# Introduction to SAS

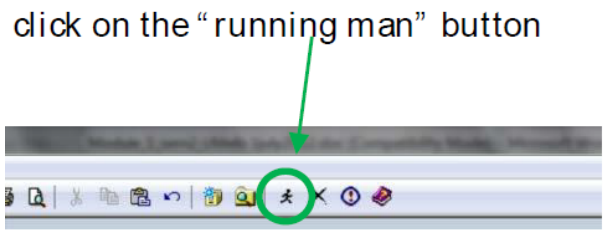
## 4. Rules for SAS variable names

- SAS names must be 33 characters long
- The first character must be an English letter (A, B, C, . . ., Z) or underscore (_). Subsequent characters can be letters, numeric digits (O, 1, . . ., 9), or underscores.
- You can use upper or lowercase letters.
- Blanks cannot appear in SAS names.
- Special characters, except for the underscore, are not allowed.
- SAS reserves a few names for automatic variables and variable lists, SAS data sets, and librefs.
- SAS When creating variables, do not use the names of special SAS automatic variables (for example, _N_ and _ERROR_) or special variable list names (for example, _CHARACTER_, _NUMERIC_, and _ALL_).

When entering data manually, it's important to identify character values so SAS knows to input the data as a string variable. For example, if we wish to enter data some data collected from male and female patients, gender must be identified as a character, this is done using '$' as summarised below:

```
data cholesterol;
input ID age gender $ systolic diastolic height weight;
datalines;
1 42 m 110 65 64.3 147.45
2 53 f 130 72 69.2 167.35
3 53 f 120 90 70.8 157.12
4 48 f 120 80 72.5 229.21
5 53 m 118 74 66.1 134.96;
run;
```

This code enters a data set with 5 patients (IDs 1-5) where each patient has their age, gender (as a character) systolic BP, diastolic BP, height and weight recorded. You will notice a semi-colon ';' is at the end of each command line. This is critical, if the semi-colon is omitted the command line will not run. Once the code has been typed in the Editor window, click the "running man" to execute the code




Most of the time we will have large data sets in spreadsheets like Excel. You are able import .csv data from Excel directly into SAS. You first need to tell SAS which directly you are using. If I'm working on my desktop I would type:

Next step is the name the data that you'll be importing. This allows SAS to associated a name with the stored data so you can easily use data from this set. What it also means is that you can have multiple data sets running at once, each with a different name. Lets consider a new example of data from a large study looking plasma iron levels in patients (file called healthiron2015.csv). Let's import the data and understand the code:

There are a number of code lines included above which we will define: 

**firstobs** tells SAS that the actual observations are in the 2nd row of the spreadsheet. This is included if and only if your first row contains column labels such as ID, weight etc. 

**delimiter=',** tells SAS that your data is separated by a comma. 

**missover** prevents an INPUT statement from reading a new input data record if it does not find values in the current input line for all the variables in the statement. When an INPUT statement reaches the end of the current input data record, variables without any values assigned are set to missing. 

**dsd** tells SAS that two comma in a row should be read as a missing value. 

**informat** is an instruction that SAS uses to read data values into a variable. Unless you explicitly define a variable first, SAS uses the informat to determine whether the variable is numeric or character. SAS also uses the informat to determine the length of character variables. In this example we've told SAS what each column variable is.

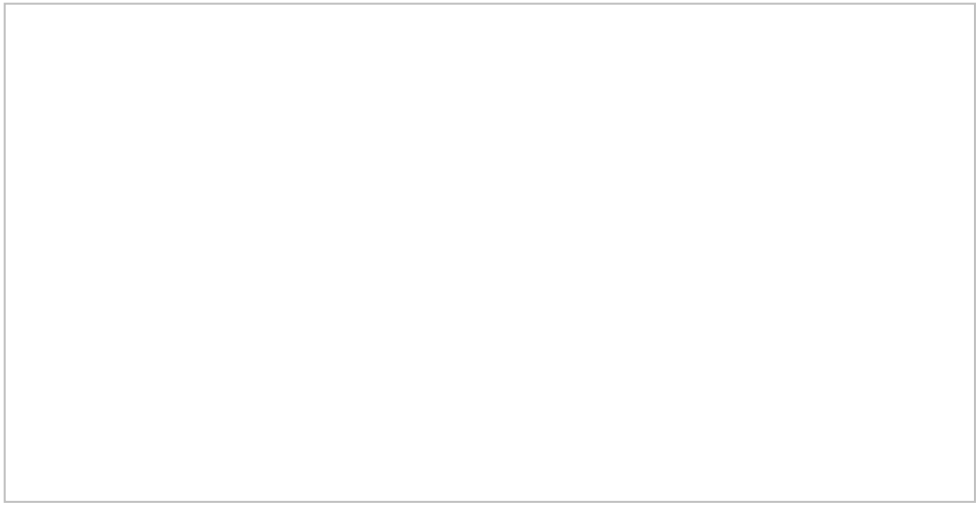
input describes the arrangement of values in the input data record and assigns input values to the corresponding SAS variables.

