Introduction

How to Use this Document

This document is an introduction to SAS for Windows. SAS is a large software package with scores of modules and utilities. It is impossible to present all its features in a brief document so we focus on the basics of getting started. If you do not own a personal copy of SAS for Windows, you may access the software from various UITS Student Technology Centers (STC's). Vendor-supplied documents are available for reference at the Swain Hall, Business/SPEA and Education Libraries' Reserve Collections, IUPUI University Library, and at the UITS Center for Statistical and Mathematical Computing (Stat/Math Center). Online manuals are available at the vdndor's site: http://support.sas.com/documentation/onlinedoc/91pdf/index.html. Faculty may arrange special introductory workshops for their classes in using SAS for Windows by contacting the UITS Stat/Math Center. If you want to buy a copy of SAS for Windows at an educational discount, contact the UITS Stat/Math Center.

What is SAS?

The SAS System is a comprehensive and flexible information delivery system that supports data access, data management, data analysis, and data presentation. It enables you to perform many analyses on your PCs that were once possible only on much larger machines. SAS for Windows reads data files from a variety of file formats including Excel, dBase, Lotus, and ASCII Text.

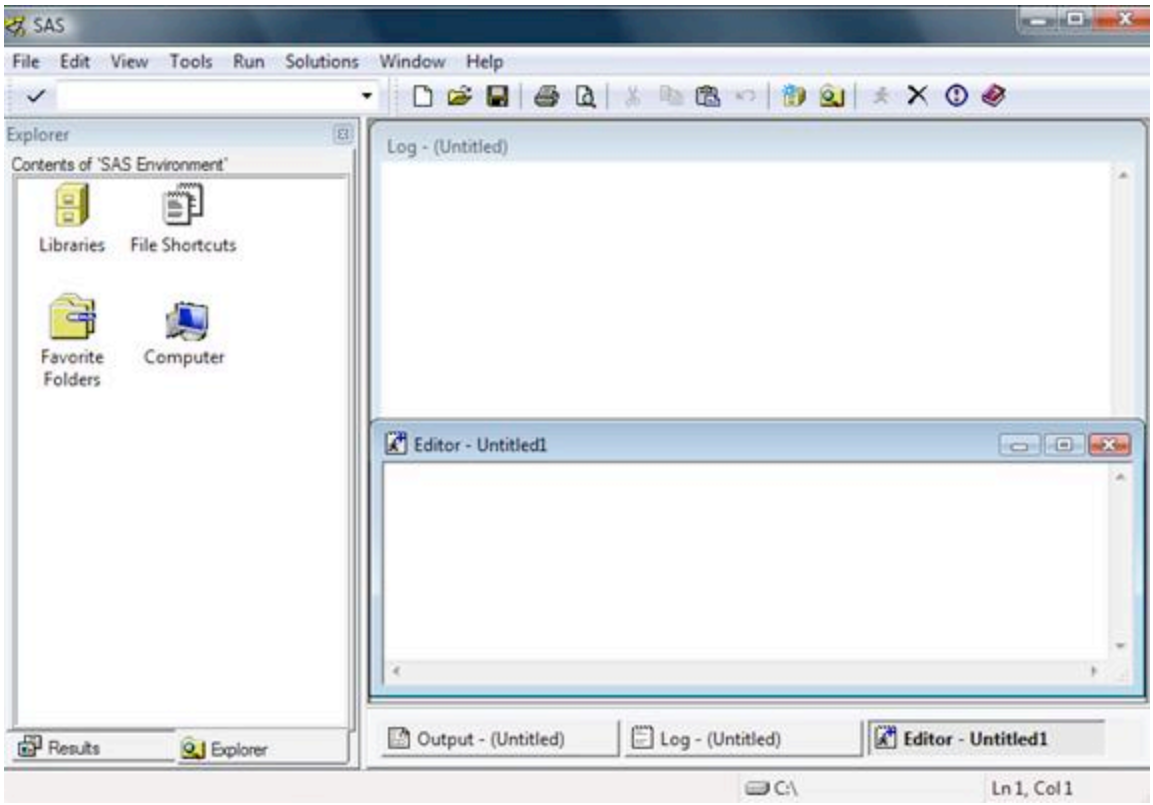
Launching SAS

The first task is to access SAS for Windows. If SAS for Windows is installed on your computer, start the Windows session and click the SAS for Windows icon. If you are accessing SAS for Windows from a computer in one of the UITS Student Technology Centers:

1. Find an available workstation.
2. Click **Start → All Programs → Departmentally Sponsored → Statistics-Math → SAS 9.13 SP4.**

Windows in SAS for Windows

The SAS System under the Windows environment has five basic types of windows.



