

5. More about the SAS language

Use of SAS
January 2013

Lists of variables

- Sometimes you need to refer to many variables at once
- E.g., if you have repeated measurements or just many similar variables

```
proc freq;  
  tables spm1-spm392;  
run;
```

- similar names x1-x20
- All character variables: `_CHARACTER_`
- All numerical variables: `_NUMERIC_`
- All variables: `_ALL_`

Labels

```
libname juul 'p:\sas\data\juul';  
proc freq data=juul.juul2;  
    table tanner;  
run;  
data hope;  
    set juul.juul2;  
    label tanner="Tanner stage";  
run;  
proc freq data=hope;  
    table tanner;  
run;
```

Formats

- Information about how to read or print variable
- Built-in formats (Numerical, dates, character)
- User defined formats (1=Male 2=Female)
- Pretty printouts
- Grouping in tables and analyses

Formats, cont.

- Standard formats: 10.3, best12., E12., \$10., date10., yymmdd10..
- Always contain a dot (Do not forget it!)
- Can be associated permanently with variable in DATA step, or:
- Specified ad hoc with FORMAT statement in PROC steps.
- User defined formats are created and listed by PROC FORMAT

Example with proc format

```
proc format;
  value sexfor 1="male" 2="female";
run;
data a;
  input sex;
  datalines;
  1
  2
  3
  .
  ;
run;
proc print data=a;
run;
proc print data=a;
  format sex sexfor.;
run;
```

Exercise using formats

1. Get the bissau data into SAS using a libname statement
2. Use bissau data, generate a sas program creating formats for variables
 - dead: 1=died 2=Survived
 - bcg: 1=yes 2=No
 - dtp: 1=yes 2=No

Help for the first one:

```
proc format;  
    value deadfmt 1=Died 2=Survived;  
run;
```

3. Make a `proc freq` of the three variables using your formats.

Using formats

Generate data

```
data test;
reply=1; do i=1 to 25; output; end;
reply=2; do i=1 to 21; output; end;
reply=3; do i=1 to 35; output; end;
run;
proc print data=test;
run;
```

Generate format for the variable reply

```
proc format;
value replyfor 1='Yes      '
                2='No       '
                3='Maybe  ';
run;
```

Distribution of the variable reply

```
proc freq data=test;
table reply; run;
```

Formats in PROC and DATA steps

The format used in a proc step

```
proc freq data=test;  
table reply;  
format reply replyfor.;  
run;
```

or the format can be associated with the variable in a data step

```
data testfor;  
set test;  
format reply replyfor.;  
run;
```

now the format will be used every time we use the data: testfor

```
proc freq data=testfor;  
  table reply;  
run;
```

Save data in a permanent data set

```
libname pdrev 'p:\';  
data pdrev.testfor;  
set testfor;  
run;
```

Restart SAS and run the program

```
libname pdrev 'p:\';  
proc freq data=pdrev.testfor;  
table reply;  
run;
```

SAS cannot find the format!

```
proc freq data=pdrev.testfor;  
table reply;  
format reply; *format _all_;  
run;
```

`format _all_;` removes the formats. Can be very useful

A more advanced PROC FORMAT example:

```
proc format;
  value agegrpfmt 0-1="0-1" 2-4="2-4" 5-6="5-6";
run;
proc format fmtlib;run;
proc freq data=afrika.bissau;
  table age;
run;
proc freq data=afrika.bissau;
  table age;
  format age agegrpfmt.;
run;
```

NB: To be able to use your created formats next time you start SAS, you can save the SAS code in a file, and run this the next time you will use the data set.