Next you may want to check that the data has imported. To do this we use a ***proc print*** statement. This is a really useful command line as it tells SAS to return a table (with all or some of the observations). To limit the number of observations printed we can say ***obs=10***. We are also able to give the table a title. The command line would be:

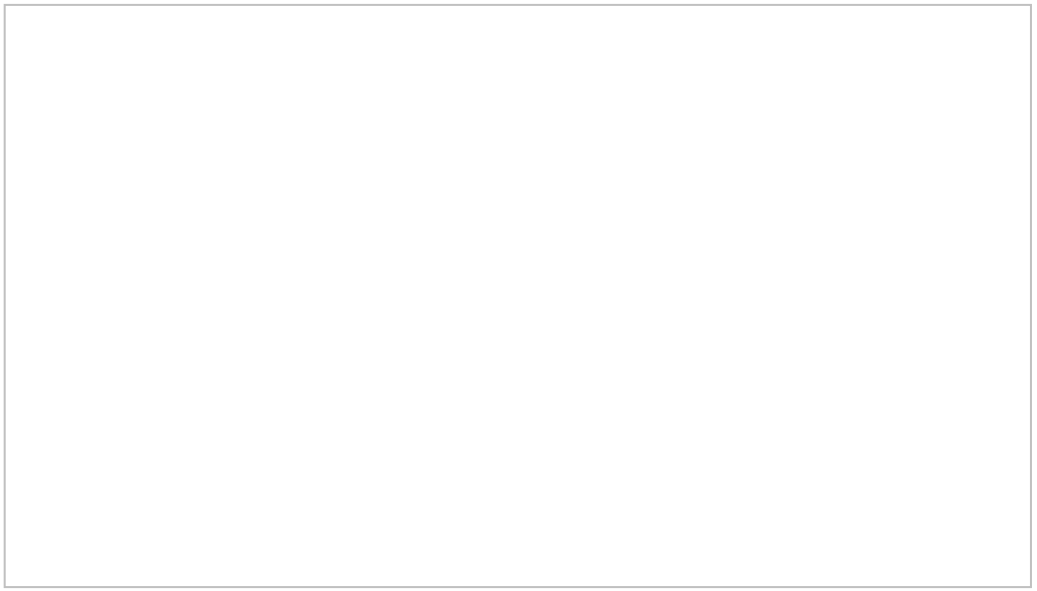
The following table is produced.

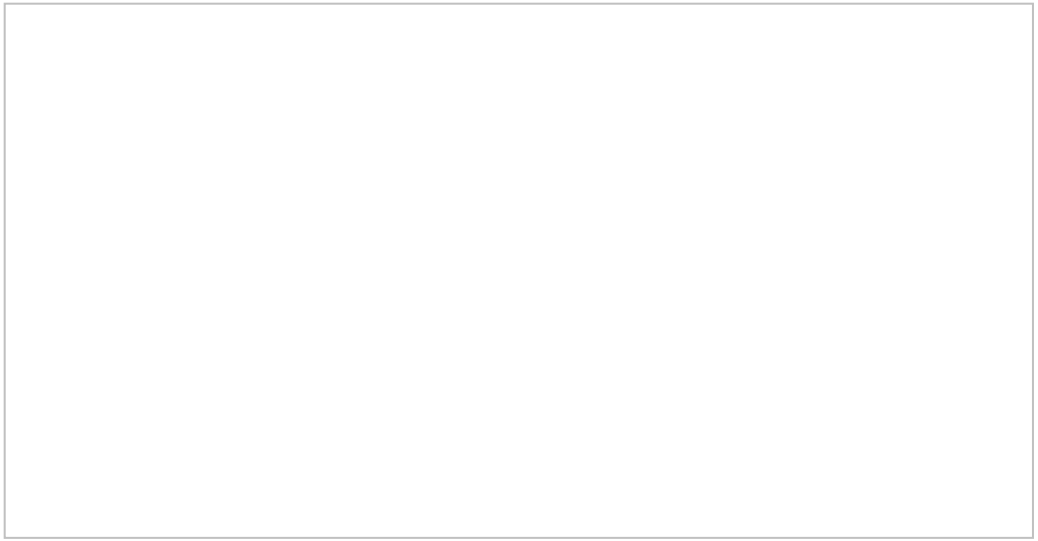
You can also do a 'global' check of the data set which has been imported using:

which returns the following output:



