

Lecture 4

1. Editor keyboard shortcuts
2. More on merging
3. First and Last variables
4. Data set options
5. Finding unmatched or discrepant observations in a merge
6. Date and time data

1

Keyboard shortcuts: Tools > Options > Enhanced Editor Keys

- Comment out lines of code using CTRL+/
• Uncomment using CTRL+SHIFT+/
• Create/remove bookmark: CTRL+F2
• Move to next/previous bookmark with F2/SHIFT+F2
• Create abbreviations using CTRL+SHIFT+A
• Convert the selected text to lowercase CTRL+SHIFT+L,
uppercase CTRL+SHIFT+U

Select columns: hold ALT key while selecting

Default list on course website (from SAS)

2

Structural limitations in SAS processing

1. SAS can only work with data contained in a dataset.

Data set: rows are observations, columns are variables.

2. SAS can only process one dataset at a time.

To work with 2 datasets, combine them (merge or set).

3. Within a data step, SAS only works with one observation at a time.

(There are ways to retain information from one observation to the next.)

3

More on MERGE

1. **One-to-one merge:** single observation for each ID in each data set (LSB §6.4)

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

2. **One-to-many merge:** single observation (xbar, SD) in first data set merged with every observation in second data set (LSB §6.5)

4

Many-to-many MERGE: multiple observations per ID

Want to merge two spreadsheets with clinical data recorded at each visit.

Data E			Data F		
id	visit	DBP	id	visit	weight
101	1	77	101	1	145
101	2	75	101	2	149
.	3	80	101	3	152
102	1	71	102	1	181
102	2	74	102	2	176
102	3	68			

5

Let's try code that worked before:

```
proc sort data=E; by id; sort by ID first
```

```
proc sort data=F; by id;
```

```
data G;  
  merge E F;  
  by id;
```

SAS LOG NOTES:

MERGE statement has more than one data set with repeats of BY values.

There were 6 observations read from the data set WORK.E.

There were 5 observations read from the data set WORK.F.

The data set WORK.G has 7 observations and 4 variables.

Extra observation?

6

Data E			Data F		
id	visit	DBP	id	visit	weight
101	1	77	101	1	145
101	2	75	101	2	149
.	3	80	101	3	152
102	1	71	102	1	181
102	2	74	102	2	176
102	3	68			

Data G:

Obs	id	visit	dbp	weight	
1	.	3	80	.	
2	101	1	77	145	
3	101	2	75	149	
4	101	3	75	152	
5	102	1	71	181	
6	102	2	74	176	
7	102	3	68	176	where did these come from?

7

Within each ID, SAS is merging *without any matching variable*.

Sort by as many variables as needed to identify each observation.

Then merge by all of them.

Here, ID and visit are enough.

Data E			Data F		
id	visit	DBP	id	visit	weight
101	1	77	101	1	145
101	2	75	101	2	149
.	3	80	101	3	152
102	1	71	102	1	181
102	2	74	102	2	176
102	3	68			

8

```
proc sort data=E; by id visit;
proc sort data=F; by id visit;
```

```
data K;
  merge E F;
  by id visit;
```

```
proc print data=K;
```

Obs	id	visit	dbp	weight
1	.	3	80	.
2	101	1	77	145
3	101	2	75	149
4	101	3	.	152
5	102	1	71	181
6	102	2	74	176
7	102	3	68	.

9

First and Last BY variables (LSB §6.15)

When you MERGE BY *variable* or SET BY *variable*, two secret variables are created:

`first.variable` = 1 for the first observation with this value of variable, = 0 otherwise

`last.variable` = 1 for the last observation with this value of variable, = 0 otherwise

Obs	id	visit	weight	first_id	last_id
1	101	1	145	1	0
2	101	2	149	0	0
3	101	3	152	0	1
4	102	1	181	1	0
5	102	2	176	0	1

These variables are temporary, and exist only during the data step.

Make a copy to save them.

Data F		
id	visit	weight
101	1	145
101	2	149
101	3	152
102	1	181
102	2	176

```
data H;
```

```
  set F;
```

```
  by id; this creates first.id, last.id
```

```
  first_id = first.id; make a copy to save these variables
```

```
  last_id = last.id;
```

11

How many observations per ID?

We can use first.ID and last.ID to count the number of observations for each ID.

Make a variable called count and increment by 1 for each observation.

```
  if (first.id = 1) then count=0;
```

```
  count = count + 1;
```

```
  if (last.id = 1) then output ; = write to new dataset
```

Output data will have one observation per ID, with count = number of obs in original data

12

Obs	id	visit	weight	count
1	101	3	152	.
2	102	2	176	.

Problem: SAS starts processing an observation by setting all variable to missing, then reading the new observation.

count is set to missing after each observation.