Date formats

- In the SAS data set `bissau2.sas7bdat` (in the `africa` directory) the first 200 observations are from the original Bissau data. Variables are:

`id` = ID of child
`dob` = Date of birth
`visitdate` = Date of visit
`agedays` = Age in days at visit

```
proc print data=afrika.bissau2;  
run;
```

| Obs | ID | dob | visitdate | agedays |
|-----|---------|----------|-----------|---------|
| 1 | 2190104 | 13/05/91 | 11/11/91 | 182 |
| 2 | 1150302 | 08/09/90 | 11/01/91 | 125 |
| 3 | 1182208 | 01/11/90 | 09/01/91 | 69 |
| 4 | 1182215 | 05/10/90 | 09/01/91 | 96 |
| 5 | 1210207 | 01/09/90 | 10/01/91 | 131 |

How does SAS handle dates?

```
proc contents data=afrika.bissua2;  
run;
```

| # | Variable | Type | Len | Format | Informat | Label |
|---|-----------|------|-----|----------|----------|---------------------|
| 1 | ID | Num | 8 | 7. | 7. | ID |
| 4 | agedays | Num | 8 | | | Age at visit (days) |
| 2 | dob | Num | 8 | DDMMYY8. | YYMMDD8. | Date of birth |
| 3 | visitdate | Num | 8 | DDMMYY8. | YYMMDD8. | Date of visit |

Variables `dob` and `visitdate` are numerical and have format `DDMMYY8.` - first the day, then the month, then the year, total length is 8. `Informat` was used when reading the data into SAS.

We remove the formats:

```
proc print data=afrika.bissau2;  
format _all_;  
run;
```

| Obs | ID | dob | visitdate | agedays |
|-----|---------|-------|-----------|---------|
| 1 | 2190104 | 11455 | 11637 | 182 |
| 2 | 1150302 | 11208 | 11333 | 125 |
| 3 | 1182208 | 11262 | 11331 | 69 |
| 4 | 1182215 | 11235 | 11331 | 96 |
| 5 | 1210207 | 11201 | 11332 | 131 |

**Actual value stored is the number of days since 1 January 1960:
Dates are numerical variables**

Working with dates

As dates internally are stored as days since 1 Jan 1960, one can add and subtract dates and constants:

Assume Bissau children were re-visited March 1st, 1992. We want to calculate number of years since `vistidate`.

```
data prv;
  set afrika.bissau2;
  visit3=mdy(3,1,1992);
  time_in_days=visit3-visitdate;
  time_in_years=time_in_days/365.25;
run;

proc print data=prv;
run;
```

| Obs | ID | dob | visitdate | agedays | visit3 | time_in_ days | time_in_ years |
|-----|---------|----------|-----------|---------|--------|------------------|-------------------|
| 1 | 2190104 | 13/05/91 | 11/11/91 | 182 | 11748 | 111 | 0.30390 |
| 2 | 1150302 | 08/09/90 | 11/01/91 | 125 | 11748 | 415 | 1.13621 |
| 3 | 1182208 | 01/11/90 | 09/01/91 | 69 | 11748 | 417 | 1.14168 |
| 4 | 1182215 | 05/10/90 | 09/01/91 | 96 | 11748 | 417 | 1.14168 |
| 5 | 1210207 | 01/09/90 | 10/01/91 | 131 | 11748 | 416 | 1.13895 |

Exercise with dates

In the SAS data set `bissau2.sas7bdat` (in the `africa` directory) the first 200 observations are from the original Bissau data. Variables are:

`id` = ID of child
`dob` = Date of birth
`visitdate` = Date of visit
`agedays` = Age in days at visit

Please, check that the variable `agedays` was correctly calculated.

Appending data sets: SET

more cases, same variables

Two data sets: `other.hosp1` and `other.hosp2`

| Obs | id | sex | bp | weight | bmi | teatment | hospital |
|-----|----|-----|-----|--------|------|----------|----------|
| 1 | 1 | 1 | 110 | 75.8 | 24.5 | 0 | herlev |
| 2 | 2 | 2 | 125 | 89.8 | 26.8 | 0 | herlev |

| Obs | id | day | sex | bp | weight | teatment | hospital |
|-----|----|----------|-----|-----|--------|----------|----------|
| 1 | 3 | 11/11/97 | 2 | 131 | 57.8 | 1 | gentofte |
| 2 | 4 | 12/11/97 | 2 | 121 | 98.8 | 1 | gentofte |

are put together using SET:

```
data all;  
  set other.hosp1 other.hosp2;  
run;  
proc print data=all;  
run;
```

Output from PROC PRINT

| Obs | id | sex | bp | weight | bmi | teatment | hospital | day |
|-----|----|-----|-----|--------|------|----------|----------|----------|
| 1 | 1 | 1 | 110 | 75.8 | 24.5 | 0 | herlev | . |
| 2 | 2 | 2 | 125 | 89.8 | 26.8 | 0 | herlev | . |
| 3 | 3 | 2 | 131 | 57.8 | . | 1 | gentofte | 11/11/97 |
| 4 | 4 | 2 | 121 | 98.8 | . | 1 | gentofte | 12/11/97 |

Note, that missing values are generated for variables not present in all data sets.

