Inputting Raw Data

Shannon Pileggi

STAT 330



Moving the pointer

OUTLINE

List and Column Input

Formatted Input

Moving the pointer

Overview

So far we have learned three methods for working with data in SAS:

- use a LIBNAME statement to access a SAS library that contains SAS data
- 2. use DATALINES in a DATA step to enter data (this is an example of *list input* with instream data)
- use PROC IMPORT for structured data files like CSV or EXCEL

Today - importing "raw" /less structured data (think .txt or .dat extension) with INFILE.

The Pointer

List and Column Input

0.00000000

- ► SAS has a virtual *pointer* which keeps track of the 'reading' location in the data file
- ▶ Pointer location depends upon method of data input
 - List pointer moves to next non-empty column to begin reading
 - Column pointer moves to the explicitly designated column location
 - Formatted pointer moves to column depending upon specified informat length

INFILE statement

```
DATA mydata;

INFILE "C:/ComputerLocation/datasetname.ext";

INPUT var1 var2 var3;

RUN;

SAS Code
```

- ▶ the INFILE statement specifies the computer location of the data file
 - be sure to include the data set name at the end of the path
 - be sure to include the data set extension at the end of the path
 - the path goes in quotes
- the INPUT statement names the variables

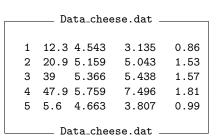
About the cheese data

List and Column Input

0000000000

In a study of cheddar cheese from the LaTrobe Valley of Victoria, Australia, samples of cheese were analyzed for their chemical composition and were subjected to taste tests.

case	sample number						
taste	subjective averaged taste						
	test score						
acetic	natural log of concentration						
	of acetic acid						
h2S	natural log of concentration						
	of hydrogen sulfide						
lactic	concentration of lactic acid						



4□ > 4□ > 4 ≥ > 4 ≥ > ≥ 90

Importing the cheese data

```
DATA cheese;

INFILE "&path.Data_cheese.dat";

INPUT case taste acetic h2s lactic;

RUN;

SAS Code
```

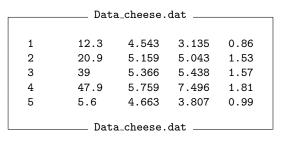
Which method determined the pointer location?

- 1. list
- 2. column
- 3. formatted
- 4. none of these

We can't use list input when we

- do not have a delimiter (ie, no space or comma or something)
 between values
- do not have periods for missing values
- ▶ have non-standard data (ie, dates)
- have data with embedded spaces
- have character variables with width > 8 characters

- ▶ limitations: data must be lined up in the same columns
- advantages: can work with embedded spaces, character variables > 8, missing data indicated by spaces

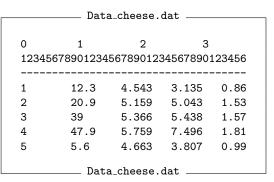


Can we use column input for the cheese data?

- 1. Yes
- 2. No

4□ > 4□ > 4 = > 4 = > = 9 < 0</p>

Column input



In which columns is the second variable (taste) located?

- 1. 2 through 9
- 2. 8 through 13
- 3. 9 through 12
- 4. 9 through 16
- 5. 6 though 16

```
DATA cheese2;
INFILE "&path.Data_cheese.dat";
INPUT case 1-2 taste 9-12 acetic 17-21 h2s 25-29 lactic 33-36;
RUN;
SAS Code
```

after each variable name, specify the numeric range of column position

Discussion

```
SAS Code _____
*Example 3 - change input order;
DATA cheese3:
   INFILE "&path.Data_cheese.dat";
   INPUT lactic 33-36 case 1-2 taste 9-12 acetic 17-21 h2s 25-29:
RUN:
*Example 4 - mix input methods;
DATA cheese4:
   INFILE "&path.Data_cheese.dat";
   INPUT case taste acetic 17-21 h2s 25-29 lactic 33-36;
RUN:
                        __ SAS Code _____
```

On your own: Will these examples work or will there be an error?

