Introduction

SAS is a statistical programming package that is extremely useful and recognized worldwide. The entire SAS System is quite large, consisting of many, many different components for statistical analysis, data management, and graphics. An introduction into SAS for these purposes is presented in this course.

SAS users do not have to be programmers. Although most people recognize the SAS programming is done in data analysis or for other purposes. There are point-and-click approaches in SAS to analysis and statistical graphics. Within SAS one could use Enterprise Guide. JMP (pronounced "jump") is a point-and-click software produced by SAS.

This course primarily addresses writing programs in SAS. The following rules apply to interpreting the SAS programming code printed. These rules are also consistent with the SAS literature. Any item that is typed in UPPER CASE is a SAS statement and must be spelled exactly as it appears. Items in lower case or *lower case italics* are items you can define, such as variable names or optional text. When you type your programs, it is not necessary to use upper and lower case letters as is done here in the examples. The two cases are used here to distinguish between SAS commands and user-defined text. The punctuation in SAS is a semicolon (;). It is necessary at the end of each SAS statement in order for your program to execute. Omitting a semicolon causes run-on statements in the same way the omitting a period does when writing a composition.

When this text refers to items on the screen that can be selected using the computer's mouse, the item will appear in boldface print. For example: To save a program, select the **File – Save As** sequence from the pull-down menus at the top of the screen. This boldface notation will be used in the chapters presenting the point-and-click procedures.

Programming in SAS

A typical SAS program consists of a data manipulation portion and an analysis. For the data manipulation, the DATA step is used. Using the DATA step, data sets can be created or "read" into the program, and variables in the data set can be modified. Once a data set is recognized by SAS then analysis procedures can be performed on the data set. The procedures are identified by SAS as "PROC's." A SAS program would generally include at least one DATA step and one or more procedures. In this text you will learn some of the basic statistical procedures (or PROC's).

Important in SAS programming is the inclusion of RUN and QUIT statements. After DATA steps or PROC's a RUN statement can be included, or a single RUN statement can appear at the end of the SAS procedures. The last line of a SAS program is the QUIT statement.

Data Step Information I - Creating a Data table Within Your Program Using the INPUT Statement

A data table may be created within your SAS program. The simplest way to do this is:

```
DATA tablename;
INPUT variable1 variable2 variable3;
DATALINES;
```

Any item typed in italics indicates variable or table names that are up to you to define. When selecting names for these items, the first character must be alphabetic. In earlier versions of SAS, the name could be no longer than 8 characters, but currently you can use longer names. The names **cannot** contain spaces. A single space is necessary to separate the items in each line of the program, such as variable names and data.

The following rules apply to interpreting the SAS code printed in this class. These rules are also consistent with the SAS literature. Any item that is typed in UPPER CASE is a SAS statement and must be spelled exactly as it appears. Items in lower case or *lower case italics* are items you can define as you prefer. When you type your programs, it is not necessary to use upper and lower case letters as is done here in the examples. The two cases are used here to distinguish between SAS commands and variable names. The punctuation in SAS is a semicolon (;). It is necessary at the end of each SAS statement in order for your program to execute.

tablename - You should get in the habit of naming each data table you work with in your SAS programs. Although naming the data table is optional, an unnamed data table can cause problems in debugging programs that are more complicated. The DATA step is where you indicate that you are creating a data table that is to be analyzed. This line must end in a semicolon.

variable1, etc. - The INPUT statement follows the DATA statement. In the INPUT statement you must indicate each variable that is in your data table. The order you list the variables on the INPUT line must be the same order that you enter information in the data lines following the DATALINES or CARDS statement. Each variable name corresponds to a column of information. This line must end in a semicolon. This illustration shows only three variables being "read" into the data table. Many more variables could be read into your data table.

Variables can be either numeric (prefix + and – signs allowed) or character (string) variables. You should identify character variables on the INPUT line of your program by following the character variable name with a space and a dollar sign. The INPUT line is the only place where you will need to specify the dollar sign for the character variables.

Example 1

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
```

The data follow the DATALINES statement. The DATALINES statement is necessary in order for your program to read in any data. NOTE: There are no semicolons at the ends of the lines of the data, and spaces separate the various pieces of data. Earlier versions of SAS used a CARDS statement rather than the DATALINES statement. The CARDS statement will still work as shown below.

