STAT 2005 – Programming Languages for Statistics

Tutorial 8 SAS data set

2020

LIU Ran

Department of Statistics, The Chinese University of Hong Kong

1 SAS Studio

SAS ONDEMAND FOR ACADEMICS:

1.1 How to get permission

1. Use the CUHK email account to register
2. Apply for permission of SAS ONDEMAND FOR ACADEMICS (It may take an hour to get the permission email.)
3. Check the email and go to the link to use SAS.

Welcome to SAS(r) OnDemand for Academics. Your user ID is:

[redacted]

You can use either this user ID or your email address (1155135697@link.cuhk.edu.hk), along with your SAS Profile password, to sign in to SAS OnDemand for Academics:
https://welcome.oda.sas.com

After signing in, you can:

* Access SAS Studio software.
* Register a course and share course information with students.
* Review reference and support information. This information is also available from the support site:
https://support.sas.com/ondemand

Regards,
The SAS OnDemand for Academics Team

Remark 1.1. Notice the Reference, Quotas, Applications.

1.2 Panes interface

Files, Libraries; Code, Log, Result;

Remark 1.2. Notice that we cannot change the attributes of columns and create a new library by using commands in this version.

2 Data Sets and Libraries

2.1 Types of SAS Data Sets

- Temporary data sets: saved in `WORK` library, will be deleted when session terminates.
- Permanent data sets: saved in other libraries, will not be deleted when session terminates.

2.2 Data Set Name: libraryname.filename

- By default, libraryname = WORK. Change libraryname if you want to create / refer to a permanent data set.
- filename = Name of the data set.
- If both libraryname and filename are not sepecified, SAS will generate data sets named Data1, Data2, Data3 and so on in the WORK library.

```
DATA A;  * DATA WORK.A;
DATA; * DATA WORK.Data1;
DATA work.; * Nothing, must define the name of data;
DATA ABC.A; * first level: ABC, second level: A;
```

Remark 2.1. A comment for the method * ; cannot appear the semicolon in the middle.

```
DATA ABC.A; * first level: ABC; second level: A;
```

SAS will regard the second half part as the command instead of the comment.

2.3 Create A New Library

- **Explorer** window → **Libraries** → Right click → **New** → Input the library name and the corresponding path (click **Browse...**).
- LIBNAME command: LIBNAME libraryname 'pathname';

```
LIBNAME A 'D:\';
```

3 Reading Data

3.1 Data Step

3.1.1 In-stream Data

- Data must be put at the end of the DATA step using CARDS; or DATALINES; or CARDS4; or DATALINES4; to indicate the start of data.

1. CARDS; and DATALINES; End when it encounters a semicolon (;). Therefore, we cannot input data with semicolon contained inside.

```
DATA A;
INPUT B C D;
CARDS;
1 2 3
4 5 6
[;] * Can be omitted;
RUN;
```

2. CARDS4; and DATALINES4; End when it encounters a line containing ;;;; in the first four columns.

```
DATA E;
INPUT F $ G $ H $;
CARDS4;
1; 2;; 3;;;
4;;;; 5;;;;;; 6;;;;;;     * 4 before the four semicolons ;
;;;;
RUN;
```

Remark 3.1. What if we add four semicolons under the CARDS:

```
DATA A;
INPUT B C D;
CARDS;
1 2 3
4 5 6
;;;;
RUN;
```

In fact, it still works. Because it regards the first semicolon as the ending. And regard the remaining three semicolons as just blank commands. But I really don't recommend this misleading coding:)

3.1.2 Missing values input

Many collections of data contain some missing values. SAS can recognize these values as missing when it reads them. You use the following characters to represent missing values when reading raw data:

1. numeric missing values: are represented by a single decimal point (.).
2. character missing values are represented by a blank, with one exception: list input requires that you use a period (.) to represent a missing value.

```
data temp;
input x $  y $;
cards;
1 2
. s
run;
```

3.1.3 External Text File

`INFILE 'filename';`: Read existing data at the very beginning of `Data` step to creat SAS data sets.

```
INFILE 'D:\sample.txt';
```

4 Three ways for data input

4.1 List Input

Are the data values separated by at least one blank space so that they can be read in using list input?

- `INPUT {variable [$ [&]] [/]} ...  [@@];`
  - `$`: Inform SAS the variable before `$` is a character variable.
  - `&`: Allow the variable to save blank column and stop at consecutive blanks ($\geq$ 2 blanks).
  - `/`: Inform SAS to jump to the next line to read the data.
  - `@@`: Inform SAS to keep on reading data from the same data line (must be put at the end).
- `[LENGTH varlist [$] length ...  ;]`: Set the maximal allowed length of variables. By default, the character variables can only save 8 characters and the numeric variable is 8 bit. So, if the charcter string to be saved is longer than 8 characters, the size of the variable should be modified.

- The data type and size of a variable are determined at first time it appears in the code. So, the `LENGTH` should be put before the `INPUT` statement.

