Paper 102-25

Flexibility in Data Management Through the SAS® System

Shawn P. Thomas, University of Arkansas, Fayetteville, AR
Brandon L. Welch, University of Arkansas, Fayetteville, AR
Keely P. Casey, University of Arkansas, Fayetteville, AR

Abstract

Researchers in education and other areas of the social science arena have a need for the ability to access, store, organize, and manipulate potentially large data sets. Useful data exist in a variety of formats and originate from a broad range of sources including the Internet, past research, and data gathering agencies. While increasing flexibility in solving problems, this variety of formats and sources also increases the level of difficulty involved in reading and controlling data. Further, once the data are accessed, stored, and organized, they frequently need to be cleaned and manipulated or reorganized to produce the necessary output or results. This paper presents a variety of methods via the SAS® system which may increase the level of flexibility and control afforded investigators when faced with data management problems.

Introduction

The increased availabilities of information and the progressively rapid growth of technology have subsequently created a need for improved ability to control and read wide varieties and potentially large amounts of data or information. This ability may be enhanced by the accessibility of a number of software packages. However, with these packages comes a need for knowledge of their capabilities for achieving analyzation and communication goals. Additionally, an understanding of how various software packages interact can enhance their individual capabilities. The SAS® system provides researchers with a plethora of tools aimed at facilitating the actualization of data driven goals. These tools include a wide range of statements and procedures with options which allow individuals to customize their data management and analysis operations. Also included in the SAS® toolbox are a number of point and click procedures which are easy to follow and often more efficient than syntax. The purpose of this paper is to present ideas aimed at using various tools to overcome specific issues related to data management.

Data Access and Storage

The first step in any data analyzation process is to place data in some format which the software package in question can read and utilize. Data may emanate from a broad range of sources including various sites on the Internet, a UNIX system, a floppy disk, a CD-ROM, or a zip disk. They may exist in a wide range of formats, including a spreadsheet or a flat text format with comma, tab, or space delimiters. They could also exist in a flat text format with no delimiters separating the variables, or in a format not readable by humans.

In SAS® a multitude of tools facilitate the transformation of data to a more easily readable format helping to minimize the difficulty associated with completing analyses on data taken from outside sources. These tools include windows where individuals point and click their way through the process of loading data from any of several formats into a data library. Another tool available in SAS/CONNECT® software makes a connection between the SAS® system and a UNIX system. Access to this software is available through programs which must be written by the researcher or through SAS/ASSIST®. If data are read from a floppy disk, CD-ROM, or a hard drive, syntax in the form of INFILE and INPUT statements must be written. The INFILE statement gives information on the location of the data set in question and may provide some descriptive characteristics of that data set. The INPUT statement contains information on variables in the data set including locations, names, formats, and lengths. The availability of these procedures provides easy access to a multitude of problem solving tools geared toward data access.

Once access to data is gained, the question of storage for future use becomes pertinent. One major advantage of the SAS® system is the potential for permanent data storage through data libraries that provide easy access to any combination of variables including previously written labels and formats. To create a data library, a user must first create a folder on some file storage system then utilize an option on the toolbar to place a library in that folder. Development of a library is achieved through either SAS/ACCESS®, SAS/INSIGHT®, or syntax in base SAS®. SAS/ACCESS® is an easy-to-use procedure for importing data from a wide range of platforms. It provides point and click options thereby not requiring the composition of syntax, but is limited in its ability to customize the data set. SAS/INSIGHT® has a broad range of capabilities which includes a spreadsheet format for entering and customizing data. If the user chooses to create syntax for the purpose of loading data into a library, they may use a combination of options including a data statement, an infile or statement, an import Procedure, an input statement, a formatting Procedure, a label statement, and a format statement.