Editor Window
: This window allows you to write SAS programs and submit your programs to execute.

Log Window
: This window displays the notes of SAS sessions, and tells you any errors, warnings after you submit your SAS programs.

Output Window
: This window returns the output of results from SAS executions.

Explorer Window
: This window shows SAS files and libraries in the Windows Explore like display.

Results Window
: This window illustrates a table of contents of SAS Output Window.

Writing a SAS Program: the DATA Step

A SAS program consists of two steps: DATA steps and PROC steps. In the DATA step a user may include commands to create data sets, and programming statements to perform data manipulations. The DATA step begins with a DATA statement. In the PROC (Procedure) step you invoke SAS procedures from its library to run the statistical analysis on a given data set. The PROC step begins with a PROC statement. These steps contain SAS statements. An important feature of SAS language is that every SAS statement ends with a semicolon (;). Without a semicolon a SAS statement is incomplete.

Organizing your Data for Analysis

SAS uses data organized in rows and columns. Observations are represented in rows and variables are represented in columns. An observation contains information for one unit of analysis (e.g., a person, an animal, a machine). Variables are information collected for each observation, such as name, score, age, income, educational level, etc. When data in files are arranged in rows and columns, they are called observations-by-variables, or rectangular data files.

Suppose you have three test scores collected from a class of 10 students (five males, and five females). Each student was assigned an identification number. The information for each student you have is an identification number, gender, and score for test one, test two, and test three. Suppose the data layout is as follows:

```
01 f 83 85 91
02 f 65 72 68
03 f 90 94 90
04 f 87 80 82
05 f 78 86 80
06 m 60 74 64
07 m 88 96 92
08 m 84 79 82
09 m 90 87 93
10 m 76 73 70
```

Notice that in the above data layout at least one blank space is left after each variable. In this instance you do not have to specify in which column a variable appears. SAS automatically considers a blank after each variable as a delimiter. In other words, if the data are space delimited, SAS will automatically know where each variable begins and ends. It is optional whether to leave a space between variable values. For example, you may choose to enter the data as following:

```
01f838591
02f657268
```

In this instance you need to assign column numbers to variables when using SAS to read the data. This is called a fixed format style input (where the variables across subjects are consistently in the same column). Format styles are discussed later in this document. Whichever format you choose, as long as you convey the format correctly to SAS, should not have any impact on the analysis. In the above layout each observation has only one line (record) of data. In another situation you may have multiple records per observation.

DATA Statement

DATA *dataname*;

The first word, DATA, tells SAS that you want to read a data file and store the data in a SAS data set you specify. Replace dataname with an appropriate SAS name (32 or fewer characters), i.e., trial, company, drug, behavior. In the example given below dataname is replaced by the name ANXIETY. Note the semicolon at the end of the statement.

DATA anxiety;

INFILE Statement

INFILE '*pathname\filename*';

If the data file is stored in a separate file an INFILE command is used to read an ASCII text file into the SAS program. Replace pathname with the name of the directory in which the data are stored. SAS can read several data files from within the same program file, so you can have multiple DATA steps in a single SAS program file.

The INFILE command is entered immediately after the DATA line.

```
DATA test;
  INFILE 'pathname\filename';
```

Replace pathname with the drive and path to your data file, and filename with the name of the file. For example, if you were reading a file called clas.dat from the c:\temp\ , the syntax would be:

```
DATA test;
  INFILE 'c:\temp\clas.dat';
```

INPUT Statement

INPUT *var1 column# var2 column# var3 column# ..... varn column#*;

The INPUT statement tells SAS the names of the variables and the column numbers that can be read on a specified line. Variable names in SAS can contain from 1 to 32 characters. They may contain numbers but must begin with a letter. If your data contains more than one line per observation, indicate the line number before specifying the variables on that line, for example:

INPUT id 1-3 company 8-10 #2 insal 6-10 finalsal 18-23 #3 retire 15-19;

The above INPUT statement informs SAS that there are 3 lines of data for each observation. The lines are indicated by a # sign.

