SAS Basic Operations
ECO 590 – Reed Olsen

SAS Basics

SAS requires you to write a code in a program that can be run either in whole or part. The program consists of a set of executable statements. Each statement must end with a semi-colon (;). If there is no semi-colon then SAS will continue reading until it finds one – failing to put in semi-colon's is the most basic mistake.

We will be using PC SAS. Once loaded on your computer, PC SAS can be started. You will then get a program screen that allows you to input commands. There will actually be a minimum of 3 windows:

- An Output window, where output of executable commands go. Files are saved as .sas files.
- A Log window, that recreates the commands and also gives error messages. Files are saved as .log files.
- A command window, where executable commands are inputted. Files are saved as .log files.

You can save all 3 windows – and should always save your command window. Each window saves as a different type of file (see above). The first two (output and log) are in text format so can be brought into many programs, such as spreadsheets and word processing programs.

Once you have the commands entered you can execute them by simply clicking the submit button on the main toolbar. It looks like a human figure running. You can submit the program in one of two methods:

- The entire file can be submitted by simply clicking the submit button without any of the commands highlighted.
- You can submit subsets of the commands by highlighting those you wish to submit and then clicking the submit button.

Arithmetic Operators in SAS

1. Addition: +
2. Subtraction: -
3. Multiplication: *
4. Division: /
5. raise to a power: **

Comparison Operators in SAS

1. equal to: = or EQ
2. not equal to: NE
3. greater than: > or GT
4. not greater than: NG
5. less than: < or LT
6. not less than: NL

7. greater than or equal to: >= or GE
8. less than or equal to: <= or LE

Logical Operators in SAS

1. And or &
2. Or
3. Not

Logical operators are most commonly used in if-then statements to create new variables. For example:

If race = 2 then ethnic = 1; else ethnic = 0;

creates a dummy variable (a 0/1 variable) which equals 1 if the variable "race" has the value of 2 and equals 0 otherwise. If-then statements are extremely useful in SAS to create variables from pre-existing variables. The general form of if-then statements are as follows:

If (argument) then (operation);

If the argument is true then the statement carries out the operation. If then statements may be connected to an "else" statement in the following general format (similar to that given above):

If (argument) then (operation1); else (operation2);

Here if the argument is true then operation1 is carried out, otherwise operation2 is carried out.

Here are some examples of using the logical operators with if-then statements:

If x <= 25 and y > 32 then z = 1; else z = 0;

Here both of the arguments must be met before z is set equal to 1, otherwise z will be set equal to 0. However, the statement

If x <= 25 or y > 32 then z = 1; else z = 0;

has z being set equal to 1 if either argument is met, rather than requiring that both be met.

Functions in SAS

1.	ABS	returns the absolute value
2.	max	returns the largest value
3.	min	returns the smallest value
4.	SQRT	calculates the square root
5.	EXP	raises e (2.71828) to a specified power.
6.	LOG	calculates the natural logarithm (base e)
7.	LOG2	calculates the logarithm to the base 2
8.	LOG10	calculates the logarithm to the base 10 (decimal)
9.	SUM	calculates the sum of the arguments

There exist more functions of course, but these are the basic ones and should get you going. Functions are used in the following standard way. Suppose that you want to create a variable, x, which is the natural log of a second variable y. You would do this in the following way:

X = log(y);

Hence, all arithmetic operators are used in that general format – Y = operator(argument).

The term:

Z = X + Y2 – EXP(A);**

Adds variable X to the square of variable Y and subtracts the exponent of variable A (i.e., e raised to the power represented by variable A.)

PC Operations in SAS

MSU has a site license for the PC version of SAS. This includes a personal copy for students to use. You can get a copy of SAS by going to either of the open labs in Cheek Hall, Glass Hall or the Library.

Each SAS program that loads data from outside the SAS program (as opposed to inputting it within the file) must have a statement to tell the program where to find the file. For example:

```
filename sasdata "c:\SAS\data.txt";
Data temp;
infile sasdata;
input X1 X2 X3 X4;
```

identifies a data file called "data.txt" and assigns it a SAS file identifier (sasdata). Thereafter, whenever using this data file the sas program must refer to it as sasdata. The next three lines do the following in order, (1) creates a temporary sasdata set (any name can be used here), (2) tells sas to input data from the data file to the temporary sas data set and (3) inputs 4 variables, X1, X2, X3, X4, from the file into the temporary sas data set. This is the procedure that you should use to input data that exists in a text file.

