SAS tips and tricks (with a focus on data cleaning)

Paul W. Dickman
Department of Medical Epidemiology and Biostatistics
Karolinska Institutet

paul.dickman@mep.ki.se

April 8, 2003

Outline

1. Working with dates in SAS
2. Introduction to arrays
3. Checking the uniqueness of ID numbers
4. Verifying the check digit on PNRs
5. Customised notes and warnings in the log
6. Counting the number of deleted observations by writing them to a SAS data set

Working with dates in SAS

- From its inception, the SAS System has stored date values as an offset in days from January 1, 1960.
- Leap years, century, and fourth-century adjustments are made automatically. Leap seconds are ignored, and the SAS System does not adjust for daylight saving time.
- This method of date representation means that calculations and comparisons of SAS date values will produce correct results, regardless of century.
- SAS users can convert external data to or from SAS date values by the use of various informats, formats, and functions.

Reading raw data into SAS date variables

- Raw data can be read into SAS date variables using an appropriate informat.

```
data temp;
input date yymmdd8.;
cards;
19310317
19681224
19651128
19990914
;
run;
                                Obs     DATE
proc print;     ==>
var date;                        1     -10517
run;                             2       3280
                                 3       2158
                                 4      14501
```

- If you want to be able to understand printouts of these dates, it is necessary to assign an appropriate format to the variable.

```
proc print;         ==>      Obs     DATE
var date;
format date date.;            1     17MAR31
run;                          2     24DEC68
                              3     28NOV65
                              4     14SEP99
```

Converting character or numeric variables to SAS date variables

- This can be done using the `INPUT` function. The following line of code extracts date of birth from PNR and writes it out as a SAS date variable.

```
birth=input(substr(pnr,2,6),yymmdd.);
```

- If the values of year, month, and day are stored in separate variables, these can be written to a single SAS date variable using the MDY function:

```
datevar=mdy(month, day, year);
```

Before matching your data with registry data

- We often match our cohorts with registry data to obtain information on exposures and/or outcomes.
- Before performing such matching it is advisable to verify that all identification numbers (*personnummer*) are valid.
- An individual with an invalid *personnummer* cannot, for example, be identified as having a diagnosis of cancer.
- Should perform the following checks:
  1. Check that all personal identity numbers are unique.
  2. Check that the 'birth number' is appropriate for the sex of the patient (e.g. in a cohort of prostate cancer patients all birth numbers should be odd).
  3. Verify the check digit on each *personnummer*.

Checking the uniqueness of ID numbers

```
data temp;
input id start end sex;
informat start end ddmmyy8.;
format start end date8.;
cards;
1 281165 070599 1
2 230489 120193 2
3 011295 .      .
3 011295 181089 2
4 020773 010399 1
;
run;
```

Can use the nodupkey option on PROC SORT

```
proc sort data=temp out=sorted nodupkey;
by id;
run;

NOTE: 1 observations with duplicate key values were deleted.
NOTE: There were 5 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.SORTED has 4 observations and 4 variables.
```

- But we have no control over which observation is deleted!

```
Obs    ID       START        END    SEX

 1      1     28NOV65     07MAY99    1
 2      2     23APR89     12JAN93    2
 3      3     01DEC95           .    .
 4      4     02JUL73     01MAR99    1
```

Preferable to write the duplicate observations to a data file so we can study them

```
data unique dups;
set temp;
by id;
if first.id and last.id then output unique;
else output dups;
run;

NOTE: There were 5 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.UNIQUE has 3 observations and 4 variables.
NOTE: The data set WORK.DUPS has 2 observations and 4 variables.
```

- Anna Torrång described the `first.` and `last.` variables and the `output` statement.
- If an ID number if unique then it must be both the first and the last observation in the data set with the ID number.

Verifying the check digit on PNRs

- Swedish personal identification numbers (PNRs) comprise 10 digits (12 if century of birth is included).
- The first 6 digits represent date of birth (YYMMDD), followed by a 3-digit birth number (*födelsenummer*), which is even for females and odd for males, and the tenth digit is a check digit which can be constructed from the preceding nine digits.
- Further details of the *personnummer* along with the algorithm for verifying the check digit can be found at *Riksskatteverkets* web site.

