Getting Your Data into SAS
The Basics

Math 3210
Dr. Zeng
Department of Mathematics
California State University, Bakersfield

Outline

- Getting data into SAS

  -Entering data directly into SAS

  -Creating SAS data sets from external data files

- Reading raw data: three basic input styles

  -List

  -Column

  -Formatted

  -Mixed Inputs

Methods for Getting Data into SAS

***Method ONE*: Entering data directly into SAS (internal data).**

You may want to type the raw data directly into your SAS program when you have small amount of data. In this chapter, you will learn about

- ✓ Generating SAS data using DATA step
- ✓ The DATALINES statement
- ✓ The INPUT statement

***Method TWO*: Creating SAS data sets from raw data files (external data).**

This method allows you to import your data externally from other software's data files into SAS such as excel or text files. In this chapter, you will learn about

- ✓ Importing excel data to SAS Enterprise Guide
- ✓ Importing text data to the SAS Studio
- ✓ Importing excel data to the SAS Studio
- ✓ The INFILE statement
- ✓ The IMPORT procedure

Entering data directly into SAS (internal data)

- Sometimes, you may want to type the raw data directly into SAS
- Especially when you have small amounts of data or when you are testing a program with small test data set.
- This method works the same for both SAS Enterprise Guide and SAS Studio

The DATALINES statement

- The DATALINES statement indicates internal data
- The DATALINES statement must be the last statement in the DATA step, before the RUN statement
- The DATALINES keyword marks the start of data input, while the semicolon indicates the end of data
- The CARDS statement and the DATALINES statement are synonymous.

Example

```
data sampledata;
input name $ gender $ age;
datalines;
Adam M 35
Mary F 32
Kent M 40
Zoe F 28
Eric M .
;
proc print  data=sampledata;
title ‘this is a sample data’;
run;
```

‘this is a sample data’

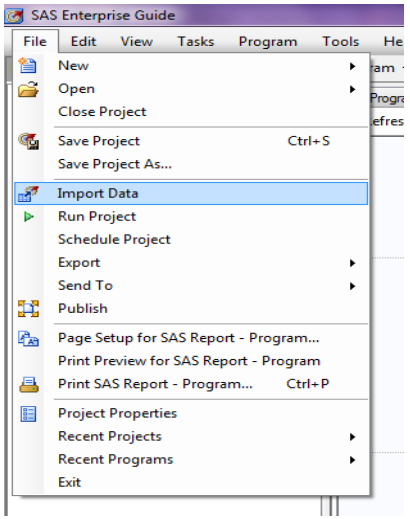
Obs	name	gender	age
1	Adam	M	35
2	Mary	F	32
3	Kent	M	40
4	Zoe	F	28
5	Eric	M	.

Note:

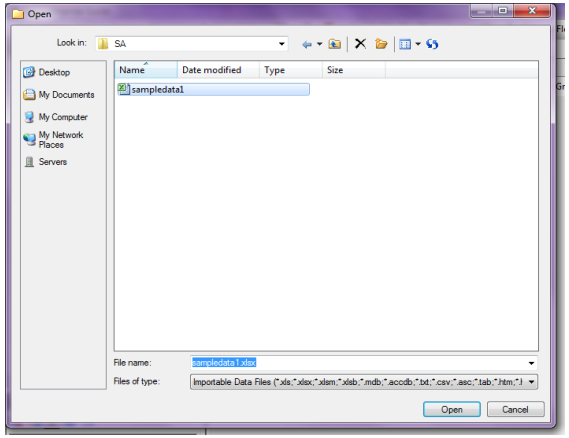
- The INPUT statement tells SAS how to read your raw data. Simply list the variable names after the INPUT keyword in the order they appear in the data file.
- Remember to place a dollar sign ($) after character variables.
- A (.) means missing values for numerical variables.

Creating SAS data sets from Excel files: SAS Enterprise Guide

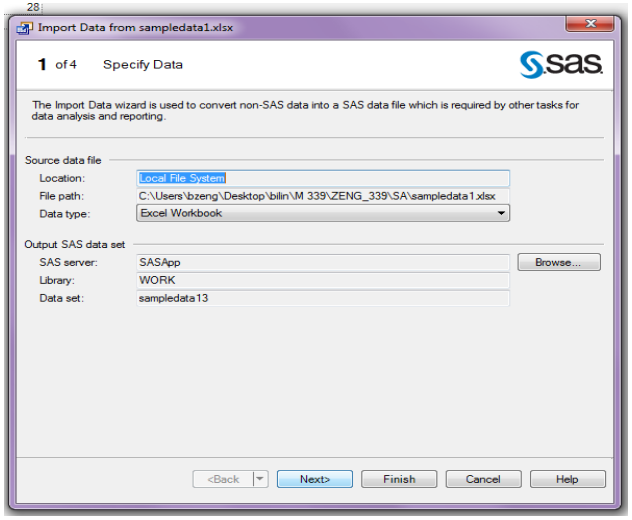
1. First click File and select import data.