```
DATA  test;
INPUT  name $ id score grade $ ;
CARDS;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
```

When an INPUT statement of the previous form is used, one observation per line must appear in each line after the DATALINES or CARDS statement. At least one space is needed between pieces of data. Do not use any punctuation to separate pieces of data.

You can also specify which column(s) the responses to each variable appears. The previous example can be rewritten as:

Example 2

```
DATA  test;
INPUT  name $ 1-5 id 9-17 score 21-22 grade $ 24 ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
```

```
or

DATA  test;
INPUT  name $ 1-5 id 6-14 score 15-16 grade $ 17;
DATALINES;
Larry00000012385B
Curly00000023496A
Moe  00000034565C
Huey 00000045680B
Dewey00000056797A
Louie00000078981B
```

NOTE: When column formatting is specified in the input statement, it is not necessary to leave spaces between pieces of information in the data lines.

If there are a number of variables it may become necessary to use more than one line for each observation. Although there are not a large number of variables in the current example, this example can be used to illustrate the concept. #n identifies line n of the data record. In this example, line 1 contains the variables name and id, and line 2 contains score and grade.

Example 3

```
DATA  test;
INPUT #1 name $ 1-5 id 9-17
      #2 score 1-2 grade $ 4 ;
DATALINES;
Larry   000000123
85 B
Curly   000000234
96 A
Moe     000000345
65 C
Huey    000000456
80 B
Dewey   000000567
97 A
Louie   000000789
81 B
```

When you have only a few variables, you can put several observations on one data line. You indicate that you have done this using the @@ at the end of your INPUT line and before the semicolon.

Example 4

```
DATA  test;
INPUT  name $ id score grade $ @@;
DATALINES;
Larry 000000123 85 B Curly 000000234 96 A Moe 000000345 65 C
Huey    000000456   80 B Dewey   000000567   97 A Louie   000000789   81 B
```

If you do not specify @@ on the INPUT line, only the first four pieces of information will be read into your data table. When using the @@ convention, you cannot use column formatting.

MISSING DATA

When entering data, if the response to a variable is missing or unknown, you can enter the other pieces of information in the data lines and put a single period (.) in the position of the missing observation.

Example 5

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 .
.       000000234   96 A
Moe     000000345   65 C
Huey    000000456   .  .
Dewey   000000567   97 A
Louie   000000789   .  B
;
```

It is sometimes reported that a semicolon must appear on a line by itself after the last line of data. This is not necessary for SAS to read in all of your data. SAS will correctly read in the data whether you decide to use the semicolon or not.

Also, data **are** case sensitive. If you enter the grade of 'a' and the grade of 'A', these will be regarded differently. So while case sensitivity does not matter with respect to the SAS code, it **does** matter in the data.

These are just a few of the simpler ways that you can create a data table in SAS. Later, you will learn how to import an external data file into SAS.

When programmers and SAS literature refer to a DATA step, the reference is to the DATA statement and the lines following it. The INPUT line or DATALINES by themselves do not have any utility. They are a part of the DATA step operation. Other DATA step operations will be covered later.

SAS programs can be written that create data sets only. Other programs may create data sets and include some analyses of that data. Later, there is more discussion on why a program would be written to create a data set only. A DATA step operation in a SAS program can be closed with a RUN statement. This can be done whether there are analyses steps in the program or not. For example,

```
DATA one;   INPUT x y z; DATALINES;
. . .
;
RUN;
```

Printing a Data Set

In addition to proofreading your SAS program, one of the simplest ways to examine the information in a data set for errors is to have SAS print the information. The SAS/PRINT procedure can be used to do this.

The syntax of the print procedure is:

```
PROC PRINT  <options> ;
VAR variablelist ;       (optional)
ID variablelist ;        (optional)
BY variablelist ;        (optional)
RUN;
```

PROC PRINT statement options are:

DATA =*tablename*	names the SAS data set to print
DOUBLE	double-spaces the printed LISTING output
NOOBS	suppresses the observation number in the output
UNIFORM	formats all pages uniformly
LABEL	uses variables' labels as column headings
SPLIT=	splits labels used as column headings across multiple lines
N	prints the number of observations in the data set, in each BY group, or both
ROUND	rounds values to the number of decimal places specified for a variable in a FORMAT statement or, if you do not specify a format, to two decimal places

VAR *variablelist:*

The VAR statement allows you to specify which variables you want printed. If the VAR statement is not used, all of the variables in the data set will be printed.