The INPUT statement does not have to contain column numbers provided there is a space between each variable value on the data line. This is referred to as free format as opposed to the fixed format in which you specify the column numbers. If a variable contains character value, indicate it by a $ sign after the variable name. If you are choosing the free format, a character variable should not exceed 8 characters and should not contain embedded blanks. Free format may not be a good idea if you have a large number of variables.

If there are decimal points in your data, you may enter the decimal points as they are or leave them out when entering the data and later indicate in the INPUT statement that a given variable has specified number of decimal points. Suppose you have a variable GPA in your study and the value is to be indicated with 3 digits, the last two of which are decimal places. e.g. 3.89 If you decide to enter the decimal points in your data file indicate this in your INPUT statement as: INPUT GPA 1-4; Another choice is to leave out the decimal (389) and later indicate in the INPUT statement that the

variable GPA has 2 decimal points: INPUT GPA 3.2;. This means that the variable GPA is given in col. 1-3, and the last 2 places are decimal places.

The input format can also be written in a shorter form with a mixed style column and formatted input.

INPUT ID 1-2 SEX $ 3 (EXP SCHOOL) (1.) (C1-C10) (1.) (M1-M10) (1.) (MATHSCOR COMPSCOR) (2.);

In this case the program will read the variable ID from columns 1-2 and SEX from column 3. The next two variables, EXP and SCHOOL, have a width of 1 column each and start at column 4. The variables C1 through C10 (10 variables in sequential order) have a width of 1 column each and start immediately after the variable SCHOOL. If you want to read only the variables ID and the last 2 variables (MATHSCOR, COMPSCOR) you could write the INPUT statement as:

INPUT ID 1-2 @26 (MATHSCOR COMPSCOR) (2.);

The @ moves the pointer to column 26 and reads two variables with a width of 2 columns each.

DATALINES Statement

DATALINES;

The DATALINES statement is used only if the data are embedded within the SAS program file. The DATALINES statement tells SAS that data lines are included in the command file. Indicate the end of data lines by a semicolon at the beginning of a new line. For example:

```
DATALINES;
25 32 82 32 1
22 42 .  36 2
;
```

The DATALINES statement is entered toward the end of the DATA step.

IF-THEN and SAS Functions

Missing values in a data set can be represented either by a blank or by a period. If you choose a free format (leaving a space after each variable in the data set and not specifying the column numbers in the INPUT statement) make sure you represent missing values with a period. When SAS encounters a blank or a period in a data set the system regards it as a missing value. You can assign a missing value to a variable (e.g. 9, 99, 999, 000) and let SAS know which value for a given variable is assigned as missing. Suppose, for a variable MATHSCOR, 99 is assigned as a missing value. Immediately after the INPUT statement you may specify:

IF test1=99 THEN test1=.;

This statement will assign a missing value whenever it encounters a value of 99 within the variable test1.

You can also use the IF-THEN statement to recode any variable. For example, if you wanted to collapse the test scores into a dichotomous variable with students who scored 90 points receiving a score of one and the rest being assigned a score of zero, the syntax would be:

```
IF test1 >=90 THEN test1=1;
   ELSE test1=0;
```

There are a number of SAS operators which could be used in a DATA step, e.g. ** (raise to a power), * (multiplication), / (division), + (addition), - (subtraction), = or EQ (equal to), >= or GE (greater than or equal to), AND, OR, NOT.

In the DATA step you can also use a number of SAS functions to transform existing variables or create new variables. There are too many different SAS function to list them all here, but some of the most commonly used ones are: MEAN (arithmetic mean), SUM (sum of arguments), VAR (variance), ABS (absolute value), SIN (sine), LOG (natural logarithm), SQRT (square root).

For instance, to create a new variable called final with the arithmetic mean (average) of the 3 scores, you would type:

```
final=MEAN(test1,test2,test3);
```

For further details refer to SAS Language Reference: Concepts, Version 8.

LABEL Statement

You can use LABEL statements either in the DATA step or in the PROC step to give labels to variables.