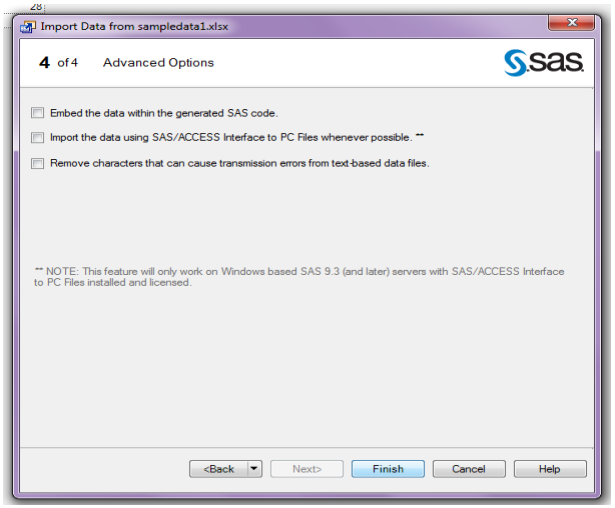


2. Select the file ‘sampledata1’



3. Hit ‘next’ for four times and hit ‘finish’



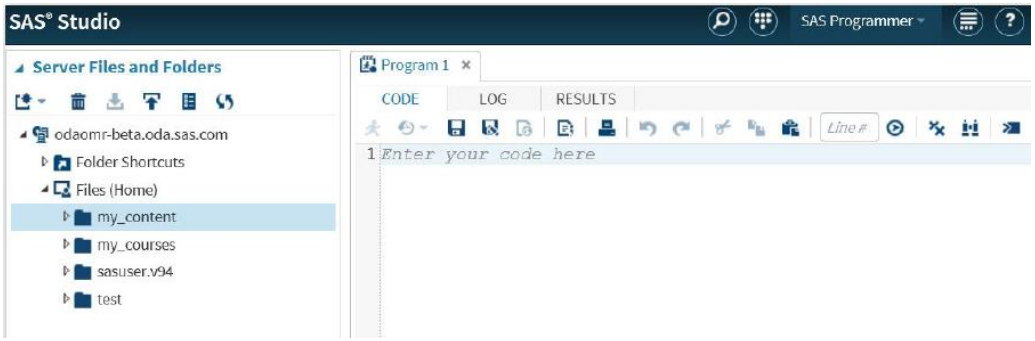


4. Print your data set

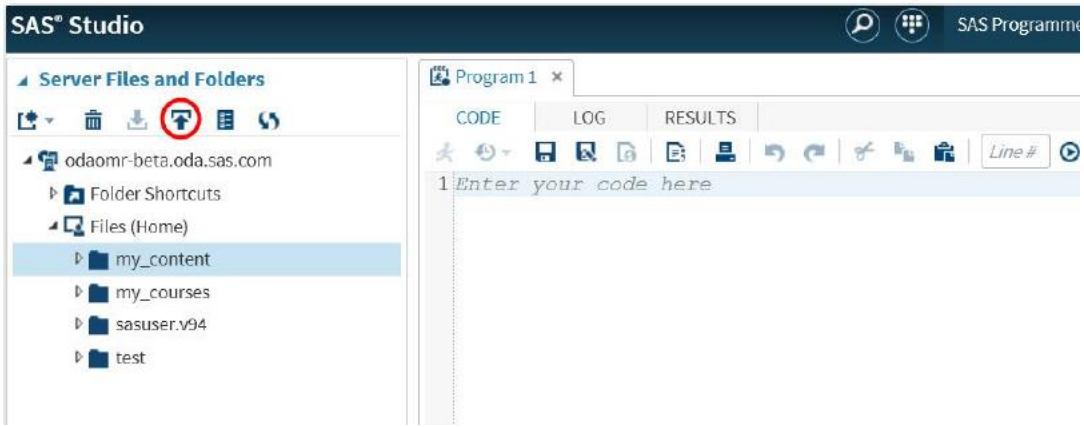
```
proc print  data=sampledata1;
title ‘this is a sample data 1’;
run;
```

Creating SAS data sets from text files: The SAS Studio

1. From within SAS Studio, expand **Files (Home)** and then select the **my_content** folder.



2. Click the **Upload** icon to upload ‘sampledata1.txt’



3. Now this raw data is stored under the '/home/bzeng/my_content' folder. In your case, this folder should be '/home/your_user_name/my_content' . To import the external data, we need to use the **INFILE** statement to tell SAS the location of the raw data files in your computer.

4. Type the following SAS code in the editor window and submit

```
data sampledata1;
infile '/home/bzeng/my_content/sampledata1.txt';
input name $  gender $  score;
run;
proc print data=sampledata1;
run;
```

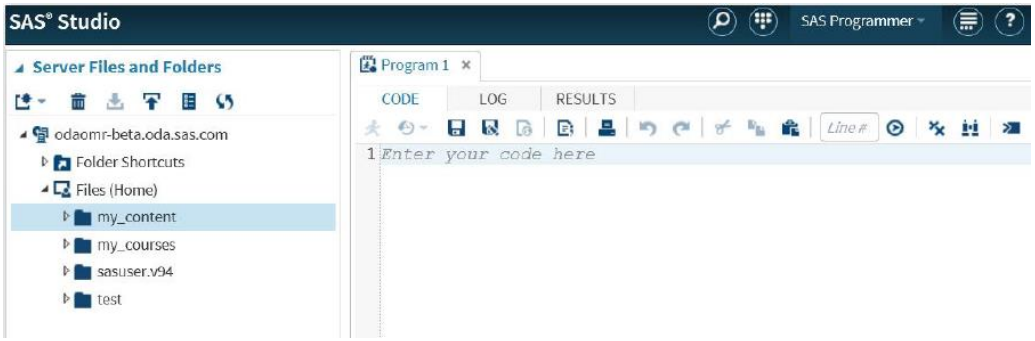
Obs	name	gender	score
1	Adam	M	35
2	Mary	F	32
3	Kent	M	40
4	Zoe	F	28
5	Eric	M	.

