SAS seminar

April 15, 2003

Åsa Klint

The little SAS book
Chapters 3 & 4

By LD Delwiche and SJ Slaughter

- Data step
  - read and modify data
  - create a new dataset
  - performs actions on rows
- Proc step
  - use an existing dataset
  - produce an output/results
  - performs actions on columns

3.1 Creating and Redefining variables

Creating and redefining variables is straightforward in a SAS data step

variable = expression;

Examples

Newvar=constant;

*Newvar=oldvar * constant;*

Use DROP or KEEP statement to decrease number of variables in your dataset

Ordinary operators

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

Standard mathematical rules of precedence.
Use parentheses to override that order.

3.2 / 3.3 Using SAS functions

- Functions perform calculations or transformations and are used to simplify your work as the programming has already been taken care of by SAS.
- Write *help functions* in the command line in SAS for information on available functions

General form

Variable=function-name(argument, argument, ...);

Note: an argument can be a literal value, variable name or an expression

Example

Newvar=SUBSTR(arg, position, n)

- pnr='19970530-0765' (**13** positions)
- birthdate=substr(pnr, **3**, **6**)
  (from position **3** I get six positions)
  ... which results in ...
- birthdate='970530'
  Note: this is still a CHAR variable

- 3.7 Working with SAS Dates

  The SAS system for working with dates is to count the number of days since January 1, 1960.

Date	SAS date value
Jan 1, 1959	-365
Jan 1, 1960	0
Jan 1, 1961	366
April 15, 2003	15810

  This can then be transformed into something we can interpret by using formats/functions

7

3.4 Using IF-THEN statements

Often we want to create variables on the form

IF condition THEN action;

Example

```
if job='banker' then highses=1;
```

IF condition AND condition THEN action;

Example2

```
if job='banker' and age>65 then ret_banker=1;
```

8

Comparisons and Logical operators

symbol	mnemonic	meaning
=	EQ	equals
¬=, ^=, ~=	NE	not equal
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal
<=	LE	less than or equal
&	AND	all comparisons must be true
|, ¦, !	OR	only one comparison must be true

9

- DO-END statement

  a DO-END statement is useful if you want to make several changes or create new variables for a subgroup or under certain conditions.

  Note that the DO-loop continues until you end it (using END;)

  Example

```
If sex='female' then do;
   if ICD=175 then ovarian=1;
   else ovarian=0;
   age_ovarian=diadat-birthdat;
   end;
Else if sex EQ 'male' then do;
   ...
   ...
   end;
```

10

3.5 Grouping/categorizing observations with IF-THEN-ELSE statements.

Say we want to compare bankers to the other two occupations

```
If job='banker' then highses=1;
   else if job='conductor' or job='driver' then highses=0;
   else highses=. ;
```

When the condition is true, SAS assigns the stated value to highses and then leaves the loop. The last ELSE is a trash-bin: anything that is not covered by the previous conditions is put to missing.

11

- Note that if you use a 'less than' expression, for example X<350, this also includes X=. (missing)
- Make sure the condition part includes all possibilities, else you might get missings or hidden errors.

  Looking at the log reveals the number of missing values created, but there is no indication for observations that were not covered by your programming!

12

- IF-THEN-ELSE statements are used for grouping/categorizing variables

  example

  if 0<=bmi<20 then bmicat=1;

  else if 20<=bmi<25 then bmicat=2;

  else if 25<=bmi then bmicat=3;

  else if bmi=. then bmicat=.;

13

3.6 Subsetting your data

There are two ways to subset a dataset if you only want to use some of the observations in the dataset

Basic form (tells SAS which observations to include):

IF expression;

Alternative (tells SAS which observations to exclude):

IF expression then delete;

14

Chapter 4: sorting, printing and summarizing your data

SAS procedures are used for actions on columns.

- Proc Sort
- Proc Print
- Proc Format
- Proc Means
- Proc Freq

15

4.1 Using SAS procedures

The statements TITLE, FOOTNOTE and LABEL

- titles and footnotes are useful to keep order among your outputs. They stay in effect until you cancel them (by creating a blank title/footnote alt *goptions reset=title*) or end your SAS session.

  Ex

  FOOTNOTE 'SAS seminar April 15, 2003';
- By using the LABEL statement you can create a longer and more detailed (<256 characters) description for each variable.