ID *variablelist;*

When the data set is printed, the variables or variables in the ID statement will be printed at the left of the printout. If one variable in the data set serves as an identifier, it is important to list it in the ID statement.

BY *variablelist;*

If one or more variables is used to define subgroups within the data set, these variables should be listed on the BY statement. In order for the BY statement to function properly, the data set must first be sorted by the variables in the BY statement. See the SORT procedure.

SAS has many procedures, and these are referred to as PROC's. So, PROC PRINT is the PRINT procedure. Each SAS procedure requires a RUN statement at the end of the procedure. For programs that contain several procedures, a single RUN statement at the end of the program is all that is necessary; however, some programmers use a RUN statement at the end of each DATA

step operation and each procedure step. At the end of a SAS program, a QUIT statement should always be included.
In this text you will find objectives or demonstrations. These are activities that you should do in SAS and observe the log window and output or Results Viewer windows for your results. These exercises are intended to demonstrate some of the capabilities of the current SAS topic.

Objective 1: Create a data set containing the names, identification numbers, test scores and grades for a class of six students using **Example 1** for the data set creation. Print the resulting data set.

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
;
PROC PRINT DATA=test;
RUN;
QUIT;
```

In this illustration, the input data set is to be printed. When a data set is printed, SAS will assign an observation number to each line of the data. This observation number is given the variable name OBS by SAS. It is not necessary for you to specify DATA=*tablename*. If you do not, the SAS procedure will operate on the most recently created data set. *Option:* You can use RUN; on the line after the data rather than the lone semicolon.

Modification: Run this objective using **Examples 2 - 5** previously given for the DATA step.

You can suppress the printing of an observation number by using an option on the PROC PRINT line. Using the coding below, the observation numbers will not appear when the data are printed and the NOOBS option below is specified.

Objective 2: Create the data set as in Objective 1. Print the data set without the SAS-defined observation numbers.

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
```

```
Dewey   000000567   97 A
Louie   000000789   81 B
;
PROC PRINT DATA=test NOOBS;
RUN;
QUIT;
```

When you have a large number of variables (or columns) in a data table and only wish to print some of the variables, you can specify which variables are to be printed and the order in which the variables will be printed using a VAR statement. If you do not use a VAR statement, all variables in the data table will be printed.

Objective 3: Create the data set as in Objective 1, but only print the students' names and letter grades. Double-space the LISTING output.

```
ODS LISTING ;
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
;
PROC PRINT DATA=test DOUBLE;
VAR name grade;
RUN;
QUIT;
```

TITLE statement

The TITLE statement can be used with almost all SAS procedures. The function of the TITLE statement is to label the top of each page of output with a title that you select. The syntax of the TITLE statement is:

TITLE " text for title here " ;

You may use either a pair of double quotes (") or a pair of single quotes (') to designate the text of the title. Whichever symbol you use, be certain that the quotation is closed. Unbalanced quotation marks in your programs can render your entire program unexecutable. For the text of a title, you can use any letters, numbers, or symbols you like. SAS can distinguish an apostrophe (') as in can't if the text of the title is enclosed in double quotes ("), such as, TITLE "   can't " ;

Using the Enhanced Editor, you can detect unbalanced quotation marks. Text for titles should appear in a different color on screen. If several lines of your program appear in the same color as the title quotations, then you need to correct the punctuation in the program.

This title will appear on every page of output that your program generates. You may have up to ten lines of title on every page. TITLE and TITLE1 both specify the first line of the title. For more than one line in the title, you must type TITLEn where n is a number from 1 to 10. The text you specify in the TITLE statement is centered at the top of the page of output.

Objective 4: Create the data set as in Objective 1. Print the contents of the data set with two lines of titles. Double space the lines in the title.

