Chapters 5, 17 CREATING SAS DATA SETS: Advanced Options

There are several important options that may be used when creating a data set.

(1) When data are inputted in columns (called **column input**), then in INPUT statement, besides variable names and types (character or numeric), it is wise to specify ranges (starting column and ending column) for each variable. For example, to describe the variables in the data set

1–––+––––10 –––

2810 61 MOD F

2804 38 HIGH F

2807 42 LOW M

2816 26 HIGH M

2833 32 MOD F

2823 29 HIGH M

type

input id $ 1-4 age 6-7 actlevel $ 9-12 gender $ 14;

This implies that variable id occupies columns 1 through 4; age, columns 6 through 7; actlevel, columns 9 through 12; and gender occupies column 14.

* Note that the line on top of the data set (called a **ruler**) is used for identifying individual columns. It is not part of the dataset, and is added conventionally to raw data in examples.
* Note that when using column input, it is not necessary to specify variables in the order of appearance. The following INPUT statement is acceptable

input actlevel $ 9-12 age 6-7 id $ 1-4 gender $ 14;

SAS reads these variables in the order actlevel, age, id, gender.

* Note that the column range must be placed **after** the dollar sign, if it exists. For example, typing

input id **1-4 $** age 6-7 actlevel **9-12 $** gender **14 $**;

produces an error message, and is not understood by SAS.

* Note that it is not necessary to give the column range for all variables listed in the INPUT statements. For example,

input id $ age 6-7 actlevel $ 9-12 gender $;

reads the data correctly (verify!)

Example. Column input is especially useful when a numeric variable contains a special character, for example, a dollar sign. The following DATA step reads the variable payment as a numeric variable ignoring the dollar sign.

data payments;

input person $ payment 4-6;

cards;

1 $350

2 $580

3 $470

;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sometimes variables in a data set are separated by a **single space**, and therefore, are not inputted in columns. For example, consider the data set

1–––+––––10 –––

2810 61 MOD F

2804 38 HIGH F

2807 42 LOW M

2816 26 HIGH M

2833 32 MOD F

2823 29 HIGH M

The entries for variable gender do not occupy the same column, and hence it is not possible to specify the column range for this variable in this data set (but the column ranges may be specified for the other variables). For example, typing

input id $ 1-4 age 6-7 actlevel $ 9-12 **gender $;**

reads the data correctly.

(2) It is possible to specify only the **starting column**. To do so, place the “at” sign and the starting column number @*n* (called **@*n* column pointer control**) in front of the variable name in the INPUT statement. For example, when the data are inputted in columns, the code line

input @1 id $ @6 age @9 actlevel $ @14 gender $;

reads the data correctly (verify!)

* Note that the INPUT statement admits any mixture of column ranges and starting columns. For example, the line

input id $ 1-4 @6 age @9 actlevel $ gender $ 14;

reads the data correctly (verify!)

(3) It is possible to specify only the **starting column relative to the current position**. To do so, place +*n* (called **+*n* pointer control**) in front of the variable name in the INPUT statement. For example, when the data are inputted in columns, the code line

input @1 id $ age +5 gender $ @9 actlevel $;

or

input @1 id $ age +5 gender $ +(-7) actlevel $;

reads the variables in the order id, age, gender, actlevel. The initial pointer position is at column 1. After reading id and age, the pointer moves to column 8. The control pointer +5 tells SAS to move to column 13 and read gender. Then the pointer @9 instructs SAS to move back to column 9 and read actlevel. Instead of @9 it is possible to use +(-7) to move the pointer back from column 15, where it ends up after reading variable gender, to column 15-7=8 and read actlevel.

1 56 8 14

1–––+––––10 –––

2810 61 MOD F

2804 38 HIGH F

2807 42 LOW M

2816 26 HIGH M

2833 32 MOD F

2823 29 HIGH M

(4) If variables in a data set are named *NAME1, NAME2, …, NAMEn* (or *name\_1*, *name\_2*, …, *name\_n*), then it is possible to list the variables in the INPUT statement as *NAME1-NAMEn.* For example,when creating a data set containing class number (class), student id (id), and scores on quizzes 1 through 12 (quiz1, …, quiz12), type

input class $ id $ quiz1-quiz12;

(5) It is possible to place an **assignment statement** inside the DATA step to create a new variable. For example, consider the following data on running and biking times (in minutes) of a group of athletes in training.



