Introduction to Using SAS on UNIX

The purpose of this document is to explain how to use SAS, a statistical software package, on systems running UNIX

Original Author: Carolynne White

Table of Contents

Terminology

In order to learn about using the SAS System, you must first understand a few of the basic definitions. The tables below provide a brief description of some terms that will be used throughout this lesson.

Data Terminology

Data	Pieces of information like city, state, or population
Raw	Unformatted Data
Data Value	A single value, such as the name of a city
Missing Value	When data is not available for a particular variable in an observation, or illegal characters have been entered, the value is considered missing and will not ordinarily be used in calculations performed by PROC statements.
Observation	A set of data describing a single occurrence of a set of variables. For example, it could be population and location information on a single city, or name, address, and grade information on a single student.
SAS Data Set	A collection of all the data and formatting information with which you are currently working.
Variable Name	A name used to refer to a category of data values, it usually is a descriptive term relating to those values.
Variable	A set of data values for the same measurement. For example, it could refer to the populations of cities or addresses of students. The value of a variable in a single observation is a measurable aspect of the phenomena under observation. For example, it could refer to the population of a specific city or address of a student.

Program Terminology

Data Step	Commands used to create and manipulate a data set
Procedure Step	Commands indicating algorithms or transformation applied to the data. Many procedures are available in SAS. Specify the desired procedure by use of the appropriate PROC statement.

File Terminology

Program File	Set of instructions to SAS.
Listing/Output File	This screen displays the output that you requested by PROC steps. This could be a listing of a data set, the calculated means of your variables, or a myriad of other possibilities.
Log File	Displays your SAS statements as the computer encountered them as well as any error messages.

SAS Software

SAS is a computer programming language that can easily manipulate both small and large sets of data. Commands are available that allow you to use or view all or part of your data set in a variety of ways. Data can be reshaped, merged, redefined, updated, edited, and analyzed using a variety of procedures. Elaborate reports and forms can be generated and data may be graphed. These characteristics make SAS one of the most popular statistical software packages available.

The Data

Data is the key to any analysis. Both text and numerical data can be used. SAS data must be well defined and entered in a consistent manner. Depending on the method of presenting the data, some flexibility is allowed, but each input mode has its own characteristics which must be adhered to. Not only is there more than one method of presentation, but data may be used from a separate file or be entered within the program when only a few observations are necessary. In most cases, a file is used. In either case, data may be saved as a different file by SAS and used over and over again. SAS can also use data files created by other software.

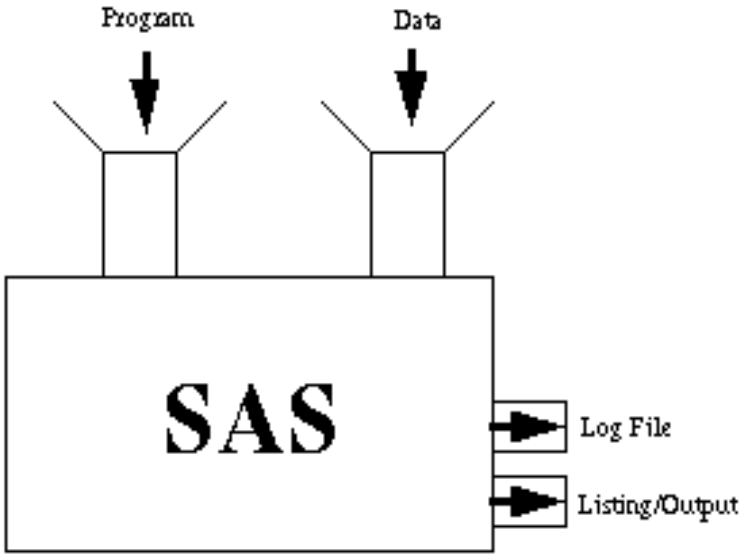

FIGURE 1. How SAS works

Observations and Variables

An easy way to visualize data is as a chart or table of information with the data organized by columns and rows. The whole chart is called a data set. Each column heading is a **variable**, while each row of data is an **observation**. Looking at the example below, each observation is composed of entries in the following fields: Last name, Initial, Sex, Level, Grade1, Grade2, and Grade3.

Jones A m 3 88 79 89

Smith K f 2 92 85 74

Jackson B m 3 95 82 88

Abrams D m 4 78 85 92

The DATA Step

Each SAS procedure requires you to indicate the data set to be used for that process. Data may be either text or numerical. Text are characters, numbers, and symbols. Numerical data are numbers that will be used mathematically. A data set is named by the user with the DATA statement. Data is then entered as a part of the program using the CARDS statement or as a file using the INFILE statement. SAS reads each line of your input and writes it to a SAS data set as an observation. Entering RUN; at the end of

your DATA step establishes a block terminator after that section and is good programming technique.

