

Paper BB-202

Creating Test Data Using SAS Hash Tables

Gwen Babcock, New York State Department of Health, Albany, NY

ABSTRACT

SAS® programmers sometimes need to simulate real data. Real data often cannot be used because it contains confidential information or is unavailable. This paper describes generating test data for the New York State Congenital Malformations Registry. The data was required to be in a flat text file format and to contain a medical record number, last name, first name, street address, city, state, ZIP code, sex, two dates, and from one to 12 diagnosis codes with descriptions. The data was created in a data step using SAS random number generating functions to generate values directly or to select items from hash tables. The selected items were written to a text file with PUT statements. The code was written in SAS 9.4 under Windows 7.

INTRODUCTION

There are many methods that can be used in SAS to create data. Entire books have been written about simulating numeric data for analysis (Wicklin, 2013). This paper will touch on several of these methods, but focus on the use of hash tables to simulate non-numeric data. A hash table can be thought of as a data set that is stored in computer memory. However, the hash table differs from a data set in that instead of being accessed sequentially, observation by observation, you directly select the desired observation based on its (typically) unique identifying information. This identifying information is called the key, and the remaining variables of interest are called data.

Hash tables will be used to simulate hospital reports to the Congenital Malformations Registry (CMR). The CMR is a database of reports from hospitals of children born or residing in New York State diagnosed before the age of two with certain birth defects. The registry collects confidential identifying variables, such as name and medical record number. To test hospital reporting, we want to generate plausible data which does not contain any identifying information of any actual persons. This paper shows how to generate records with medical record numbers, children's names, diagnosis codes, and dates.

A RANDOM NUMBER

A medical record number is assigned by a hospital, and is typically a series of digits. So to create a random record number, we can use the RANUNI function. This function outputs a number between zero and one. Since I want an integer between 1 and 100000, I multiply it by 100000 and round it to the nearest integer. Since all of the output is to be text, I converted it from a number to text using the PUT function.

```
x=round(ranuni(-1)*100000,1);
mrec=put(x,20.);
```

WHAT'S IN A NAME

Is a name just a random collection of letters? If so, we can generate it (SAS Institute, 2016) (**Table 1**):

Code	Result
do p=1 to max(round(ranuni(-1)*20,1),3);	LNAME
substr(lname,p)=	P
byte(floor(65+26*ranuni(0)));	JVMO 5
end;	RHZ 4
	XFT 4
	MXYZPTLK 9
	ZQHKZWVQBOV 12
	IRRRLSJIANHWRGZVKX 19
	IDNHTZUPGTZ 12
	RFELDEMVSNDWONODN 18
	UBGYLWKXVUUD 13
	LWMBNYSGEPMMA 13

Table 1. Code to Produce Random Strings and the Results

The first line determines the length of the name, which should be between 3 and 20 characters. The second line creates the name using the SUBSTR and BYTE functions. The SUBSTR function replaces the character values in the

'lname' variable with the character returned by the BYTE function. (In Windows, The BYTE function requires an integer between 0 and 255 and returns a character). Since the number given to the BYTE function is random the character replaced is random, and a random string of characters is generated.

If you want to make your names more plausible and pronounceable, you can pick from a list, such as a list of frequent surnames provided by the US Census Bureau (2000) (**Table 2**). You don't even have to unzip the file to use it in SAS 9.4:

NAME	RANK	COUNT	CUM_PROP100K	PCTWHITE	PCTBLACK	PCTHISPANIC
SMITH	1	2376206	880.85	73.35	22.22	1.56
JOHNSON	2	1857160	1569.3	61.55	33.8	1.5
WILLIAMS	3	1534042	2137.96	48.52	46.72	1.6
BROWN	4	1380145	2649.58	60.71	34.54	1.64
JONES	5	1362755	3154.75	57.69	37.73	1.44
MILLER	6	1127803	3572.82	85.81	10.41	1.43
DAVIS	7	1072335	3970.33	64.73	30.77	1.58
GARCIA	8	858289	4288.5	6.17	0.49	90.81
RODRIGUEZ	9	804240	4586.62	5.52	0.54	92.7

Table 2. Sample Data

Using FILENAME ZIP allows text data to be read in directly from a ZIP archive. You must provide the path to the ZIP archive and the name of the file in the archive.

```
filename mysur ZIP "C:\folder\names\surnames.zip" mem-
ber="app_c.csv";
/*from http://www.census.gov/topics/population/geneal-
ogy/data/2000_surnames.html*/
data surnames;
format surname $15. pctwhite pctblack pctapi pctasian
pct2prace pcthispanic $6.;
infile mysur dsd firstobs=2;
input surname rank count prop100k cum_prop pctwhite pctblack
pctapi pctasian pct2prace pcthispanic;
recno=_n_;
run;
```

When reading in the data, we add a sequential number ('RECNO') which we will use later as the key to a hash table.

Next, we put in the surname data in a hash table, then pick from it a random name. The 'DECLARE HASH' statement tells SAS to create a hash table called 'lnames' containing the data from the data set 'surnames'. The next two statements tell SAS that we will find observations in the hash table 'lnames' using RECNO (the key), and the information we wish to retrieve is SURNAME (the data). These types of statements, which have a <hashtable name>.<keyword> syntax, are also called methods. The DEFINEDONE() method statement lets SAS know that we are finished giving information about the structure of this hash table.

```
data names;
format surname $15.;
```

```

if _n_=1 then do;
  declare hash lnames(data set:"surnames");
  lnames.definekey("recno");
  lnames.definedata("surname");
  lnames.definedone();
  call missing(surname);
end;

do n=1 to 10;
  /*pick a random last name*/
  recno=round(ranuni(0)*151671,1);
  rc=lnames.find();
  output;
end;

run;

```

In the second half of this data step, we generated a random recno, and use the find() statement to retrieve the corresponding surname (Table 3).