Remark:

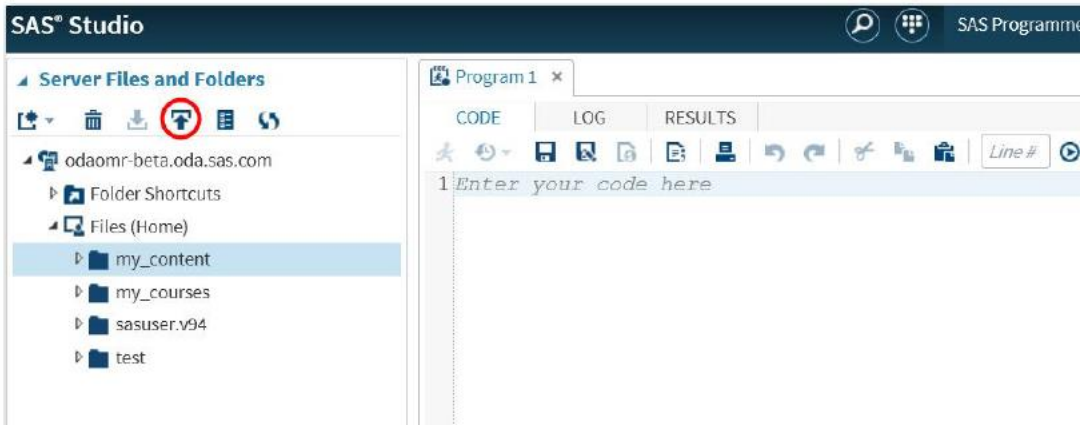
- The INFILE statement tells SAS where to find the raw data in your computer
- You must indicate the path of your raw data in single quotes immediately after the INFILE keyword
- The INPUT statement tells SAS how to read your raw data. Simply list the variable names after the INPUT keyword in the order they appear in the data file.
- Remember to place a dollar sign ($) after character variables.
- This method is also useful if the external file has the .dat extension, say “sampledata1.dat”.
- You can easily convert a text file to a DAT file by changing the extension name from .txt to .dat

Creating SAS data sets from Excel files: The SAS Studio

1. From within SAS Studio, expand **Files (Home)** and then select the **my_content** folder.



2. Click the **Upload** icon to upload ‘sampledata1.txt’



3. To read an excel file to SAS, we need to use the **IMPORT** procedure. Here is the general form of the IMPORT procedure for reading Excel files:

```
proc import datafile='filename'  options
  dbms=identifier  out=data-set    replace;
```

- filename is the file you want to read
- data-set is the name of data set you want to create
- The REPLACE option tells SAS to replace the SAS data set named in the OUT=option if it already exists
- The dbms=option tells SAS the type of Excel file to read, such as .xls, .xlsx,
- The IMPORT procedure treats the first line of your data file as the header. So the INPUT statement is not needed here

4. In this example:

```
proc import datafile='/home/bzeng/my_content/sampledata1.xlsx'
dbms=xlsx  out=sampledata2 replace;
run;
proc print data=sampledata2;
title SAS data set read from excel files;
run;
```

Sometimes, the **IMPORT** procedure makes it easy to read external data:

- ✓ It treats two consecutive delimiters as a missing value, will read values enclosed by quotation marks, and assign missing values to variables when it runs out of data on a line.
- ✓ The IMPORT procedure treats the first line of your data file as the header. So the INPUT statement is not needed here
- ✓ It can recognize some date formats
- ✓ It scans the first 20 rows in your data file and automatically determine the variable type (numeric or character)
- ✓ It assigns lengths to the character variables
- ✓ We can use the CONTENTS procedure to verify the variable type and length. The general form of CONTENTS procedure is:

```
proc contents data=data-set; run;
```

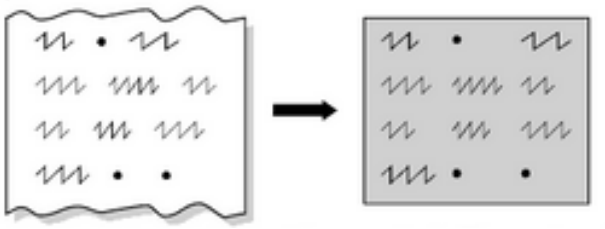
Introduction to Raw Data

To create a SAS data set from raw data, you must examine the data records first to determine how the data values that you want to read are arranged. Then you can look at the styles of reading input that are available in the INPUT statement. SAS provides three basic input styles:

- List (free formatted) input
- Column input
- Formatted input

Reading Unaligned Data: List Input

Understanding List Input:



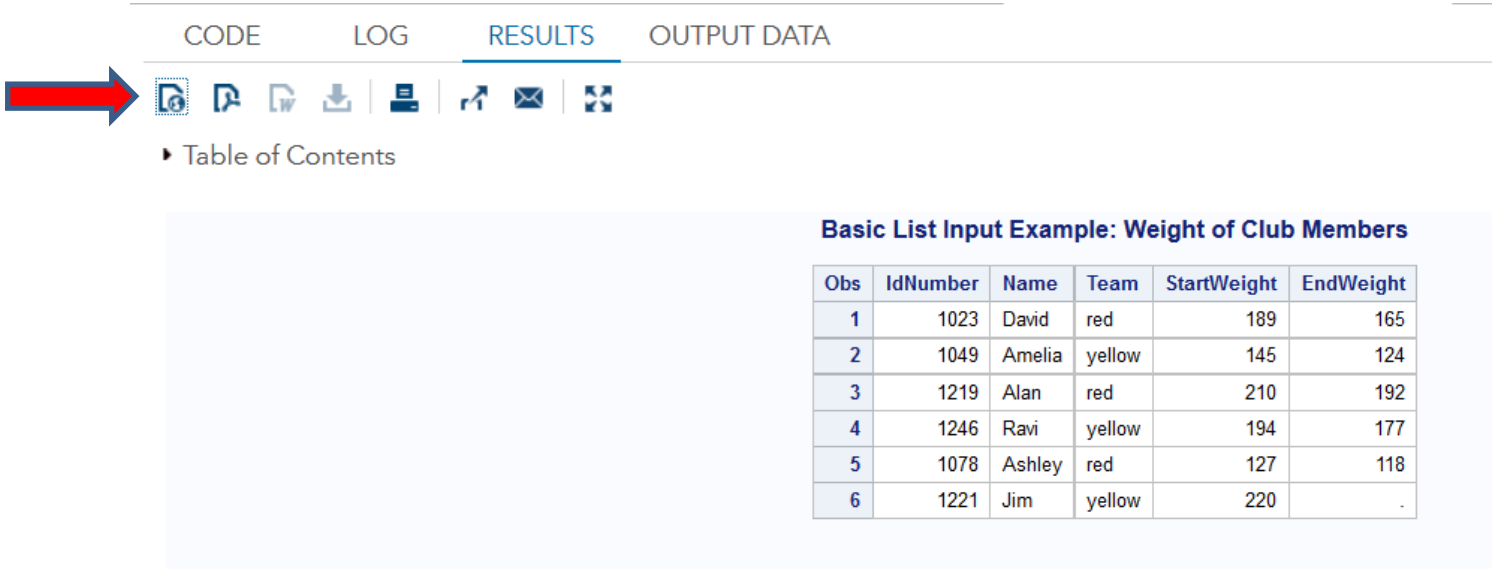
- List input is the simplest form of the INPUT statement
- List input is used to read your raw data files that are separated by a delimiter character (by default, a blank space)
- With list input, SAS reads a data value until it encounters a blank space or the end of the input record

Restrictions with list input:

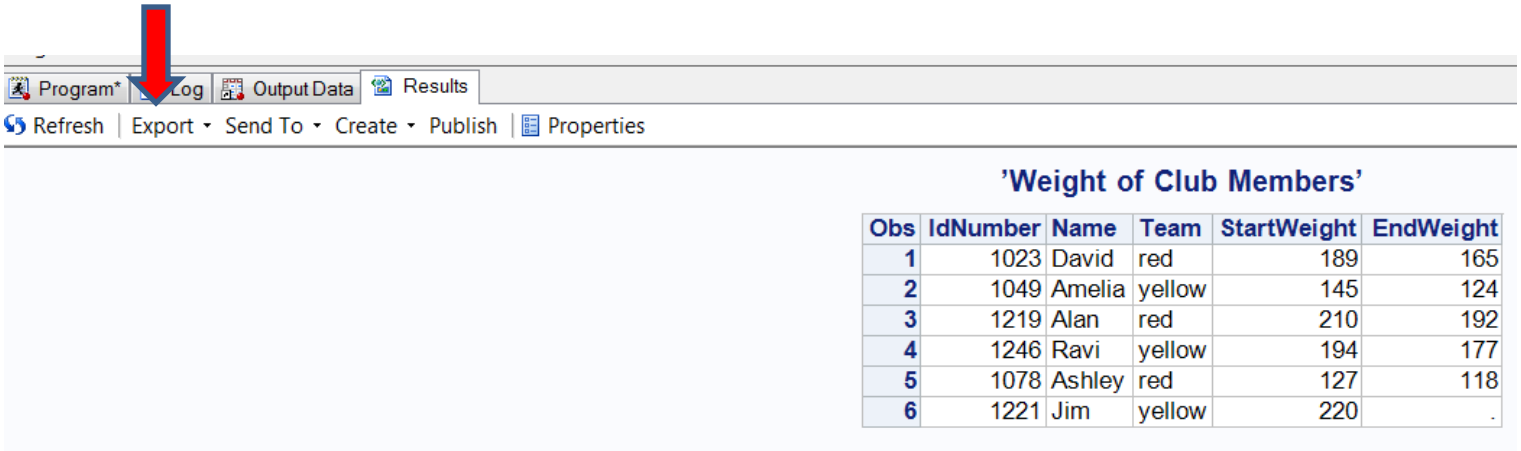
- You must read all the data in a record-no skipping over unwanted values
- Any missing data must be indicated with a period
- Character data, if present, must be simple: no embedded spaces, and no values greater than 8 characters in length
- List input is not appropriate for the data file that contains date or other values which need special treatment