Using the CARDS Statement

Data that is included as part of the program is entered immediately after the statement CARDS;. Each line represents an observation. However, using CARDS is not an efficient way to enter data unless it is a very small data set.

Data or Variable names

Variable Name Restrictions

Variable names must be one to eight characters (letters, numbers, or underscore) beginning with a letter or underscore. If the name represents text, the name must be followed by a space and a dollar sign ($).

```
name $
```

Data Restrictions

- Text - up to 250 characters (letters, numbers, spaces, punctuation)
- Numerical - Stored as floating point double precision numbers. Without special formatting, numbers may not include dollar signs ($), commas (,), blanks, and trailing plus (+) or minus (-) signs. Leading plus (+) and minus (-) signs are fine.

Data formats

The INPUT statement describes the data lines. The following data formats, list and column, can be used to read data with the INPUT statement from either CARDS or from a file that was previously created. If an INPUT statement is too long for one line, continue on the next card without splitting in the middle of a variable name.

List Input

Exercise 1: Practice

Place the cursor on line 1 of the Program Editor. Type the following data step, pressing enter after each complete line. Select Submit under the Locals menu of the Program Editor.

SAS statements must end with a semicolon (;) and can continue on another line.

```
DATA class;

INPUT name $ initial $ sex $ year grade1 grade2 grade3;

CARDS;

Jones A m 3 88 79 88

Smith K f 2 92 85 92

Jackson B m 3 95 . 82

Abrams D m 4 78 85 88

RUN;
```

Notice the format of the INPUT statement. This program is entering data in the LIST format. SAS scans the first card for the first nonblank column, reads the value through the next blank, and assigns it to the first variable name. The next nonblank column is the beginning of the next value and so on until the card is finished.

- Each variable name is separated by a blank from the name preceding it.
- Every field in the data must be named in the input statement because there is a one to one match as it reads the data.
- All character data must be included.
- If numerical data is missing for a single variable, a period (.) must be included as a place holder.
- Character field values have a limit of eight characters in List input format and may not contain blanks.
- If each observation requires more than one card for input, the number sign (#) can be used to move to another card. The same method can be used to skip lines of data.

```
INPUT  a   b   $   #2 c   d;

INPUT  a   b   $   #2 c   d  #4;
```

Column Input

```
DATA class;

INPUT name $ 1-10 initial $ 12 sex $ 14 year 16 grade1 18-19 grade2 21-22 grade3
24-25;

CARDS;

Jones    A  m  3   88 79 83

Smith    K  f   2   92 85 76

Jackson B   m  3   95     89

Abrams  D   m  4   78 85 78
```

```
RUN;
```

Notice the format of the INPUT statement. The beginning column for each variable is listed, followed by a dash (-) and then the ending column.

- A column range is defined by the lower column number followed by the higher.
- Any column range can be a variable.
- Any part of an observation may be used, reused, or skipped.
- Portions of a variable may be reread and assigned to different variable names.
- If a field is blank, SAS reads it as missing data.
- Individual character values can be up to 250 characters long and can contain blanks.
- A period (.) may be entered to indicate that a field is blank.
- If each observation requires more than one card for input, the number sign (#) can be used to move to another card.

Naming Data Sets

When you create a data set, you designate the name with the statement DATA scores;. Data set names have similar restrictions to variable names: one to eight characters (letters, numbers, or underlines) beginning with a letter or underline. If a data set is named *scores*, it is actually called *work.scores* by SAS. This is an example of SAS's use of a two-level naming system for SAS data sets. SAS stores data sets in groups called *libraries*. Each library usually contains data sets that have related information although the data sets do not have to be related in any way. The second level is the actual name of the data set. If you do not specify a first level for a data set name, SAS will always assume that *work* is the first level. The work library is a special library for temporary data sets. Work data sets are only available for use for the duration of your current SAS session. When you leave SAS, the work data sets are deleted. You must recreate them to use them in another session. Later we will see how to create your own libraries so that data sets can be stored permanently on your account. To specify a data set for use in a PROC statement, simply add **DATA=*scores*;** (or whatever the data set name is) to the PROC statement.

Data Step Summary (Note the order)

Cards	File
DATA statement;	DATA statement;
INPUT statement;	INFILE statement;
CARDS (if needed);	INPUT statement;

SAS statements must end with a semicolon (;) and can continue on another line.

Data Step Processing

When a Data Step is submitted to SAS for processing, two steps take place. The first of these is compilation. In this stage, each statement is examined for syntax errors including punctuation and key word spelling. SAS checks for resources to find the files that you include in your program. The last part of compilation is to set up the overall logic or flow of the data step. When compilation is complete, the program actually runs.

Interactive SAS

Now that you have learned how to structure data sets for use with SAS, you can use SAS to manipulate that data by writing programs that perform procedures on previously entered datasets.

Starting Interactive SAS

SAS can be run in graphical interactive mode or text interactive mode.