4日 > 4日 > 4目 > 4目 > 4目 > 99(で

STAT 330: Lecture 18

Moving the pointer

Formatted Input

•000000

List and Column Input

Formatted Input

Moving the pointe

INFILI

List and Column Input

List and Column Input

Review

Informats are used to read non-standard data.

Character: \$name_of_informatw.

Numeric: name_of_informatw.d

Date: name_of_informatw.

- w is the width of the *entire* field, including special characters
- d is the number of decimals
- period indicates that we are establishing an informat (rather than a variable name)

Note: the default width for character variables is 8.

Discussion

The DOLLARw.d informat removes embedded characters for numeric data.

```
SAS Code
DATA test;
INPUT name $11. money ? ;
DATALINES;
Constantine $15,000.35
Billy
            $8,000.05
            $3,000.63
Sue
            $400.45
Megan
RUN;
            SAS Code _
```

Which is the correct informat for money?

- 1. DOLLAR5.2
- 2. DOLLAR7.2
- 3. DOLLAR10.2
- 4. DOLLAR12.0

STAT 330: Lecture 18

The pointer with formatted input

```
DATA test;
INPUT name $11. money DOLLAR10.2;
DATALINES;
Constantine $15,000.35
Billy $8,000.05
Sue $3,000.63
Megan $400.45
;
RUN;
SAS Code
```

- ➤ SAS looks for name in columns 1 through 11.
- ► The pointer moves to column 12.
- ➤ SAS looks for money in columns 13 through 22.

List and Column Input

Discussion

```
DATA test2;
INPUT name $11. money DOLLAR10.2;
DATALINES;
Constantine $15,000.35
Billy $8,000.05
Sue $3,000.63
Megan $400.45;
RUN;
SAS Code
```

SAS Code -

If my data look like this, will it still import correctly?

- 1. Yes
- 2. No

The colon modifier

List and Column Input

- ► A colon modifies an informat
- examples: :COMMA6. :MMDDYY10. :\$10.
- ► A colon allows you to use an informat for reading data in an otherwise list input process. Why?
 - ▶ If you assign an informat like \$10., SAS will read 10 columns every time, and may include unwanted characters
- ▶ Applying the colon modifier tells SAS to read a value *until* it encounters a space (so it doesn't use a set column position)

The ampersand modifier

- An ampersand modifies an informat
- examples: &COMMA6. | &MMDDYY10. | &\$10.
- Continues to read a character value, even if it contains blanks
- ► Two or more blanks indicates the data value is complete

On your own: How can we correctly read in the test data from the previous slide?

Moving the pointer

Formatted data can cause issues with the pointer because it forces the pointer to look in specific columns. You can manually move the pointer by specifying pointer location <u>before</u> the variable name.

- \triangleright +n Move pointer ahead *n* columns
- On Move pointer directly to column n



age of home number of features NorthEast Location Customer Label

Corner Location

Annual taxes

Formatted Input

List and Column Input

age

numfeat.

CustLab

137,500 STAT 330: Lecture 18

CorLoc

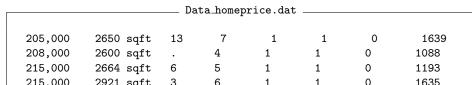
tax

NEloc

On your own: What features do you notice in the data?

Moving the pointer

INFILE



	2000 2420	•	-	-	-	•	2000	
215,000	2664 sqft	6	5	1	1	0	1193	
215,000	2921 sqft	3	6	1	1	0	1635	
199,900	2580 sqft	4	4	1	1	0	1732	
190,000	2580 sqft	4	4	1	0	0	1534	
180,000	2774 sqft	2	4	1	0	0	1765	

156,000 1161 1920 sqft 145,000 2150 sqft

144,900 1710 sqft 1010 1837 sqft 5

Discussion

DATA homes;
INFILE "&path.Data_homeprice.dat";
INPUT price COMMA7. size 12-15 @23 age
 numfeat NEloc CustLab CorLoc tax;
RUN;

SAS Code

Which method(s) did I use to import the data?

