An Introduction to SAS-Lecture 1

Konstantinos Fokianos
University of Cyprus

What is SAS

Some facts about SAS:

- Developed in the early 1970s in North Carolina State University.
- Originally intended for management and analysis of agricultural field experiments.
- Now, it is one of the most widely used statistical software.
- SAS: acronym for *Statistical Analysis System*.

What is SAS

SAS is a statistical software package that allows the user to manipulate and analyze data in many different ways. Because of its capabilities, this software package is used in many disciplines (not just statistics!). It is widely used in medical sciences, biological sciences and social sciences. Knowing the SAS programming language will benefit you not only in your current class but also possibly obtaining a job.

What is SAS

Overview of SAS products:

- **Base SAS**: Data management and basic procedures.
- **SAS/STAT**: Statistical Analysis
- **SAS/GRAPH**: Presentation quality graphics
- SAS/OR: Operations Research.
- SAS/ETS: Econometric and Time Series Analysis.
- SAS/IML: Interactive Matrix Language.

and many other specialized products.

SAS Language

SAS programming language is made up of statements that specify how data are to be processed and analyzed.
The statements correspond to operations, manipulations and instructions for data analysis.
A SAS program consist of a sequence of SAS statements grouped together into blocks (*steps*).
There are two type of steps:

- data steps
- proc steps (procedure steps)

SAS Language

- The data step reads data from external sources, manipulates and combines it with other data set and prints reports. The data step is used to prepare your data for use by one of the procedures.
- SAS is not sensitive about the format of its input. In other words, statements can be broken up across lines, multiple statements can appear on a single line, and blank spaces and lines can be added to make the program more readable.
- The procedure steps perform analysis on the data, and produce the output.
- *The most effective strategy for learning SAS is to concentrate on the details of the data step, and learn the details of each procedure as you have a need for them.*

Accessing SAS

When SAS starts on Windows, it has its three main windows open:

- The program editor window;
- the log window;
- the output window.

Accessing SAS

- The program editor window is for typing in programs, editing them and running them. When a SAS program is run there are two types of output which generated: the log output and the procedure output. Both of them are displayed in their corresponding windows.
- The log shows the SAS statements that were submitted, together with messages from the SAS system about the execution of the program, including warning and error messages.
- The output window shows the printed results of any procedures.
- If graphical procedures are run, then there exists a fourth window to display the resulting graphs.

Concepts and Rules

- SAS variable names must be 32 characters or less. The variable names can be constructed by using letters, digits and the underscore character.
- Do not start variable names with underscore because SAS is using this convention for special system variables.
- Data set names follow similar rules as variables, but they have a different name space.
- SAS is not case sensitive, except inside of quoted strings.
- Missing values are denoted ny a period (.).
- **Each SAS statement must end in semicolumn (;)**

Basic Structure

- Lines beginning with an asterisk ($\star$) are treated as comments. Or you can use $/\star$ and $\star/$.
- You can combine as many data and proc steps in whatever order you want.
- Data steps begin with the word `data` and procedure steps begin with the word `proc`.
- The `run ;` command tells SAS that all previous commands should be executed.
- There are global statements (linesize, pagesize as well as many other).

Data Step

- The data step provides a wide range of capabilities, among them reading data from external sources, reshaping and manipulating data, transforming data and producing printed reports.
- The data step is actually an implied do loop whose statements will be executed for each observation either read from an external source, or accessed from a previously processed data set.
- For each iteration, the data step starts with a vector of missing values for all the variables to be placed in the new observation. It then overwrites the missing value for any variables either input or defined by the data step statements. Finally, it outputs the observation to the newly created data set.
- *The true power of the data step is illustrated by the fact that all of these defaults may be overridden if necessary.*

Data Step

- Each data step begins with the word `data` and optionally one or more data set names (and associated options) followed by a semicolon. The name(s) given on the data step are the names of data sets which will be created within the data step.
- If you do not include any names on the data step, SAS will create default data set names of the form `datan`, where `n` is an integer which starts at 1 and is incremented so that each data set created has a unique name within the current session. Since it becomes difficult to keep track of the default names, it is recommended that you always explicitly specify a data set name on the `data` statement.
- When you are running a data step to simply generate a report, and do not need to create a data set, you can use the special data set name `_null_` to eliminate the output of observations.

