

Effectively Using the SAS® System for Survey Research

Kim L. Kolbe, Systems Seminar Consultants

Introduction

Individuals involved with survey research have specific demands upon a software package which assists them in exploratory data analysis. When used correctly, SAS is a very powerful data analysis tool. Using SAS both efficiently and effectively will assure both timely results to research questions as well as reflect positively upon our ability to quickly and accurately analyze survey data. This paper will explore some areas within the SAS system that can greatly improve the quality and time it takes to analyze survey research data.

Reading Data In

The first obstacle most survey researchers must overcome is reading the data in efficiently and effectively. Typically researchers receive the data as a raw or flat file which can come from any number of sources. Some of the more common sources are:

- * **Service vendor** a market research firm who conducts a survey and provides a tape of the keyed results for in-house data analyses
- * **Establish external data source** government data: census, housing starts, etc.
- * **Internal data** system files, in-house key punch of data, etc.

A description of the file is usually provided and it is our job to code an input statement and build a SAS data set. Generally, it is a good idea to read the data once, build a SAS data set and use the SAS data set for analyses.

Example:

```
DATA SASDATA.VOTERDAT;  
  INFILE VOTER;  
  INPUT @1 NAME $10. @11 VOTED $1.  
        @12 REGISTER $1. @13 CONTRIB 5.  
        @18 PARTYMEN $1.;  
RUN;
```

How many times have you coded an input statement only to find out later it is incorrect because you excluded some variable?

An input statement that can easily add a forgotten variable:

```
DATA SASDATA.VOTERDAT;  
  INFILE VOTER;  
  INPUT @1 NAME $10. +0 VOTED $1.  
        +0 REGISTER $1. +0 DISTRICT $2.  
        +0 CONTRIB 5. +0 PARTYMEN $1.;  
RUN;
```

What do you do if you don't have a file description?

You can use the LIST statement:

```
DATA SASDATA.VOTERDAT;  
  INFILE VOTER;  
  INPUT;  
  LIST;  
RUN;
```

Reading Multiple Card Data

Commonly survey research data spans multiple lines or cards. There are a number of ways in which to read this type of data:

```
DATA SASDATA.TREEDATA;  
  INFILE TREES;  
  INPUT @1 NUTRIENT 5. @6 AVG_SUN 5.;  
  INPUT @1 AVGMOIST 5. @6 SPECIES $10.;  
  INPUT @1 HEIGHT 6.2;  
RUN;
```

Another possible solution:

```
DATA SASDATA.TREEDATA;  
  INFILE TREES;  
  INPUT @1 NUTRIENT 5. @6 AVG_SUN 5. /  
        @1 AVGMOIST 5. @6 SPECIES $10. /  
        @1 HEIGHT 6.2;  
RUN;
```

Yet, another possible solution:

```
DATA SASDATA.TREEDATA;  
  INFILE TREES;  
  INPUT #1 @1 NUTRIENT 5. @6 AVG_SUN 5.  
        #2 @1 AVGMOIST 5. @6 SPECIES $10.  
        #3 @1 HEIGHT 6.2;  
RUN;
```

Which is the best solution?

While they all provide the same results, the last solution takes slightly more memory. It is, however, the most flexible if you want jump around the input buffer and selectively read the data:

```
DATA SASDATA.TREEDATA;
  INFILE TREES;
  INPUT #2 @6 SPECIES $10. @;
  IF SPECIES='OAK';
  INPUT #1 @1 NUTRIENT 5. @6 AVG_SUN 5.
         #2 @1 AVGMOIST 5.
         #3 @1 HEIGHT 6.2;
RUN;
```

Efficient Input Statements

- * Read the data selectively
- * Read numeric as character if you are never going to use the variable in mathematical operations
- * Use informats to re-code whenever possible

Handling and Analyzing the Data

There are a number of ways in which data can be re-coded:

Using the SELECT statement:

```
DATA PRODUCTS;
  LENGTH MANUF $10.;
  SET SASDATA.PRODFILE(KEEP=MAKE);
  SELECT (MAKE);
    WHEN ('1') MANUF='FORD';
    WHEN ('2') MANUF='CHRYSLER';
    OTHERWISE MANUF='FOREIGN';
END; RUN;
```