Example: Basic List Input

```
data club1;
input IdNumber Name $ Team $ StartWeight EndWeight;
datalines;
1023 David red 189 165
1049 Amelia yellow 145 124
1219 Alan red 210 192
1246 Ravi yellow 194 177
1078 Ashley red 127 118
1221 Jim yellow 220 .
;
proc print data=club1;
title 'Basic List Input Example: Weight of Club Members ';
run;
```



Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.



Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

Note: SAS allows you to download your result as a PDF file or an HTML file.

Comments:

- The variable names in the INPUT statement are specified in exactly the same order as the fields in the raw data records.
- The DATALINES statement marks the beginning of the data lines while the semicolon that follows the data lines marks the end of data lines and the end of the DATA step.
- Each data value in the raw data record is separated from the next by at least one blank space. The last record contains a missing value.

Example: When the Data is Delimited by Characters, Not Blanks

```
options pagesize=60 linesize=80 pageno=1 nodate;
data club1;
infile datalines dlm=’,’;
input IdNumber Name $ Team $ StartWeight EndWeight;
datalines;
1023,David,red,189,165
1049,Amelia,yellow,145,124
1219,Alan,red,210,192
1246,Ravi,yellow,194,177
1078,Ashley,red,127,118
1221,Jim,yellow,220,.
;
proc print data=club1;
title ’Example: data delimited by characters’;
run;
```

’Example: data delimited by characters’

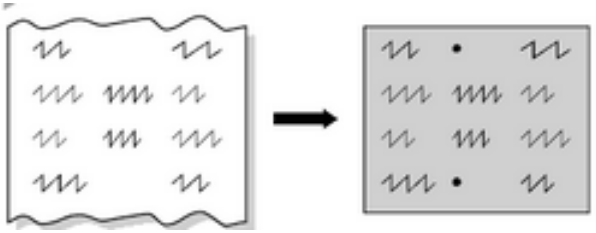
Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

Comments:

- List input, by default, scans the input records, looking for blank spaces to delimit each data value. The DLM= option enables list input to recognize a character, here a comma, as the delimiter.
- These values are separated by commas instead of blanks.
- The DLM=option is only available in the INFILE statement.
- If you read the data from an external data file, in the INFILE statement, we need to replace the datalines keyword by the location of your external raw data file, such as '/home/bzeng/my_content/sampledata1.txt'

Reading Data That is Aligned in Columns

Understanding Column Input:



- Some raw data files do not have spaces or other delimiters between all the values or periods for missing data; so the files cannot be read using list input.
- But if each of the variable's values is always found in the same place in the data line (e.g. survey data), then you can use column input as long as all the values are character or standard numeric.
- With column input, data values occupy the same field within each data record.

Example: Reading Data Aligned in Columns

```
data club1;
input IdNumber 1-4 Name $ 6-11 Team $ 13-18 StartWeight 20-22
EndWeight 24-26;
datalines;
1023 David    red      189 165
1049 Amelia yellow 145
1219 Alan      red      210 192
1246 Ravi      yellow         177
1078 Ashley  red       127 118
1221 Jim       yellow   220
;
proc print data=club1;
title Example: reading data aligned in columns;
run;
```

Example: reading data aligned in columns

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	.
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	.	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

Comments:

- When you use column input in the INPUT statement, list the variable names and specify column positions that identify the location of the corresponding data field.
- With column input, the INPUT statement takes the following form. After the INPUT keyword, list the first variable's name. If the variable is character, leave a space; then place a $. After the dollar sign, or variable name if it is numeric, leave a space; then list the column or range of columns for that variable. Repeat for all variables.
- If you read the data from an external data file, use the INFILE statement before the INPUT statement and remove the DATALINES statement.

Advantages of Column Input over List Input

- With column input, character variables can contain embedded blanks
- Missing values can be left blank
- Spaces are not required between values
- Column input also enables the creation of variables that are longer than eight bytes
- Skip some data field when reading records of raw data

Practice 1

Learn about reading embedded blanks and creating longer variables. Create the following data set by using column input.

reading embedded blanks and creating longer variables

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David Shaw	red	189	165
2	1049	Amelia Serrano	yellow	145	124
3	1219	Alan Nance	red	210	192
4	1246	Ravi Sinha	yellow	194	177
5	1078	Ashley McKnight	red	127	118
6	1221	Jim Brown	yellow	220	.

Practice 2

Column input also enables you to skip over fields or to read the field in any order. Create the following data set by only modifying the INPUT statement that you wrote in Practice 1.

skipping fields when reading data records

Obs	Team	Name	StartWeight	EndWeight
1	r	David Shaw	189	165
2	y	Amelia Serrano	145	124
3	r	Alan Nance	210	192
4	y	Ravi Sinha	194	177
5	r	Ashley McKnight	127	118
6	y	Jim Brown	220	.