Starting SAS in Graphical Interactive Mode

To use the graphical interface you must be logged into vet using the X-windows system. This could be MacX on the Macintosh, MicroX or PC Xview on the PC or a UNIX box in one of the campus labs. If you are using a telnet or terminal connection to vet you must run SAS in text interactive mode.

Interactive SAS gives you three windows. All three windows appear on your screen although one of them may be covered by the other two. You should at least see the Program Editor and Log windows. The Output window may be hidden by the other two. To bring up graphical interactive SAS,

- Login to your vet account
- Enter **sas &** at the command prompt

Shortly, the three windows will appear. At the top of each window is a menu bar containing several menus. An important thing to remember is that you choose Exit under the File menu on either the Program Editor or Log window to quit SAS.

Starting SAS in Text Interactive Mode

Text interactive mode provides a windowing system that will allow you to use SAS interactively from a terminal or telnet connection. To use SAS in text interactive mode

- Login to your vet account
- Enter **sas -dms -fsdevice ascii** at the command prompt

You will have a Program Editor window and a Log window upon execution of the above. The prompt will be in the Program Editor window. This windowing system will be familiar to you if you previously used SAS on the RiceVM1 mainframe.

To get a menu similar to that of graphical interactive SAS type “pmenu” at the Program Editor prompt.

```
Command ===> pmenu
```

This will load a pull-down menu at the top of each window. You can tab between options and hit [Enter] to pull down the menu. Use arrow keys and [Enter] to select an option.

When you are ready to quit interactive sas type “endsas” at any command prompt

```
Command ===> endsas
```

or if you are using the pmenu system Tab to File in the pull-down menu, hit [Enter], arrow to Exit and hit [Enter]. Press [Enter] again when SAS asks if you want to end your SAS session.

Running a SAS Program

- Enter the SAS statements
- Submit the program
- Check your output
- Edit and resubmit as needed
- Select Exit from File menu to quit SAS

Recalling Previously Submitted Programs

To recall your previously submitted program, select Recall Text under the Locals menu of the Program Editor.

Editing in Interactive Mode

Edit your program to include two more lines. You can use the following commands in the numeric prefix area of the Program Editor.

A, B	Mark the target position of a copy or move. B= before; A= after
C, CC	Copy a single line or a block or lines. For a single line, place a C in the prefix area. For a block, place a CC on the first statement’s prefix and another CC on the last
D, DD	Delete a single line or a block of lines. For a single line, place a D in the prefix area. For a block, place a DD on the first statement’s prefix and another DD on the last.

M, MM	Move a single line or a block of lines. For a single line, place an M in the prefix area. For a block, place an MM on the first statement's prefix and another MM on the last.
I[n]	Inserts a line (or lines) immediately following the marked line.
IB[n]	Inserts a line (or lines) immediately preceding the marked line.
R, RR[n]	Repeat a single line or a block of lines a specified number of times (default is 1). For a single line, place an R in the prefix area. For a block, place a RR on the first statement's prefix and another RR on the last.

Saving a Program in Interactive Mode

Save the program by selecting the Save As menu under the File menu and then selecting the Write to File option. A window will appear that will allow you to select the directory the file is to be stored in as well as the name of the file to be created. When you have entered the correct filename, click on the OK button. Use the Filter button to change directories to your target directory. You can also save the listing in the Log window by using the same process with the File menu on the Log window.

Using a Pre-existing Program File in Interactive Mode

You can recall a previously existing program by selecting the Read File option on the Open menu which is under the File menu. This must be done from the Program Editor window. When you select this menu item, the now familiar window will appear that will allow you to select a file from any directory.

Printing a Window

The easiest way to print the contents of the SAS Program Editor, Log, or Output window is to save the contents of the window to a file and then use the UNIX printing commands to print the file. Printing a graph window takes some special setup. Call the Consulting Center (527-4983) to speak to the SAS support person for information on how to set up your account to print graphs.

Using a Pre-existing Data File as Input

Data can be used in file format either as a raw data file or as a permanent SAS data set.

Reading Data Using INFILE

In SAS, you must use the INFILE statement to specify which file you wish to read data from. The most common use of the INFILE statement is to have it specify a file reference, or *fileref*. To allocate a fileref,

you must use the FILENAME statement. An example of using the FILENAME and INFILE statements appears below.

FILENAME *fileref* [**DISK**] '*pathname*';

DATA *data set name*;

INFILE *fileref*;

INPUT ...;

The *fileref* can be any legal SAS name as described previously. Also, the DISK keyword is optional because it is the default. There are other keywords available. See the SAS manuals for details. The *pathname* is the UNIX pathname of the file to be accessed. The pathname can be an absolute pathname or a reference pathname where the reference is to the current SAS directory. You may also use the tilde (~) operator in the pathname. Once the FILENAME statement has been submitted, that fileref is available for use for the rest of the SAS session until either the fileref is explicitly cleared or SAS is exited. To clear a fileref, use the following statement:

FILENAME *fileref* **CLEAR**;

To clear all existing filerefs, submit the following statement:

FILENAME _ALL_ CLEAR;

To display a window showing all existing filerefs and the files that they are allocated to, select the Filename List option under the Data Management menu which is under the Globals menu. To exit the Filename window, select End under the Edit menu on the Filename window.

