Ch2. Getting your Data into SAS

1. Reading row data

1.1 List input

Use: The values in your raw data file are all separated by at least one space

Limitations: (1) Any missing data must be indicated with a period `.`. (2) Character data must be simple: no embedded spaces, and no values greater than 8 characters in length. (3) You must read all the data in a record—no skipping over unwanted values.

Example:

Lucky 2.3 1.9 . 3.0 Spot 4.6 2.5 3.1 .5 Tubs 7.1 . . 3.8 Hop 4.5 3.2 1.9 2.6 Noisy 3.8 1.3 1.8 1.5
Winner 5.7 . . .

```
* Create a SAS data set named toads;
* Read the data file ToadJump.dat using list input;
DATA toads;
       INFILE 'c:\MyRawData\ToadJump.dat';
       INPUT ToadName $ Weight Jump1 Jump2 Jump3;
RUN;
    * Print the data to make sure the file was read correctly;
PROC PRINT DATA = toads;
       TITLE 'SAS Data Set Toads';
RUN;
```

SAS Data Set Toads 1

Obs	Toad Name	Weight	Jump1	Jump2	Jump3
1	Lucky	2.3	1.9	.	3.0
2	Spot	4.6	2.5	3.1	0.5
3	Tubs	7.1	.	.	3.8
4	Hop	4.5	3.2	1.9	2.6
5	Noisy	3.8	1.3	1.8	1.5
6	Winner	5.7	.	.	.

***

1.2 Column input

Use: if each of the variable's values is always found in the same place in the data line

Limitations: All the values are character or standard numeric.

> Standard numeric data contain only numerals, decimal points, plus and minus signs, and E for scientific notation. Numbers with embedded commas or dates, for example, are not standard.

Example:

```
----+----1----+----2----+----3----+----4
Columbia Peaches       35  67  1 10  2  1
Plains Peanuts        210      2  5  0  2
Gilroy Garlics         151035 12 11  7  6
Sacramento Tomatoes   124  85 15  4  9  1
```

```
* Create a SAS data set named sales;
* Read the data file OnionRing.dat using column input;
DATA sales;
       INFILE 'c:\MyRawData\OnionRing.dat';
       INPUT VisitingTeam $ 1-20 ConcessionSales 21-24 BleacherSales 25-28
             OurHits 29-31 TheirHits 32-34 OurRuns 35-37 TheirRuns 38-40;
RUN;
* Print the data to make sure the file was read correctly;
PROC PRINT DATA = sales;
       TITLE 'SAS Data Set Sales';
RUN;
```

```
                          SAS Data Set Sales                          1

                          Concession Bleacher  Our Their  Our Their
    Obs    VisitingTeam      Sales     Sales  Hits  Hits Runs  Runs

     1  Columbia Peaches        35        67     1    10    2    1
     2  Plains Peanuts         210         .     2     5    0    2
     3  Gilroy Garlics          15      1035    12    11    7    6
     4  Sacramento Tomatoes    124        85    15     4    9    1
```

***

1.3 Formatted input

General form: `Input var-name informat`

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

***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
$HEX*w*.	Converts hexadecimal data to character data	1–32,767	2
$*w*.	Reads character data—trims leading blanks	1–32,767	none
Date, Time, and Datetime[1]			
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
TIME*w*.	Reads time in form: *hh:mm:ss.ss* (hours:minutes:seconds—24-hour clock)	5–32	8
Numeric			
COMMA*w.d*	Removes embedded commas and $, converts left parentheses to minus sign	1–32	1
HEX*w*.	Converts hexadecimal to floating-point values if *w* is 16. Otherwise, converts to fixed-point.	1–16	8
IB*w.d*	Reads integer binary data	1–8	4
PD*w.d*	Reads packed decimal data	1–16	1
PERCENT*w*.	Converts percentages to numbers	1–32	6
w.d	Reads standard numeric data	1–32	none

[1] SAS date values are the number of days since January 1, 1960. Time values are the number of seconds past midnight, and datetime values are the number of seconds past midnight January 1, 1960.

Informat	Input data	INPUT statement	Results
Character			
$CHAR*w*.	my cat my cat	INPUT Animal $CHAR10.;	my cat my cat
$HEX*w*.	6C6C	INPUT Name $HEX4.;	11 (ASCII)or %% (EBCDIC)[2]
$*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
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
HEX*w*.	F0F3	INPUT Value HEX4.;	61683
IB*w.d*	[3]	INPUT Value IB4.;	255
PD*w.d*	[3]	INPUT Value PD4.;	255
PERCENT*w*.	5% (20%)	INPUT Value PERCENT5.;	0.05 -0.2
w.d	1234 -12.3	INPUT Value 5.1;	123.4 -12.3

***

[2] The EBCDIC character set is used on most IBM mainframe computers, while the ASCII character set is used on most other computers. So, depending on the computer you are using, you will get one or the other.

[3] These values cannot be printed.