SURNAME	N	RECNO	RC
PYCH	1	118041	0
DIPISA	2	100281	0
DUMOUCHEL	3	69035	0
KENDZORA	4	130846	0
SARAGOZA	5	49536	0
HAMBROOK	6	107220	0
DOMINGUE	7	8406	0
NESTLE	8	39218	0
WAMACK	9	89145	0
CARCIERI	10	101210	0

Table 3. Results of Randomly Selecting Names using Hash Tables

JUST WHAT AILS YOU

Each child can have up to 12 diagnoses of birth defects. Each one gets a special code, called an ICD code. Fortunately, it is more common to have just one birth defect than multiple defects. To reflect this in the test data, I will generate random numbers in the Poisson distribution, with a floor of one and a ceiling of 12.

```

icdno=ranpoi(0,1.5);/*one to 12 codes, random number*/
if icdno<1 then icdno=1; if icdno>12 then icdno=12;

```

The actual diagnosis codes with their descriptions are available from the Centers for Medicare and Medicaid Services (CMS) (2015). Most codes for birth defects begin with the letter 'Q'. There are 944 valid codes in the CMS file that begin with 'Q'. I picked codes and descriptions from this list using hash tables, similar to what was done for names.

```

declare hash icd(data set:"icd10_order");
icd.definekey("recno");
icd.definedata("ICD10","short_desc10");
icd.definedone();

...

do p=1 to icdno;
  recno=round(ranuni(-1)*944,1); icd.find();

```

```

    put @(474+55*(p-1)) short_desc10 @;
    put @(1293+7*(p-1)) ICD10 @;
end;

```

GETTING A DATE

Generating a random date is similar to generating a random number. SAS dates are values that represent the number of days between January 1, 1960 and a specified date. So to get a random date, we take a valid SAS date and add or subtract a random number of days from it (Cary, 1997). In the example shown in **Table 4**, I wanted to generate a random date within about a month prior to today:

Code	Results when run on July 28, 2016
<pre> data dates (drop=n); format myday worddate32.; do n=1 to 5; myday=date()-round(ranuni(8)*30,1); output; end; run; </pre>	myday July 14, 2016 July 5, 2016 July 22, 2016 July 26, 2016 July 17, 2016

Table 4. Code to Generate Random Dates and the Results

PUTTING IT ALL TOGETHER

Once the random data was generated, I used a FILE statement along with PUT statements to create a text file containing the data.

```
file "G:\CMR\cmrfake&pfi._&now..txt";
```

```
...
```

```

put @1 pfi @;
put @5 mrec @;
put @22 lname @;
put @42 fname @;
put @73 street @;
put @109 city @;
put @124 state @;
put @126 zip @;
put @135 dob @;
put @148 sex @;
put @158 dodis @;

```

```
...
```

CONCLUSION

Hash tables combined with random number generating functions are an alternative way to generate random data, especially for data that is selected from a finite list. Hash tables are extremely fast, but limited by available computer memory. They are especially useful if several dependent or related fields must be simulated. Combined with SAS random number functions and other techniques, they allow the rapid simulation of plausible data.

REFERENCES

- Cary, Andrew JL. (1997) *Generating Data With the SAS Data set*. Proceedings of Twenty-Second Annual SAS Users Group International Conference. <http://www2.sas.com/proceedings/sugi22/CODERS/PAPER74.PDF> Accessed May 24, 2016.
- Centers for Medicare and Medicaid Services.(2015). 2016 ICD-10-CM and GEMS. <https://www.cms.gov/Medicaid/Coding/ICD10/2016-ICD-10-CM-and-GEMS.html>
- SAS Institute, Inc. (2016) *Sample 51386: Creating character variables that contain random alphabetic values* <http://support.sas.com/kb/51/386.html> Accessed May 24, 2016.
- United States Census Bureau (2000). Frequently Occurring Surnames from the Census 2000, File B: Surnames Occuring 100 or more times. http://www.census.gov/topics/population/genealogy/data/2000_surnames.html Accessed May 24,
- Wicklin, Rick (2013) *Simulating Data with SAS*. Cary, NC: SAS Institute, Inc.

ACKNOWLEDGEMENTS

Thanks to Deborah Fox, Neil Muscatiello, and the New York State Department of Health, Center for Environmental Health for supporting this work.

RECOMMENDED READING

- Varney, Brian (2010) *The Super Genius Guide to Generating Dummy Data*. Proceedings of PharmaSUG 2010. <http://www.lexjansen.com/pharmasug/2010/AD/AD24.pdf>
- Wicklin, Rick (2015). Ten Tips for Simulating Data with SAS. Proceedings of SAS Global Forum 2015. <http://support.sas.com/resources/papers/proceedings15/SAS1387-2015.pdf>. Accessed Aug 24, 2016.

CONTACT INFORMATION

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

Gwen D LaSelva
New York State Department of Health
Bureau of Environmental and Occupational Epidemiology
Empire State Plaza-Corning Tower, Room 1203
Albany, NY 12237
Work Phone: 518-402-7785
Fax: 518-402-7959
Email: Gwen.Laselva@health.ny.gov

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.