Reading Data That Requires Special Instructions

Understanding Formatted Input:

- Standard numeric data contain only numerals, decimal points, plus and minus signs, and E for scientific notation.
- Sometimes raw data are not in standard numeric or character form.
- Examples such as 2,500 (read as twenty-five hundreds), $8 (eight dollars) and 06/10/2017 (June 10th, 2017), are in non-standard format.
- The INPUT statement requires special instructions to read the data correctly.
- In these cases, use formatted input which has the ability to read nonstandard values.

Three general types of informats:

Character	Numeric	Date
$informatw.	*informatw.d*	*informatw.*

- *informat* is the name of the informat
- *w* is the total width of the input field (including signs)
- *d* is the number of decimal places (numeric informats only)
- The $ indicates character informats
- The period (.) is a very important part of informats
- Two informats do not have names: $w. which reads standard character data, and w.d which reads standard numeric data

Selected Informats

Informat	Definition	Width range	Default width
Character			
$CHAR*w*.	Reads character data—does not trim leading or trailing blanks	1–32,767	8 or length of variable
$UPCASE*w*.	Converts character data to uppercase	1–32,767	8
$*w*.	Reads character data—trims leading blanks	1–32,767	none
Date, Time, and Datetime[8]			
ANYDTDTE*w*.	Reads dates in various date forms	5–32	9
DATE*w*.	Reads dates in form: *ddmmmyy* or *ddmmmyyyy*	7–32	7
DATETIME*w*.	Reads datetime values in the form: *ddmmmyy hh:mm:ss.ss*	13–40	18
DDMMYY*w*.	Reads dates in form: *ddmmyy* or *ddmmyyyy*	6–32	6
JULIAN*w*.	Reads Julian dates in form: *yyddd* or *yyyyddd*	5–32	5
MMDDYY*w*.	Reads dates in form: *mmddyy* or *mmddyyyy*	6–32	6
STIMER*w*.	Reads time in form: *hh:mm:ss.ss* (or *mm:ss.ss*, or *ss.ss*)	1–32	10
TIME*w*.	Reads time in form: *hh:mm:ss.ss* (or *hh:mm*)	5–32	8
Numeric			
COMMA*w.d*	Removes embedded commas and $, converts left parentheses to minus sign	1–32	1
COMMAX*w*.	Like COMMA*w.d* but reverses role of comma and period	1–32	1
PERCENT*w*.	Converts percentages to numbers	1–32	6
w.d	Reads standard numeric data	1–32	none

Informat	Input data	INPUT statement	Results
Character			
$CHAR*w*.	my cat my cat	INPUT Animal $CHAR10.;	my cat my cat
$UPCASE*w*.	my cat	INPUT Name $UPCASE10.;	MY CAT
$*w*.	my cat my cat	INPUT Animal $10.;	my cat my cat
Date, Time, and Datetime			
ANYDTDTE*w*.	1jan1961 01/01/61	INPUT Day ANYDTDTE10.;	366 366
DATE*w*.	1jan1961 1 jan 61	INPUT Day DATE10.;	366 366
DATETIME*w*.	1jan1960 10:30:15 1jan1961,10:30:15	INPUT Dt DATETIME18.;	37815 31660215
DDMMYY*w*.	01.01.61 02/01/61	INPUT Day DDMMYY8.;	366 367
JULIAN*w*.	61001 1961001	INPUT Day JULIAN7.;	366 366
MMDDYY*w*.	01-01-61 01/01/61	INPUT Day MMDDYY8.;	366 366
STIMER*w*.	10:30 10:30:15	INPUT Time STIMER8.;	630 37815
TIME*w*.	10:30 10:30:15	INPUT Time TIME8.;	37800 37815
Numeric			
COMMA*w.d*	$1,000,001 (1,234)	INPUT Income COMMA10.;	1000001 -1234
COMMAX*w*.	$1.000.001 (1.234,25)	INPUT Value COMMAX10.;	1000001 -1234.25
PERCENT*w*.	5% (20%)	INPUT Value PERCENT5.;	0.05 -0.2
w.d	1234 -12.3	INPUT Value 5.1;	123.4 -12.3

For a complete list of informats by category, see
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a001239776.htm

$CHARw.

1) It reads character informats.
2) w specifies the width of the input field (default is 8)
3) The $CHARw. informat does not trim leading and trailing blanks
4) The $CHARw. Informat does not convert a single period in the input data field to a blank before storing values.

Examples

```
input @1 name $char5.;
```

Data Line	Results*
----+----1	
XYZ	XYZ##
XYZ	#XYZ#
.	##.##
X YZ	#X#YZ

* The character # represents a blank space.

$w.

1) It reads standard characters.
2) w specifies the width of the input field
3) The $w. informat trims leading and trailing blanks
4) The $w. Informat converts a single period in the input data field to a blank before storing values.

Examples

```
input @1 name $5.;
```

Data Line	Results*
----+----1	
XYZ	XYZ##
XYZ	XYZ##
.	
X YZ	X#YZ#

* The character # represents a blank space.