Data labels like ‘sex’ and code 1 and 2 for males and female isn’t very helpful. Creating variable labels makes data easier to read and work with. The following code will be used to create a value label for genotype, sex, menopause etc and this label will be added to the variables imported:

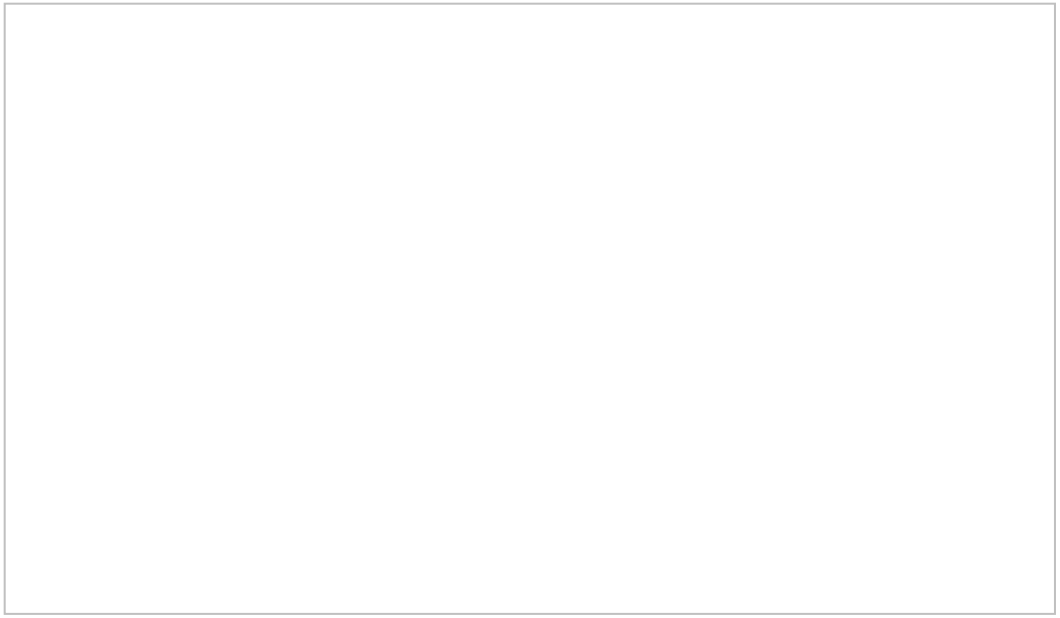




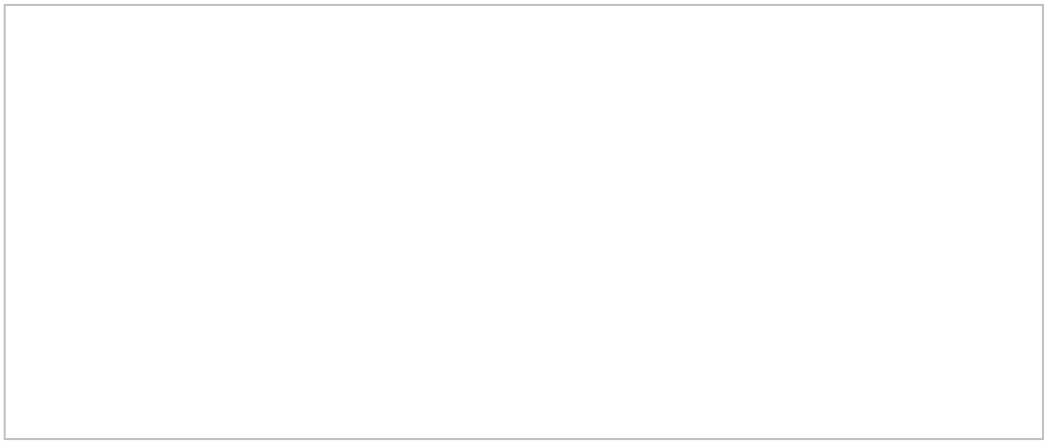
Running the ***proc print*** command will now show **Male** and **Female** as opposed to 1 and 2 in addition to text for other variables. The numerical code is still there, but we now see it as text.