  Ex

  LABEL HIENG =' Energy-intake per day.
  High=1, Low=0';

16

Subsetting in procedures with the WHERE statement

- The WHERE statement can be used to select a subgroup of your dataset for a procedure in the same way as an IF-statement in a data step.
- The same logical operators as previously shown can be used here. Other useful phrases

  … IS NOT MISSING
  BETWEEN … AND …
  CONTAINS …
  IN (list)

Example

WHERE weight IS NOT MISSING;

17

4.3 Sorting your data with Proc Sort

- Some procedures demands that the data is sorted for them to be able to work. A BY-statement is used in Proc Sort to specify what variable(s) you want SAS to sort your data by.

  By default, the dataset you sort is replaced by your new, now sorted, dataset. However, you can create a new dataset by the OUT-option.

18

Useful options in Proc Sort

- NODUPKEY eliminates any duplicate observations with the same values for the BY-variables. Practical in situations with multiple observations for each individual and you only want, for example, the first diagnosis in your dataset.

  Ex *Proc sort data=cancer NODUPKEY;*
  by pnr diadat;
- By default SAS sorts in ascending order (from lowest to highest alt from A to Z). Add the keyword DESCENDING before the relevant variable to change this.

19

4.4 Printing your data with Proc Print

- Proc Print is used to look at the data in your output window. In its simplest form SAS prints all observations and all variables in your dataset. However, you can restrict and group the output by statements

  VAR variable-list - specifies what variables to print and in what order

  BY variable-list - groups the data into sections. Demands sorting according to BY-variable

20

Changing the appearance of printed values with Formats

- SAS decides what format it thinks is most appropriate for your data in terms of number of decimals, how much space for each value etc. There are many formats for numeric, character and date values.
- Note the period in the format as this distinguishes a format from a variable name.
- General forms of a SAS format

Character	$formatw.	Ex $6.
Numeric	formatw.d	Ex BEST6.3
Date	formatw.	Ex DATE7.

21

Creating your own formats using Proc Format

- Data are often coded with numbers in order to save disc space. In Proc Format you create formats that fits your purposes and you associate them with your data by using a format statement in the procedure.

  Ex *Proc format*
  value sex
  1='male'
  2='female'
  ;

  general form (for assigning format to a variable)
  format varname formatname. ;

22

Note:

- characters must be enclosed in quotes
- several values can be assigned the same value label. Separate them with a comma (,) or use the hyphen (-) for a continuous range.
- the keyword OTHER can be used to assign a value label to any values not listed in the value statement
- the keywords HIGH and LOW can be used in ranges to indicate the highest and the lowest non-missing value for the variable.

23

4.9 Summarizing your data using Proc Means

- Proc Means provides simple statistics for numeric variables so that you get a feel for your data and might discover errors that has occurred while reading in the data.

  Available options (among others)
  - N (number of non-missing observations)
  - NMISS (number of missing observations)
  - RANGE
  - SUM
  - MEDIAN
  - MEAN
  - STDDEV

24

- If you use Proc Means with no other statements, you'll get statistics for all observations and all numeric variables in the dataset.

  To restrict this use statements such as
- *VAR variable-list;*

  specifies which numeric variables to use
- *BY variable-list;*

  Performs separate analyses for each level of variables in the list (note: sorting required)
- *CLASS variable-list;*

  Performs separate analyses for each level of variables in the list. More compact output than in the BY statement (note: no sorting required)

25

- Instead of having the statistics displayed in the output window we can create a new dataset with the selected statistics.
- The NOPRINT option is used to stop SAS from writing in the output-window
- OUTPUT *OUT=newdataset* creates a new dataset

  - The new dataset will contain the variables listed in the output-statistic list, the variables in the by/class-statement (if any) and two new variables (*_TYPE_* and *_FREQ_*)

26

4.11 counting your data with Proc Freq

- Proc Freq is used to produce frequency tables of categorical data.

  General form

  Proc Freq data=dataset;

  tables variable-combinations;

You can create from *one-way* to *n-way* tables.

Options include

* List	* Missing
* Norow	* Nocol
* Nopercent	* Out=dataset

27

28

Next seminar: April 22

THANK YOU FOR LISTENING!

Åsa Klint
Asa.Klint@mep.ki.se

29