Introduction to SAS

Elly Kaizar
132M Baker Hall
ekaizar@stat.cmu.edu

July 15, 2002

1 Communicating with SAS

There are two ways to communicate with SAS. The interactive way is similar to SPSS in that you use wizards to create/modify data and generate output using this data. However, a commonly more efficient way to communicate with SAS is to write mini programs called *batch files* that you can run through SAS all at once. This second method is what we will be working with. In order to do this, we will be using SAS's text editor.

2 SAS Structural Overview

SAS is used to analyze data. It does this in two basic steps - input steps called *data steps* and output steps called *procedure steps*. In addition to these, SAS has settings you can change in what I will call *option/setting steps*. *Data steps* are used to create and/or modify SAS data sets. These data sets must be defined before any *procedure steps*, which will in turn use your created SAS data sets to calculate the output you request.

3 The Data Step

3.1 Naming a Data Set

To name a data set and begin the data step, use the data command:

DATA *mylib.mydata*;

The second part of the data's name is a descriptive title you can make up. The first part of the data's name is the library in which it is stored. You can either use the default library (WORK), or you can name your own **before the data step** with the command:

LIBNAME *mylib* *'file path/name'*;

If you choose to use the default, you do not need to include the library name in the name of the data set (i.e., do not include *mylib.* in front of *mydata*). However, the data set will not be permanent. That is, the SAS data set will be deleted when you quit SAS.

3.2 Getting Data

There are four ways to get data into a SAS data set:

1. **From a tab or space delimited file**

   INFILE *'file path/name'*;

   INPUT *numvar charvar $ ...*;

   'INFILE' tells SAS where to look for the data, and 'INPUT' tells SAS what is in the file and how it is organized. For a simple tab or space delimited file, the 'INPUT' statement simply consists of a list of the variable (column) names in the order in which they appear, and a $ after any variable name that is non-numeric.

2. **From another sort of file**

   INFILE *'file path & name'* <delimiter=",">;

   INPUT <see help menu>;

   'INFILE' and 'INPUT' have the same meaning as above, but the syntax differs according to the organization of the source file.

3. **From previously created SAS data sets**

   SET *dataset1* <*dataset2*>;

   -or-

   MERGE *dataset1 dataset2*;

   BY *idnumber*;

   SET treats the two data sets as two sets of observations with the same variables. MERGE treats the two data sets as two sets of variables for the same observations (or patients). MERGE requires a unique identifier (idnumber) for each observation so that it can properly match up observations.

4. **Direct typing**

   INPUT *var1 var2 $*;

   CARDS;

   <input, using . for missing data>;

A final note about reading data from a file. SAS has a default input record (line) length; if the record length is greater than that, you must specify how long the longest line is allowed to be using ‘LRECL=#’. Also, using the option ‘MISSOVER’ instructs SAS to recognize as missing any variables for which it did not find values in the current line.

3.3 Modifying The Data Set

- **Labels and Formats** To give variables nicer (longer, more descriptive) names in the SAS output, you can use the label command:

  LABEL *myvar1*=”*mylabel1*”
  myvar2=”*mylabel2*”;

  You will also most likely want to format any categorical variables. For example, a variable GENDER will often be coded 0 for female and 1 for male. SAS will use female/male rather than 0/1 in the output if you specify an appropriate format. First, you must define a format **before the data step**:

  PROC FORMAT;
  VALUE *myformat1* 0=”*label0*” 1=”*label1*”;
  VALUE *myformat2* 1=”*label1*” 2=”*label2*”;
  RUN;

  Then, include as an option **in the data step**: FORMAT *myvar1 myformat1*. *myvar2 myformat2*.;

- **Selecting Subsets**

  - **Saving and Deleting Entire Variables** To keep only some of the variables in a data set, use the command:

    KEEP *myvar1 myvar2 myvar3*;

    To drop some of the variables from a data set, use the command:

    DROP *myvar1 myvar2 myvar3*;

  - **Saving and Deleting Certain Observations** To keep only those observations that satisfy some test (such as *myvar*=2), use the command:

    IF *test*;

    To drop only those observations that satisfy some test, use the command:

    If *test* then DELETE;

- **Creating New Variables**

  - **Creating Constants** Type the new variable name and what it equals. To create a new variable called *newvar* that equals the same number or string for every observation, use:

    newvar = #;

- **Combining Current Variables** Type the new variable name and what it equals. For example, to create a new variable called *newvar* that is the ratio of *myvar1* and *myvar2*, use:

  newvar=*myvar1*/*myvar2*;

- **if...then** To create a new variable *newvar* that depends on the existing variable *myvar1*, use:

  IF *myvar1* < 5 then *newvar*=0;

  ELSE IF *myvar1* < 10 then em newvar=1;

  ELSE *newvar*=2;

  IF *myvar1* = . then *newvar*=.;

  SAS believes that a missing value is included in the first statement, so the fourth is needed to write over the erroneously coded missing values.

4 Procedure Step Syntax

Procedure steps are much more variable than data steps. Their syntax in general follows:

PROC *PROCNAME* <DATA=*mylib.mydata* > <options>;
    <options>;
    RUN;

4.1 Some Common Procedures

In the following procedures, *quantvar* indicates a quantitative variable, *catvar* indicates a categorical variable, *respvar* indicates a response variable and *expvar* indicates an explanatory variable.

4.1.1 General

- PROC SORT;
    BY *myvar*;
    RUN;

- PROC PRINT;
    VAR *myvar*;
    RUN;

4.1.2 Data Description

- PROC UNIVARIATE;

  HISTOGRAM *myvar*;

  VAR *myvar*;

  RUN;

- PROC BOXPLOT;

  PLOT *quantvar*constant*;

  RUN;

  (single boxplot)

- PROC BOXPLOT;

  PLOT *quantvar*catvar*;

  RUN;

  (split boxplot)

- PROC GPLOT;

  PLOT *quantvar1*quantvar2*;

  RUN;

- PROC FREQ;

  TABLES *catvar*;

  RUN;

  (one-way tabulation)

- PROC FREQ;

  TABLES *catvar1*catvar2*;

  RUN;

  (two-way tabulation)

4.1.3 Statistical Procedures

- PROC MEANS;

  VAR *quantvar*;

  RUN:

- PROC FREQ;

  TABLES *catvar* /BINOMIAL;

  RUN;

  (test of proportions)

- PROC FREQ;

  TABLES *catvar1*catvar2* /CHISQ;

  RUN;

- PROC TTEST;
  CLASS *catvar*;
  VAR *quantvar*;
  RUN;

- PROC CORR;
  VAR *myvar1 myvar2*;
  RUN;
  (correlation)

- PROC REG;
  MODEL *respvar* = *expvar1 expvar2*;
  RUN;

  1. plot observed values vs. fitted values
     PLOT *respvar**p.;
     RUN;
  2. qq plot residuals
     PLOT r.*nqq.;
     RUN;
  3. plot residuals vs. fitted values
     PLOT r.*p.;
     RUN;

- PROC LOGISTIC;
  CLASS *respvar*;
  MODEL *respvar* = *expvar1 expvar2*;
  RUN;

5 Option/Setting Steps

SAS has built-in options and settings for various reasons. You can change these default settings. Several common options/setting steps are:

- LIBNAME ...;
- OPTIONS NOCENTER;
- OPTIONS LINESIZE=80 PAGESIZE=50;
- TITLE '*mytitle*'; TITLE2 '*mytitle2*';