It's often a good idea to check for and identify invalid or missing data. Lets look at frequency counts for the variable menopause in females and check that men haven't been coded as menopausal!

In this example you'll notice that we have included two **proc** statements before running the code. You'll also notice we used **where sex=1** in the command (remember the numerical code is still there even though we see **Male**). If you have binary data, you can limit your search to one group using the **where** command. Once the code has run, the following tables will be produced.

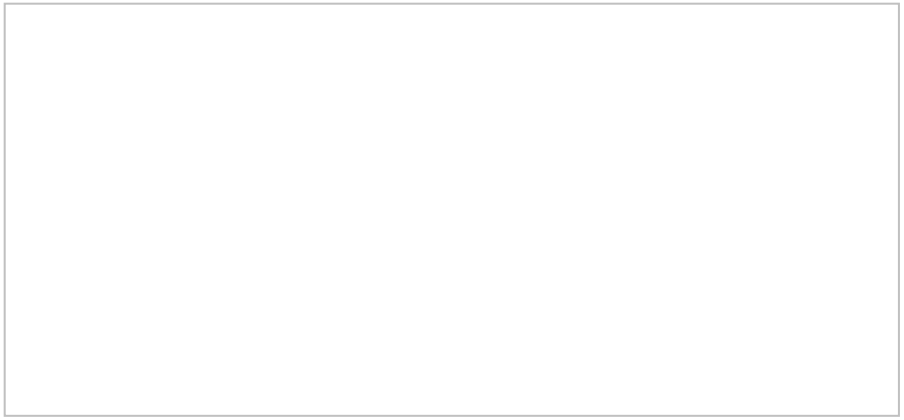


We can see form the output that there are 2 missing data point in the menopausal data for females. Let's identify the ID number of these patients:

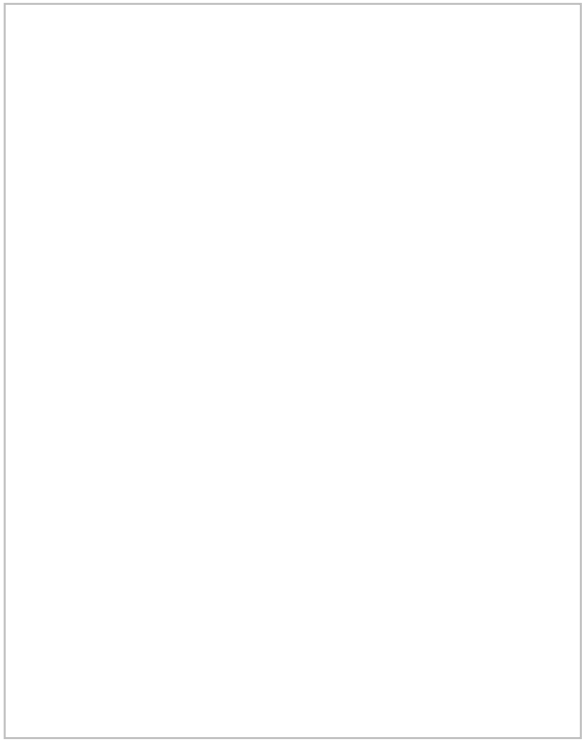


In this code we used the **where sex=2** command again, but we've added some additional code to filter the data to ensure only the missing IDs are reported. The extra code **menopause ne O and menopause ne 1** tell SAS to filter for sex=2 (females) and when menopause does not equal (ne) O AND when menopause does not equal 1; this will filter for missing data.

Being able to easily present descriptive statistic is one of the strength of all statistical packages. In SAS the MEANS procedure provides descriptive statistics for variables across all observations and within groups of observations. The code is **proc means.** If we want to present the number of observations, number of missing observations, the minimum and maximum of a the variables agebase, agefup, ferrbase, ferrfup, tsatbase, and tsatfup, the code would be as follows:



In addition to those shown above, the **proc means** statement outputs a number of descriptive statistics using the following key words:



## ADMINISTRATION

Book administration

Print book

Print this chapter

Disclaimer | Copyright | Privacy | Web accessibility | CRICOS Provider no OO121B | Contact | Student Help | Staff Help | Read this page

You are currently using guest access (Log in)