Reading Raw Data Separated by Spaces

Suppose that we have the following data and we want to create a SAS data set:

Obs	ID	HEIGHT	WEIGHT	GENDER	AGE
1	1	68	144	M	23
2	2	78	202	M	34
3	3	62	99	F	37
4	4	61	101	F	45

This can be done by the following program:

```
DATA LISTINP;
INPUT ID HEIGHT WEIGHT GENDER $ AGE;
DATALINES;
1 68 144 M 23
2 78 202 M 34
3 62 99 F 37
4 61 101 F 45
;
PROC PRINT;
   TITLE 'Example 1';
RUN;
```

Reading Raw Data Separated by Spaces

- Raw data lines are read beginning with the line immediately following the `DATALINES` statement.
- You specify the end of your data input with a stand alone semicolumn.
- The `INPUT` statement lists the variables you wish to create in the same order as the order appearing after the statement `DATALINES`.
- SAS reads data as either numeric or character. In this example `GENDER` is a character (indicated by the dollar sign).
- Data values have to be separated by more than one space

Reading Raw Data Separated by Spaces

You get the same results with:

```
DATA LISTINP;
INPUT ID HEIGHT WEIGHT GENDER $ AGE;
DATALINES;
1  68   144   M     23
2     78     202    M    34
3   62    99   F              37
4   61   101   F    45
```

`PRINT` procedure lists the data. In this way we can check whether the data have been read in the right order.

Reading Raw Data Separated by Spaces

Missing Values: Use

```
DATA LISTINP;
INPUT ID HEIGHT WEIGHT GENDER $ AGE;
DATALINES;
1 68 144 M 23
2 78 202 M 34
3 62 99 F 37
4 . 101 F 45
;
PROC PRINT;
   TITLE 'Example 1';
RUN;
```

Reading Raw Data Separated by Commas

A common practice is to consider data set with values separated by a comma (,).

```
ATA COMMAS;
   INFILE DATALINES DLM=',';
   INPUT ID HEIGHT WEIGHT GENDER $ AGE;
DATALINES;
1, 68, 144, M, 23
2, 78, 202, M, 34
3, 62, 99,  F, 37
4, 61, 101, F, 45
;

PROC PRINT DATA=COMMAS;
   TITLE 'Example 2';
RUN;
```

Reading Raw Data Separated by Commas

- The `INFILE` statement is used to indicate that the raw data are stored in, and are being read, from an external file. We will see more applications of `INFILE` statement next.
- The option `DLM=","` tells the program to use commas rather than spaces for delimiters. You may choose any data delimiter you wish by using this option.
- An improvement over the option `DLM=","` is the statement `DSD` which allows you to treat two consecutive delimiters as containing a missing value. You can also read a text string that contains the delimiter if it is contained in quotes.

Reading Raw Data Separated by Commas

```
DATA COMMAS;
   INFILE DATALINES DSD;
   INPUT X Y TEXT $;
DATALINES;
1,2,XYZ
3,,STRING
4,5,"TESTING"
6,,"ABC,XYZ"
;

PROC PRINT;
   TITLE 'Example 3';
RUN;
```

Reading data arranged in columns

SAS provides two methods of reading data values that are uniformly aligned in columns: column input and formatted input. Both provide the ability to read data from fixed locations in the input record and both expect to find data in those locations.

```
DATA COLINPUT;
   INPUT ID 1 HEIGHT 2-3 WEIGHT 4-6 GENDER $ 7 AGE 8-9;
DATALINES;
168144M23
278202M34
362 99F37
461101F45
;

PROC PRINT;
   TITLE 'Example 4';
RUN;
```

- In this example, we do not need to leave any space between data values.
- The column specifications in the `INPUT` statement provide instruction as to where to find the data values.
- Notice that the value `99` (WEIGHT variable) is placed in columns `5-6` and not `4-5`.
- However SAS will read the right numbers because the columns `4-6` have been specified for this variable.