Permanent SAS data sets

You can create a permanent SAS data file on your account. As mentioned before, SAS stores data sets in libraries. So far we have only used the work library, which is deleted at the end of the SAS session. To create permanent SAS data sets, you must create a library on your account. This is actually very simple because in SAS on UNIX, a library is just a directory. Any directory can be a library. As an example, the work library that we have been using is a directory that SAS creates in the **/tmp** directory. When you exit SAS, it deletes this directory. To see this, issue an **ls -l /tmp** at your UNIX command prompt. You should see a directory whose name begins with SAS that is allocated to your userid. Any work data sets that you create during your session are stored here until the end of your session. You may be wondering why we explained that the names of the work data sets where things like *work.scores* instead of using the complicated names that appear in the /tmp directory. This is because in the two level data set names, the first level is actually a library reference, or *libref*, just like the filerefs described above. Therefore, the two level names are actually in the form *libref.datasetname.* When you start up SAS, it creates the appropriate directory in /tmp and allocates the libref *work* to that directory. To create your own libraries, simply make a directory on your account then use the LIBNAME statement to allocate a libref to that directory. An example appears below.

LIBNAME *libref* '*pathname*';

DATA *libref.datasetname*;

...

The same rules apply to the libref and pathname as applied for the FILENAME statement. Like the filerefs allocated by the FILENAME statement, librefs allocated by the LIBNAME statement remain in effect for the entire session. Likewise, you can have multiple librefs allocated to the same library. Also, the name used for the libref does not have to be the same across SAS sessions. You may use any name. The only thing that remains the same is the pathname of the directory that is the needed library. You can clear librefs and get a listing of currently existing librefs. To clear a libref or all librefs, use one of the following statements.

```
LIBNAME libref CLEAR;

LIBNAME _ALL_ CLEAR;
```

To display a window listing all existing librefs and the directories they are allocated to, select the Libname List option under the Data Management menu which is under the Globals menu. To bring up a window listing all of the SAS files in a library, single click in the space to the left of the libref. To bring up a listing of all of the variables in a particular data set, single click to the left of a data set name. To exit any of these windows, chose the End option under either the Edit or File menus.

Running a SAS Program in Batch Mode

After you have created, debugged, and saved a program in Interactive Mode, you do not have to use Interactive Mode to retrieve and run the program again. You can run a SAS session in the background and tell it to execute the statements in your program file. No windows will appear and SAS will exit once it has finished executing the statements in your program file. You will be notified when it is completed. The log and output/listing of the program will be written to files that you can examine. Note that since SAS exits upon finishing executing your program statements, any librefs, filerefs, and work data sets do not exist and cannot be accessed in another SAS session unless recreated. To run a program in batch mode, issue the following command from your UNIX command prompt:

sas *program_filename* **&**

You may do other things while it is running. A message will appear when it is complete. The log results will be saved in a file called *program_filename***.log** and the output/listing results will be written to a file called *program_filename***.lst**. You can also specify the names of these output files by using the following options:

sas *filename* **-print** *listing_file* **-log** *log_file* **&**

Exercise2: Running a program

First use a UNIX editor or the SAS editor to save the following raw data into a file named *test.txt*. Be careful to keep the data aligned in columns.

```
Smith      14    95
Jones      12    81
Kennedy    13    85
Street     11    76
```

Duncan	11	78
Brown	12	84
Thomas	14	4
Clowes	14	97
Peters	13	89

Next, enter the following program in Interactive mode. Once you have it entered, submit it and look at the results in both the Output and ALog windows. Then clear the Output and Log windows (hint: look under the Edit menu). Now recall the program and try submitting it again, only see if you can submit the program one line at a time (hint: it is under the Locals menu). Watch the sequence of events in the Log window. Also, note how the titles of the window change. Can you figure out when a step is being compiled and when it is being executed? You may have to clear, recall, and submit this program several times to get the hang of it. Once you are done, recall the program once more and save it to a file called *test.pgm*. Then try submitting the program in Batch Mode. Look at the results in the log and listing files. Try giving the log and listing files names other than the default ones. Save this program for later exercises.

```
FILENAME xxx DISK 'test.txt';

DATA scores;

INFILE xxx;

INPUT name $ 1-8 age 10-12 score 14-15;

RUN;

PROC PRINT;

RUN;
```

Creating Subsets of the Data

One of the most popular ways to use SAS is to create data sets that contain subsets of the data in a large data set. These subsets are then used in an analysis or to print a report. There are a number of ways to subset a data set. A few of them are explained below.