We want to hold on to the value of count from one observation to the next, breaking the rule about processing only one observation at a time.

13

RETAIN statement (LSB §3.10)

RETAIN variable holds the value of variable from the last observation

```

data M; * count obs per ID;
  set F;
  by id;

  retain count;
  if (first.id = 1) then count=0;
  count = count + 1;
  if (last.id = 1) then output;

```

14

Obs	id	visit	weight	count
1	101	3	152	3
2	102	2	176	2

Values of visit and weight are from observation with (last.ID = 1)

```

data M; * count obs per ID;
  set F;
  by id;
  retain count;
  if (first.id = 1) then count=0;
  count = count + 1;
  if (last.id = 1) then output;
  keep id count; list variables to keep in data

```

Also `drop` statement: use the one that has a shorter list

15

Data Set Options (LSB §6.11)

Data set options change the way SAS reads a data set.

Option(s) go in parentheses after the data set name:

```

Data new_data;
  set old_data (option) ;

```

1. `Options that work on rows` (observations):

`FIRSTOBS` = tells SAS where to start processing the data

`OBS` = tells SAS where to stop processing the data

Need `FIRSTOBS < OBS` to read any observations

16

Very helpful in printing part of a long dataset:

```
Proc Print data = ph6470.child_IQ (obs = 5) ;
```

Obs	ID	child_IQ	mom_HS_ grad	mom_age	mom_IQ	male
1	1	65	1	27	121	1
2	2	98	1	25	89	1
3	3	85	1	27	115	0
4	4	83	1	25	99	1
5	5	115	1	27	93	0

17

2. Options that work on columns (variables):

DROP = *variable list*: specifies *variables* to be excluded

Data B;

```
set A (drop = First_Name Last_Name); no commas
```

KEEP = *variable list*: specifies *variables* to be included

```
RENAME = ( oldname1=newname1 oldname2=newname2 )
```

changes the names of variables oldname1 and oldname2.

Data B;

```
set A (rename = (mask_patient = ID)); double parentheses
```

18

`IN` = *new-variable name*:

Creates an indicator variable that tells whether the current observation has variables from this data set or not.

This variable only exists during the dataset: if you print the data it won't appear. To keep it, create a new variable equal to the IN-variable.

Merge A and B, but keep track of which datasets contribute to each row.

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

19

```
data D;
  merge A (in = in_a)  B (in = in_b);
  by id;
  save_in_a = in_a; save_in_a will appear in D, in_a will not
  save_in_b = in_b;
  in_both = (in_a=1 AND in_b=1);
```

```
Proc Print data=D;
```

Obs	id	color	mass	pH	save_ in_a	save_ in_b	in_both
1	12	orange	3650	.	1	0	0
2	13	blue	11267	7.8	1	1	1
3	14		3568	8.2	0	1	0
4	15	yellow	4103	5.1	1	1	1

20

Finding unmatched observations in a MERGE

An “unmatched observation” appears in only one of the data sets being merged:

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

We would like to merge A with B by `id`, and then get a list of the unmatched observations that tells which data set they came from.

We use a dataset option (`IN = variable`) to create an indicator that labels the source.

21

```
data D;
  merge A (in = in_a)  B (in = in_b);
  by id;
  save_in_a = in_a;
  save_in_b = in_b;
  in_both = (in_a=1 AND in_b=1);
```

```
Proc Print data=D;
```

Obs	id	color	mass	pH	save_ in_a	save_ in_b	in_both
1	12	orange	3650	.	1	0	0
2	13	blue	11267	7.8	1	1	1
3	14		3568	8.2	0	1	0
4	15	yellow	4103	5.1	1	1	1

22

To get a list of the unmatched observations:

```
data D;  
  merge A (in = in_a)  B (in = in_b);  
  by id;  
  save_in_a = in_a;  
  save_in_b = in_b;  
  IF (in_a NE in_b);
```

```
Proc Print data=D;
```

Obs	id	color	mass	pH	save_ in_a	save_ in_b	in_both
1	12	orange	3650	.	1	0	0
2	14		3568	8.2	0	1	0

23

Finding discrepant values for the same observation in a MERGE

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

List observations that have discrepant values for mass.

