

Lecture 12

1. Reading spreadsheets with problems
2. CSV form of data
3. INFORMATS and FORMATS
4. Structuring programs and project code
5. Review of Proc GLM models

1

Reading spreadsheets with problems

Download `bad_spreadsheet.xls` from the course website.

	A	B	C	D	E	F	G	H	I	J
1	subject	city	gender	BMI	visit date	0 min	30 min	120 min	PredDose	Lymphs
2	0001065	Mankato	F	25.6X	lost	12	NA	105	5	0.2
3	0001066	Minneapolis	M	27.8		15		112	10	NA
4	0001067	Cedar Rapids Iowa	male	31.2	11/13/2002	7	61.00	220	0	1.8
5	0001068	Kansas City MO	F	29.4	10/24/81	19	143.00	386	10	0.7*

2

Import the spreadsheet into SAS as "import_xls" using *SAS Import Wizard*.

Equivalent: Proc IMPORT (see *LSB* §2.16–2.17).

From the SAS log:

NOTE: PH6470.BAD_XLS data set was successfully created.

```
Proc PRINT data = PH6470.bad_xls;
```

```
Proc CONTENTS data = PH6470.bad_xls;
```

3

```

                                v
                                i
                                s
                                P
s      -      i      r
u      -      g      t      2 e L
b      e      -      0 0 d y
j      c      n      d      - - - D m
0 e    i      d      B a m m m o p F F F
b c    t      e      M t i i i s h 1 1 1
s t    y      r      I e n n n e s 1 2 3

1 0001065 Mankato      F      . lost      12      . 105 5 0.2
2 0001066 Minneapolis  M      27.8      15      . 112 10 .
3 0001067 Cedar Rapids Iowa male 31.2      7 61 220 0 1.8
4 0001068 Kansas City MO F      29.4 10/24/81 19 143 386 10 .
5

```

	A	B	C	D	E	F	G	H	I	J
1	subject	city	gender	BMI	visit date	0 min	30 min	120 min	PredDose	Lymphs
2	0001065	Mankato	F	25.6X	lost	12	NA	105	5	0.2
3	0001066	Minneapolis	M	27.8		15		112	10	NA
4	0001067	Cedar Rapids Iowa	male	31.2	11/13/2002	7	61.00	220	0	1.8
5	0001068	Kansas City MO	F	29.4	10/24/81	19	143.00	386	10	0.7*

4

Data Set Name		PH6470.BAD_XLS		Observations		5
Member Type		DATA		Variables		12
#	Variable	Type	Len	Format	Informat	Label
3	BMI	Num	8			BMI
10	F10	Char	1	\$1.	\$1.	F10
11	F11	Char	1	\$1.	\$1.	F11
12	F12	Char	1	\$1.	\$1.	F12
9	Lymphs	Num	8			Lymphs
8	PredDose	Num	8			PredDose
6	_0_min	Num	8			30 min
7	_20_min	Num	8			120 min
5	__min	Num	8			0 min
2	city	Char	17	\$17.	\$17.	city
1	subject	Num	8			subject
4	visit_date	Char	8	\$8.	\$8.	visit date

5

Fixing the CSV file

When the Data Import Wizard cannot correctly read an Excel spreadsheet, save it as a CSV file and tell SAS exactly how to read each variable.

CSV means *Comma Separated Values*: each vertical line separating columns is replaced with a comma

- Check green marks on cells
- Use *Find and Select* (search and replace) within columns to remove characters from numeric data (<0.02, 358*)

In Excel save the spreadsheet in CSV format as bad_spreadsheet_1.csv.

Viewed in Excel, we have lost the leading zeros in the subject IDs:

	A	B	C	D	E	F	G	H	I	J
1	subject	city	gender	BMI	visit date	0 min	30 min	120 min	PredDose	Lymphs
2	1065	Mankato	F	25.6X	lost	12	NA	105	5	0.2
3	1066	Minneapo	M	27.8		15		112	10	NA
4	1067	Cedar Ra	male	31.2	11/13/2002	7	61	220	0	1.8
5	1068	Kansas C	F	29.4	10/24/1981	19	143	386	10	0.7*

They're not really gone.

