```
/*************************************************************************
                    Week 03. Methods of Data Inputs
     Instructor: C. Peng
           Date: 2/6/2021
     Previous V: 9/4/2020
         Topics: 1. Review: SAS display management, library, options,
                    permanent/temporary data set
                 2. SAS INPUT styles: column input, list input, formatted input;
                 3. Variable LENGTH specification
                 4. INFILE - reading external data file
                 5. SAS Labels
                 6. SAS informat/format
*************************************************************************/

/*********************************
1. Review: Options, LIBNAME, DM
*********************************/
OPTIONS NODATE NONUMBER PS = 80 LS = 64 nodate nonumber;   * to specify the form of sas output;
LIBNAME  sasw03  'C:\STA311\w03';          * permenent library;

/* clear log and output windows */
DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';


/************************************************
 2. SAS INPUT styles: column, list and formatted
************************************************/


/* 2.1. column input with inline data - values of variables must be
        placed within the corresponding range and values of adjacent
        variables must separated by at least one blank.           */
DATA Orange;
INPUT state $ 1-10 early 12-14 late 16-18;
DATALINES;
```

```
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
;
RUN;

PROC PRINT DATA = Orange;
TITLE "Output of theTesting Data";
RUN;

/**------- key takeaways -------
1. flag character variable with $
2. INPUT statement: specify variable type and length.
3. TITLE is a global statement. To delete previous title,
   use statement TITLE "";
4. TITLE can be placed either inside PROC or outside PROC.
----------------------------*/

/* 2.2. List input  */
DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

/* List input - No blanks between the values of chracter variables
                values between adjacent variables must be separates
                by at leasnt one blank.                          */
DATA Orange_list;
LENGTH state $ 10.;
INPUT state $ early late;
DATALINES;
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
;
RUN;

PROC PRINT DATA = orange_list;
```

```
TITLE 'Include data values in the sas program';
RUN;

/*--------- Key Takeaways  ------------------------------------------------------------------
1. list is appropriate when no blanks are in the values of the character variable.
2. if the maximum length of character variable value is big than 8 (i.e., more than
   8 characters including blanks BETWEEN the strings, we need to define the length
   of the variable using statement: LENGTH char_name $ specified-length
3. one can also specify the length within the INPUT statement: char_name $ specificed_length.
   Example, INPUT State_name $ 18.;
-------------------------------------------------------------------------------------------*/


/*******************************************************
  2.3. column pointer-formatted input -
       values of all variables must be placed
       within the corresponding column range.
       There is NO need to separate the values
       of adjacent variables in column pointer input.
*******************************************************/
DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

DATA kids;
  INPUT @1 firstnam $ 11.     /* @ <== column pointer  */
        @12 lastname $11.    /* $ <== char variable indicator */
        @23 birthday $10.    /* in $10, 10 <== length of birthday  */
        @ 33 gender $1.       /* @33  33 <== value of gender starts at column 33  */
        @34 wklyrate;        /* close the INPUT statement with a semi-colon. */
DATALINES;
Douglas    Lindgren   08/29/1996M115
Elizabeth  Wilkerson  01/13/1997F95
Evangeline Chambers   03/11/1997F100
Arthur     Hollander  07/19/1996M.
ChristopherKalbfleisch04/13/1995M115
Stacy      Siegel     11/15/1996F100
;
RUN;
```

```
PROC PRINT DATA =kids;
 TITLE 'Daycare roster';
RUN;

/*-------------------- Key Takeaways  -------------------
1. In INPUT statement, a column pointer must be placed in
    front of the corresponding variable.
2. length-specifier: $20 and $ 20 both work fine!
3. column pointer: @33 and @ 33 both work fine!
-----------------------------------------------------------*/


/*****************************************************************
 3. INFILE - specify whether part of whole of an inline external
             data file should be read into SAS.
*****************************************************************/

DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

LIBNAME  sasw03  'C:\STA311\w03';          * permenent library;


/* 3.1 read in part of an external data file */
DATA sasw03.Orange;
INFILE "C:\STA311\w03\w03-Orange.txt" FIRSTOBS = 3 OBS = 6;  /* use the explicit path, not library
reference,
                                                              the input data is NOT as SAS format data
set!*/
INPUT state $ 1-10 early 12-14 late 16-18;
RUN;

/*---------------  Key Takeaways  -------------------
1. FIRSTOBS = 3: <== start reading data from row 3.
  (first 2 rows contain data descriptions)
2. OBS = 6:  <== the last row to read in SAS. (rest
   of the rows are part of the data file).