- 1. list input
- 2. column input
- 3. formatted input

- 4. list + column5. list + formatted
- 6. column + formatted
 - 7. list + column + formatted

Wrap up

List and Column Input

- ▶ Importing data correctly may take some trial and error and there can be multiple correct methods.
- ▶ TIP: the pointer moves from left to right. Variable 10 will likely not appear correctly if Variable 1 does not appear correctly. Get variables to appear correctly, one at time, according to their input order.
- ► TIP: Always check PROC CONTENTS to make sure that variables have the correct type (numeric vs character).
- ► TIP: If data file is open in a separate application, you may need to close it before you import it.
- ▶ Input features must go before/after variable names

Feature	Before	After	
Column position (e.g., 3-4)	Х	√	
Informat (e.g., COMMA7.2)	X	\checkmark	
+n/@n	\checkmark	< □ X · < 5	p •

4 = > 4 = > = 900

Formatted Input

•0000000000

Moving the pointer

INFILE

List and Column Inpu

Formatted Inpu

Moving the pointer

INFILI

List and Column Input

@"XXXX" column pointer

Data_Dogs.dat _

Name- Kia Breed: Shepherd Vet Bills: 325.25 Name- Sam Breed: Beagle Vet Bills: 478.78 Name- Sydney Breed: Boxer Vet Bills: 733.54 Name- Bugsy Breed: Pug Vet Bills: 518.09

Data_Dogs.dat ____

- When data have a consistent prefix use the @"XXXX" column pointer
- XXXX represents the prefix

What are the *exact* prefixes for...

- 1. Name of the dog
- 2. Breed of the dog
- 3. Amount spent at the vet

Moving the pointer

0000000000

mailing.dat _____

Brenda Smith email: Bsmith@charter.net 123 Grand Ave.

Arroyo Grande CA 93420

David White email: david6060@gmail.com 456 Traffic Wy.

Arroyo Grande CA 93420

Alexandra Jones email: AJJ43@yahoo.com 789 Foothill Blvd.

San Luis Obispo CA 93405

mailing.dat ____

On your own: What features does this data have that we will need to address when importing?

- ▶ Raw data sets typically consist of one observation per line
- ▶ Line pointers tells SAS to to skip to a new line
 - ▶ / skip to next line
 - ▶ #n skip to line n
- This is used to read multiple lines of data into a single observation

Reading the mail data

```
____ SAS Code _____
DATA mailing;
INFILE "&path.mailing.dat";
INPUT fname :$10. lname $ @"email: " email :$20.
     / street &$30.
     / city &$30. state $2. zip ;
/*this also works:
INPUT fname :$10. lname $ @"email: " email :$20.
     #2 street &$30.
     #3 city &$30. state $2. zip;
*/
RUN:
                        SAS Code _____
```



Double trailing at @@

- ► SAS assumes that a single line of data corresponds to a single observation
- If a single line of a data corresponds to multiple observations, need to use
- Tells SAS to keep reading data into new observations until it runs out

Trailing at 0

- Use when interested in specific records from raw data
- Tells SAS to wait for more information
- Syntax is typically
 - input statement with @
 - if-then statement to select obs
 - new input statement

@@ example

```
_____ baggagefees.dat _____
```

- 1 Delta 863,608 2 American 593,465
- 3 US Airways 506,339 4 Continental 353,416
- 5 United 276,817 6 AirTran 164,670
- 7 Alaska 157,013 8 Spirit 133,970 9 JetBlue 64,078 10 Hawaiian 56,590
- 11 Frontier 54,862 12 Allegiant 53,562
- 13 Virgin America 33,482 14 Southwest 32,035
- 15 Sun Country 13,398 16 Mesa 1,683
- 17 USA 3000 1,650

baggagefees.dat _____

SAS Code _____

DATA baggage ; INFILE "&path.baggagefees.dat";