Another option is:

```
DATA PRODUCTS;
  LENGTH MANUF $10.;
  SET SASDATA.PRODFILE(KEEP=MAKE);
  SELECT;
    WHEN (MAKE='1') MANUF='FORD';
    WHEN (MAKE='2') MANUF='CHRYSLER';
    OTHERWISE MANUF='FOREIGN';
END;
RUN;
```

Multiple values:

```
DATA PRODUCTS;
  LENGTH MANUF $10.;
  SET SASDATA.PRODFILE(KEEP=MAKE);
  SELECT;
    WHEN (MAKE IN('1','2','3','4')) USMADE='YES';
    OTHERWISE USMADE='NO';
  END;
RUN;
```

Do-Loop Processing:

```
DATA PRODUCTS;
  LENGTH MANUF $10.;
  SET SASDATA.PRODFILE(KEEP=MAKE AIR);
  SELECT;
    WHEN (MAKE='1')
      DO;
        MANUF='FORD';
        IF AIR='Y' THEN OPTION1='AIR';
      END;
    WHEN (MAKE='2') MANUF='CHRYSLER';
    OTHERWISE MANUF='FOREIGN';
  END;
RUN;
```

Using the Select Statement:

- * Arrange select values in descending order of probability to improve efficiency.
- * There must be at least one WHEN statement.
- * If more than one WHEN statement is true, only the first WHEN statement is executed.
- * The OTHERWISE statement specifies a statement to be executed if no WHEN condition is met.
- * An error message is issued if all SELECT-WHEN comparisons are false and no OTHERWISE statement is present.
- * END statement ends a SELECT group.

When to use the select statement versus IF-THEN statements:

- * When you have a long series of mutually exclusive conditions. SAS stops checking statements when the condition is true.
- * SELECT groups makes the program easier to read and debug.

Re-coding Using a User-Defined Format

The above problem could have been similarly solved by the following:

```
PROC FORMAT;
  VALUE $ MAKEFMT;
    '1'='FORD'
    '2'='CHRYSLER'
    OTHER='FOREIGN';
RUN;

DATA PRODUCTS;
  SET SASDATA.PRODFILE(KEEP=MAKE);
  MANUF=PUT(MAKE,$MAKEFMT.);
RUN;
```

SELECT statement versus a user-defined format:

- * Once a user-defined format is created it can be permanently stored and re-used.
- * Real efficiency can be gained by re-coding the value upon reading it in by using a user-defined INFORMAT.
- * SELECT statements are nice for one-shot re-codes.

Using Informats for Re-coding

When we created the SASDATA.PRODFILE data set we could have used informats to transform, or re-code the data upon reading it in:

```
PROC FORMAT;
  INVALUE $ MAKEFMT;
    '1'='FORD'
    '2'='CHRYSLER'
    OTHER='FOREIGN';
RUN;

DATA SASDATA.PRODFILE;
  LENGTH MAKE $10 ;
  INFILE PRODUCT;
  INPUT @1 MAKE $MAKEFMT2.;
RUN;
```

The MISSING Statement

MISSING declares special missing values in the input data for numeric values. Typically in survey data, a no-response (contact established, but refused to respond to the question) is distinctly different from a not available (no contact established). The MISSING statement is a convenient, multi-purpose way to handle variables of this type.

```
DATA GUNS;
  MISSING R A;
  SET SASDATA.VIEWS(KEEP=GUNCONT);
RUN;
```

```
PROC FREQ;
  TABLES GUNCONT/MISSING;
RUN;
```

```
PROC MEANS;
  VAR GUNCONT;
RUN;
```

The resulting output will show a separate frequency statistic for the values R and A. And in the case of the means, R and A will not be counted in the calculation of the mean.

MISSING statement versus MISSING= option:

The MISSING= system option specifies a character to be printed when an ordinary numeric missing value is encountered (.). If your data contains characters representing special missing values such as 'a' and 'r' **DO NOT** use the MISSING= option.

The MISSING statement allows us to perform numeric operations on the variable even though theoretically the field contains character values.

Efficient Use of SAS PROCs

When using SAS Procedures make sure you are aware of the defaults and know when to override them:

- * **Select only the variables needed for the analysis.** If all you want to know is the average price of breakfast cereal, why allow PROC MEANS by default to generate the average price of all food items? Limit the analysis variables by using the VAR statement.
- * **Suppress the calculation of unnecessary statistics.** If you are only truly interested in the mean, or average price, of breakfast cereal, suppress the calculation of other default statistics that PROC MEANS generates, such as MIN, MAX, STD DEV. Use the options on the PROC statement to limit the statistics.

