

Class 3

1. Character data, character variables
2. SET and MERGE
3. DO-loops
4. Using output from a procedure: standardizing a variable

1

Data Types

SAS distinguishes three types of data, and has different methods for dealing with each type.

- Numeric data: integer or floating point
- Character data: any combination of letters, numbers, spaces, punctuation—character string
- Date-time data: calendar dates and times of day

2

Character variable is created by

```
color = 'red';    or, color = "red";
```

(matching single or double quotes).

Within a character variable, SAS distinguishes upper and lower case:

"Female" is not equal to "female"

Numeric variables: missing is indicated by a period, dbp = .

Character variables: missing is indicated by no characters, color="

(two single quote marks with no space between them)

3

Length of character variables

Data: $x = 15, 16, 12, 4, 5, 3$

```
data example;  
  set X_data;  
  if (x > 10) then size="big" ;  
  else size="small" ;
```

4

```
Proc Print data = example;
```

Obs	X	size
1	15	big
2	16	big
3	12	big
4	4	sma
5	5	sma
6	3	sma

5

SAS assigns a length (in characters) to each character variable. Default is 8.

When you define a character variable in a data-step statement

```
if (x > 10) then size="big" ;  
else size="small" ;
```

SAS sets the variable-length equal to the length of the first value assigned.

First value is x = 15

⇒ size="big"

⇒ length of size is 3

6

To prevent SAS from truncating, LENGTH statement sets the variable length:

```
data example;
  set X_data;
  LENGTH size $5 ;    $ indicates CHARACTER, number gives length
  if (x > 10) then size="big";
  else size="small";
```

Obs	x	size
1	15	big
2	16	big
3	12	big
4	4	small
5	5	small
6	3	small

See *LSB* §10.13

7

Pasting character variables together

|| or !! is the concatenation operator that combines character strings.

Example: Study measures IQ in children at ages 8, 12, 24, and 30 months.

What is the pattern of missing data?

```
data zero;
  set lead.data ;
  iq8 = "8_"; if mentdi8=. then iq8="__";    mentdiN is IQ score
  iq12 = "12_"; if mentdi12=. then iq12="___";
  iq18 = "18_"; if mentdi18=. then iq18="___";
  iq24 = "24_"; if mentdi24=. then iq24="___";
  IQ_missing_pattern = iq8||iq12||iq18||iq24 ;
```

8

Here are the first 10 observations:

Obs	childid	IQ_missing_ pattern	MENTDI8	MENTDI12	MENTDI18	MENTDI24
1	2	__12_18_24_	.	107	89	84
2	4	_____24_	.	.	.	92
3	6	__12_18_24_	.	90	109	113
4	10	__12_18____	.	94	99	.
5	11	__12_18_24_	.	109	116	118
6	18	__12_18_24_	.	99	87	84
7	21	__12____24_	.	110	.	76
8	22	8_12_18_24_	81	109	105	108
9	24	8_12_____	96	80	.	.
10	39	__12_18____	.	108	113	.

9

Applied to a single variable, Proc Freq makes a table of the values of that variable.

```
Proc Freq data=zero;  
  tables IQ_missing_pattern ;
```

gives count of each missing-data pattern.

The FREQ Procedure

IQ_missing_ pattern	Frequency	Percent	Cumulative Frequency	Cumulative Percent
8_12_18_24_	23	7.40	23	7.40
8_12_18_	2	0.64	25	8.04
8_12_24_	53	17.04	78	25.08
8_12_	55	17.68	133	42.77
8_18_24_	2	0.64	135	43.41
8_24_	7	2.25	142	45.66
8_	37	11.90	179	57.56
_12_18_24_	22	7.07	201	64.63
_12_18_	7	2.25	208	66.88
_12_24_	28	9.00	236	75.88
12	39	12.54	275	88.42
_18_24_	4	1.29	279	89.71
18	4	1.29	283	91.00
24	13	4.18	296	95.18
_	15	4.82	311	100.00

Functions to work with character strings

SUBSTR(variable, position, n) takes substring starting at 'position' and extending for 'n' characters

TRANSLATE replaces characters

Reading in LSB:

§3.3 List of character functions

§10.8 converting character values to numeric

Combining data sets with SET

When only one data set is SET

```
Data new;
  SET old;
```

the data set `new` is a copy of `old`.