COMMAw.d

1) The COMMAw.d informat reads numeric values and removes embedded commas, blanks, dollar signs, percent signs, dashes, and close parentheses from the input data.
2) w specifies the width of the input field (default is 8)
3) d specifies the power of 10 by which to divide the value. If the data contain decimal points, the d value is ignored
4) It converts an open parenthesis at the beginning of a field to a minus sign

Examples

```
input @1 x comma10.;
```

Data Line	Results
----+----1----+	
$1,000,000	1000000
(500)	-500

w.d

1) It reads standard numeric data.
2) The w.d informat reads values with decimal points and values in scientific E-notation
3) w specifies the width of the input field
4) The w.d informat reads numeric values that are located anywhere in the field. Blanks can precede or follow a numeric value with no effect.
5) A minus sign with no separating blank should immediately precede a negative value.
6) It interprets a single period as a missing value.

Practice 3

If we use the following input statement, what is the corresponding output for the input data below? Write a simple SAS program to verify your answer.

```
INPUT Value 5.1;
```

a) 123
b) 1234
c) 123456
d) 1.25
e) 12.2567

Example 1: date and comma

```
data total_sales;
input Date mmddyy10. +2 Amount comma5.;
datalines;
09/05/2013  1,382
10/19/2013  1,235
11/30/2013  2,391
;
run;
proc print data=total_sales;
title Reading Raw Data not in Standard Format: date and comma;
run;
```

Obs	Date	Amount
1	19606	1382
2	19650	1235
3	19692	2391

Note:

- The MMDDYY10. informat for the variable date tells SAS to interpret the raw data as a month, day, and year, ignoring the slashes. 10 is the length.
- Notice that these dates are printed as the number of days since January 1, 1960. In later chapters, we will talk about how to format these values into readable dates.
- The comma5. informat for the variable amount tells SAS to interpret the raw data as a number, ignoring the comma. Note that the length of the variable is 5.
- The +2 is a pointer control that tells SAS where to look for the next item.

Example 2: standard character and numeric data

```
data example2;
input name $10. Age 3. Height 5.1 BirthDate MMDDYY8.;
datalines;
Jane Matt 35 175.6 03-21-82
Jose Lee    32 172.8 06-15-85
;
run;
proc print data =example2;
title 'Example 2-Reading Raw Data not in Standard Format: standard character and numeric data';
run;
```

Obs	name	Age	Height	BirthDate
1	Jane Matt	35	175.6	17612
2	Jose Lee	32	172.8	17698

Note:

- Name $10. tells SAS to read the first variable “name” from columns 1 through 10.
- Then the starting point for the second variable is column 11, and SAS reads values for “age” in columns 11 through 13.
- The third variable “height” are in columns 14 through 18. It contains a decimal place.
- The last variable “BirthDate” starts in column 19 and is in a date form.

Example 3: standard and non-standard data

```
data contest;
input name $16. Age 3. +1 Type $1. +1 Date MMDDYY10. (Score1 Score2
Score3 Score4 Score5) (4.1);
datalines;
Alicia Grossman 13 c 10-28-1999 7.8 6.5 7.2 8.0 7.9
Matthew Lee       9 D 10-30-1999 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-1999 8.9 7.9 8.5 9.0 8.8
Lori Newcombe    6 D 10-30-1999 6.7 5.6 4.9 5.2 6.1
Jose Martinez      7 d 10-31-1999 8.9 9.510.0 9.7 9.0
Brian Williams    11 C 10-29-1999 7.8 8.4 8.5 7.9 8.0
;
proc print data=contest;
title Example3;
run;
```

Obs	name	Age	Type	Date	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	14545	7.8	6.5	7.2	8.0	7.9
2	Matthew Lee	9	D	14547	6.5	5.9	6.8	6.0	8.1
3	Elizabeth Garcia	10	C	14546	8.9	7.9	8.5	9.0	8.8
4	Lori Newcombe	6	D	14547	6.7	5.6	4.9	5.2	6.1
5	Jose Martinez	7	d	14548	8.9	9.5	10.0	9.7	9.0
6	Brian Williams	11	C	14546	7.8	8.4	8.5	7.9	8.0

Note:

- The variable “name” is a standard character. $16. means it is in column 1 through column 16.
- The variable “age” is also standard numeric data. 3. means it is three columns wide, and has no decimal places.
- The +1 skips over one column.
- $1. means the variable “type” is a standard character which is one column wide.
- MMDDYY10. reads date in the form 10-31-2016 or 10/31-2016, each 10 columns wide.
- Score 1-score5 require the same informat 4.1.
- By putting the variables and the informat in separate sets of parentheses, you only need to list the informat once.

Working with SAS Dates: The FORMAT Statement

If you print a SAS date value, SAS will by default print the actual value-the number of days since January 1, 1960. In most cases, this is not very meaningful. In fact, SAS has a variety of formats for printing dates in different forms. Here is a list of selected SAS date formats.