What about this?

```
data I;  
  merge A  B ;  
  by id;  
  if (mass NE mass);
```

24

We need to distinguish mass variables from each data set. Instead of creating new datasets and renaming these variables, use dataset option:

```
Proc Sort data = A; by id;
Proc Sort data = B; by id;
data E;
  merge A(rename = (mass = mass_a)) B(rename = (mass = mass_b)) ;
  by id;
Proc Print;
```

Obs	id	color	mass_a	mass_b	pH
1	12	orange	3650	.	.
2	13	blue	3877	11267	7.8
3	14		.	3568	8.2
4	15	yellow	4103	4103	5.1

25

List of observations with discrepant values:

```
data E;
  merge A(rename = (mass = mass_a)) B(rename = (mass = mass_b)) ;
  by id;
  if (mass_a NE mass_b) ;
```

```
Proc Print data = E;
```

Obs	id	color	mass_a	mass_b	pH
1	12	orange	3650	.	.
2	13	blue	3877	11267	7.8
3	14		.	3568	8.2

26

Working with dates in SAS (LSB §3.8–3.9)

SAS converts a calendar date to an integer = number of days from 1 January 1960 to the date.

SAS maps time to the real number line by setting midnight, 1 January 1960, to zero.

Example: 11 June 1995 is represented as 12945. Units are days.

27

Getting dates in and out of SAS

Most basic way to input data is to list it inside a data step.

Good example, because shows all the moving parts.

```
Data one;  
  INPUT list of variables ;  
  CARDS; or datalines;  
    data lines  
  ;
```

INPUT statement must list all variables

Character and date variables must have format defined:

SAS calls these INFORMAT (goes with IN-put) (LSB §2.8)

28

```

data one;
  input ID gender $ birthdate ;
  cards;
  4833 F 5/16/1978
  4834 F 7/4/1980
  4855 M 12/14/1988
  ;

```

INFormat for gender goes **after** the variable name:

INFORMAT for character variables: **\$**

default length up to 8 characters, defined by first value (F)

or **\$w.** period is required, w = length in characters

29

INFORMAT for birthdate **MMDDYY10.** (see *LSB §2.8*)

10 characters wide: *mm/dd/yyyy* (slashes count as character)

```

data one;
  input ID gender $ birthdate MMDDYY10. ;
  cards;
  4833 F 5/16/1978
  4834 F 7/4/1980
  4855 F 12/14/1988
  ;

```

Period at the end of **MMDDYY10.** is required,
otherwise SAS thinks it's another variable.

30

Here's what we have now:

```
Proc Print data = one;
```

Obs	ID	gender	birthdate
1	4833	F	6710
2	4834	F	7490
3	4855	F	10575

Values of birthdate are number of days since 1/1/1960.

31

We want SAS to display the dates in MMDDYY format.

For input, use INFORMAT; for output, use FORMAT.

```
data one;
  input ID gender $ birthdate MMDDYY10. ;
  birthdate1 = birthdate; new variable is copy of birthdate
  birthdate2 = birthdate;
  format birthdate1 MMDDYY10. ;
  format birthdate2 WORDDATE. ;
  cards;
  4833 F 5/16/1978
  4834 F 7/4/1980
  4855 F 12/14/1988
  ;
```

32

Here are the two date formats:

ID	gender	birthdate	birthdate1	birthdate2
4833	F	6710	05/16/1978	May 16, 1978
4834	F	7490	07/04/1980	July 4, 1980
4855	F	10575	12/14/1988	December 14, 1988

For date informats, formats see *LSB §3.9*

33

Calculating with dates

The difference between two dates in SAS: `interval = date2 - date1;`
is the number of days between the two dates.

Find a person's age at a clinic visit from `visit_date` and `birth_date`

`(visit_date - birth_date)` (unit is *days*)

convert to years:

`age1 = (visit_date - birth_date) / 365.25;`

`age2 = floor(age1);` `floor(x)` gives the largest integer $\leq x$

34

```

data one;
  input ID gender $ birthdate MMDDYY10. ;
  format birthdate MMDDYY10. ;
  now = TODAY(); today as SAS date
  current_age = (today() - birthdate)/365.25;
  current_age2 = floor(current_age);
  cards;
  4833 F 5/16/1978
  4834 F 7/4/1980
  4855 F 12/14/1988
  ;

```

35

ID	gender	birthdate	now	current_ age	current_ age2
4833	F	05/16/1978	18528	32.3559	32
4834	F	07/04/1980	18528	30.2204	30
4855	F	12/14/1988	18528	21.7741	21

36

Minutes

SAS Import Wizard does not handle time of day (11:15 AM) well

[Convert to CSV, read with INPUT statement to specify format; we'll do this later]

Time within days: SAS converts times to number of seconds since midnight.

*15:45:10 is represented as $56710 = 15*60*60 + 45*60 + 10$. Units are seconds.*

SAS converts date-times to the number of seconds since midnight,
1 January 1960.

37

Date constant (LSB §3.8)

Example: use study start date (March 4, 2008) to calculate time in study

```
Data B;  
  set study.outcomes;  
  study_start = '04MAR2008' D ;  
  study_days = death_date - study_start;  
  
date_constant = 'ddMMMyyyy' D;
```

38