To include only some of the variables, use the following example.

```
DATA newlibref.newdatasetname;

SET oldlibref.olddatasetname;

KEEP var1 var6;
```

or

```
DROP var6 var7;
```

You can also select the data by observation number in the SET statement. For example,

```
SET name (obs = 5);
```

Selects the first 5 observations

or

SET *name* **(firstobs = 5 obs = 10);**

Keeps observations 5 through 14

The data may also be screened to retain only those observations meeting a certain criteria. To include only some observations by content,

IF *variable* **EQ** *amount;*

numeric selection

or

IF *variable* **EQ** *'text';*

text selection

Other operations that are available are

Equal	EQ	=
Not Equal	NE	^=
Greater Than	GT	>
Less Than	LT	<
Greater than or equal	GE	>=
Less than or equal	LE	<=

You may also combine conditions.

IF *condition 1* **AND**/**OR** *condition 2*;

Exercise 3: More Practice

Retrieve the program called *test.pgm* that you saved earlier. Add the following lines to the end of the program.

```
DATA subset1;
SET scores (DROP = name);
PROC PRINT;
RUN;
PROC PRINT DATA=scores;
DATA subset2;
SET scores (OBS=5);
PROC PRINT;
```

```
RUN;

DATA subset3;

SET scores;

IF score >= 80;

PROC PRINT;

RUN;
```

Submit the new program and examine the results. Make sure that you understand the flow of the new parts of the program. Submit the program a line at a time if you need to. Next, see if you can get to a window with a listing of all of the data sets that you just created. (Hint: look under the work libref). Also, notice that you have to scroll around in the output window to look at all of the print-outs. SAS has another window called the Output Manager that is designed to simplify this process. To bring it up, select the Output Manager item under the Globals menu. A window will appear that contains a listing of each section of output currently in the Output Window. The listing will contain the name of the procedure that created the output, the page the output begins on, the number of pages consumed by the output, and a title for the output. The default title is The SAS System but this can be set by the user. You can select which section of output you wish to view by entering an S beside the desired output. You can also delete and print output from here.

More on Programming

SAS has many PROCEDURES for data display and analysis. They are usually referred to as PROC and then the name of the requested activity and a semicolon (;). Each PROC is actually a program that has already been written. Each PROC needs to use data and this data must be a SAS data set. You need to specify which SAS data set to use and if there are any options or restrictions on the data set. The lines following the PROC may hold more instructions. Statements all end with a semi-colon (;). More than one statement may be on the same line, but the statements must be separated by at least one semicolon (;).

Documenting Your Program

You can document your program by placing comments in the text. Comments may appear anywhere and are not compiled when the program is submitted. Comments take the form

```
/* comment */
```

Procedures

The basic form of a procedure is

```
PROC xxx DATA=name <OPTIONS>;

control statements;
```

```
RUN;
```

Most common uses of data sets

- Use the most recently created SAS data set
- Use all variables in their given order
- Use the entire data set at once

Things you can specify

- What data set to process
  Add DATA=name

  ```
  Example:  PROC PRINT DATA=old;
  ```
- Which variables to use and in which order
  List the variables with a VAR statement

  ```
  Example:  PROC MEANS;

  VAR age weight;
  ```
- How to process the data set, i.e. subsets
  Use the BY statement with PROC SORT and then PROC xxx

  ```
  PROC SORT;

  BY age;

  PROC MEANS;

  PROC PRINT;
  ```

Examples of Sorting and Using Sorted Data

```
PROC SORT DATA=scores;

BY grade1;
```

- Two Level Sort

  ```
  PROC SORT DATA=pop;

  BY state city;
  ```
- Ascending/Descending

  ```
  PROC SORT DATA=scores;

  BY DESCENDING grade1;
  ```
- Create New Data Set and Retain the Unsorted Data Set

  ```
  PROC SORT DATA=pop OUT=newpop;

  BY state city;
  ```

To Generate Statistics

```
PROC MEANS DATA=scores;
BY grade1;
PROC FREQ DATA=scores;
```

Statement Order

You should note that SAS always executes statements in the order in which they are received. This can sometimes cause results different from what you expected.

```
DATA name1;                    Usually first
/* comment */                 Document your work
INFILE fn;                     Next if pertinent
INPUT var1 var2 var3;          Describe the data format
IF ...;                        Restrict data to be used
RUN;                           Block terminator
DATA name2;                    Establish a second data set
/* comment */                 Comment your work
INPUT var1 var2;               Describe data format
CARDS;                         if pertinent
 cards
RUN;                           Block terminator
PROC xxx DATA=name1;           Procedures on name1 data set
RUN;                           Block terminator
PROC xxx DATA=name1;           Procedures on name1 data set
 BY var1;
RUN;                           Block terminator
DATA name3;                    Establish a third data set
/* comment */                 Document your work
SET  name1 (DROP=var1);        Establish a source of data
PROC xxx DATA=name3;           Procedures on name3 data set
RUN;                           Block terminator
PROC xxx DATA=name2;           Procedures on name2 data set
RUN;                           Block terminator
```