LABEL variable='*variable label*';

Replace variable with the name of the variable, and variable label with the label you want to assign to the variable. A SAS variable is limited to 32 characters, whereas a label assigned to a SAS variable can have up to 256 characters including blanks. Labels should be enclosed in quotes, and the LABEL step is terminated by a semicolon. For example,

```
LABEL exp='years of computer experience';
LABEL mathscor='score in mathematics';
```

PROC FORMAT Statement

FORMAT statements associate formats with variables in a DATA step. For example, in a data set, the variable SEX has two values (f,m) and the variable SCHOOL has 3 values (1,2,3). To associate these values with appropriate value labels use the format statement. The FORMAT statement may be used in a DATA step or in a PROC step. However, when you define format with PROC FORMAT it appears as the very first line in a SAS program. Note that a dollar sign ($) required for a character variable format and character variable values are entered in single quotes.

```
PROC FORMAT;
  value $sex 'm'='male' 'f'='female';
  value school 1='rural' 2='city' 3='suburban';
```

Once you defined the format as above, you should associate the format with the variables through a format statement after the INPUT line.

```
DATA anxiety;
  INPUT id 1-3 sex $ 4 school 5 test 6-7;
  FORMAT sex $sex. school school.;
```

RUN Statement

To execute any series of statements that read or transform data in any way, you must also include a RUN statement to execute those commands unless they are followed by a PROC step (explained below). For example:

```
DATA anxiety;
  INPUT id 1-3 sex $ 4 school 5 test 6-7;
  FORMAT sex $sex. school school.;
RUN;
```

Comment Statements

Comments are provided for documentation purposes. Statements enclosed in /* .... */ or *..... are ignored by SAS and will not be used while executing the program.

```
/* This is a comment */
* So is this
/* This comment
spans several lines and ends with the asterisk-slash */
```

Writing a SAS program: the PROC Step

The next step is to create PROC (procedure) steps. SAS procedures read the SAS data and perform various computations and print the results of these computations. The FREQ procedure computes the frequencies on specified variables; the TTEST procedure performs a t-test analysis on the specified variables. In short, the statements that ask SAS to process or analyze a specified data set are known as PROC steps. The DATA steps and PROC steps can be used in any order within a SAS program. As the DATA step starts with a DATA statement the PROC step starts with a PROC statement.

```
PROC PRINT DATA=dataname;
  VAR var1 var2;
```

This procedure requests SAS to print data values for variables 1 and 2. If the VAR statement is omitted, data values for each variable in the data set will be printed. DATA=dataname is an optional

statement. If this is omitted, SAS selects the most recently created data set within the program. It is a good practice to specify the dataname along with the PROC statement. Replace the dataname with the name of the data created in the DATA step. Printing out a few variables before doing any analysis is a good way to check whether the data are being read by SAS as you want them to be read.

PROC FREQ Statement

PROC FREQ;
  TABLES *var1 var2 var1*var2*;

This statement produces tables showing distribution of variable values. In the above example SAS will display variable values for var1 and var2, and the combined frequency distributions for var1 and var2. For example, if you wanted to get the gender breakdown for the test scores discussed previously, you would use the following command:

PROC FREQ;
  TABLES sex;

The output generated by this command would look like this:

SEX	Frequency	Percent	Cumulative Frequency	Cumulative Percent
f	5	50.0	5	50.0
m	5	50.0	10	100.0

PROC MEANS Statement

PROC MEANS;
  VAR *var1 var2*;

This statement computes descriptive statistics for specified variables. If the VAR statement is omitted, descriptive statistics for each variable in the data set will be calculated.

Again, referring back to the grades data, if you wanted to generate some basic descriptive statistics for the students' test scores, the commands and output would look like this:

PROC MEANS;
  VAR test1 test2 test3;

The SAS System 10:00 Thursday, November 20, 2008

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
TEST1	10	80.1000000	10.4504120	60.0000000	90.0000000
TEST2	10	82.6000000	8.4616783	72.0000000	96.0000000
TEST3	10	81.2000000	10.6854002	64.0000000	93.0000000

PROC CORR Statement

PROC CORR;
  VAR *var1 var2*;

A correlation analysis is performed to quantify the strength of association between two numeric variables. For example, if you wanted to see if there were a correlation between student test scores, the commands and output would look like this:

PROC CORR;
  VAR test1 test2 test3;

```
                  Pearson Correlation Coefficients, N=10
                          Prob > |r| under Ho: Rho=0

                    TEST1             TEST2             TEST3
TEST1             1.00000           0.75692           0.91522
                   0.0               0.0113            0.0002

TEST2             0.75692           1.00000           0.86857
                   0.0113            0.0               0.0011

TEST3             0.91522           0.86857           1.00000
                   0.0002            0.0011            0.0
```

ENDSAS statement

By this statement you indicate that there are no more data steps or procedure steps to be read or processed. ENDSAS should be followed by a semicolon (e.g., ENDSAS;). However, an ENDSAS statement during a SAS for Windows session will exit the SAS session and take you to the Microsoft Windows environment.