Merging data set: MERGE

new variables, same cases. Normally there is a key, say `id`, and all data sets must be sorted by `id`

Data `other.quest` concerns the same patients

```
proc print data=other.quest;  
run;
```

| Obs | id | income | employed |
|-----|----|--------|----------|
| 1 | 3 | 35000 | 1 |
| 2 | 2 | 44000 | 1 |
| 3 | 5 | 25500 | 1 |

We want to merge it with data `all`

Code for merging data sets

```
data quest;  
    set other.quest;  
run;  
proc sort data=quest;  
    by id;  
run;  
proc sort data=all;  
    by id;  
run;  
data all1;  
    merge all quest;  
    by id;  
run;  
proc print data=all1;  
run;
```

Output from PROC PRINT

| Obs | id | sex | bp | weight | bmi | teatment | hospital | day | income | employed |
|-----|----|-----|-----|--------|------|----------|----------|----------|--------|----------|
| 1 | 1 | 1 | 110 | 75.8 | 24.5 | 0 | herlev | . | . | . |
| 2 | 2 | 2 | 125 | 89.8 | 26.8 | 0 | herlev | . | 44000 | 1 |
| 3 | 3 | 2 | 131 | 57.8 | . | 1 | gentofte | 11/11/97 | 35000 | 1 |
| 4 | 4 | 2 | 121 | 98.8 | . | 1 | gentofte | 12/11/97 | . | . |
| 5 | 5 | . | . | . | . | . | . | . | 25500 | 1 |

Note, that subject with `id=5` who is present only in `other.quest` has missing values for variables not in that data set.

Exercise: More about MERGE

In the library 'p:\sas\prg' you will find the file 'exercise_merge11.sas'. Run the first part of the code.

```
data cop;  
  set other.copenhagen;  
run;
```

```
proc print data=cop;  
run;
```

This gives the output:

| Obs | hospital | staff | ranking |
|-----|----------|-------|---------|
| 1 | gentofte | 51 | 13 |
| 2 | herlev | 45 | 5 |

This is a hospital-level data set. We want to merge this data with the individual-level data `all1` so that the information about e.g., `gentofte` hospital is repeated for all patients from that hospital. The following code gives two possible solutions. Run the code and explain the differences.

```
*solution 1;  
data all2;  
  merge other.all1 cop;  
run;  
proc print data=all2;  
run;
```

```
*solution 2;  
proc sort data=other.all1;  
  by hospital;  
run;  
proc sort data=cop;  
  by hospital;  
run;
```

```
data all3;  
  merge other.all1 cop;  
  by hospital;  
run;  
proc print data=all3;  
run;
```

Exercise: Paired t -test

Find the data set `pain` in `'p:\sas\data\other'`. In short data holds information about a study where each patient were given two treatments 'a' and 'b'. The aim was to decrease the 'outcome', which is a measure of pain. The two treatments should be compared using a paired t -test. This test is done by first calculating the outcome difference between the two treatments for each patient. Then we test whether the differences have mean zero. This can be done using e.g. `proc univariate`. The following steps will take you through the calculation.

1. Print the data and explain why the outcome difference cannot be directly calculated.
2. Make two data sets from the `pain` data: one containing treatment 'a' results, the other containing treatment 'b' results.
3. In the data containing treatment 'a' results, make a new variable `res_a` which is equal to `outcome`. Similarly, make a new variable `res_b` which is equal to `outcome` in the data containing treatment 'b'

results.

4. Merge these data sets so that information from the same patient are in the same row.
5. Calculate the outcome-difference between the treatments.
6. Run a `proc univariate` for the difference and try to find the appropriate p -value in the output.