Exercise 5: Procedure Practice

Try adding the following procedures to the test.pgm program and look at their results.

```
PROC SORT;
BY age;
PROC PRINT;
RUN;
PROC MEANS;
RUN;
```

Getting Your Results

Adding a Title

```
PROC PRINT DATA=scores;
TITLE 'Class Standings';
```

Printing Data in Subsets

```
PROC SORT DATA=scores;
BY year;
PROC PRINT DATA=scores;
BY year;        Separate groups
TITLE1 'Class Standings by Year';
        Add up to ten titles (centered)
FOOTNOTE1 '1990 Registrar List';
        Add up to ten footnotes
```

To Select or Order Certain Variables

```
PROC PRINT;
VAR name year grade; Select variables and order
BY year;          Separate groups
```

Using the PUT Statement

The PUT statement allows you to customize your report. PUT is in some ways the opposite of the INPUT statement. PUT can print either variable values or constants. Unless otherwise specified, PUT always begins a new line. PUT statements must be used inside a data step.

PUT /;	Skips a line
PUT @10 ’text’;	Prints text starting in column 10
PUT var1 1-10;	Prints the values of var1 columns 1-10
PUT var1 1-10 .2;	Prints the values of var1 to 2 decimal places in columns 1-10
PUT ‘Name:’ var1;	Prints the text in quotes followed by the value of var1

More on DATA

SAS has several rules about the way it handles data. Leading and trailing blanks are ignored. Missing numerical data is specified by a decimal point (.). Without special formatting, a numerical value that contains an alphabetic character is also considered blank and the SAS variable _ERROR_ is set at 1. The ‘bad’ observation will be printed in the LOG and the field with the error will be pointed out.

- To test for missing text values, use the following statement:

  IF _ERROR_ NE 1;
- If you specify a decimal (.) in the value, it will override the decimal format.
- To screen out observations with missing text values, use the following statement:

  IF name NE ’’;
- To screen out observations with missing numerical values use the following statement:

  IF value NE .;

Numerical Data

Here are some examples of handling decimal input data

- Entering the decimal point

INPUT valuea valueb	valuea	valueb
12.5 54.98	12.50	54.98
1.45 49.76	1.45	49.76
3.52 .59	3.52	0.59
5.6 65	5.60	65.00

- Enter Number of Decimal Places

INPUT valuea 1-4 2 valueb 6-11 3;	valuea	valueb
125 5498	1.25	5.498
145 4976	1.45	4.976
352 59	3.52	.059
56 65	5.60	.065

- Mixed, Entered Decimal Takes Preference

INPUT valuea 1-4 2 valueb 6-11 3;	valuea	valueb
12.5 54.98	12.50	54.980
1.45 49.76	1.45	49.760
352 59	3.52	.059
5.6 65	5.60	.065

Adding New Variables

A new variable may be created or an old variable modified by using a mathematical expression.

newvar = *expression*;

The mathematical operations will be done in a hierarchial order, not from left to right. Mathematical expressions within parentheses are performed first. The actual order of operation within parentheses or in the absence of parentheses is exponentiation, multiplication and division, and lastly addition and subtraction. Use parentheses to ensure that the expressions are evaluated according to your specifications.

- New Variable

```
var12 = var11 + 5;
var12 = var10 - var11;
var12 = (var8 + var9 + var10 + var11) / 4;
```

- Modify an old variable

```
var3 = var3*1.05;
```

- Use in a program

```
DATA newds;
INPUT var1 var2;
var3 = var1 + (var2 * 4);
CARDS;
```

- Use with IF statement

```
IF var3 = 5 THEN var10 = 6;
IF var3 = 5 THEN var10 = 6;
ELSE var10 = 1;
```

More on PMENUs

In the exercises and sections above, you learned how to use many of the functions in the menu bar in your SAS session. This section will describe some of the other functions that are available to you through the menus plus show you some shortcuts on using these functions. We will go through each menu in turn and explain the functions that are in each menu. The menus and functions that will be described are the ones that you will see on the Program Editor, Log, and Output Windows. It should be noted that other windows displayed by SAS may have different functions and menu bars. Also, some menu items cannot be selected from some windows. This is because it does not make sense to execute that function from that window. When this is the case, SAS will 'grey out' the particular items. You are encouraged to explore the PMENU system and examine the functions that are available. In this section, the term 'current window' will refer to the window from which the menu item was chosen.

The File Menu

Open

- Read File - brings up a window that allows you to load in a text file from disk into the current window.
- Read Object - Reads in files that were saved as objects.

Save

Saves the contents of the current window into the default file if one exists.