Writing and executing a SAS Program

Writing a SAS Program

Now that we have looked at various steps in creating a SAS program, the next step is to write one. A SAS statement may begin in any column on a line. However, for the sake of clarity, lines starting with DATA or PROC statements begin in column 1; all other statements are indented in the following SAS programs. In the following example, SAS is asked to create a data set named grades after reading the data steps. The data are embedded within the SAS program. You may use the Notepad or your preferred editor to create the following program, grade1.sas, and save it to a disk.

A SAS program to read the data, and execute some basic descriptive statistics, in a simple form would be:

DATA grades;
  INPUT id 1-2 sex $ 3 test1 4-5 test2 6-7 test3 8-9;
DATALINES;

```
01f838591
02f657268
03f909490
04f878082
05f788680
;
RUN;

PROC PRINT;
  VAR sex test1 test2 test3;

PROC FREQ DATA=grades;
  TABLES sex test1 test2 test3;
RUN;
```

If your data contain at least one blank space after each variable value, you can skip the column specifications in the INPUT statement. In this case the program will read the variable ID from columns 1-2 and SEX from column 4, and test1 from column 6-7, test2 from column 9-10, and test3 from column 12-13. The procedure PRINT will print out the variables specified with the VAR (abbreviation for VARIABLES) statement.

```
DATA grades;
  INPUT id sex $ test1 test2 test3;
  DATALINES;
01 f 83 85 91
02 f 65 72 68
03 f 90 94 90
04 f 87 80 82
05 f 78 86 80
;
/* Other Command lines */
RUN;
```

This type of format is called free format. Missing values cannot be left blank in this type of data input as the SAS System will fill that blank with the succeeding variable value. Always assign some value to missing values when you use free format. Usually periods "." are used to represent missing values in SAS. With fixed format, you may leave the missing values blank.

Suppose you decided that you want to keep your data file as an external file, then make the following changes to the above program:

```
DATA grades;
  INFILE 'c:\temp\grade.dat';
  INPUT id 1-2 sex $ 4 test1 6-7 test2 9-10 test3 12-13;
RUN;
```

```
PROC PRINT;
  VAR sex test1 test2 test3;

PROC FREQ DATA=grades;
  TABLES sex test1 test2 test3;
RUN;
```

Replace c:\temp\grade.dat with your own parameter. The INFILE command points out the location of the data file during program execution. Note that the DATALINES statement followed by the data lines are not needed anymore in the above example because the data file is stored as an external file.

Executing a SAS Program

Suppose that you saved the above program into a file, grade1.sas, on C:\TEMP\ directory. If you have saved your data file, grade.dat, as an external file let us assume that file is also on C:\TEMP directory.

To retrieve the program file, grade1.sas, you created and saved on C:\TEMP directory, launch SAS and follow as below:

1. select File/Open

Note: If you haven't already created and saved the command file you may move the cursor to the EDITOR window and type in the lines. A dialog box appears which enables you to make an appropriate file selection. Once the file is selected click OK. The file will be opened into the EDITOR window. Next, tell SAS which commands you wish to execute. In SAS you can run an entire program file at once or just portions of it. Select the commands you want by either clicking and dragging over those commands, or select the entire program file by:

2. select Edit/Select All in the Editor

To run the program you can click the Submit button, right click and open the pop-up menu in the EDITOR window, or go to the main menu and:

3. select Run/Submit

The program will run and LOG and OUTPUT windows will appear (if there are errors no OUTPUT window will appear).

If there are errors, return to the EDITOR (select Windows/Editor or click anywhere in the EDITOR window), edit the appropriate commands, and resubmit the job.

Note that you do not have to select the text before you use the submit command. If you submit the program without selecting any command specifically, SAS will run the whole program.

You may print the contents of the LOG and OUTPUT window by selecting File/Print from the menu. You may also save the contents in any of the windows by selecting File/Save from the pop-up menu.

Sample Data Sets

So far we have looked at the SAS System to develop a basic idea of how SAS for Windows works. The next step is to examine a few other data analysis techniques that you might employ for your own data analysis. All the statistical procedures available from the SAS System under other operating systems are also available from SAS for Windows. Refer to the SAS documentation for further information.

