using a proc univariate statement on dataset that was output from the frequency procedure. The maximum and minimum can then be stored into macro variables using call symput.

```
Proc Univariate Data=Table noprint;
Var rank;
Output out=TMP min=MIN max=MAX
run;

data Tmp;
set tmp;
call symput('MAX',max);
call symput('MIN', min);
run;
```

There are now 2 macro variables “max” and “min” which contain the minimum and maximum rank of an AVM which could value the property.

If there are discrete numbers between the minimum and maximum values of the dataset which are not present in the dataset, the values need to be accounted for by assigning a zero probability. For instance, in the real estate example, the ranks of 9, 10, 11, 13, and 14 are missing. These numbers can be added by creating a list of numbers that are

completely populated between the minimum and maximum, merging the datasets together, and then assigning a 0% probability that the missing numbers are chosen. This is done using the code below:

```
data nums;
do rank=&min to &max;
    output;
end;
run;

data cdf;
merge TABLE nums;
by rank;
percent=CUM_PCT/100;
run;
```

The dataset *cdf* contains the same information as *TABLE*; however, the missing values are included in the rank fields, and containing missing data points for COUNT, PERCENT, CUM_FREQ CUM_PCT.

Obs	rank	COUNT	PERCENT	CUM_FREQ	CUM_PCT
1	1	100	.02199	100	2.199
2	2	130	.05057	230	5.057
3	3	558	.17326	788	17.326
4	4	512	.28584	1300	28.584
5	5	351	.36302	1651	36.302
6	6	612	.49758	2263	49.758
7	7	654	.64138	2917	64.138
8	8	891	.83729	3808	83.729
9	9	.	.	.	.
10	10	.	.	.	.
11	11	.	.	.	.
12	12	687	.98835	4495	98.835
13	13	.	.	.	.
14	14	.	.	.	.
15	15	53	1	4548	100.000

The concept of the next set of steps is simple; although the coding is not straightforward. Assume there is an additional data set of “future” records which need to be assigned a randomly generated variable. The previous steps provided an Empirical Cumulative Distribution Function (CDF) of this variable. Simulation theory states that any CDF is distributed with a uniform[0,1] distribution. Therefore, the SAS® uniform random number generator can be used to generate a number from this distribution. This will be done in the following steps:

1) Generate a Uniform random number (u) for each “future” record (In the real estate case, the record is a loan, and the variable is the rank of the AVM which provided a valuation)

- 2) Loop through each discrete number and compare u to the cumulative distribution value.
- 3) If u lies between two values, say X and Y, then assign the value of the variable to be the larger of the two values.

1) Generate a Uniform random number (u) for each new record

```
data future_loans;
set future_loans;
u=ranuni(&seed);
run;
```

2) Loop through each discrete number and compare u to the cumulative distribution

Because SAS® lends itself to looping easiest at the record level, it's best to merge the CDF with the future records, therefore the cdf must first be transposed.

```
proc transpose data=cdf out=cdf_x;
id rank;
var percent;
run;
```

Obs	_NAME_	_LABEL_	_1	_2	_3	_4	_5			
1	PERCENT	Percent of Total Frequency	0.0219	0.0506	0.1733	0.2858	0.3630			
Obs	_6	_7	_8	_9	_10	_11	_12	_13	_14	_15
1	0.4976	0.6413	0.8373	.	.	.	.9884	.	.	1

Then, the missing values of the discrete distribution must be filled in. Because this is a cumulative distribution (which is the sum of the probability distribution or pdf), the missing discrete values should to be the same as the previous value plus the probability of the missing value (which is 0). Therefore, the missing values have the same cumulative probability as the cumulative probability of the value immediately preceding it. For instance, the CDF at 8 is .8373 and 9 is a missing value, so the CDF at 9 should also be .8373. A value of 9 will never be assigned by the random number generator because a uniform[0,1] variable will never be between .8373 and .8373. In other words, there is a 0 probability of generating a 9.

```
%do i=(&max-1) %to &min %by -1;
%let j=%sysevalf(&i-1);
data cdf_x;
set cdf_x;
if _&i=. then _&i=_&j;
run;
%end;
```

Obs	_NAME_	_LABEL_			_1	_2	_3	_4	_5	
1	PERCENT	Percent of Total Frequency			0.0219	0.0506	0.1733	0.2858	0.3630	
Obs	_6	_7	_8	_9	_10	_11	_12	_13	_14	_15
1	0.4976	0.6413	0.8373	0.8373	0.8373	0.8373	0.9884	0.9884	0.9884	1

In order to merge the cdf distribution to the dataset of future loans, there needs to be a consistent “by” variable in both datasets. Since there is already a variable “_NAME_” in the cdf dataset, its easiest to add that same variable to the future loans dataset.

```
data future_loans;
set future_loans;
_NAME_='PERCENT';
run;
```

Then merge on the new variable:

```
data simulated_data;
merge future_loans cdf_x;
by _NAME_;
run;
```

3) If u lies between two values, say X and Y, then assign the value of the variable to be the larger of the two values.

Once this dataset, *cdf_x*, is merged with the generated record dataset (*simulated data*), and each record in *simulated_data* has been given a value of u, using the ranuni function, a value of the variable can be assigned to each record . These values will come from the historical distribution of values.

```
%do i=&min %to &max;
%let j=%sysevalf(&i-1);
%if %sysevalf(&j=-1) %then %let j=0;
data simulated_data;
set simulated_data;
if &i=&min and u<&i then rank=&i;
else if u<&i and u>&j then rank=&i;
run;
%end;
```

Obs	loanno	u	rank
1	1	0.55806	7
2	2	0.47756	6

3	3	0.62659	7
4	4	0.02646	2
5	5	0.62174	7
6	6	0.67674	8
7	7	0.92044	12
8	8	0.77973	8
9	9	0.14670	3
10	10	0.70119	8

Based on the computations above, each record in the simulated future data has been assigned a variable rank, which is the randomly generated rank of the AVM in the cascade based on historical distribution of model ranks. Calculations can now be performed on the simulated future dataset and the loans analyzed just as current loans would be.

CONCLUSION

Using the code above, it should be just as easy to randomly generate a loan from a “known” distribution as a user-defined distribution. This increases the opportunities for simulation to be used to analyze scenarios where data does not currently exist.

CONTACT INFORMATION

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

Amy Boulris
Bank of America
NC1-014-11-16
201 South College Street
Charlotte, NC 28255
Work Phone: 704-386-3518
Fax: 704-386-1213
E-mail: Amy.Boulris@bankofamerica.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.