Save As

- Write to File - Brings up a dialog box that allows you to save the contents of the current window to a specific file.
- Write to Object - Saves the file as an object instead of text.

End

Close this window

Exit

Quit SAS

The Edit Menu

The Edit menu provides many common editing functions. You can mark text in the current window by dragging your mouse cursor over the desired text and then cut, copy, or clear the marked text. You can position the cursor at a location and paste text from the clipboard. You may also perform searches for particular text strings and change the text strings to other text strings. You can configure a whole menu of editing options, and there is even a menu that will allow you to spell check your programs.

The View Menu

Colors

This item brings up a menu that allows you to set the colors for all of the different windows that SAS displays. Experiment with the colors to find those that suit your tastes.

Save Attributes

This item allows you to permanently save the colors that you have just chosen.

The Locals Menu

The Locals menu contains items that are specific to the current window. Sometimes there are no functions specific to the current window and, therefore, the Locals menu will not appear on that window. One place that it does appear and is most useful is in the Program Editor window. The items described below are specific to the Program Editor window.

- Submit - Submits the entire contents of the Program Editor window for execution.
- Recall Text - Recalls previously submitted text to the window beginning with the last submission and going backwards. Successive recalls bring back successive submissions. The recalls are placed in the order that they were submitted.
- Submit Top Line - Allows you to selectively submit the top n lines from the Program Editor window for execution.
- Other Items - The remaining items in the Locals menu are used with SAS/CONNECT to establish connections to other systems. Consult the SAS manuals if this function is needed.

The Globals Menu

The items in the Globals menu allow you to bring up other SAS windows and execute the global Display Manager Commands. The items are described below.

ASSIST

Brings up the SAS/ASSIST window. SAS/ASSIST is a task-oriented, menu-driven windowing system that simplifies use of the SAS System.

Program Editor

Brings up the Program Editor window if iconified or hidden**.**

Log

Brings up the Program Editor window if iconified or hidden.

Output

Brings up the Program Editor window if iconified or hidden.

Output Manager

Brings up the Output Manager window. The Output Manager provides a listing and the ability to manipulate all output that is currently in the Output window.

Graph Manager

The Graph Manager window performs the same function as the Output Manager except for the Graph window.

Invoke Application

This item allows you to bring up SAS/AF or SAS/INSIGHT. SAS/AF allows the development and execution of user-written applications. You can create your own screens, menus, programs, data entry and data display screens. SAS/INSIGHT is a highly interactive tool for data analysis. With it you can explore data through a variety of interactive graphs including bar charts, scatter plots, and 3D rotating plots. Other data analysis tools are also available.

Data Management

The Data Management item brings up another menu which has several very useful items for managing your data. These items are described below.

- Directory - The directory item brings up a listing of the contents of the last accessed SAS library. The Work library is displayed by default.
- Libname List - This item brings up a window with a listing of all currently defined librefs and the libraries they point to. By selecting any of the librefs, you can bring up a directory window for that libref.
- Variable Definition - This item brings up a window showing a listing of all of the variables defined for the last used data set. This window can also be accessed for any data set by selecting that data set in the directory window.
- Catalog Directory - This item brings up a window displaying the contents of the last accessed SAS catalog. This window may also be displayed by selecting a catalog from the directory window.
- Edit - This item brings up a menu that allows you to interactively edit your data sets. You can edit your data in Table or Record format. Table format displays a screenful of observations at once in a table using FSVIEW. Record format uses FSEDIT to display a single observation at a time. To learn more about FSEDIT and FSVIEW, consult the SAS/FSP manual.
- Browse - This item is similar to the Edit item except that you can only view the data instead of editing it.
- Filename List - This item brings up a window showing a listing of all of the currently defined filerefs and the files or directories that they point to.
- View File - This item allows you to view a file listing similar to that produced by the **more** command in UNIX.

Desktop

The Desktop item displays a menu of several helpful SAS applications. The uses of these applications should be obvious.

Command

The command item allows you to issue Display Manager and host system commands. The options available are described below.

- Command - Allows you to type in a Display Manager command
- Command Line - Turns off PMENUs for the current window only. Enter COMMAND to restore PMENUs for the window.
- Host Command - Allows you to execute a UNIX command.

Global Options

This item allows you to access a number of items that affect all windows under the Display Manager System. These options are described below.

- SAS Options - This gives a list of setable global SAS options. Only advanced SAS users will need to use these.
- Titles - Brings up a window that lets you modify the title lines that are used when output is written to the Output window.
- Footnotes - Similar in effect to the Titles option.
- Graphics Axis, Legend, Symbol, Pattern - Bring up windows that allow you to modify how graphs are produced.
- Action Bar Off - Turns off the PMENUS for **all** windows. PMENUs can be restored by entering the PMENU command.

