SESUG Paper 90-2021

Using PROC SURVEYSELECT to Create Data Files with All Pairwise Combinations of Data

Jason Brinkley, Abt Associates

ABSTRACT

PROC SURVEYSELECT provides an easy mechanism to create datafiles that are all pairwise combinations of two observations in a dataset. While the procedure is traditionally used for creating subsamples of data, there are options that allow one to use the entire data. This paper illustrates how to take an address-based dataset and create a new dataset that has all pairwise combinations of each set of addresses so that additional analyses can be done.

INTRODUCTION

It can sometimes be the case that analysts need to do calculations or matching inside of an entire dataset to elucidate some new insights or novel information. In such cases it can be a challenge to explore all pairwise combinations of the data to perform such calculations. Presented here is a simple use-case of the SURVEYSELECT procedure in SAS® to create a novel dataset that is all pairwise combinations of the data. The technique is likely not the most efficient but can be extremely effective with limited programming. This paper will be broken up into three parts: a reminder of the SURVEYSELECT procedure, discussion of the use case, and an example where we look at publicly available hospital data to determine which hospitals are closest to one another within the state of North Carolina.

SURVEYSELECT PROCEDURE

The SURVEYSELECT Procedure is in the SAS/STAT platform and is designed to help users select probability-based random samples. The online documentation overviews states that the '*procedure can select a simple random sample or can sample according to a complex multistage design that includes stratification, clustering, and unequal probabilities of selection*'. This can be very useful in a lot of scenarios related to sampling design and experimentation where one might have a large census of all individuals from which a sample has to be selected for outreach. Therefore, most users of this procedure only want a subset of the overall data. The true power in the approach though is the ability to select multiple subsamples within the same procedure. Therefore, the SURVEYSELECT procedure is a favorite among individuals who need to do replicate sampling for statistical analysis. Replicate sample analyses include cross-over validation techniques such as the popular bootstrap technique for statistical analysis. The reader can see Efron and Tibshirani (1994) for a thorough overview of such statistical procedures.

USING SURVEYSELECT TO OBTAIN ALL PAIRWISE COMBINATIONS

The focus of this paper is to extend the use case of the SURVEYSELECT procedure with a simple observation, instead of using the procedure to obtain one or more random subsets of the original data, we can instead use it to replicate the full sample many times. Suppose you have a dataset with 1000 observations, we could do replicate subsampling and draw a random subset of say 100 observations from the overall data one time. Or the procedure can be used to draw many different subsamples of 100 observations from the overall 1000. Those subsamples could be done either with or without *replacement* where sampling with replacement would mean potentially having the same original observation show up in subsamples many times. If one creates without replacement samples of 100 from a larger pool of 1000 then we can only create 10 such datasets until all the data have been exhausted. Sampling with replacement means that one can create as many subsamples as one desires, with the notion that observations are eligible to be sampled many times in repeated samples. But one may also use the SURVEYSELECT procedure to sample exactly 1000 observations without replacement and one obtains a new dataset with all original 1000 observations. While that may not sound useful, the same procedure will allow me to draw 1000 copies of that same dataset quickly. Therefore, the procedure will quickly allow me to gain 1000 datasets that are 1000 observations each, in essence making 1000 quick copies of the original data and stacking them up into a new dataset that is 1000 x 1000 = 1,000,000 observations. By merging this dataset with the original datafile in a clever way, one can easily create a new file that is all pairwise matches of the original data.

EXAMPLE

The Cecil G Sheps Center for Health Services Research maintains a yearly list of all hospitals operating in the United States that is downloadable and available to the public. Suppose we want to extract out all acute care hospitals in the state of North Carolina and then calculate the distance between each hospital and then determine the average distance between NC hospitals and/or we want to limit the data to each hospital and its closest neighbor.

After downloading and importing the data into SAS (a simple Proc Import will do) we can work with the list of 4614 acute care hospitals. For simplicity we will pair down to just hospitals inside of the state of North Carolina which leaves 104 acute care hospitals in the dataset. Given that we have to geocode the addresses, we will use the Proc Geocode procedure to find the longitude and latitude of these hospitals. Hadden (2021) has a terrific overview on how to use Proc Geocode with public use data to get street level long/lat metrics. Note that for this datafile, the zip codes must be altered from character to numeric to process via Proc Geocode. Regardless, we find that of the 104 addresses in the data that 82 (79%) hospitals match at the street level while the remaining 22 match at the zip code level (SAS will use the center of the zip code region as the long/lat when a specific address can not be found). We will use this initial matching for the rest of this analysis.

Start by indexing your observations. This can be done quickly by including a do loop with your set command in the data statement:

```
Data INPUT;
do i = 1 to 104;
set INPUT;
output;
end;
run;
```

The key to creating the all pairwise combinations is a simple implementation of the SURVEYSELECT procedure as demonstrated below:

```
proc surveyselect data=INPUT
n=104 reps=104 out=OUTPUT;
run;
```

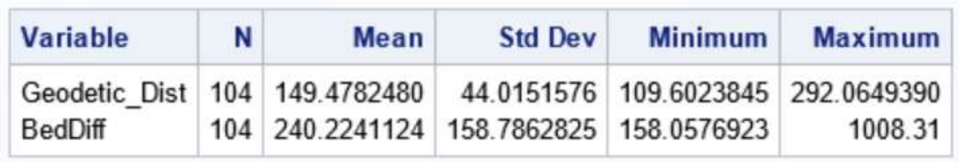
The names of the input and output files will change depending on users choice. However, the basic procedure creates a simple random sample of 104 observations (the full data) exactly 104 times. Therefore, we do not have a random sample but have 104 copies of the current data indexed by the new variable called *Replicate*. Working with the OUTPUT file one needs to rename and keep important indicators for comparison. For the hospital data that may involve renaming the NAME, ADDRESS, ZIP, and any metrics in the output file along with your X (long) and Y (lat) variables.