`http://www.rsv.se/pdf/70407.pdf`

Algorithm for calculating the check digit

- First multiply each of the first nine digits in the PNR by the digits 2,1,2,1,2,1,2,1,2 and calculate the cumulative sum of each of these 9 calculations.
- If the product results in a number greater than 10, then add the individual digits.
- For example, $9 \times 2 = 18$, so we add 9 (the sum of 1 and 8) to the cumulative sum in the example below.

```
3 1 0 3 1 7 0 9 9
2 1 2 1 2 1 2 1 2
6 1 0 3 2 7 0 9 18 37
```

- In the example above, the cumulative sum is 37. The check digit is the number we would have to add to the product sum in order to obtain a multiple of 10.

- In the above example, we would need to add 3, so the check digit is 3.
- If the cumulative sum is a multiple of 10 then the check digit is 0.

Introduction to arrays

- SAS arrays are useful when we wish to perform a similar operation on a set of variables. For example, if we have 50 variables where missing values have been coded as 999.

```
array weight wt1-wt50;

do i=1 to 50;
if weight{i}=999 then weight{i}=.;
end;
```

- The variables do not have to be named wt1, wt2, wt3, ... wt50.

```
array vars pnr height weight bmi sex;
array vars pnr--sex;
array vars _numeric_;
```

- A SAS array is nothing more than a collection of variables (of the same type), in which each variable can be identified by referring to the array and, by means of an index, to the location of the variable within the array.
- SAS arrays are defined using the ARRAY statement, and are only valid within the data step in which they are defined.
- The syntax for the array statement is:

```
ARRAY array-name {subscript} <$> < length >
<< array-elements > <( initial-values )>>
```

- `array-name` must follow the naming rules for SAS variables.
- `{ subscript }` is the dimension (possibly multiple) of the array, and can be omitted or specified as * in which case SAS infers the dimension from the number of array elements.

- `< array-elements >` is the list of array elements (variables) which can be omitted if the dimension is given, in which case SAS creates variables called `array-name1` to `array-name{n}` where `{n}` is the dimension of the array.
- For example:

```
array wt {50};
```

will cause the variables wt1-wt50 to be created.

Example using data from the Swedish Birth Register

- For each record, we have information on up to 12 'events'. The event type (usually a birth) is stored in the variables type1-type12 and the corresponding date is stored in the variables date1-date12.
- The coding for the 'type' variables is:

```
0=stillbirth
1=live boy
2=live girl
6=immigration
```

- For each woman, we want to count the total number of live births, the total number of completed pregnancies (live births plus still births), and extract the emigration date for the women who emigrated.

```
array type type1-type12;
array datum date1-date12;
births=0; comppreg=0; emigrate=0;
do i = 1 to 12;
if type[i] in (1,2) then births=sum(births,1);
if type[i] in (0,1,2) then comppreg=sum(comppreg,1);
if type[i] in (6) then do;
    emigrate=1;
    emi_date=input(datum[i],yymmdd.);
    end;
end;
label
births='No. live births'
comppreg='No. completed pregnancies'
emigrate='Emigration indicator'
emi_date='Date of emigration (SAS date)'
;
```

SAS code for calculating the check digit

- Available at
  `http://www.pauldickman.com/teaching/sas/pnr_check.html`

```
/* generate some test data */
data temp;
input pnr $ 1-10 sex 12;
cards;
3103170993 1
3103170999 1
6812241450 1
6812241457 1
7511281896 2
7805062242 2
;
run;
```

```
data pnr_chk;
set temp;
length product $ 18 result $ 3;
array two_one {9} (2 1 2 1 2 1 2 1 2);
/*********************************************
multiply each of the first 9 digits in PNR by
the corresponding digit in the array two_one
and concatenate the result.
The COMPRESS function removes blanks.
*********************************************/
do i = 1 to 9;
product=compress(product||(substr(pnr,i,1)*two_one{i}));
end;

/** Now we sum the digits **/
do i = 1 to length(product);
sum=sum(sum,substr(product,i,1));
end;
```

```
/** extract the check digit from PNR **/
chk=substr(pnr,10,1);

/** calculate the correct check digit **/
corr_chk=mod(10-mod(sum,10),10);

if chk=corr_chk then result='ok';
else result='bad';
label
chk='Actual check number'
corr_chk='Correct check number'
pnr='Personnummer (10 digits)'
;
run;

proc print data=pnr_chk;
var pnr chk product sum corr_chk result;
run;
```