Here is bad_spreadsheet_1.csv viewed in a text editor:

```
subject,city, gender,BMI,visit date,0 min,30 min,120 min,PredDose,Lymphs,,
0001065,Mankato,F,25.6X,lost,12,NA,105,5,0.2,,
0001066,Minneapolis,M,27.8,,15,,112,10,NA,,
0001067,Cedar Rapids Iowa,male,31.2,11/13/2002,7,61.00,220,0,1.8,,
0001068,Kansas City MO,F,29.4,10/24/81,19,143.00,386,10,0.7*,,,
,,,,,,,,
```

7

Reading a .CSV file (LSB §2.15)

Use a data step to read in the CSV file:

```
Data new;
```

```
  INFILE "path to file" firstobs=2 DLM="," DSD missover
    lrecl=100;
```

```
  INPUT list of variables, with required INFORMATS;
```

8

These are the options for the INFILE statement:

`firstobs=2` skip the first line with variable names and start reading at line 2

`DLM =", "` specifies the *delimiter*, the thing that separates variables (a comma)

`DSD` treats `,,` as a missing value

`missover` If there are more variables to read at the end of the data line, set them to missing instead of continuing on to the next line for them.

`lrecl` = logical record length. If your data lines are longer than 100 characters, use a large number for this. SAS log will tell you the actual maximum record length and you can correct it.

9

How to get the path to the file

Find the file. While its folder is open as the current window:

Windows7: Hold the shift-key and right-click on the file. In the context menu that appears, select *Copy as Path*. The path to the file is copied in the clipboard.

Paste the path into your program, check that the file's name is included at the end.

Windows XP: Right-click on the file. In the context menu that appears, select *Properties*. Copy the path at Location.

Paste the path into your program, and add the name of the file.

10

Data new;

```
INFILE "path to file" firstobs=2 DLM="," DSD missover  
lrecl=100;
```

```
INPUT list of variables, with required INFORMATS;
```

INFORMAT is a format for input data

FORMAT is a format for output data

See *LSB §2.8* for table of useful INFORMATS.

11

Character data: specify width = number w of characters. $\$w.$

Trims leading blanks.

Colon ($:\$w.$) read until w characters or delimiter

Dates:

$DATEw.$ reads dates in form $ddmmmyy$ or $ddmmmyyyy$

e.g. $DATE10.$ reads 05sept1998 or 1 JAN 07

$MMDDYYw.$ read dates in form $mm/dd/yy$ or $mm-dd-yyyy$

e.g. 05/21/1998 or 02-01-74

12

```

data read_csv;
  infile
"C:\Documents and Settings\Administrator\Desktop\bad_spreadsheet1.csv"
  DSD dlm="," firstobs=2 missover lrecl=300;

input  subject :$8. city :$17. gender $ BMI visit_date :mddy10.
      min0 min30 min120 PredDose Lymphs;

if (subject NE ''); * delete blank lines;
format  visit_date mddy10. ;

```

13

```

v
i
s
P
s
i
r
L
t
m
e
D
m
j
c
n
d
m
i
n
D
m
0
e
i
d
B
a
i
n
1
o
p
b
c
t
e
M
t
n
3
2
s
h
s
t
y
r
I
e
0
0
0
e
s

1 0001065 Mankato F . . 12 . 105 5 0.2
2 0001066 Minneapolis M 27.8 . 15 . 112 10 .
3 0001067 Cedar Rapids Iowa male 31.2 11/13/2002 7 61 220 0 1.8
4 0001068 Kansas City MO F 29.4 10/24/1981 19 143 386 10 .

```

	A	B	C	D	E	F	G	H	I	J
1	subject	city	gender	BMI	visit date	0 min	30 min	120 min	PredDose	Lymphs
2	1065	Mankato	F	25.6X	lost	12	NA	105	5	0.2
3	1066	Minneapo	M	27.8		15		112	10	NA
4	1067	Cedar Ra	male	31.2	11/13/2002	7	61	220	0	1.8
5	1068	Kansas C	F	29.4	10/24/1981	19	143	386	10	0.7*

14

NOTE: Invalid data for BMI in line 2 19-23 .

NOTE: Invalid data for visit_date in line 2 25-28.

NOTE: Invalid data for min30 in line 2 33-34.

```
RULE:      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
2          0001065,Mankato,F,25.6X,lost,12,NA,105,5,0.2,,, 47
subject=0001065 city=Mankato gender=F BMI=. visit_date=. min0=12 min30=.
min120=105 PredDose=5 Lymphs=0.2 _ERROR_=1 _N_=1
```

NOTE: Invalid data for Lymphs in line 3 40-41.

```
3          0001066,Minneapolis,M,27.8,,15,,112,10,NA,,, 45
subject=0001066 city=Minneapolis gender=M BMI=27.8 visit_date=. min0=15 min30=.
min120=112 PredDose=10 Lymphs=. _ERROR_=1 _N_=2
```

NOTE: Invalid data for Lymphs in line 5 57-60.

```
5          0001068,Kansas City MO,F,29.4,10/24/81,19,143.00,386,10,0.7*,,, 63
subject=0001068 city=Kansas City MO gender=F BMI=29.4 visit_date=10/24/1981
min0=19 min30=143 min120=386 PredDose=10 Lymphs=. _ERROR_=1 _N_=4
```

NOTE: 5 records were read from the infile "C: . . bad_spreadsheet1.csv".