INPUT rank airline &\$20. revenue :COMMA. @@;

@ example

List and Column Input

```
potassium.dat _____
Mollusk, clams
                     534 85 3 oz.
Cod
                     439
                          85 3 oz.
Halibut
                     490 85 3 oz.
Salmon
                     319 85 3 02.
                     375 85 3 oz.
Trout
                     484 85 3 oz.
Tuna
Apricots, dried
                     814 70 10 med.
```

Objective: retain observations with potassium (K) greater than 500

```
DATA potassium;
INFILE "&path.potassium.dat";
INPUT @21 K COMMA5. @;
IF K < 500 THEN DELETE;
INPUT food $ 1-20 @28 weight measure &$10.;
RUN;
```

Discussion

Which symbol best represents the opposite operation performed by @@?

- 1. @
- 2. @"XXX"
- 3. @n, +n
- 4. /, #n
- 5. &
- 6. :

Moving the pointer

Formatted Input

List and Column Input

Formatted Inpu

Moving the pointe

INFILE

List and Column Input

- ► FIRSTOBS= tells SAS at which line to *start* reading useful to skip variable names
- ▶ OBS= tells SAS at which line to stop reading useful to read in part of data file
- ► FLOWOVER default value; SAS jumps to next line if current line does not have enough values to read. After a jump, SAS reads the next line regardless of whether it has enough values.
- ► MISSOVER Set a variable to missing value if missing or if length is too short
- ► TRUNCOVER Allows SAS to handle data values of varying lengths appropriately with column or formatted input

```
names.txt

FirstName LastName
Allison Allen
Billy Bryson
Carmen Cottle
David Decker
Enrique Edwards
Faith Firth

names.txt
```

```
What values should I use below to read in the A through D names?
```

- 1. 1, 5
 2. 2, 5
- 3. 1. 4
- 4. 2. 4
- **5**. 4, 2

```
SAS Code

DATA names;
INFILE "&path.names.txt" FIRSTOBS= OBS=;
INPUT fname $ lname $;
RUN;

SAS Code
```

Try it

List and Column Input

```
SAS Code _
%MACRO checkoptions(option);
DATA test:
   INFILE "&path.numbers.txt" &option;
   INPUT numbers 6.;
RUN;
TITLE "&option";
PROC PRINT; RUN;
%MEND;
%checkoptions(flowover);
%checkoptions(missover);
%checkoptions(truncover);
              SAS Code
```

On your own: Try changing the 6. numeric format to

1. (nothing)

2. :6.

What differences do you observe?

- ► A delimited file contains a specific character that separates data values
- with list input, SAS assumes the delimiter is a space
- ▶ DLM= option in the infile statement specifies the delimiter
 - ▶ DLM=',' for commas
 - ▶ DLM='09'x for tabs
- ▶ by default, SAS assumes that ≥ 2 delimiters in a row is a single delimiter; to override this, the $\boxed{\texttt{DSD}}$ option:
 - treats two delimeters in a row as a missing value
 - ignores delimiters enclosed in quotes
 - does not read quotes as part of the data value



beer.csv ____

INFILE

Example

List and Column Input

```
Consumption per capita [1],,,
Country, Consumption (liters), 20092010 (change 633-ml bottles), Total
Vietnam, 19,,
Venezuela, 83, -4.7, 2259
Uzbekistan, 11,,
United States, 78, -2.5, 24138
United Kingdom, 74, -3.4, 4587
```

____ beer.csv _____

```
DATA beer;
INFILE "&path.beer.csv" DSD FIRSTOBS=3 DLM=",";
INPUT country :$30. consumption change total;
RUN;

SAS Code
```

STAT 330: Lecture 18 39 / 38