Output from PROC PRINT

```
Obs     PNR       CHK    PRODUCT      SUM   CORR_CHK   RESULT

 1   3103170993    3   6103270918     37        3       ok
 2   3103170999    9   6103270918     37        3       bad
 3   6812241450    0   12822442410    30        0       ok
 4   6812241457    7   12822442410    30        0       bad
 5   7511281896    6   14521482818    44        6       ok
 6   7805062245    5   1480506428     38        2       bad
```

How you could use this code in practice

- Instead of

```
if chk=corr_chk then result='ok';
else result='bad';
```

- You might use

```
data all pnr_bad;
set mydata;
....
if chk=corr_chk then output pnr_bad;
output all;
run;
```

- Or you might just use the original code and then use

```
proc print data=pnr_chk(where=(result='bad'));
var pnr chk product sum corr_chk result;
run;
```

Verifying that gender agrees with that implied by the PNR

- Assume we have a variable called SEX which takes the values 1 for males and 2 for females.

```
/* 11th digit of PNR odd => male (sex=1), else female (sex=2) */
if mod(input(substr(pnr,11,1),1.),2)=1 then sex_chk=1;
else sex_chk=2;
if sex ne sex_chk then output badsex;
```

Customised notes and warnings in the log

- If you ‘put’ a text string to the log which begins with 'ERROR:', 'WARNING:', or 'NOTE:', then SAS will format the text as an ERROR, WARNING, or NOTE respectively.
- That is the text will appear in the log file using the designated colour (red, green, and blue by default).
- This provides an efficient way of performing and documenting data checking/cleaning.

```
data temp;
input id start end sex;
informat start end ddmmyy8.;
format start end date8.;
cards;
1 281165 070599 1
2 230489 120193 2
3 011295 181089 2
4 020773 010399 1
;
run;

data new;
set temp;
if start gt end then
put 'ERROR: end before start: ' id= start= end= ;
run;
```

Extract from the log file

```
235  data new;
236  set temp;
237  if start gt end then
238  put 'ERROR: end before start: ' id= start= end= ;
239  run;

ERROR: end before start: ID=3 START=01DEC95 END=18OCT89
NOTE: There were 4 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.NEW has 4 observations and 4 variables.
NOTE: DATA statement used:
      real time           0.09 seconds
      cpu time            0.09 seconds
```

Another example

```
if sex=1 then hi_bmi=(bmi>29);
else if sex=2 then hi_bmi=(bmi>28);
else put 'WARNING: Invalid value for sex: ' pnr= sex= ;
```

- This is a simple way of putting logic checks in your program.

Counting the number of deleted observations by writing them to a SAS data set

```
data analysis noninv prevcan premeno no_age;
set emma.main;

/* non-invasive */
if 11<=histop<=19 then do; output noninv; delete; end;

/* previous cancer */
if 1<=prevcanc<=2 then do; output prevcan; delete; end;

/* premenopausal */
if mptype=0 then do; output premeno; delete; end;

/* age at first birth unknown */
if agefb=. and parity>0 then do; output no_age; delete; end;
output analysis;
run;
```

Extract from the log file

```
286  data analysis noninv prevcan premeno no_age no_ht;
287  set emma.main;
288  /* non-invasive */
289  if 11<=histop<=19 or histop>22 then do; output noninv; delete; end;

300  /* height unknown */
301  if f2=. then do; output no_ht; delete; end;
302  output analysis; run;

NOTE: There were 7420 observations read from the data set EMMA.MAIN.
NOTE: The data set WORK.ANALYSIS has 6207 observations and 89 variables.
NOTE: The data set WORK.NONINV has 352 observations and 89 variables.
NOTE: The data set WORK.PREVCAN has 471 observations and 89 variables.
NOTE: The data set WORK.PREMENO has 350 observations and 89 variables.
NOTE: The data set WORK.NO_AGE has 6 observations and 89 variables.
NOTE: The data set WORK.NO_HT has 34 observations and 89 variables.
NOTE: DATA statement used:
      real time           2:22.59
      cpu time            0.53 seconds
```