```

```
3. INPUT is used as usual to tell the information of the variables!
4. INFILE statement tells the location of the data file.
   For an external file, the path to the file must be given. FIRSTOBS
   and OBS are optional.
-----------------------------------------------------*/

PROC PRINT DATA = sasw03.Orange;
TITLE 'Read in data from an external data file';
RUN;
QUIT;


DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';


/* 3.2. inline text data with descriptions - read part of the data file:
        FIRSTOBS=  OBS =  */
DATA Inline_Orange;
INFILE DATALINES FIRSTOBS = 3 OBS = 6;
INPUT state $ 1-10 early 12-14 late 16-18;
DATALINES;
Projected Orange Yields in October 1997
State    Early Late
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
Based on information obtained from the
Florida Agricultural Statistics Service
RUN;

PROC PRINT DATA =Inline_Orange;
TITLE 'Read in data from an inline data file';
RUN;
QUIT;

/*------------------ Key Takeaways  ----------------
```

```
1. DATALINES in the INFILE statement points to the location
   of the inline data file which is below the DATALINES statement.
2. INPUT statement specifies the information of variables as usual
   regardless of using an inline data file or an external data file.
-------------------------------------------------*/



/*************************************************************************
 4. Complex data layout: (1). single record was split into multiple lines;
                         (2). multiple records were put in the sample line.
*************************************************************************/

DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

/** 4.1 Read in specially layout data: one record in multiple lines
        In the old version of SAS, we need to use '\' or '#2' to tell sas
        to continue to read data in the next row. In 9.0 or later, SAS
        automatically will go to the next line to read data until completing
        the record.
    **/
DATA inline_grades_multi_line;
INPUT NAME $ / QUIZ TEST PROJECT $ ABSENCES;
DATALINES;
Ann
84 90 A- 0
Bill
78 84 B 0
Cathy
95 89 A 1
David
84 88 B+ 1
;
RUN;

PROC PRINT DATA = inline_grades_multi_line;
RUN;
```

```
/*** 4.2. External txt file - one record in multiple lines ***/
DATA grades_multi_line01;
INFILE "C:\STA311\w03\w03-multi-line-grades.txt";
INPUT NAME $ / QUIZ TEST PROJECT $ ABSENCES;
RUN;

PROC PRINT; RUN;    /* If the data set is not specified, PROC PRINT will print
                       the most currently created SAS data set!                */


DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

/** 4.3. Multiple records in one line: double trailings:
         @@  must be used as in the following to read the
         data correctly in SAS.                          **/

DATA grades_multiple_obs;
  INPUT name $ quiz test project $ absences;
DATALINES;
Ann 84 90 A- 0 Bill 78 84 B 0 Cathy 95 89 A 1
David 84 88 B+ 1
;
RUN;

PROC PRINT; RUN;


DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

/* INFILE-DATALINES combination ;   */
/* 4.4. It will create a wrong data set without if '@@' is used!  */
DATA grades_infile_dataline;
INFILE DATALINES;
  INPUT name $ quiz test project $ absences;
```

```
DATALINES;
Ann 84 90 A- 0 Bill 78 84 B 0 Cathy 95 89 A 1
David 84 88 B+ 1
;
RUN;

TITLE " grades_infile_dataline;";
PROC PRINT; RUN;
TITLE "";

/*------------------ Key Takeaways  ----------------------------
1. The types of errors that SAS log reports are related to syntax.
   However, seeing no error in the log window does not mean
   you created a correct data set! You need to the content of the
   DATA SET before moving to the next step.
2. When you create multiple data sets, add a meaningful title
   PROC PRINT step so you can see the clear title in each output.
   It is a good practice to CLEAR the old title for the next new title
   by typing TITLE ""; after the PROC PRINT step.
-----------------------------------------------------------------*/



/* ========================================================
 5. Review: SAS Permanent SAS library - create a permanent
            SAS library to store SAS  *data set*!!!
             Caution - you can save your SAS
             file to folder you used for the SAS library
             without using library reference!!