SAS Date Format	Displayed Date
DATE.	07APR11
DATE9.	07APR2011
DAY.	7
DDMMYY.	07/04/11
DDMMYY10.	07/04/2011
DDMMYYB10.	07 04 2011
DDMMYYC10.	07:04:2011
DOWNAME	Thursday
MMDDYY.	04/07/11
MMDDYY10.	04/07/2011
MMDDYYD10.	04-07-2011
MMDDYYP10.	04.07.2011
MMYYD.	04-2011
MONNAME.	April
MONTH.	4
MONYY.	APR11
WEEKDATE.	Thursday, April 7, 2011
WEEKDAY.	5
WORDDATE.	April 7, 2011
WORDDATX.	07 April 2011

Example: The FORMAT Statement

The FORMAT statement example below tells SAS to print the variable date using the MMDDYY8. format.

```
proc print data=contest;
format date mmddyy8. ;
title Example 3;
run;
```

Obs	name	Age	Type	Date	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	10/28/99	7.8	6.5	7.2	8.0	7.9
2	Matthew Lee	9	D	10/30/99	6.5	5.9	6.8	6.0	8.1
3	Elizabeth Garcia	10	C	10/29/99	8.9	7.9	8.5	9.0	8.8
4	Lori Newcombe	6	D	10/30/99	6.7	5.6	4.9	5.2	6.1
5	Jose Martinez	7	d	10/31/99	8.9	9.5	10.0	9.7	9.0
6	Brian Williams	11	C	10/29/99	7.8	8.4	8.5	7.9	8.0

Next example tells SAS to print the variable date using the worddate18. ;

```
proc print data=contest;
format date worddate18.;
title Example 3;
run;
```

Obs	name	Age	Type	Date	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	October 28, 1999	7.8	6.5	7.2	8.0	7.9
2	Matthew Lee	9	D	October 30, 1999	6.5	5.9	6.8	6.0	8.1
3	Elizabeth Garcia	10	C	October 29, 1999	8.9	7.9	8.5	9.0	8.8
4	Lori Newcombe	6	D	October 30, 1999	6.7	5.6	4.9	5.2	6.1
5	Jose Martinez	7	d	October 31, 1999	8.9	9.5	10.0	9.7	9.0
6	Brian Williams	11	C	October 29, 1999	7.8	8.4	8.5	7.9	8.0

Listing the Contents of a SAS Data Set

An easy way to get a description of a SAS data set is using the CONTENTS procedure. You just need to type the keywords **PROC CONTENTS** and specify the data set you want with the DATA=option. In Example 3 with the contest data set, we should use

```
proc contents data=contest;
run;
```

Description of Data

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	Age	Num	8
4	Date	Num	8
5	Score1	Num	8
6	Score2	Num	8
7	Score3	Num	8
8	Score4	Num	8
9	Score5	Num	8
3	Type	Char	1
1	name	Char	16

Understanding How to Control the Position of the Pointer

There are at least three ways to keep track of the position of the data in the input buffer.

- variable $ n-m: means this variable is in column n through column m
- +n: moves the pointer forward n columns in the input buffer
- @n variable: directs the pointer to move to column n in the input buffer.

Examples: Control the Position of the Pointer

/*The following three SAS program produce the same result. */

/*Example 1: column input*/

```
data january_sales;
input Item $ 1-16  Amount comma5.; datalines;
Trucks        1,382
vans          1,235
sedans        2,391
; run;
```

Obs	Item	Amount
1	trucks	1382
2	vans	1235
3	sedans	2391

/*Example 2: formatted input with the @n pointer*/

```
data january_sales;
input Item $10.  @17  Amount comma5.; datalines;
trucks        1,382
vans          1,235
sedans        2,391
; run;
```

/*Example 3: formatted input with the +n pointer*/

```
data january_sales;
input Item $10. +6  Amount comma5.;
datalines;
trucks        1,382
vans          1,235
sedans        2,391
; run;
```

```
----+----1----+----2----+----3
trucks              1,382
```

Practice 4: Position of Pointers

Follow the instructions from the previous example, please record the following data by using

1. column input
2. formatted input with the +n pointer
3. formatted input with @n pointer.

```
----+----1----+----2

Red             59
```

Obs	Team	Score
1	red	59
2	blue	95
3	yellow	63
4	green	76

Mixing Styles of Inputs

- You are not restricted to use one of the three styles (list, column, or formatted) alone.
- You can mix up input styles in a single INPUT statement as long as it properly record the raw data
- However, each style of input uses the pointer a little differently.

Notes:

1. With list style input, SAS automatically scans to the next non-blank field and starts reading
2. With column style input, SAS starts reading in the exact column you specify. It reads embedded blanks.
3. But with formatted input, SAS just starts reading-wherever the pointer is, that is where SAS reads. You are recommended to use the pointer @n or +n for the formatted input.

Example:

The following raw data contain information about U.S. national parks: name, state (or states as the case may be), year established, and size in acres.

```
----+----1----+----2----+----3----+----4----+----5----+----6
Yellowstone            ID/MT/WY 1872    4,065,493 48
Everglades             FL 1934          1,398,800 48
Yosemite               CA 1864            760,917 48
Great Smoky Mountains NC/TN 1926          520,269 48
Wolf Trap Farm         VA 1966                130 48
```

```
DATA nationalparks;
INFILE '/home/bzeng/my_content/NatPark.dat';
INPUT ParkName $ 1-22 State $ Year @40 Acreage COMMA9.;
RUN;
```

Practice 5: Mix Inputs

Create the data set club1 by following the instructions below:

- Use list input for variables IdNumber, StartWeight, and EndWeight
- Use formmated input for variable Name
- Use column input for variable Team

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

```
----+----1----+----2----+----3----+----

1023 David Shaw              red      189 165
```