STAT COMPUTING
BANA 6043

Lecture 1

Getting Started with SAS/Basic Data Manipulation

Let us do the quiz using SAS!

Get Started with SAS

Two options:

- Remotely access SAS using ucvlab
- Use the SAS installed on your own laptops

To-do list:

1. Open SAS
2. Get familiar with all the windows in SAS

SAS windows

- Explore window --- where you can access your SAS files and libraries
- Editor window --- where you input, edit and submit SAS codes
- Output/Result window --- where you get the results
- Log window --- where you check how SAS executes the program (notes, warnings or errors)
  - **Note**: Beginners often ignore the SAS log and go straight to the output – which is a bad SAS programming habit.

Tips and Tricks

- Always take a quick glance of the log window to assure that you do not lose any observation or variable.

- Read the messages in the log window carefully if you suspect something is wrong with the output or you see warning or error messages.

- If you are not able to figure out what is wrong with you code, copy and paste the warning or error messages to Google and see how others solve the problem.

How to input data?

- Example (Question 2 in quiz)

```
DATA TABLE;                    /*names the data set*/
    INPUT LENGTH;              /*names the variable*/
CARDS;                         /*signal that data follow*/
    8                          /*data begin*/
    10
    10
    8
;                              /*signal the end of data*/
RUN;
PROC PRINT;                    /*print the data*/
RUN;                           /*tells SAS to run all the preceding lines of steps*/
```

- Run it on your computer. Check the result.

Basic SAS Syntax

SAS programs are comprised of DATA steps and PROC steps.

- **DATA** steps
  - Begin with DATA statements
  - Read and modify data
  - Create new SAS data sets

  - Example:
```
DATA distance;
    Miles=23.8;
    Kilometers=1.61*Miles;
```

- **PROC** steps
  - Begin with PROC statements
  - Performs specific analysis or functions
  - Yield analysis results (tables, figures, reports…..)

  - Example:
```
PROC PRINT DATA=distance;
RUN;
```

Structure of SAS program

- **DATA** *data_name*;
  INPUT *variables;*
  CARDS;

  lines of data
  ;
  RUN;

  PROC *procedure options;*
  statements;

  RUN;

Tips and Tricks

- Statements begin with keywords --- DATA, PROC, INPUT, CARDS, VAR, INFILE…
- Use space to separate codes
- Statements **always** end up with a **semicolon** (;)
- Case **insensitive** - UPPER or lower cases are treated as the same
- Insert your programming notes /* ABC…*/ into your SAS codes to make your code **readable** to yourself and others.

Back to Question 2 in the quiz

- Example

```
DATA TABLE;
    INPUT LENGTH;
CARDS;
    8
    10
    10
    8
;
RUN;
PROC UNIVARIATE;                    gives a statistical summary of the data
RUN;
```

- Run it on your computer. Check the output.
  Do you have the mean and standard deviation?

Exercise 1

To-do list:

1.Use **PROC UNIVARIATE** to do Question 4.

2.Check the output.

1. What are the mean and the standard deviation? Compare them to the those in Question 2.
2. What else statistics you get?

Question 4: We have another scale and we use it to measure the same table four times. The result is as follows

8.9, 9.1, 9.1, 8.9

How would you estimate the actual length of the table now?

Which scale is more accurate in your opinion? State your reason.

More about INPUT

Multiple variables in a data set

- Example

```
DATA TABLE;
    INPUT PLANT $ LENGTH WIDTH HEIGHT @@;
CARDS;
    MARK  12  6  8   ABC  11  7  8   LATE  11  8  9
    YORK   6   .  5   AAA  .   7  7
;
RUN;
PROC PRINT DATA=TABLE;
RUN;
```

- Questions to figure out yourselves
  - How many variables in the data set “TABLE”?
  - Why do we need “$” sign after “PLANT”?
  - Why do we need “@@” sign at the end of the statement?

Tips and Tricks

- Structure: INPUT VAR1 VAR2 $ VAR3;

- $ indicates the type of the variable is a character string. It is placed right after the corresponding variable.

  The default type of a variable is a real number.

- @@ tells SAS that more than one observations will be entered on a line instead of a single observation as before.

- Period(.) indicates missing value.

Exercise 2

- Frog Jump Competition.

  The data includes the name of the frog, its weight and scores in 3 jumps. Some scores are missing (disqualified jump), they're labeled with a period.