The minimum record length was 12.

The maximum record length was 67.

NOTE: The data set WORK.READ_CSV has 4 observations and 10 variables.

15

Fix original XLS file: remove X, *, NA, lost

Re-save as CSV

If you edit the CSV file and re-save, the leading zeros on IDs are lost.

```
data read_csv;
  infile
"C:\Documents and Settings\Administrator\Desktop\bad_spreadsheet2.csv"
  DSD dlm="," firstobs=2 missover lrecl=300;
input  subject :$8. city :$17. gender $ BMI visit_date :mmdyy10.
      min0 min30 min120 PredDose Lymphs;
if (subject NE ''); * delete blank lines;
format  visit_date mmdyy10. ;
```

16

```

v
i
s
i
r
P
s
u
b
j
e
c
t
g
e
n
d
e
r
B
M
I
v
i
s
i
t
_
d
a
t
e
0
m
i
n
3
0
m
i
n
1
2
0
m
i
n
P
r
e
d
D
o
s
e
L
y
m
p
h
s
1
0
0
0
1
0
6
5
M
a
n
k
a
t
o
F
2
5
.
6
.
1
2
.
1
0
5
5
0
.
2
2
0
0
0
1
0
6
6
M
i
n
n
e
a
p
o
l
i
s
M
2
7
.
8
.
1
5
.
1
1
2
1
0
.
3
0
0
0
1
0
6
7
C
e
d
a
r
R
a
p
i
d
s
I
o
w
a
m
a
l
e
3
1
.
2
1
1
/
1
3
/
2
0
0
2
7
6
1
2
2
0
0
1
.
8
4
0
0
0
1
0
6
8
K
a
n
s
a
s
C
i
t
y
M
O
F
2
9
.
4
1
0
/
2
4
/
1
9
8
1
1
9
1
4
3
3
8
6
1
0
0
.
7

```

	A	B	C	D	E	F	G	H	I	J
1	subject	city	gender	BMI	visit date	0 min	30 min	120 min	PredDose	Lymphs
2	1065	Mankato	F	25.6		12		105	5	0.2
3	1066	Minneapolis	M	27.8		15		112	10	
4	1067	Cedar Rapids	male	31.2	11/13/02	7	61	220	0	1.8
5	1068	Kansas City	F	29.4	10/24/81	19	143	386	10	0.7

No errors in log file. Are we done?

17

Editing the SAS file:

1. Make corrections in data step.

```
IF gender = "male" then gender="M";
```

2. If necessary, convert variables from numeric to character:

```
new variable = INPUT(old_variable, informal);
```

or character to numeric:

```
new variable = PUT(old_variable, informal);
```

See LSB §10.8

18

Structuring SAS programs

The primary aim in writing SAS code is to get a job done. But the job is not just to produce some output. You should structure your SAS programs:

- make finding errors as easy as possible
- make the code easy to understand later by yourself or someone else
- document the time and stage of the analysis
- document data edits and corrections

Most projects have an initial data cleaning phase, followed by an analysis to write a paper. After the paper is submitted, there is a delay until the referee reports arrive, followed by another analysis with revisions.

Interruptions of weeks or months in a project are common—write your code to make it easy to pick up later.

19

1. List program name, date, author, and revision date(s), and *what the program does* at the top.

If it writes a permanent dataset, describe the dataset and list the sources (eg. path to spreadsheet)

Use comments to break code into sections.

20

```
options ls=80 nodate pageno=1 NOFMterr;

* vitd00.sas 16 Feb 2011;
* reads VitD_PrelimPFTs.XLS as cfvitd.data00

writes cfvitd.data01 with n=678 vit D measurements (multiple per child)

      CFVITD.BASELINE has 165 observations,
      only use vit A, E within 1 yr of vit D

cfvitd.MRN_CFFid = link between CFF_ID and MRN

4-year changes in PFT found in vitd03.sas;

proc contents data=cfvitd.data00;
* Vit D measurement is 'Result1', character variable with '<n' for many entries;
```

21

2. Edit the data and create new variables in a single data step at the beginning of the program, as far as possible. This makes it easy to find and correct problems.
3. Use as few data steps as possible. Data steps are the most confusing part of a program.

22

4. Use comments to explain data edits and identify data problems. Hard code all data corrections in these programs.

If there is email or other documentation, include it within a comment in the code, so you don't need to search your mail files later to figure out what happened.

5. Try to keep programs only a few pages long. Makes them easier to debug.

23

This advice also applies to a collection of programs written for a project:

1. Number the programs within their names as you write them:

PlanB_01.sas, PlanB_02.sas, etc.