The CLAS Data Set

The data set we discussed in our earlier example was designed to get you started. A more sophisticated data set and command file with statistical examples is available on the World Wide Web for you to download, peruse, and execute. The program file has many examples of the various procedures and commands discussed previously and you should probably read through this file closely to make sure you understand the syntax. Comments are provided throughout the program file which includes some statistical examples to demonstrate how these procedures work.

In this sample program file, you will read an ASCII data file, clas.dat, created with a word processor and saved as a text file, into the SAS session. The data collected from 40 middle school students contains 28 variables. The first four variables (id, sex, exp, school) are background variables. The variable sex has two levels (M=male, F=female). Exp (prior computer experience) has three levels (1=less than one year, 2=1-2 years, 3=more than 2 years), school (type of school system) has three levels (1=rural school, 2=suburban school, 3=urban school). The next 20 variables (C1...C10, M1...M10) are Likert type responses to a computer opinion survey, and a mathematics anxiety survey. The last four variables (mathscor, compscor, mattati, compopi) are scores on the math test, computer test, math anxiety score, and computer opinion survey score.

To obtain a copy of the sample files:

1. Launch a Web browser (e.g. Internet Explorer or Mozilla Firefox)
2. Go to the URL: http://www.indiana.edu/~statmath/stat/sas/clas_win.sas.
3. Save this as a text file (e.g. in Mozilla Firefox, right click on clas.dat and Save Link As...) to a flash drive or C:\TEMP\ directory.
4. Repeat for http://www.indiana.edu/~statmath/stat/sas/clas.dat.

To run the sample program file:

1. Launch SAS for Windows
2. Retrieve the program file by selecting File/Open
3. Check to make sure the INFILE command corresponds to the location and name of your data file. For example, if the data file is called clas.dat and it is saved on C:\TEMP directory, the INFILE command should be:

```
INFILE 'C:\TEMP\clas.dat';
```

4. Submit the command file by selecting Run/Submit.

Once you submit the commands, SAS takes you to the output window where you can review your results. You can scroll through the output by either using the PageUp or PageDown keys or the scroll bar. To print the results, click the printer icon on the toolbar.

Other Sample Files

A number of sample files are available for each of the SAS add-on modules. If you are working in a UITS Student Technology Centers, you may access the sample files during a SAS session by specifying the path C:\Program Files\SAS\SAS 9.1\core\sample (replace "core" with "stats" if you want to access SAS/STAT samples). You can also copy the files to your disk.

IU's involvement in the TeraGrid, and the presentation of this material is based upon work supported by the National Science Foundation under Grants No. 0833618, SCI451237, SCI535258, and SCI504075. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

SAS Data Sets

Creating a SAS data set

If you are handling a large data file (large numbers of cases/variables) it is advisable to create a SAS data set and work with that. A SAS data set is a specially structured files readable only by the SAS System from the operating system where the file was created. A stored SAS file cannot be edited. SAS data sets are referenced with a one- or two-level name. The two-level name is of the form libref.member-name, where libref refers to the directory in which the data set is to be stored or read, and member-name refers to the name of the SAS data file to be created or read. If a two-level name is not used, SAS stores files in a temporary work library that is deleted when you exit SAS.

Suppose you wished to read in an external file, and store the data in a permanent SAS data set. The program file you would create would look something like the one below, which is a subset of the sample data and program files described previously. Note the first two lines in the following program where the libname command is issued to reference the directory in which the SAS file is to be created, and the two-level name links the member-name (e.g., anxiety) to the libref (e.g., try1).

```
LIBNAME try1 'C:\TEMP\';
DATA try1.anxiety;
INFILE 'C:\TEMP\clas.dat';
   INPUT ID 1-2 SEX $ 3 EXP 4 SCHOOL 5
   (C1-C10) (1.) (M1-M10) (1.) (MATHSCOR COMPSCOR) (2.);

IF MATHSCOR=99 THEN MATHSCOR=.;
IF COMPSCOR=99 THEN COMPSCOR=.;

C3=6-C3; C5=6-C5; C6=6-C6; C10=6-C10;
M3=6-M3; M7=6-M7; M8=6-M8; M9=6-M9;
COMPOPI = SUM(OF C1-C10);
```

```
MATHATTI = M1+M2+M3+M4+M5+M6+M7+M8+M9+M10;
LABEL ID='STUDENT IDENTIFICATION' SEX='STUDENT GENDER'
  EXP='YRS OF COMP EXPERIENCE' SCHOOL='SCHOOL REPRESENTING'
  MATHSCOR='SCORE IN MATHEMATICS'
  COMPSCOR='SCORE IN COMPUTER SCIENCE'
  COMPOPI='TOTAL FOR COMP SURVEY'
  MATHATTI='TOTAL FOR MATHATTI SCALE';

RUN;
```