1.4 @ (Column pointer) and : (Column modifier)

`@` : move to a certain column

`@n`	move to column n
`@`	“Stay tuned for more information. Don’t touchthat dial!”
`@'character'`	move to the column after a particular character or word

`:` : read until it encounters a delimiter (default space)

Statements	Value of variable DogBreed
`INPUT @'Breed:' DogBreed $;`	Rottweil
`INPUT @'Breed:' DogBreed $20.;`	Rottweiler Vet Bill
`INPUT @'Breed:' DogBreed :$20.;`	Rottweiler

`&`: have two or more delimiters (default space) between two variables

Example:

```
130.192.70.235 - - [08/Jun/2008:23:51:32 -0700] "GET /rover.jpg HTTP/1.1" 200 66820
128.32.236.8 - - [08/Jun/2008:23:51:40 -0700] "GET /grooming.html HTTP/1.0" 200 8471
128.32.236.8 - - [08/Jun/2008:23:51:40 -0700] "GET /Icons/brush.gif HTTP/1.0" 200 89
128.32.236.8 - - [08/Jun/2008:23:51:40 -0700] "GET /H_poodle.gif HTTP/1.0" 200 1852
118.171.121.37 - - [08/Jun/2008:23:56:46 -0700] "GET /bath.gif HTTP/1.0" 200 14079
128.123.121.37 - - [09/Jun/2008:00:57:49 -0700] "GET /lobo.gif HTTP/1.0" 200 18312
128.123.121.37 - - [09/Jun/2008:00:57:49 -0700] "GET /statemnt.htm HTTP/1.0" 200 238
128.75.226.8 - - [09/Jun/2008:01:59:40 -0700] "GET /Icons/leash.gif HTTP/1.0" 200 98
```

```
DATA weblogs;
      INFILE 'c:\MyWebLogs\dogweblogs.txt';
      INPUT @'[' AccessDate DATE11. @'GET' File :$20.;
RUN;
PROC PRINT DATA = weblogs;
      TITLE 'Dog Care Web Logs';
RUN;
```

```
                          Dog Care Web Logs                          1

                    Obs    AccessDate[3]    File

                     1         17691        /rover.jpg
                     2         17691        /grooming.html
                     3         17691        /Icons/brush.gif
                     4         17691        /H_poodle.gif
                     5         17691        /bath.gif
                     6         17692        /lobo.gif
                     7         17692        /statemnt.htm
                     8         17692        /Icons/leash.gif
```

2. Permanent Data Sets

General form: `LIBNAME libref 'your-SAS-data-library';`

***Example (continued)*:**

```
LIBNAME stat 'c:\MyWebLogs';
DATA stat.weblogs;
      INFILE 'c:\MyWebLogs\dogweblogs.txt';
      INPUT @'[' AccessDate DATE11. @'GET' File :$20.;
RUN;
PROC PRINT DATA = weblogs;
      TITLE 'Dog Care Web Logs';
RUN;
```

> *Temporary data sets:* `DATA WORK.data`

3. PROC IMPORT

***General form*:** `PROC IMPORT DATAFILE = 'filename' OUT = data-set DBMS = identifier REPLACE;`

- `DBMS`:

Type of File	Extension	DBMS Identifier
Microsoft Excel	.xls .xlsx	`EXCEL` or `XLS` `XLSX`
dBase	.dbf	`DBF`
JMP	.jmp	`JMP`
Lotus	.wk4	`WK4`
Paradox	.db	`PARADOX`
SPSS Save file	.sav	`SAV`
Stata	.dta	`DTA`

- `REPLACE`: replace the SAS data set named in the OUT= option if it already exists.

> Microsoft Excel Files: `SHEET="sheet-name";` `RANGE="sheet-name$UL:LR";` `GETNAMES=NO;`

***Example*:**

```
PROC IMPORT DATAFILE = 'J:\CLASSES\STAT46\PAMSAS.xlsx' DBMS=XLSX OUT = PAM
SHEET = 'PAM';
RUN;
```

Equivalent:

```
LIBNAME Lab XLSX PATH='J:\CLASSES\STAT46\PAMSAS.xlsx';

DATA PAM;
   SET Lab.'PAM$';  /*non-numeric sheet name followed with $*/
RUN;
```

4. Contents of the Data Set (descriptor portion)

General form: `PROC CONTENTS DATA = data-set options;` Options:

Statement	Task
`NODS`	Suppress the printing of individual files
`NOPRINT`	Suppress the printing of the output
`OUT=SAS-data-set`	Specify the name for an output data set
`SHORT`	Print abbreviated output
`VARNUM`	Print a list of the variables by their position in the data set. By default, the CONTENTS statement lists the variables alphabetically

Example:

```
DATA funnies (LABEL = 'Comics Character Data');
              INPUT Id Name $ Height Weight DoB MMDDYY8. @@;
              LABEL Id  = 'Identification no.'
                 Height = 'Height in inches'
                 Weight = 'Weight in pounds'
                 DoB    = 'Date of birth';
              INFORMAT DoB MMDDYY8.;
              FORMAT DoB WORDDATE18.;
              DATALINES;
           53 Susie    42  41  07-11-81 54 Charlie  46  55  10-26-54
           55 Calvin   40  35  01-10-81 56 Lucy     46  52  01-13-55
```

```
              ;
* Use PROC CONTENTS to describe data set funnies;
PROC CONTENTS DATA = funnies;
RUN;
```

> `LABEL` or `FORMAT` statement within a PROC step, the label or format applies only to the output from that step.