Sample Program 1: (Creates data sets in some example libraries.)

```
options fmtsearch = (formats);
libname = example1  'c:\example1';
libname = example2  'c:\example2';
libname = formats    'c:\sas formats';
```

(FMTSEARCH and LIBNAME give SAS® information on where to store data and formats. Note: An autoexec program could also be used for this.)

```
proc format  library = formats;
value   formata   00 = "Not Available";
value $formatb "F" = "Female"
               "M" = "Male"; run;
```

(PROC FORMAT loads a set of formats into a library.)

```
proc import out = new datafile = "c:\olddata1.xls"
dbms = excel5 replace;
getnames=yes; run;
```

(PROC IMPORT reads a Microsoft® Excel file and stores it as a SAS® data set in the Example Library. The following syntax reads the data set that was placed in the work library by the import procedure and adds labels to each variable.)

```
data example1.newdata1;
set new (keep = ssn lastname firstnme gender district
score1);

label
ssn          = "Student's Social Security Number"
lastname     = "Student's Last Name"
firstnme     = "Student's First Name"
district     = "District Name"
gender       = "Student's Gender"
score1       = "Mathematics Test Score 1";run;
```

(The following syntax reads a data file on a CD-ROM, stores it in the Example Library and adds labels to each of the variables.)

```
data example2.newdata2;
infile "e:\olddata2.dat"     lrecl = 1500;
input @1 (ssn) ($9.) @10 (lastname) ($15.)
      @25 (firstnme) ($10.)  @35 (district) ($20.)
      @80 (gender) ($1.)  @81 (score2) (2.)
      ....continuing to column 1500;

label
ssn        = "Student's Social Security Number"
lastname   = "Student's Last Name"
firstnme   = "Student's First Name"
district   = "District Name"
gender     = "Student's Gender"
score2     = "Mathematics Test Score 2"; run;
```

(The following syntax associates variables in the indicated data sets with formats which have previously been stored in the Example Library.)

```
data example1.newdata1;
set example1.newdata1;
format gender  $formatb.   score1 formata;
run;

data example2.newdata2;
set example2.newdata2;
format gender  $formatb.   score2 formata.;
run;
```

Data Cleaning

Once the data are stored and easily accessible, one may begin the necessary statistical analyses, however, it is very important to consider the state of the data being used. The data must be clean. Cleaning a data set refers to the process of minimizing the possibility of committing errors in the statistical analysis that may follow, thereby increasing the level of control investigators have over their analysis. Cleaning a data set is an extremely tedious and complicated process, however due to SAS®'s flexibility, a series of techniques can be utilized to control various aspects of the data.

This section addresses three problems and their associated solutions frequently encountered in the social sciences. The first problem is one of dealing with missing observations. Missing observations are coded with a dot or period when inputting data with a dot or period when manipulating data manually with a CARDS statement, or a zero when using Microsoft® Excel or Lotus® Spreadsheet. This will dramatically affect the conclusions of any statistical analysis. Thus, one may implement simple IF/THEN statements to alleviate the problem. This is illustrated in the following source code:

Sample Program 2a:

```
options nocenter ls=80;
data example2.newdata3;
input name $ mathscor readscor;
```

```
if mathscor = 0 then mathscor =.;
If readscor  = 0 then readscor  =.;
cards;
subject1 23 50
subject2 45 0
subject3 0  14
;
proc print;
run;
```

The aforementioned code is similar when one uses a data library. The only modifications are the removal of INPUT and CARDS statements, and the addition of a SET statement as demonstrated below:

Sample Program 2b:

```
options nocenter ls=80;
data one;
set example2.newdata2;
if mathscor = 0 then mathscor =.;
if readscor = 0 then readscor =.;
proc print; run;
```

Another tool useful in data cleaning involves the manipulation or reorganization of data. Applying a fictional situation, a researcher has a data set which is a longitudinal combination of two data sets from two different years of standardized test data for his particular state. S/he is concerned with determining the number of students in each school for each year. Further, suppose that some of the schools have slight changes in their reported names within the same years and between the two years, so that the data set's smallest unit of observation is the student, and contains 200,000 students. To gain an accurate interpretation of each school's population, s/he must first make the school names concise by reading all of the school's names, then generating syntax to develop one name for each school. Without reorganizing the data s/he would have to sift through 200,000 observations to determine which school names needed to be changed. Simplification of this process is achieved by creating another data set with school names as the smallest unit of observation. This is accomplished through the use of SAS® procedures PROC FREQ and PROC SORT.