2. Do all the data editing and creation of permanent datasets in the first program—no analysis.

Hard code all data corrections in these programs.

Include notes and email correspondence as comments in the code.

24

3. Perform analysis in later programs that simply call the permanent datasets.

Analysis programs should only create *temporary* datasets.

Multiple versions of the same data invite trouble.

All data changes are made in the first (data edit) program.

Analysis programs can be run again without modification.

4. Write a “table of contents” file that gives program name and a summary of which datasets it creates or what it does. Keep this up to date.

25

Review of Interactions

Suppose we have a continuous response y , two continuous predictors X_1 and X_2 , and two categorical predictors A_1 and A_2 .

Obs	y	X1	X2	A1	A2
1	3.4	5	212	red	large
2	4.8	8	165	red	small
3	7.1	2	121	blue	small
4	1.4	9	156	blue	small
5	6.6	1	243	blue	large

```
Proc GLM;  
  class A1 A2;  
  model y =
```

26

Describe model $y = X1$

Obs	y	X1	A1	red	blue
1	3.4	5	red	1	0
2	4.8	8	red	1	0
3	7.1	2	blue	0	1
4	1.4	9	blue	0	1
5	6.6	1	blue	0	1

27

Describe model $y = A1$

Obs	y	X1	A1	red	blue
1	3.4	5	red	1	0
2	4.8	8	red	1	0
3	7.1	2	blue	0	1
4	1.4	9	blue	0	1
5	6.6	1	blue	0	1

28

Describe model $y = X1 A1$

Obs	y	X1	A1	red	blue
1	3.4	5	red	1	0
2	4.8	8	red	1	0
3	7.1	2	blue	0	1
4	1.4	9	blue	0	1
5	6.6	1	blue	0	1

29

Describe model $y = X1 A1 X1*A1$

Obs	y	X1	A1	red	blue	X1*red	X1*blue
1	3.4	5	red	1	0	5	0
2	4.8	8	red	1	0	8	0
3	7.1	2	blue	0	1	0	2
4	1.4	9	blue	0	1	0	9
5	6.6	1	blue	0	1	0	1

30

Describe model $y = A1 A2 A2*A1$

Obs	y	A1	red	blue	A2	large	small	large*red	small*red	large*blue	small*blue
1	3.4	red	1	0	large	1	0	1	0	0	0
2	4.8	red	1	0	small	0	1	0	1	0	0
3	7.1	blue	0	1	small	0	1	0	0	0	1
4	1.4	blue	0	1	small	0	1	0	0	0	1
5	6.6	blue	0	1	large	1	0	0	0	1	0

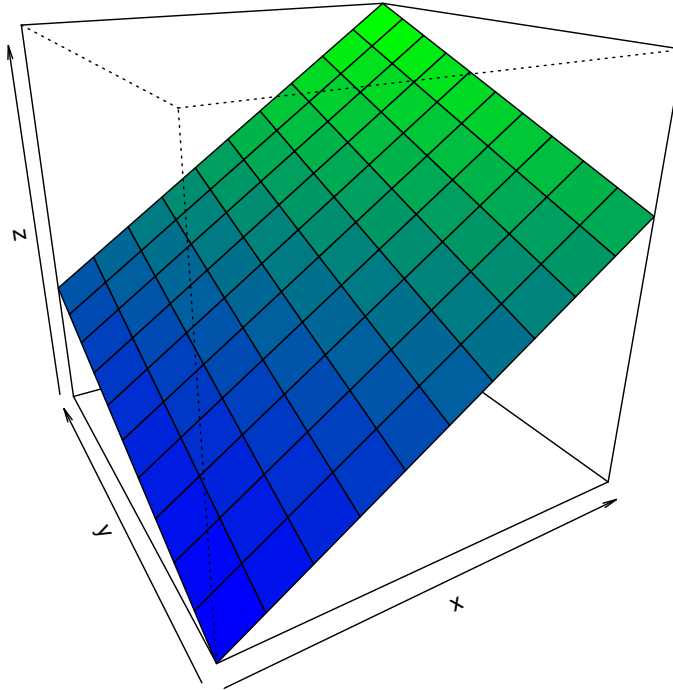
31

Describe model $y = X1 X2$

Obs	y	X1	X2
1	3.4	5	212
2	4.8	8	165
3	7.1	2	121
4	1.4	9	156
5	6.6	1	243

32

model $y = X1 X2$



33

Describe model $y = X1 X2 X1*X2$

Obs	y	X1	X2	X1*X2
1	3.4	5	212	1060
2	4.8	8	165	1320
3	7.1	2	121	242
4	1.4	9	156	1404
5	6.6	1	243	243

34

model $y = X_1 X_2 X_1 * X_2$