Replace C:\TEMP\ with the appropriate directory. When the job is executed a SAS data set named anxiety.sas7bdat will be stored in the directory assigned.

Accessing a SAS data set

To read an existing SAS data set, use a two-level name of the form libref.member-name. The following example illustrates how to access the SAS data set (e.g., anxiety.sas7bdat) created and run some SAS procedures with it. (Note that try2 is given as libref and anxiety is given as member-name. The SET command used below reads the SAS data set called try2.anxiety into the data area called test.

```
LIBNAME try2 'C:\TEMP\';
DATA TEST;
  SET try2.anxiety;

PROC TTEST;
  CLASS SEX;
  VAR COMPOPI;
  TITLE 'T-TEST';
PROC CORR;
  VAR COMPSCOR MATHSCOR COMPOPI MATHATTI;
```

It is also possible to retrieve only a subset of the original data using an IF statement. For example, if you wanted to retrieve only the female respondents from the anxiety.ssd file, the DATA step of your program would look something like this:

```
LIBNAME try2 'C:\TEMP\';
DATA TEST;
  SET try2.anxiety;
  IF sex='F';
RUN;
```

The IF statement in this example tells SAS to only read in those observations where the character variable SEX is equal to "f." All other cases will be ignored. Several conditions can be set with an IF statement.

SAS transport libraries

SAS also handles transport format data sets. You create a transport format file when you want to move your SAS data set to another operating system (e.g., UNIX). Also, if you are bringing SAS data sets from another operating system into SAS for Windows, you must first save the file in transport format.

Suppose you want to create a SAS transport format file from the SAS data file, anxiety.sas7bdat. Define a libname to read the SAS data set, and another libname to write a SAS transport file. The XPORT (for transport engine) parameter is used to indicate that you want to create a transport format file. A transport format file always has a fixed block size with a record length of 80, and a block size of 8000. The select statement may be omitted if there is only one SAS file stored in the directory or if you want to convert all the members of a single SAS data library into SAS transport format file.

```
LIBNAME test1 'C:\TEMP\';
LIBNAME test2 XPORT 'C:\TEMP\trans.dat';
PROC COPY IN=test1 OUT=test2;
  SELECT anxiety;
RUN;
```

Once the job is executed, a file, trans.dat, will be created and stored in the directory specified.

To read a transport format file (e.g., trans.dat) stored on the disk and create a SAS data file, as in the example given above, define two libnames (one for reading, and one for writing) as in:

```
LIBNAME test1 'C:\TEMP\';
LIBNAME test2 XPORT 'C:\TEMP\trans.dat';
PROC COPY IN=test2 OUT=test1;
  SELECT anxiety;
RUN;
```

The select statement may be dropped if you want to convert all the members of the transport library into a SAS file, or if there is only one member in the specified transport library. If you want to read/write SAS transport files involving format library use the CIMPORT/CPORT procedures instead of the COPY procedure.

Using SAS/GRAPH

SAS has excellent graphics capabilities. To make a hard copy of the graphics output you may create a PDF format file and print it using the printers at various Student Technology Centers.

To create a PDF file, type the command lines:

```
FILENAME stat2 'C:\TEMP\graph1.pdf';
  GOPTIONS DEV=PDF GSFNAME=stat2 GSFLEN=132 GSFMODE=REPLACE;
RUN;
```

If you need to produce other format of graphs (e.g. GIF or JPEG format), type the followig command:

```
PROC GDEVICE;
RUN;
```

Further Help

The material covered in this document illustrates some of the basic features of SAS for Windows. For further help refer to SAS for Windows documents. SAS for Windows is currently available for public access from all UITS public computing facilities. If you need help in using SAS for Windows, contact the UITS Stat/Math Center. You may also refer to the SAS Institute's website, http://www.sas.com, for further information.