Sample Program 3: (Develops a data set of all possible school names for each school then creates consistent school names and prints the population for each school.)

```
data one;
set example.19961997 (keep = schoolf6);
proc freq data=one;
tables schoolf6 / out=out1
(drop=count percent pct_col pct_row);
run;

data one;
set Example.19961997 (keep = schoolf6);

proc freq data=one;
tables schoolf6 / out=out2
(drop=count percent pct_col pct_row);
run;
```

(The above syntax generates two output sets of school names for 1996 and 1997. The following two statements print the contents of those two data sets for the researcher to investigate for differences in those school names.)

```
proc print data=out1;
proc print data=out2;
```

(The following syntax standardizes the school names.)

```
data dxample.19961997;
set example.19961997;
if schoolf6='TONI THORN ELEM' then
schoolf6='Toni Thorn Elementary';
if schoolf6='TONI THORN ELEM.' then
schoolf6='Toni Thorn Elementary';
if schoolf7='Toni Thorn Eleme' then
schoolf7='Toni Thorn Elementary';
if schoolf7='Toni Thorn Elemen' then
schoolf7='Toni Thorn Elementary';
if schoolf6='MAC MURPHY ELEM' then
schoolf6='Mac Murphy Elemenatry';
if schoolf6='MAC MURPHY ELEMENTARY' then
 schoolf6='Mac Murphy Elemenatry';
if schoolf6='JAMES WOLFE ELEM.' then
 schoolf6='James Wolfe Elementary';
if schoolf6='James Wolfe Eleme' then
schoolf6='James Wolfe Elementary';
if schoolf7='JAMES WOLFE ELEM.' then
 schoolf7='James Wolfe Elementary';
if schoolf7='James Wolfe Elemen' then
schoolf7='James Wolfe Elementary';run;
```

(The following syntax generates the population of each school in the data set.)

```
proc freq data=example.19961997;
tables schoolf6;
run;
```

```
proc freq data=example.19961997;
tables schoolf7;
run;
```

The final problem, like its predecessor, involves combining multiple years of data. An extremely useful statistical technique often used in the social sciences is the repeated measures design. In order to perform this analysis in SAS®, a master data set must be created containing subjects that have multiple measurements of some phenomenon over time. For instance, an experimenter wants to conduct a repeated measures analysis on students within a specific school over three years. These data must be merged and, in this particular case, the most effective way to achieve this is merging by social security number. This is problematic if the students did not consistently report accurate social security numbers for all three years. However, if the problem is mitigated, the social security number rubric is still more effective for matching observations than other rubrics using other descriptor variables such as grade, school, date of birth, last name, or first name.

Sample Program 4: (The following program utilizes the UPDATE command to overwrite students' social security numbers reported during the first year (1997) of data collection over those recorded in the years that followed.)

```
data ida;
set example2.school97 (keep=lastname nickname dob
ssn grade97 score1);
year97=97;
grade = grade97; run;

data idb;
set example2.school98 (keep=lastname nickname dob
ssn grade98 score2);
year98=98;
grade = grade98 - 1; run;

data idc;
set example2.school99 (keep=lastname nickname dob
ssn grade99 score3);
year99=99;
grade = grade99 - 2; run;

proc sort data=ida;by dob grade lastname nickname;
proc sort data=idb;by dob grade lastname nickname;
proc sort data=idc;by dob grade lastname nickname;
```

(The above syntax develops and sorts three data sets consisting only of variables to be used in the UPDATE and MERGE statements. The grade variable is developed from three different grade variables and should be consistent between the data sets.)

```
data idd; update idb ida;
by dob grade lastname nickname; run;
dataide; update idc idd;
by dob grade lastname nickname; run;
```

(The above syntax first merges students whose date of birth, grade, last name, and first name match between the 97 and 98 data sets then overwrites the social security numbers in the first data set with those in the second. The process is then repeated by adding the final data set.)