Now by rejoining the INPUT and modified OUTPUT files using your *i* and *Replicate* variables you have a combined file that has all pairwise combinations of the original data in a flat file for subsequent analysis. For this example, we will retain the total number of bed variable *TOTAL_BEDS_POS2* as well as create a new variable *Distance* which will be the geodetic distance (as the crow flies) and can be calculated in the data step with the 'geodist' command.

This leaves us with a dataset that 10,816 observations totally 104 by 104 pairwise combinations and a metric of distance between each pair of hospitals (obviously the distance between a hospital and itself is zero and you can remove if you wish) and we also can calculate the different in beds. From here one can either summarize the full data or summarize by hospital (as indexed by the letter i). It is likely preferred to roll the comparisons up to a single comparator for each hospital, which can be done using a Proc Means statement with the 'class i;' statement to index by each hospital. From there one can work with the just the rolled-up means (dropping the other MEANS procedure measures) and then get a distribution of average differences.

Finally, a Proc Means on the rolled-up data should produce this output table:

Table 1 – Proc Means Output for Hospital Example



Variable	N	Mean	Std Dev	Minimum	Maximum
Geodetic_Dist	104	149.4782480	44.0151576	109.6023845	292.0649390
BedDiff	104	240.2241124	158.7862825	158.0576923	1008.31

And we see that the average distance between hospitals is approximately 149.5 miles and that the *per hospital average* distance to other NC hospitals fluctuates from 109.6 to 292 miles. Likewise, there are large differences in the size of hospitals in NC with the average difference in hospital beds is around 240. A likely indicator that hospitals are either very small or very large given the wide fluctuation in the minimum and maximum (which again represent the min and max average distances between a single hospital and all the other hospitals in the state of North Carolina).

CONCLUSIONS

The SURVEYSELECT procedure is excellent at drawing all kinds of subsamples of data. But it is also good at creating replicate samples of the same data for later matching and analysis. This technique is fast and can let the analyst explore all kinds of within data differences. When paired with geospatial techniques it can be used to identify a whole slew of extant information which may be useful for planning and policy purposes.

REFERENCES

SAS Institute Inc., SAS 9.4 Help and Documentation, Cary, NC: SAS Institute Inc., 2002-2004.

Efron, Bradley, and Robert J. Tibshirani. An introduction to the bootstrap. CRC press, 1994.

Hadden, Louise. Visually Exploring Proximity Analyses Using SAS® PROC GEOCODE and SGMAP and Public Use Data Sets. Presented at PharmaSUG 2021. Available at www.pharmasug.org.

CONTACT INFORMATION

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

Jason Brinkley, PhD
Principal Data Scientist, Abt Associates
Jason_Brinkley@abtassoc.com

APPENDIX

Example Implementation Code:

```
*specify location of lookup street level geocode dataset;
libname lookup 'LOCATION';

*Import hospital data as work.hospitals;

Data Hospitals2;
set hospitals;
if State ne 'NC' then delete;
zip2=zip+0;
drop zip;
run;

Data Hospitals2;
set hospitals2;
if State ne 'NC' then delete;
zip=zip2;
drop zip2;
run;

*Index hospitals;

Data Hospitals2;
do i = 1 to 104;
set Hospitals2;
output;
end;
run;

*proc geocode with an output statement, using street as the look up method
and specifying USM as the lookup dataset;
proc geocode out=output method=street
lookupstreet=lookup.Usm;
run;

Data Output;
Set Output;
keep x y i replicate id--zip;
run;

*Create replicate datafile;

proc surveyselect data=output
n=104 reps=104 out=output2;
run;

*Screen replicates for variables for comparison;

Data Output3;
set Output2;
i=replicate+1;
if i>104 then i=i-104;
X2=X;
Y2=Y;
```

```
NAME2=NAME;
ADDRESS2=ADDRESS;
ZIP2=ZIP;
FIPS2=FIPS;
TOTAL_BEDS__POS2=TOTAL_BEDS__POS;
keep X2 Y2 NAME2 ADDRESS2 ZIP2 FIPS2 TOTAL_BEDS__POS2 replicate i;
run;

*match original and replicate file;

proc sort;
by i replicate;
run;

data New;
merge Output Output3;
by i;
*create indicators for analysis;

Geodetic_Dist = geodist(x, y, x2,y2, 'M');
BedDiff = abs(Total_beds__Pos - TOTAL_Beds__POS2);
run;

*roll up hospitals;

Proc means noprint;
class i;
var Geodetic_Dist BedDiff;
output out=summary;
run;

Data Summary;
set Summary;
if _STAT_ ne 'MEAN' then delete;
if _TYPE_ = 0 then delete;
run;

*Final analysis across all hospitals;

proc means;
var Geodetic_Dist BedDiff;
run;
```