Here is an example that both inputs the same data but outputs the data to a permanent SAS data set (as opposed to the temporary one used in the above example.) The advantage of the permanent SAS data set is that you can save all the calculations and then use it later without redoing the same calculations.

```
filename sasdata "c:\SAS\data.txt";
libname sasdat1 'c:\sas';
Data temp;
infile sasdata;
input X1 X2 X3 X4;
various calculations;
;
Data sasdat1.name;
```

The above program is the same except that now we have a permanent data set that is called "name .sas7bdat" on your computer hard drive (c:\sas\name.sas7bdat).. The four variables that you read, including any created in the calculations will be in this data set. If you wish to access the data set in future operations you need just write:

```
libname sasdat1 'c:\sas';          ←This line tells SAS where to find the data set.
Data temp; set sasdat1.name;       ←This line tells SAS to put the permanent data
                                     set into a temporary one called "temp".
```

Some Useful SAS Commands

1. Titles

   You can issue titles for your SAS Output, which is often useful in keeping track of what exactly you are doing. The form of the command is as follows:

Title 'Content of Title goes here';

Every sas operation thereafter will have the printed title listed until another title command is issued.

2. Comments and spaces

Sas allows you to issue comments within the program by leading with an asterix and ending with a semicolon. For example:

*This file runs various regressions on the class data set for assignment 2;

Be careful, because if you forget the semi-colon then SAS will consider everything until the next semi-colon to be a comment. It is extremely useful to insert comments to remind yourself what you're doing.

Spaces are useful just to keep programs ordered and to divide parts of the program from others. A line space is issued by just having a line with only a semi-colon.

3. Data Step

Each SAS program must have a data step, as are given in the examples above. In general, SAS programs must include three general types of commands. I've discussed two of these above, the commands that identify input and output files and the Data command that creates either permanent or temporary sas data sets. You must have a Data step in each program or else you can do nothing else.

You must have the data command if you have data in a separate file (you don't need it if you're inputting data within the SAS program.) Remember that the “filename” command identifies a particular file with text data while a “libname” command identifies where permanent sas data sets are either written or already exist.

It is during the Data step that calculations on the data set are performed. All manipulation of variables must occur after a Data statement and before any SAS Proc statements.

Each SAS program should have some PROC statements. PROC statements (sas procedures) are things that you tell SAS to do with the data set that you just created. Here are some common PROC statements:

PROC Means ← gives summary statistics for variables in the current data set
PROC Reg ← runs an OLS regression (must be accompanied by a model statement.)
PROC Sort ← sorts the current data set (must be accompanied by a "by" statement.)
PROC Corr ← yields correlation coefficients for variables in the current data set
PROC Plot ← Plots the variables from the current data set

We'll mostly be using PROC means and Proc Reg.

4. Labels

You can label your variables in a label statement. Label statements allow you to keep track of the definition of each variable that you are using. The format of the label statement is as follows:

Label Variable = label;

Any number of labels can be included in the same statement. For example:

Label X = sex
Y = race
Z = income;

The label can be up to 40 characters long, including blanks, and if it includes either a semicolon or an equal sign, the label must be enclosed in either single or double quotes. For example:

Label X = sex
Y = "0 if Z=2; 1 otherwise"
Z = income;