(Since the three smaller data sets are now merged into one, the must be separated then merged with the larger data sets overwriting the old social security numbers with the new. The previously generated year9_variable is utilized to complete the separation.)

```
data ida;
set ide;
if year97 = . then delete;
run;

data ida;
set ide;
if year98 = . then delete;
run;

data idka;
set ide;
if year99 = . then delete;
run;

data example2.school97;
update example2.school97 ida;
by grade lastname nickname score1; drop year97
year98 year99;
proc sort; by id; run;

data school98;
update example2.school98 idb;
by grade lastname nickname score2; drop year97
year98 year99;
proc sort; by id; run;

data school99;
update example2.school99 idc;
by grade lastname nickname score3; drop year97
year98 year99;
proc sort; by id; run;
```

Now the potential for successfully using the social security number to merge a larger number of students is often increased. Note that the UPDATE procedure which

overwrote the social security numbers used an extremely stringent rubric allowing fewer errors to be made.

Data Organization and Manipulation

While data manipulation techniques are useful in the data cleaning process, they also have value in other areas. For instance, if necessary, one may reduce the size of a data set by removing extraneous variables using a DROP statement, increase the amount and type of information available by merging two longitudinal data sets, take a random sample of a set of observations, split one variable into a set of variables or one data set into a number of data sets, create a data set of means or frequencies, or simply change the appearance of a data set by rearranging variables or transposing them. This flexibility with the data sets allows investigators more freedom to use multiple types of analyses with greater ease. Examples of several of the aforementioned data manipulations are provided in the following programs.

Sample Program 5: (Merges four data sets, rearranges some variables and splits others, then creates smaller data sets for each school in the larger data set.)

```
data one;
merge example1.cohort96 example1.cohort97
example2.cohort98 example2.cohort99;
by stdntid; run;
```

(The previous syntax merges four data sets by the students' social security numbers. The following syntax rearranges the order of the variables and splits the three session variables into 40 session score variables.)

```
data two;
set one (keep = stdntid school96 school97 school98
school99 session1 session4 session6 gender race age);
file 'c:/rearange.dat'  lrecl= 200;
put @1 (stdntID) ($9.) @ 10 (gender) ($1.)
    @13 (race) (1.) @ 15 (age) (2.)
    @18 (school96) ($30.) @ 52 (school97) ($30.)
    @85(school98) ($30.) @118 (school99) ($30.)
    @151 (session1) (20.) @174 (session4) ($12.)
    @189 (session6) (8.); run;

data example2.newsets;
infile 'c:/rearange.dat'  lrecl=200;
input @1 (stdntid) ($9.) @10 (gender) ($1.)
      @13 (race) (1.) @15 (age) (2.)
      @18 (school96) ($30.) @52 (school97) ($30.)
      @85 (school98) ($30.) @118 (school99) ($30.)
      @151 (ssn1an1-ssn1an20) (20*1.)
      @174 (ssn4an1-ssn4an12) (12*$1.)
      @189 (ssn6an1-ssn6an8) (8*1.); run;
```

(The following macro splits the new data set into smaller school based data sets consisting of all students who attended the particular school for any of the four years.)

```
%macro split(school,group);
data &group;
set example2.newsets;
if schoolf6=&school  or  schoolf7=&school  or
schoolf8=&school  or  schoolf9=&school;
proc export data=&group
            outfile="c:rogers\&group.xls"
            dbms=excel5 replace;
run;
%mend;

%split(Toni Thorn Elementary, TTE)
%split(Mac Murphy Elementary, MME)
%split(James Wolfe Elementary, JWE)
```