The Help Menu

This menu gives you access to a fairly thorough amount of on-line help. In the least, you can get the syntax of any procedure or data step statement that is available for use. You can also get a listing of PFkey definitions for the current window. Using PFkeys is a shortcut to using many of the menu functions. You may edit the key definitions to your tastes, making sure that you use Display Manager commands. When you exit the key definition window, your new definitions will be saved permanently in your sasuser profile for later sessions. Be aware that key definitions only affect the current window and some Display Manager commands may not make sense when issued in some windows. For example, submit does not make sense in the Log window. PF3 is the most commonly used key. By default it is defined to exit the current window except in the Program Editor where PF3 will submit the contents for execution. Another handy key is the PF4 key in the Program Editor. This key will recall your previous submissions. Experiment with the other keys to see what they do.

SAS Errors

How SAS Reacts to Errors

- usually underlines the error
- identifies the error by number
- enters syntax check mode

If SAS Aborts Initially

- Make sure that the SAS program file is in an F format data set
- Make sure you are in a large enough region

Syntax Errors

155: THE VARIABLE NAME IS NOT ON THE DATA SET

- Variable name is misspelled.
- A procedure is referenced without the DATA=option and the most recently created data set was used.

180: STATEMENT IS NOT VALID OR IT IS USED OUT OF PROPER ORDER.

- Semicolon omitted.
- SAS keyword is misspelled.
- Specified a DATA step statement outside the DATA step or a PROC step statement outside of a PROC step.
- Failed to specify the MACRO option and then used a SAS macro facility statement.
- Included CARDS or PARMSCARDS statement in a macro
- Entered invalid characters in columns 73-80 of the statement in error.

183: THE PROCEDURE NAME IS NOT KNOWN TO THE SYSTEM.

- Check the spelling of the name.
- Make sure that library is installed. We have SAS/Basics, Access, AF, Assist, Connect, Insight, OR, QC, Stat, Graph, ETS, IML, and FSP.

620: OBSOLETE FORM OF STATEMENT OR TEXT82 OPTION INCORRECT.

- Put text strings for Titles and Footnotes in quotes.
- Common problem with older SAS programs.
- Creates a warning message.

107: CHARACTER LITERAL HAS MORE THAN 200 CHARACTERS

- Omitted a closing quote mark

  title ’Client’s calendar for 1989’;

- Used a single quote mark within a quoted string

  title ’Client’s calendar for 1989’;

- should be

  title "Client’s calendar for 1989";

Problems with Data

NOTE: SAS WENT TO A NEW LINE WHEN INPUT STATEMENT REACHED PAST THE END OF A LINE.

- You are missing data.
- Make sure the input data set is F format. If you are using a V format data set, you may have to use $VARYING or other informats to read it.

NOTE: INVALID DATA FOR A IN LINE nn

- SAS encountered character data where numeric was expected. It lists out the line of data for your inspection with a ruler.

Execution/Logic Problems

How to check that your program produced correct results

- Run it with a small sample OBS=5
- Use PROC PRINT on a larger sample.
- Regression procedures are very data dependent.

How to find logic problems

- Use lots of PUT statements

Common Problems with PROC PRINT

How to Eliminate the Default SAS title

```
title;
```

How to Format all Pages the same way

```
proc print uniform;
```

Appendix: Additional Practice

What is wrong with these data steps?

Data Step #1

```
DATA test;
```

```
INPUT var1 var2
CARDS;
12 14;
3 4
RUN;
```

Data Step #2

```
DATA test;
INPUT var1 var2;
```

Data Step #3

```
DATA test;
INPUT var1 var2;
Smith 5
Jones 4
RUN;
```

Data Step #4

```
DATA test;
INPUT var1 var2 $;
6 Smith 5
2 Jones 4
RUN;
```

Data Step #5

```
DATA test;
INPUT var1 var2
```

```
INFILE study
```

Data Step #6

```
DATA test;
INPUT var1 1-4 var2 $ 8-7;
CARDS;
  12   2
34     4
RUN;
```

Solutions to Additional Exercises

1

```
DATA test;
INPUT var1 var2 add ;
CARDS;
12 14; remove ;
3 4
RUN;
```

2 DATA test;
INPUT var1 var2;
no data, add INFILE *or* CARDS
no RUN*;*

3 DATA test;
INPUT var1 var2; *add* $ *after var1*
add CARDS*;*
Smith 5
Jones 4
RUN;

4 DATA test;
INPUT var1 var2 $;
add CARDS*;*
6 Smith 5
2 Jones 4
RUN: *remove* :, *add* ;

Note: All data does not have to be read.

5 DATA test;
INPUT var1 var2 *add* ;
INFILE study *add* ;
no RUN;

Move the INFILE statement in front of the INPUT.

6 DATA test;
INPUT var1 1-4 var2 $ 8-7;*change to 7-8*
CARDS;
12 2
34 4
RUN;

Note: Numbers may be used as text characters.