The following DATA step creates the data file and calculates the total training time for each athlete in minutes and in hours.

data training;

input athlete $ 1-2 gender $ 4 rtime 6-7 btime 9-10;

ttime\_min=rtime+btime;

ttime\_hrs=ttime\_min/60;

cards;

1 F 52 34

2 F 56 28

3 M 58 29

4 F 67 58

5 F 66 53

6 M 58 33

7 M 49 29

8 M 51 32

9 M 53 36

10 M 55 41

;

Basic arithmetic operators that can be used to create a new variable are

* Addition z=x+y
* Subtraction z=x-y
* Multiplication z=x\*y
* Division z=x/y
* Negation y=-x
* Exponentiation z=x\*\*y
* The expression defining a new variable can contain the same variable name also on the right side of the equation sign. This statement **re-defines** the values of the variable. For example, to express the running time in hours, type rtime=rtime/60;

(6) It is possible to assign **permanent labels** to variables inside the DATA step. Labels provide each variable with more descriptive text. Labels for different variables may be assigned in a single LABEL statement or in separate statements. For example, to assign labels to the variables in the training data set, type

data training;

input athlete $ 1-2 gender $ 4 rtime 6-7 btime 9-10;

ttime\_min=rtime+btime;

ttime\_hrs=ttime\_min/60;

label athlete='Athlete Number'

rtime='Running Time'

btime='Biking Time'

ttime\_min='Total Time (in minutes)'

ttime\_hrs='Total Time (in hours)';

cards;

1 F 52 34

…

10 M 55 41

;

or, alternatively, use the LABEL statement multiple times,

data training;

input athlete $ 1-2 gender $ 4 rtime 6-7 btime 9-10;

ttime\_min=rtime+btime;

ttime\_hrs=ttime\_min/60;

label athlete='Athlete Number';

label rtime='Running Time';

label btime='Biking Time';

label ttime\_min='Total Time (in minutes)';

label ttime\_hrs='Total Time (in hours)';

cards;

1 F 52 34

…

10 M 55 41

;

These labels are used in PROC PRINT statement

proc print data=training **label**;

run;

(7) In the DATA step, sometimes it is convenient to input several data rows on one line. In this case, **DATALINES** statement must be used in place of CARDS statement, and double “at” sign @@ must be used at the end of the INPUT statement. For example, the following assignment of data is legitimate.

data training;

input athlete $ gender $ rtime btime @@;

datalines;

1 F 52 34 2 F 56 28 3 M 58 29 4 F 67 58 5 F 66 53

6 M 58 33 7 M 49 29 8 M 51 32 9 M 53 36 10 M 55 41

;

* Note that column ranges cannot be specified in this case.
* DATALINES statement is often used to shorten the length of SAS programs.
* In the DATA step when the CARDS statement is applicable, the DATALINES statement may be used instead. For example,

data training;

input athlete $ 1-2 gender $ 4 rtime 6-7 btime 9-10;

**datalines;**

1 F 52 34

…

10 M 55 41

;

SOME USEFUL BASIC FACTS ABOUT SAS

* A single semicolon on a separate line that indicates the end of the input data is termed a **null statement**.
* If there is a need to add or modify something in an existing data file, then one can refer to that file by typing

data *filename*;

set *filename*;

For example, to add a label to the variable athlete in existing data file

training, type

data training;

set training;

label athlete='Athlete Number';

run;

* The RUN statement tells SAS to execute the previous SAS statement. It must be included after every DATA step or PROC step, except when the DATA step creates a raw data. In this case, the step may end in the RUN statement or the null statement. For example, typing

data training;

set training;

label athlete='Athlete Number';

~~run;~~

makes SAS hang forever.