3.36pt

Introductory Computing for Statistics

Lecture 1: Basics of SAS Programming

Xiao Li
Rutgers University

October 9, 2017

How to get access to SAS University Edition

- Install Oracle VirtualBox
- Add the SAS University Edition vApp to VirtualBox
- Create a folder for your data and results
- Start the SAS University Edition vApp
- Open the SAS University Edition

SAS Interface

- Program Editor Window
  - Access and edit existing SAS programs;
  - Write, submit and save SAS programs;
- Output Window
  - To examine your programs results;
  - Automatically accumulates output in the order in which it is generated;

SAS Interface

- Program Editor Window
  - Access and edit existing SAS programs;
  - Write, submit and save SAS programs;
- Output Window
  - To examine your programs results;
  - Automatically accumulates output in the order in which it is generated;
- Log Window
  - Act as a records of your SAS session;
  - Messages are written to the log in the order in which they are generated by the program.

SAS Interface

- Program Editor Window
  - Access and edit existing SAS programs;
  - Write, submit and save SAS programs;
- Output Window
  - To examine your programs results;
  - Automatically accumulates output in the order in which it is generated;
- Log Window
  - Act as a records of your SAS session;
  - Messages are written to the log in the order in which they are generated by the program.

Basic Example

Example 1.1

```
DATA mydata;
        INPUT Id Height Weight;
        DATALINES;
        53 42 41
        54 46 35
        55 40 35
        ;
RUN;
PROC print data=mydata;
        var Height Weight;
RUN;
PROC means data=mydata;
        var Height;
RUN;
```

SAS Language

SAS is organized into steps, which are like paragraphs. There are two types of steps:

- DATA Step: create a data set
- PROC Step: analyze data created by DATA step.

Notes:

- SAS steps consist of statements;
- Use space to spereate codes;
- Case of commands does not matter in SAS;
- A semicolon (;) is required to denote the end of the statement;
- Using a consistent layout pattern makes SAS pramgrams easier to read, debug, and correct (highly recommended).

SAS Language

SAS is organized into steps, which are like paragraphs. There are two types of steps:

- DATA Step: create a data set
- PROC Step: analyze data created by DATA step.

Notes:

- SAS steps consist of statements;
- Use space to spereate codes;
- Case of commands does not matter in SAS;
- A semicolon (;) is required to denote the end of the statement;
- Using a consistent layout pattern makes SAS pramgrams easier to read, debug, and correct (highly recommended).

DATA Step

General Form of Simple Data Step

```
data data_set_name;
        input variables;
datalines;
<the lines of data>
;
run;
```

Naming rules:

- Combination of numbers, letters or underscores;
- Must begin with letters or underscores;
- No length limit;
- Be careful of certain special variable names reserved by SAS, such as, _N_ ,_numeric_ , _NULL_.

DATA Step

General Form of Simple Data Step

```
data data_set_name;
        input variables;
datalines;
<the lines of data>
;
run;
```

Naming rules:

- Combination of numbers, letters or underscores;
- Must begin with letters or underscores;
- No length limit;
- Be careful of certain special variable names reserved by SAS, such as, _N_ ,_numeric_ , _NULL_.

DATA Step

the INPUT statement

The INPUT statement names the variables in your data set and tells SAS where the values of the variables can be found.

- Three major variations for INPUT: list input, column input and formatted input.
- Variables can be either numeric or character (involves use of dolloar sign $).

DATA Step

Example 1.2 (List input)

Obs	name	Weight	Jump1	Jump2	Jump3
1	Lucky	2.3	1.9	.	3.0
2	Spot	4.6	2.5	3.1	0.5
3	Tubs	7.1	.	.	3.8
4	Hop	4.5	3.2	1.9	2.6
5	Noisy	3.8	1.3	1.8	1.5

List input

```
DATA ToadJump_1;
        INPUT name $ Weight Jump1 Jump2 Jump3;
        DATALINES;
        Lucky 2.3 1.9 . 3.0
        Spot 4.6 2.5 3.1 0.5
        Tubs 7.1 . . 3.8
        Hop 4.5 3.2 1.9 2.6
        Noisy 3.8 1.3 1.8 1.5
        ;
RUN;
```

Notes:

- "." denotes missing value;
- $ denotes previous variable is a character.

List input cont.