5. A Sample Program with all the steps. This is one of my actual programs.

```
*This file creates some variables from the Trinidad and Tobago Labor
Force data set and runs various income regressions on subsets of the
data;
;
OPTIONS NOLABEL;
;
libname sasdata 'c:\sas\adi';
DATA TEMP; SET SASDATA.WORKER;
KEEP LINCTOT EDUCNEW EDIN1 EDIN2 OJT INSTIT2 GOVTT GOVT6 GOVT7
EXPED EXPEDSQ EXP1 LHRS DAYSHIFT URBAN MARRIED COMMNLAW HEAD
SEX AFRICAN INDIAN OIL DISTRIB MANUFACT FINSERV AGRIC
ELECTRIC CONSTRUC COMMUNIC TECHNICN PROFESSN AGWORKER SERVICE TRADE
OPERATOR ELEMNTRY SENIOR INCTOT SECONDAR UNIVERS HRSWORK OTHER
SOCIPERS CLERK WHITECOL;       (only these variables are kept in the
data set)
;
TITLE 'FULL DATA SET';
PROC MEANS N MEAN min max std;     (you can limit what you get from
proc means, these give you the number of observations, the mean,
minimum and maximum values and the standard deviation.)

DATA TEMP1; SET TEMP;
IF SEX=1;
if married=1; (These two lines delete observations, keeping only those
for which sex = 1 and married = 1 – i.e., married males)
TITLE 'LEGAL MARRIED MALE DATA SET';
PROC MEANS N MEAN MIN MAX STD;
PROC REG;
  MODEL LINCTOT= EDUCNEW EDIN1 EDIN2 OJT INSTIT2 GOVTT
        EXPED EXPEDSQ EXP1 LHRS DAYSHIFT URBAN
        HEAD AFRICAN INDIAN
        OIL DISTRIB MANUFACT FINSERV AGRIC ELECTRIC CONSTRUC COMMUNIC
        TECHNICN PROFESSN AGWORKER SERVICE TRADE
        OPERATOR ELEMNTRY SENIOR;
  MODEL LINCTOT= EDUCNEW OJT INSTIT2
        EXPED EXPEDSQ LHRS WHITECOL SENIOR GOVTT URBAN;  (The model
statements must be in the form of dependent variable = independent
variables.)
DATA TEMP1; SET TEMP;
IF SEX=1;
if COMMNLAW=1;
TITLE 'COMMON LAW MARRIED MALE DATA SET';
```

```
PROC MEANS N MEAN MIN MAX STD;
PROC REG;
  MODEL LINCTOT= EDUCNEW EDIN1 EDIN2 OJT INSTIT2 GOVTT
        EXPED EXPEDSQ EXP1 LHRS DAYSHIFT URBAN
        HEAD AFRICAN INDIAN
        OIL DISTRIB MANUFACT FINSERV AGRIC ELECTRIC CONSTRUC COMMUNIC
        TECHNICN PROFESSN AGWORKER SERVICE TRADE
        OPERATOR ELEMNTRY SENIOR;
  MODEL LINCTOT= EDUCNEW OJT INSTIT2
        EXPED EXPEDSQ LHRS WHITECOL SENIOR GOVTT URBAN;

DATA TEMP1; SET TEMP;
IF SEX=0;
if married=1;
TITLE 'LEGAL MARRIED FEMALE DATA SET';
PROC MEANS N MEAN MIN MAX STD;
PROC REG;
  MODEL LINCTOT= EDUCNEW EDIN1 EDIN2 OJT INSTIT2 GOVTT
        EXPED EXPEDSQ EXP1 LHRS DAYSHIFT URBAN
        HEAD AFRICAN INDIAN
        OIL DISTRIB MANUFACT FINSERV AGRIC ELECTRIC CONSTRUC COMMUNIC
        TECHNICN PROFESSN AGWORKER SERVICE TRADE
        OPERATOR ELEMNTRY SENIOR;
  MODEL LINCTOT= EDUCNEW OJT INSTIT2
        EXPED EXPEDSQ LHRS WHITECOL SENIOR GOVTT URBAN;
DATA TEMP1; SET TEMP;
IF SEX=0;
if COMMNLAW=1;
TITLE 'COMMON LAW MARRIED MALE DATA SET';
PROC MEANS N MEAN MIN MAX STD;
PROC REG;
  MODEL LINCTOT= EDUCNEW EDIN1 EDIN2 OJT INSTIT2 GOVTT
        EXPED EXPEDSQ EXP1 LHRS DAYSHIFT URBAN
        HEAD AFRICAN INDIAN
        OIL DISTRIB MANUFACT FINSERV AGRIC ELECTRIC CONSTRUC COMMUNIC
        TECHNICN PROFESSN AGWORKER SERVICE TRADE
        OPERATOR ELEMNTRY SENIOR;
  MODEL LINCTOT= EDUCNEW OJT INSTIT2
        EXPED EXPEDSQ LHRS WHITECOL SENIOR GOVTT URBAN;

run;  (The run command tells SAS this is your last command.  If you
don't use this SAS expects another command and you will have to
manually stop operations using the break command – an exclamation mark
in a circle.)
```