- Write SAS codes to print the data EXACTLY the same as follows

```
Lucky  2.3  1.9  .    3.0
Spot   4.6  2.5  3.1   .5
Tubs   7.1   .    .    3.8
Noisy  3.8  1.3  1.8  3.1
```

Answer

```
DATA FROG_JUMP;
    INPUT NAME $ WEIGHT JUMP1 JUMP2 JUMP3 @@;
CARDS;
    Lucky  2.3  1.9  .    3.0      Spot   4.6  2.5  3.1    .5
    Tubs  7.1   .     .    3.8     Noisy  3.8  1.3  1.8  3.1
;
RUN;
PROC PRINT DATA=FROG_JUMP;
RUN;
```

More about INPUT

- **Column input** --- Reading data arranged in columns
- Example

The data file consists of name , age , weight and gender for several students. It is like this

```
John Adams   18 150m
Sally Busch  20 120f
Sarah Lee    21 110f
David Turner 17 190m
```

Notice:

1. There is A space in each name.
2. There is NO space between weight and gender.

Codes

```
DATA COLDATA;
    INPUT NAME $ 1-12 AGE 14-15 WEIGHT 17-19 GENDER $ 20-20;
CARDS;
John Adams   18 150m
Sally Busch   20 120f
Sarah Lee      21 110f
David Turner 17 190m
;
RUN;
PROC PRINT;
RUN;
```

- Run the codes on your computer. Be careful! Make sure you put data in the right columns.

Tips and Tricks

Column Input

- Values do not have to be separated by spaces.
- Blank spaces CAN appear in the character string, but regarded as ONE entry.
- The data MUST be arranged neatly so that each value starts at the same column in each data line.

More about INPUT

- **Define a variable yourself**

  You can define variables after the keyword “INPUT” based on existing variables.

- Example

  Reading scores before and after training

Before	After	Diff=After-Before
20	29	?
12	20	?
19	23	?
16	17	?
….	….	….

How can we obtain “Diff” ?

Code

```
DATA READING;
    INPUT BEFORE AFTER;
    DIFF=AFTER-BEFORE;
CARDS;
    20   29
    12   20
    19   23
    16   17
;
RUN;
PROC PRINT;
RUN;
```

- Run the codes on your computer.

PROC PRINT

Syntax

PROC PRINT DATA=*dataset*;

VAR *variable_1 variable_2… variable_n*;

Comment

- DATA =*dataset* tells SAS to print out the data set named "*dataset*" (optional but recommended)
- List the variables you want to print after VAR in the order you want them printed.
- Without VAR statement, by default, SAS will print all the variable present in the data set

Exercise 3

Reading scores before and after training

Before	After	Diff=After-Before
20	29	?
12	20	?
19	23	?
16	17	?

Print out “Diff” alone.

You do not have to retype the data into SAS.

Miscellaneous

Simple rules for making up names for variables and data sets:

- Names must be 32 characters or fewer in length
- Names must start with a letter or an underscore (_)
- Names can contain only letters, numerals or underscores (_). No !@#$%&*
- Names can contain upper and lower case letters

Miscellaneous

System options are the parameters that determine

- How SAS works
- What the output looks like
- How much memory is used
- And the like.

To see the default options

```
PROC OPTIONS;
RUN;
```

Some common options

- 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.
- NODATE - won’t print the date in the output.
- NONUMBER - won’t print the page number in the output
- Example
  - OPTIONS LEFTMARGIN = 1IN NODATE;

Break

You should know…

- The difference between DATA step and PROC step
- How to use the CARDS statement to input data
- PROC PRINT & PROC UNIVARIATE
- How to use column input (when we need this?)
- How to handle missing data
- How to define a new variable based on existing variables in the DATA step

Let us warm up!

- Write SAS code to input the following data set Name it “SCORE”.

name	math	sport	music
Daniel	98	87	89
Mark	76	?	90
Bill	90	91	89

Answer

```
DATA SCORE;
    INPUT NAME $ MATH SPORT MUSIC @@;
CARDS;
Daniel 98 87 89 Mark 76 . 90 Bill 90 91 89
;
RUN;
PROC PRINT DATA=SCORE;
RUN;
```

We will learn …

- How to create a new data set based on an existing data set
  - Refer to an existing data set (SET statement)
  - Select & delete variables (KEEP & DROP statement)
  - Get a subset (IF statement)
  - Sort a data set (SORT statement)

Set statement

- **Copy an existing data set**

Example

```
DATA SCORECOPY;                    create a new data set
     SET SCORE;                    refer to an existing data set
RUN;
PROC PRINT DATA=SCORECOPY;
RUN;
```

Run the codes in your computer.

Note: What you print out is a NEW data set SCORECOPY.

Tips and Tricks

- The SET statement is used in a DATA step to refer to another (existing) data set in SAS.
- The original data set still exists and can be used in SAS.
- In previous example, SCORE is the original data set, which is already stored in SAS. SCORECOPY is a new data set we created.

DROP/KEEP statement

- **Delete variables**

Example

name	math	sport	music
Daniel	98	87	89
Mark	76	.	90
Bill	90	91	89

What if we want a new data set containing solely a list of students' name?