When a researcher completes analyses on a large number of observations along with a large number of variables, the potential for reporting results can be staggering. The goal in developing tables or graphical representations of output is to present large amounts of information quickly and/or in a manner which is easy to understand. However, this tool can become problematic if the data are not organized in such a way as to be readable to the report generating procedures. In this case, the data manipulation techniques can be utilized to develop data sets which may be more accessible to the reporting procedures.

Sample Program 6: (Manipulates data for utility in developing charts then generates the charts.)

```
proc format;
value $score "1997read"   =   "1997 Reading"
             "1998read"   =   "1998 Reading"
             "9798read"   =   "Reading Growth"
             "1997math"   =   "1997 Math"
             "1998math"   =   "1998 Math"
             "9798math"   =   "Math Growth"
             "1997scie"   =   "1997 Science"
             "1998scie"   =   "1998 Science"
             "9798scie"   =   "Science Growth";
run;

title1 "Diane's Public Schools";
title2 "Transformed Reading Scores";
```

(The previous syntax creates formats and titles which will be used in the charts. The following syntax creates a data set of the means of nine reading, math, and science scores for each grade in the data set then transposes the new data

set to fit the format required by PROC GCHARTS.)

```
data one; set example2.dmerges;
proc sort data=one;by grade97;run;
proc means noprint data=one;
var  1997read  1998read  9798read  1997math
     1998math  9798math 1997scie  1998scie
     9798scie;
by grade97;
output out=temp1 (drop=_type_ _freq_) mean=;
run;

proc transpose data=temp1 out=cts1a name=typea
prefix=score;
var  1997read  1998read  9798read  1997math
     1998math  9798math 1997scie  1998scie
     9798scie;
by grade97;run;
```

(The following syntax develops the charts.)

```
pattern1 v=solid color=white repeat=3;
pattern2 v=solid color=white repeat=3;
pattern3 v=solid color=grayee repeat=3;

data charts;
set cts1;
if score1=. then score1=0;
axis1 label=none; axis2 label=none;
proc sort; by grade97;run;
proc gchart data=charts;
vbar3d typea
     / cframe=white
       inside=sum
       patterid=midpoint
       group=gtype1
       sumvar=score1
       nozeros
       width=6
       space=3.0
       gspace=5.0
       raxis=axis1
       maxis=axis1
       gaxis=axis1
       axis = 0 to 100 by 10 href=50
       caxis=black
       coutline=gray;
       by grade97;
       label grade97 = “Grade";
run; quit;
```

Conclusion

There is enormous potential for data management to be burdensome and oftentimes complicated. The SAS® system provides users with many options which remedy many of the rigors statisticians and/or data managers face. Furthermore, these options increase the level of control researchers have over their data as well as flexibility in their analyses. The methods presented in this paper are geared primarily for educational research, but they need not be restrictive. The flow of information is essential to any data-dependent field. To ease this flow, one needs sound storage/access capabilities, accurate statistical interpretations made possible through data cleaning, and the ability to organize and manipulate large data sets. It is hoped that the examples in this paper provide the readers with some level of inspiration and guidance in their data management endeavors.

References

SAS® Institute Inc. (1990), *SAS Language: Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS® Institute Inc. (1990), *SAS Procedures Guide, Version 6, Third Edition*, Cary, NC: SAS Institute Inc.

Miron, T. (1995). *The How-To Book for SAS/GRAPH Software,* Cary, NC: SAS Institute Inc.

Contact Information

Shawn Thomas
Office of Research, Measurement, and Evaluation
University of Arkansas
302 West Avenue Annex
Fayetteville, Arkansas 72701
email: **spthoma@comp.uark.edu**
(501) 575-5593