4.2 Column Input

Are the data values arranged in neatly defined columns so that they can be read in using column input?

- `INPUT variable [$] start_column [-end_column] [/]... ;`
- If the input only has 1 column, then the ending column can be omitted.

```
data grade;
input age 19-20 name $ 1-9 sex $ 21
grade 23-26;
cards;
Ann Wrong--A------15F-70.5
John Chan--B------16M-72.0
Peter Lee--C------17M-75.3
run;
```

4.3 Formatted Input

Do the data values contain special formats or characters so that they must be read in using formatted input?

- Format: P53-P56 of lecture notes Ch9. `[$]INFORMATw.d` (please note that w is the total width not the width before the decimal point.)
- `INPUT variable [$]INFORMATw.d ... ;`

```
DATA contest;
/* INPUT Name $16. Age 3. +1 Type $1. +1 Date MMDDYY10. */
/*  (Score1 Score2 Score3 Score4 Score5) (4.1); */
INPUT Name $16. +1 Age 2. +1 Type $1. +1 Date MMDDYY10.
(Score1 Score2 Score3 Score4 Score5) (4.1);
cards4;
Alicia Grossman  13 c 10-28-2008 7.8 6.5 7.2 8.0 7.9
Matthew Lee       9 D 10-30-2008 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-2008 8.9 7.9 8.5 9.0 8.8
Lori Newcombe     6 D 10-30-2008 6.7 5.6 4.9 5.2 6.1
Jose Martinez     7 d 10-31-2008 8.9 9.510.0 9.7 9.0
Brian Williams   11 C 10-29-2008 7.8 8.4 8.5 7.9 8.0
;;;;
RUN;
```

4.4 Mixed Input

- Mix the above mentioned three ways to input the raw data.
- `#n`: Move the pointer to n-th data line relative to current record(observation) .
- `+(-n)`: Skip over/Go back n columns.
- `@n`: Read data from the nth column.

```
data rec;
input student_id $ +(-2) college $1.;
cards;
04102872
04625861
run;
```

Remark 4.1. Some reference: Introduction to SAS

5 Exercises

1. Modify the following codes such that they can read all data.

```
data scores;
input name $ age sex $ score;
cards;
Ann Cheung  15 F 72
Mary Tam  16 F 80
Peter Wong  17 M 82
run;
```

```
data scores;
length name $ 10;
input name $ & age sex $ score;
cards;
Ann Cheung  15 F 72
Mary Tam  16 F 80
Peter Wong  17 M 82
run;
```

```
data scores;
length name $ 10;
input name $ & age sex $ score;
cards;
Ann Cheung  15 F 72 Mary Tam  16 F 80
Peter Wong  17 M 82
run;
```

```
data scores;
length name $ 10;
input name $ & age sex $ score @@;
cards;
Ann Cheung  15 F 72 Mary Tam  16 F 80
Peter Wong  17 M 82
run;
```

```
data scores;
length name $ 10;
input name $ & age sex $ score @@;
cards;
Ann Cheung#########
15 F 72
Mary Tam  ######
16 F 80
Peter Wong  ########
17 M 82
run;
```

```
data scores;
length name $ 10;
input name $ & / age sex $ score @@;
cards;
Ann Cheung#########
15 F 72
Mary Tam  ######
16 F 80
Peter Wong  ########
17 M 82
run;
```

Remark 5.1. Unlike the format width, LENGTH determines the maximal length to store the variables instead of reading length. So SAS continue reading the NAME including the #### until encountering the blank space. But NAME variable only allows 10 bits character, so the hash tags # are not stored.

But for format input `$10.`, SAS will only read the first 10 elements and it will not stop when encountering the blank space.

2. What is the result of the following code?

```
data d1;
input x x y y;
cards;
1 2 3 4
5 6 7 8
run;
```

```
   x  y
1  2  4
2  6  8
```

In a DATA step, it does not allow two variables to have the same name. So, in this case, SAS will only accept the last value of your input. For this example, first, x will receive the value 1 and then x will also receive the value 2, but currently it is the same observation. The value 2 will overwrite the value 1. And the rest part is similar.

3. Write a SAS program to read the raw data (dislocation) and to form the SAS data set as:

x	y	z
4	2	6
7	5	9
10	8	12

```
1 2 3
4 5 6
7 8 9
10 11 12
```

```
data d1;
input #2 x #1 y y #2 z z z @@;
cards;
1 2 3
4 5 6
7 8 9
10 11 12
run;
```

Remark 5.2. For the first record(observation), the first two data lines are:

```
1 2 3
4 5 6
```

For the second record(observation), due to the `@@`, the first two data lines are:

```
4 5 6
7 8 9
```

If no `@@`, for the second record, the first two data lines are:

```
7 8 9
10 11 12
```

So, the result without `@@` will be:

```
    x  y  z
1   4  2  6
2   10 8  12
```