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
PROC PRINT DATA=test;
TITLE 'Introductory Example';
TITLE3 'Test Scores';
RUN;
QUIT;
```

Notice that only the first and third lines of title are used to achieve the effect of double spacing in the titles. It's OK to skip one or more lines in the titles. To keep the titles single spaced, change TITLE3 to TITLE2 in the above program.

As your programs get larger and involve more procedures (PROC's), the titles can change with each procedure. Simply type the new TITLE statements in the following procedures.

If you wish to clear all lines of titles at some point in your program, you can include

TITLE;

in your program. This erases all lines from the titles. It is important to point out that once you create a title during a SAS session, that title stays active until you change it or delete it. Therefore, a title created in an earlier program can appear in the output of a current program, if the current program does not change it. If you are running several programs in one SAS session, TITLE; is a useful tool. If you place TITLE; at the top of your new program, all lines of titles generated by previous programs will be deleted.

TITLEn;
erases all lines of titles from the nth to the tenth. (Specify n.)

FOOTNOTE statement

Similar to the TITLE statement is the FOOTNOTE statement where FOOTNOTE adds lines of text to the bottom of the pages of your SAS output. There are ten lines of footnotes possible. The syntax of the FOOTNOTE statement is:

```
FOOTNOTEn  " insert footnote text here";
```

where n can be a number from 1 to 10, and either double (") or single (') quotes can be used to define the text of the footnote. The Enhanced Editor can again help you detect typographical errors in your footnotes.

Like the titles, footnotes carryover from one program to the next during the same SAS session. To clear all footnotes, include

```
FOOTNOTE;
```

in your program. (Or if you wish to remove the footnote lines from n and higher, FOOTNOTEn; where n is a number from 1 to 10.

Sorting a Data Table

After a data table is created, the data is kept in the same order as it was entered. This is evident if you create a data table and then print the data table. The data can be sorted by a specific variable in either ascending or descending order. One can also sort data using more than one variable to determine the order. The SORT procedure performs these operations. The syntax of the SORT procedure is:

```
PROC SORT  DATA=setname;
BY   variable1  variable2 ... variablen;
RUN;
```

The data table would first be placed in ascending order according to variable1. Within each level of variable1, the data is sorted according to variable 2 and so on. The SORT procedure **does not** produce any information in the output window. To view the results in the output window you can add a PRINT procedure after the SORT procedure.

Objective 5: Sort the test data by putting the student names in alphabetical order. Print only the students' names and grades.

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
;
PROC SORT DATA=test;  BY  name;
PROC PRINT DATA=test;
VAR name grade;
RUN;
QUIT;
```

The above program would produce the following output:

```
OBS   NAME    GRADE
 1    Curly     A
 2    Dewey     A
 3    Huey      B
 4    Larry     B
 5    Louie     B
 6    Moe       C
```

By default, the SORT procedure sorts items in ascending order: 1 2 3 ... or A B C ... . This may be changed in the program. If you wish to place data in descending order according to a specified variable, type DESCENDING prior to the variable in the BY statement.

Objective 6: Sort the data so that the lowest grades are listed first. For grades that are the same, list the students' names in alphabetical order. Print only the students' names and grades. Then print the students' names and grades using a BY statement on the PRINT procedure.

```
DATA  test;
INPUT  name $ id score grade $ ;
DATALINES;
Larry   000000123   85 B
Curly   000000234   96 A
Moe     000000345   65 C
Huey    000000456   80 B
Dewey   000000567   97 A
Louie   000000789   81 B
;
PROC SORT DATA=test;  BY  DESCENDING grade name;
PROC PRINT DATA=test;
VAR name grade;
PROC PRINT DATA=test; BY DESCENDING grade name;
VAR name grade;
PROC PRINT DATA=test; BY DESCENDING grade;
VAR name grade;
RUN;
QUIT;
```

The above program would produce the following output for the first PRINT procedure:

```
OBS    NAME    GRADE
 1      Moe      C
 2      Huey     B
 3      Louie    B
 4      Larry    B
 5      Curly    A
 6      Dewey    A
```

Note that the grades are in descending order and the names are in ascending order within each value of grade. Notice how the BY statement is used on the second and third PRINT procedures in the program. Observe the output for the three different formats that are printed.

The DESCENDING option is necessary in the BY statements of the PRINT procedures since the data were sorted in descending order. If the data were sorted in ascending order, all of the occurrences of the DESCENDING option would not be necessary in the program.

The SORT procedure is useful in many ways. Often calculations (rather than just print procedures) are necessary for each level of a particular variable. If the data table is first sorted by a specific variable, many SAS procedures can be invoked which can operate on the sorted data tables.