When more than one data set is SET, the result is a stack of data sets in which variables with *exactly* the same name are put in one column as a single variable.

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

13

```
Data D;
  SET A A B C;
```

Obs	id	color	mass	pH	mss
1	12	orange	3650	.	.
2	13	blue	3877	.	.
3	15	yellow	4103	.	.
4	12	orange	3650	.	.
5	13	blue	3877	.	.
6	15	yellow	4103	.	.
7	13		11267	7.8	.
8	14		3568	8.2	.
9	15		4103	5.1	.
10	13		.	7.8	11267
11	14		.	8.2	3568
12	15		.	5.1	4103

Variables which are not part of a data set are missing in the combined data.

14

Combining data sets with MERGE

If ID = 15 is the same object in both data sets A and B, then we want one observation (row) with all 15's data.

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

While SET stacks data sets, MERGE aligns rows horizontally using a matching variable identified with `BY` .

15

Both datasets must first be sorted on the BY variable(s):

```
proc sort data=a; by id;
proc sort data=b; by id;
```

```
Data D;
MERGE A B;
BY id;
```

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

Which mass does ID=13 get?

16

```
data D;
  merge A B;
  by id;
```

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

```
data D;
  merge B A;
  by id;
```

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

Second data set over-writes first for shared variables.

17

Merging *without* a BY variable simply matches observations in order—usually wrong because it mixes observations together. No error message in the log file.

```
data D;
  merge A B;
```

Proc Print:

Obs	id	color	mass	pH
1	13	orange	11267	7.8
2	14	blue	3568	8.2
3	15	yellow	4103	5.1

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

18

DO-loops

In computer programming, a DO-loop is a structure that allows the repetition of a section of code.

```
DO j = 1 to 3; j is the "counter" for the loop: 3 repetitions
  x = j * 2;
  y = j + 1;
  z = x * y;
END; signals end of loop
```

Values at each step:

j	x	y	z
1	2	2	4
2	4	3	12
3	6	4	24

19

There is an implicit DO-loop in every data step (LSB §1.4) :

Data two;

```
DO from _N_ = 1 to last-observation;
  set one;
  statement A;
  statement B;
  statement C;
END;
```

`_N_` is the internal variable that counts observations.

Data step is sequential, only looks at one observation at a time.

20

Using output from a procedure: standardizing a variable

We have scores from a questionnaire and would like to standardize them:

$$\text{standardized score } Z = \frac{\text{score} - \text{average score}}{\text{SD of scores}}$$

This is done in a DATA step. We would like to write something like this:

```
data B;  
  set A;  
  Zscore = (score - mean(score) ) / SD(score);
```

Doesn't work. SAS won't know the mean and SD until it has reached the last observation. And it cannot go back and use this information.

21

Solution 1. Calculate mean ($\bar{x} = 96.3$) and SD (7.2). "Hard code" these values:

```
data B;  
  set A;  
  Zscore = (score - 96.3 ) / 7.2 ;
```

What if we get additional observations?

How do you remind yourself to update this?

22

Solution 2. Safer: calculate current mean and SD, output and merge the results.
(LSB §4.10)

```
Proc Means mean stddev data=A;
  var score;
  output out=C mean=xbar stddev=SD;
```

```
Proc Print data=C;
```

Obs	_TYPE_	_FREQ_	xbar	SD
1	0	311	96.3240	7.20066

23

```
Proc Means mean stddev data=A;
  var score;
  output out=C mean=xbar stddev=SD;
```

```
data D;
```

```
merge A C ; no BY. xbar and SD are added to every line of A
```

```
Z_score = (score - xbar)/SD;
```

This code adapts automatically to updates in the list of scores (data A).

SAS needs the summary statistics at the beginning of the data step to standardize.

Proc STDIZE will standardize variables; offers several estimates of center and scale.

24