Approach 1

```
DATA NAMELIST;                    create a new data set
    SET SCORE;                    refer to an existing data set
DROP MATH SPORT MUSIC;
RUN;
PROC PRINT DATA=NAMELIST;
RUN;
```

- DROP statement tells SAS to delete all the variables following the keyword DROP and keep all other variables.

Approach 2

```
DATA NAMELIST;                    create a new data set
    SET SCORE;                    refer to an existing data set
KEEP NAME;
RUN;
PROC PRINT DATA=NAMELIST;
RUN;
```

- KEEP statement tells SAS to only keep variables following the keyword KEEP and delete all others.

Tips and Tricks

- You can use either KEEP or DROP to select variables, whichever is easy for you.
- We use SET statement to refer to an existing data set so as to *create* a new data set. *In this process, we do not overwrite the original data set*. In other words, the original data set does not change.

Exercise 1

ID	name	sex	math	music
02	Mark	M	78	98
12	Bill	M	89	?
23	Cathy	F	93	79

To-do list:

1. Create a data set STUDENT in SAS.
2. Create another data set without any score, using the data set you created in Step 1.

Answer

```
DATA STUDENT;
    INPUT ID NAME $ SEX $ MATH MUSIC @@;
CARDS;
02 MARK M 78 98 12 BILL M 89 .  23 CATHY F 93 79
;
RUN;
DATA NOSCORE;
     SET STUDENT;
KEEP ID NAME SEX;
RUN;
PROC PRINT DATA=NOSCORE;
RUN;
```

KEEP/DROP statements allow us to select variables (columns).

Question:
How to select a subset of observations (rows)?

IF statement

- **Get a subset of observations**

For the data set STUDENT, what if we want a list of all males?

ID	name	sex	math	music
02	Mark	M	78	98
12	Bill	M	89	?
23	Cathy	F	93	79

Codes:

```
DATA BOY;
    SET STUDENT;        refers to an existing data set
IF SEX='M';             specifies the condition of the subset to be created
RUN;
```

- Try it on your computer.

IF…THEN…ELSE statement

IF…THEN…ELSE statement can be used to create new variables based on existing variables.

```
DATA CODING;
    SET STUDENT;
IF SEX='M' THEN CODE=0;       encodes boy as 0
ELSE CODE=1;                  encodes girl as 1
RUN;
PROC PRINT DATA=CODING;
RUN;
```

To-do list:

1.Write out SAS output yourselves without running the code.

2.Run the code in SAS and check if your “guess” in Step 1 is right.

Exercise 2

- For the data set STUDENT, create a new variable GOOD in such a way that if the math score of a student is greater or equal to 90, put YES and otherwise put NO.

- Note: you are in fact creating a new data set which expands the data set STUDENT in the way that it has one more variable GOOD.

Answer

```
DATA STU;
  SET STUDENT;
IF MATH >= 90    THEN GOOD='YES';
ELSE   GOOD='NO';
RUN;
PROC PRINT DATA=STU;
RUN;
```

PROC SORT

- Rearrange the order of a SAS data set.

  Example

Make	Country	MPG
Honda	Japan	30
Buick	USA	25
Ford	USA	23
Toyota	Japan	32
Nissan	Japan	28

  Syntax: PROC SORT DATA=CAR;

  BY MPG; *sorts the data by MPG*

To-do list

1. Write SAS code to input the data
2. Sort the data set by MPG. Check the output.

Tips and Tricks

- The data set is overwritten!
- If we'd like the data set sorted to be stored in another place, use the following syntax

```
PROC SORT DATA=CAR OUT=CARSORTED;
    BY MPG;
```

"DATA=…" indicates which data set SAS should work on.

"OUT=…" tells SAS to use a new place to store the sorted data set. If omitted, the original data set will be overwritten.

Tips and Tricks (continuing)

- The default is for PROC SORT to sort from smallest to largest (ascending). If you would like to do it in a reverse way, put DESCENDING before each variable name.

```
PROC SORT DATA=CAR;
                BY DESCENDING MPG;
```

- It is possible to sort by more than one variable.

```
PROC SORT DATA=CAR;
                BY COUNTRY MPG;
```

The first variable listed after BY statement does the initial sort. The next variable sorts within the first variable.

Exercise 3

- The following data show the average length in feet of selected whales and sharks.

name	category	length	name	category	length
Beluga	Whale	15	Sperm	Whale	60
Basking	Shark	30	Dwarf	Shark	7
Gray	Whale	50	Blue	Whale	100
Mako	Shark	12	Killer	Whale	30

To-do list

1. Input the data.
2. Sort it by category and within each category put it in such a way that the length is decreasing.

Note: Do not overwrite the original data set.

Answer

```
DATA SW;
    INPUT NAME $ CATEGORY $ LENGTH;
CARDS;
….
;
RUN;
PROC SORT DATA=SW OUT=SWSORTED;
    BY CATEGORY DESCENDING LENGTH;
RUN;
```