========================================================== */

DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

LIBNAME SASWK3 'C:\STA311\w03';   /* create a folder under C drive and use the
                                     environment shortcut to search the folder,
                                     the actual PATH to the folder shown on the
                                     top-left of the Explorer window. You need to
```

```
                                    type the path inside the single quotes ' ' to
                                    define the permanent library.              */

/** 5.1 Save the created SAS data set to the permanent SAS library  **/
DATA SASWK3.libname_orage;
LENGTH state $ 10.;
INPUT state $ early late;
DATALINES;
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
;
RUN;

PROC PRINT DATA = SASWK3.libname_orage;
TITLE 'Print the  libname_ permanent data set';
RUN;


/* ====================================================
 6.     Label and Informat/Format

      It is very important to attach as much information
      as possible to the variables. Useful information
      includes

    - LABEL: a description or definition of variables
      in the SAS data set.
    - INFORMAT: tell SAS the format of the input variables
    - FORMAT: tell SAS the format you want to display in the
              SAS output.
=======================================================*/

/*****************************************************
5.1. Label - it describes each input variable
             This information will be kept in the resulting
             SAS data set. Most importantly, this information
```

```
             will be carried over to the SAS data sets derived
             from the initial data set.
*****************************************************/
DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';
DATA kids_label;
  INPUT @1 firstnam $11.
        @12 lastname $11.
        @23 birthday $10.
        @33 gender $1.
        @34 wklyrate 3.;
  LABEL firstnam='First name'
        lastname='Last name'
        birthday='Birthday in days from Jan. 1, 1960'
        gender='Gender'
        wklyrate='Rate';
DATALINES;
Douglas    Lindgren   08/29/1996M115
Elizabeth  Wilkerson  01/13/1997F95
Evangeline Chambers   03/11/1997F100
Arthur     Hollander  07/19/1996M.
ChristopherKalbfleisch04/13/1995M115
Stacy      Siegel     11/15/1996F100
;
RUN;

PROC PRINT DATA =kids_label;
 TITLE 'Daycare roster: kids Label';
RUN;


/*****************************************************************
5.2. SAS Date Informat/Format

About SAS Dates:

1. A SAS date is saved as a numeric value
  that represents the number of days since
```

```
  January 1, 1960. A negative value implies
  that the date is earlier than January 1, 1960.
  A positive value means the date is after January 1, 1960.

2.FORMAT: If you want to display a date, you need to
  specify a format. Otherwise, you will have a numerical
  value. There are many different formats for dates in SAS.
  See the following link to the SAS document:

https://documentation.sas.com/?docsetId=lrcon&docsetTarget=p1wj0wt2ebe2a0n1lv4lem9hdc0v.htm&docsetVersion=9
.4&locale=en

3. INFORMAT: different date variables in source data may have
   different formats (even in a single data file), when we use
   INPUT or INFILE to read data in SAS, we have to tell SAS the format
   of the corresponding date variable. Since the format is the
   for the incoming date variable, it is called INFORMAT.
 ***************************************************************************/

DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

DATA kids_format;
  INPUT @1 firstnam $11.
        @12 lastname $11.
        @23 birthday mmddyy10.  /* the form of date has format: mmddyy10.
                                   Because it is used in the input statement,
                                   it is called INFORMAT.                  */
        @33 gender $1.
        @34 wklyrate 3.;
  LABEL firstnam='First name'
        lastname='Last name'
        birthday='Birthday in days from Jan. 1, 1960'
        gender='Gender'
        wklyrate='Rate';
DATALINES;
Douglas    Lindgren   08/29/1996M115
Elizabeth  Wilkerson  01/13/1997F95
```

```
Evangeline Chambers   03/11/1997F100
Arthur     Hollander  07/19/1996M.
ChristopherKalbfleisch04/13/1995M115
Stacy      Siegel     11/15/1996F100
;
RUN;

PROC PRINT DATA =kids_format;
FORMAT birthday worddate18. wklyrate dollar7.1;
 TITLE 'Daycare roster';
RUN;


/* ========================================================
6.     User Defined FORMATS - More to come next week
========================================================*/
DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

PROC FORMAT;
VALUE $genderfmt 'F'='Female'
                 'M'='Male';
RUN;

PROC PRINT DATA =Kids_format  LABEL;
FORMAT GENDER $genderfmt. birthday date9.;
 TITLE 'Daycare roster';
RUN;
```