Several of the SAS PROCs report the same basic statistics. It's to your advantage to utilize a procedure to its fullest potential. Why use two procedures, like PROC SUMMARY to sum up a variable and PROC MEANS to calculate the mean when you could have used either SUMMARY or MEANS to sum it up AND calculate the mean all in one procedure.

People tend to forget that PROC UNIVARIATE can generate a frequency table identical to PROC FREQ by using the FREQ option:

```
PROC UNIVARIATE DATA=SASDATA.VOTERDAT FREQ PLOT;
  VAR DISTRICT;
RUN;
```

Other useful procedures in the Base SAS product for survey researchers are: PROC UNIVARIATE, PROC CORR, PROC FREQ, and PROC TABULATE.

The best way to maximize PROCs is to know what statistics are available in each procedure. SAS provides a useful table in their *SAS Procedures Guide* on pages 2-3.

Sampling

Every researcher at some point or another needs to sample their data. There are a number of different methods and ways in which the data can be sampled:

Nth Sampling

You want to select every nth observation.

```
DATA SAMPLE;
  DO OBSNO=10 TO TOTOB5 BY 10;
    SET SASDATA.VOTERDAT POINT=OBSNO NOBS=TOTOB5;
    OUTPUT;
  END;
STOP;
RUN;
```

Randomly Selecting an Exact Size Sample With Replacement.

```
DATA SAMPLE;
  SAMPSIZE=100;
  DO N=1 TO SAMPSIZE;
    OBSNO=CEIL(UNIFORM(0)*TOTOB5);
    SET SASDATA.VOTERDAT POINT=OBSNO NOBS=TOTOB5;
    OUTPUT;
  END;
STOP;
RUN;
```

Randomly Selecting an Exact Size Sample Without Replacement

```
DATA SAMPLE;
  SET SASDATA.VOTERDAT;
  X=RANUNI(0);
RUN;

PROC SORT DATA=TWO OUT=SAMPLE(OBS=20);
  BY X; RUN;
```

Approximate Sized Samples

You want to select approximately 10% of the data set.

```
DATA SAMPLE;
  SET SASDATA.VOTERDAT;
  IF RANUNI(0) LE .10;
RUN;
```

Generating Mailing Labels and Form Letters

SAS is one of the few mainframe packages that can do a match-merge for form letters. By using the SAS/FSP® product, survey researchers can maintain their database of survey respondent data. Names and addresses can be updated and deleted. From the database mailing labels can be generated, form letters written, tracking of responses can be maintained as well as follow up letters and mailing labels.

The following code generates 3 sets of two up labels:

```
PROC FORMS DATA=SASDATA.VOTERDAT NACROSS=2
  BETWEEN=2 LINES=3 DOWN=1 INDENT=2 NDOWN=3 SETS=3;
  LINE 1 NAME;
  LINE 2 ADDRESS1;
  LINE 3 ADDRESS/REMOVE;
  LINE 4 CITY STATE ZIP/PACK;
RUN;
```

Some of options used in PROC FORMS:

NACROSS=	number of form units across the page
BETWEEN=	number of columns between form units
LINES=	number of lines on form units
DOWN=	start this far from the top of the page
INDENT=	start this far from the left margin of the page
NDOWN=	number of form units down the page
SETS=	the number of sets of the entire list (this is like running the entire PROC FORMS step multiple times)

Optional Statements:

PACK trims trailing blanks when more than one variable is specified on the line statement

REMOVE deletes entire line if all the variable values are blank.

Other SAS Procedures and Products

There are a number of other SAS procedures that are useful to survey researchers that are used less frequently such as PROC TIMEPLOT, PROC STANDARD, and PROC RANK. While there are some graphical procedures in the Base SAS product, such as PROC PLOT and PROC CHART, the results are not presentation-ready. The SAS/GRAPH product is more likely to produce quality output for presentation.

SAS/STAT and SAS/ETS are two statistically oriented packages. SAS/STAT covers most statistics not appearing in the Base software package and SAS/ETS is for econometric time series modeling.

Trademark Notice

SAS/FSP, SAS/GRAPH, SAS/STAT, SAS/ETS are registered trademarks of the SAS Institute Inc., Cary, NC, USA.

The author will be glad to answer questions and accept suggestions at the following address:

Kim L. Kolbe
Systems Seminar Consultants
6014 Gateway Green
Madison, WI 53716
(608) 222-7081