If you have more than one data point recorded on each line, it is not necessary to reformat the data. Simply add an @@ at the end of the INPUT line.

```
DATA ToadJump_2;
        INPUT name $ Weight Jump1 Jump2 Jump3 @@;
        DATALINES;
        Lucky 2.3 1.9 . 3.0 Spot 4.6 2.5 3.1 0.5
        Tubs 7.1 . . 3.8 Hop 4.5 3.2 1.9 2.6
        Noisy 3.8 1.3 1.8
        1.5
        ;
RUN;
```

Note: This is the same exact data set as before, only the formatting has changed.

List input cont.

- Character string must be simple: no imbedded spaces, no longer than 8 characters;
- No special format is allowed: no date, time etc;
- Separated by space or spaces (extra white space doesn't matter in SAS).

Column input

Example 1.3

Obs	name	Weight	Jump1	Jump2	Jump3	birthday
1	Lucky	2.3	1.9	.	3.0	Aug 12 2013
2	Spot	4.6	2.5	3.1	0.5	Jul 31 2012
3	Tubs	7.1	.	.	3.8	Nov 13 2010
4	Hop	4.5	3.2	1.9	2.6	Aug 24 2013
5	Noisy	3.8	1.3	1.8	1.5	Sep 10 2009

Column input cont.

```
DATA ToadJump_3;
    INPUT name $ 1-5 Weight 7-9 Jump1 11-13 Jump2 15-17
            Jump3 19-21 birthday $ 23-34;
    DATALINES;
    ----+----1----+----2----+----3----
    Lucky 2.3 1.9  .  3.0 Aug 12 2013
    Spot  4.6 2.5 3.1 0.5 Jul 31 2013
    Tubs  7.1   .   .  3.8 Nov 13 2010
    Hop   4.5 3.2 1.9 2.6 Aug 24 2013
    Noisy 3.8 1.3 1.8 1.5 Sep 10 2009
    ;
RUN;
```

Column input cont.

When to use this type of input?

- Values are not all separated by spaces but each of the variable's values is always found in the same place in each data line.
- All the variables are standard numeric or character strings.
- Blank spaces appear in the character string.

Formatted input

Example 1.4 (same table as example 1.3)

```
DATA ToadJump_4;
        INPUT @1 name $5. @7 (Weight Jump1 Jump2
                Jump3) (4.1) birthday $char11.;
        DATALINES;
        ----+----1----+----2----+----3---
        Lucky 2.3 1.9  .  3.0 Aug 12 2013
        Spot  4.6 2.5 3.1 0.5 Jul 31 2013
        Tubs  7.1   .   .  3.8 Nov 13 2010
        Hop   4.5 3.2 1.9 2.6 Aug 24 2013
        Noisy 3.8 1.3 1.8 1.5 Sep 10 2009
        ;
RUN;
```

PROC Step

PROC Step General Form

```
proc <procedure options>;
        options;
run;
```

- All procedures start with the keyword PROC followed by the name of the procedure, such as PRINT, SORT or CONTENT etc.
- Each PROC has its own unique form with its list of options.

OPTIONS statement

An OPTIONS statement is used to define an environment for the program. It usually appears at the beginning of the program, outside of PROC steps and DATA steps. It will change the default settings.

- LINESIZE (LS) - defines the number of characters that will be printed across the width of each page. Acceptable ranges are from 64 through 256.
- PAGESIZE (PS) - defines the number of lines that will be printed down the length of each page. Acceptable ranges are from 15 through 32767.
- NOCENTER - writes all the output in the log and listing files flush left.
- NODATE - won't print the date in the output.
- NONUMBER - won't print the page number in the output.

PROC PRINT

PROC PRINT General Form

```
proc print data = data_set;
        var <variables>;
        by <variables>;
run;
```

PROC PRINT

Example 1.5 (print out variables "name", "Weight", "birthday")

```
DATA ToadJump;
        INPUT @1 name $5. @7 (Weight Jump1 Jump2
                Jump3) (4.1) birthday $char11.;
        DATALINES;
        Lucky 2.3 1.9  .  3.0 Aug 12 2013
        Spot  4.6 2.5 3.1 0.5 Jul 31 2013
        Tubs  7.1   .   .  3.8 Nov 13 2010
        Hop   4.5 3.2 1.9 2.6 Aug 24 2013
        Noisy 3.8 1.3 1.8 1.5 Sep 10 2009
;
RUN;
PROC print data = ToadJump;
        var name Weight birthday;
RUN;
```