SAS Seminar, MEB

2012-02-27

Working with SAS Formats and SAS Dates

Anna Johansson

MEB, Karolinska Institutet

Slides also available on

www.pauldickman.com/teaching/sas/index.html

Outline

- Formats and Informats
- SAS Date variables
- Converting CHAR and NUM into SAS Dates
- Extracting birthdate from PNR
- SAS Date functions
- Calculating age in exact years
- Calculating age at diagnosis from PNR and diagnosis date
- YEARCUTOFF option for two-digit years

Formats and Informats

A **format** is a layout specification for how a variable should be printed or displayed.

An **informat** is a specification for how raw data should be read.

SAS contains many internal (pre-defined) formats and informats.

To see a list of all internal formats and informats, type in the command line

```
help formats
```

then type in the Index window of the Help page

```
formats, by category
```

Formats

There are four different categories of formats:

Category	Description
Character	instructs SAS to write character data values from character variables.
Date and time	instructs SAS to write data values from variables that represent dates, times, and datetimes
ISO	instructs SAS to write date, time, and datetime values using the ISO 8601 standard.
Numeric	instructs SAS to write numeric data values from numeric variables.

Examples. Numeric formats

Stored Value	Format	Displayed Value
12345.9876	10.4	12345.9876
12345.9876	10.2	12345.99
12345.9876	8.2	12345.99
12345.9876	COMMA11.4	12,345.9876
12345.9876	BEST8.	12345.99
12345.9876	BEST4.	12E3 (E3=10^3)

Example. Assign formats in DATA or PROC steps

```
data bc.main;
  set bc.cancerreg;
  ... Statements ...;

  format age 4.0
         bmi 4.2
         birthyr best5.;
run;

proc print data=bc.main;
  var birthyr age bmi;
  format age 4.0
         bmi 4.2
         birthyr best5.;
run;
```

Before:

ID	age	bmi	birthyr
1	34.567	22.8677	1975
2	22.4478	24.3333	1968
3	78.004	31.1233	1956

After:

ID	age	bmi	birthyr
1	35	22.9	1975
2	22	24.3	1968
3	78	31.1	1956

Example. Character formats

Stored Value	Format	Displayed Value
‘Anna Johansson’	$20.	Anna Johansson
’Anna Johansson’	$10.	Anna Johan
‘Anna Johansson’	$UPCASE20.	ANNA JOHANSSON

Informats

An informat is used when you **read in data** from a file. It specifies how SAS should interpret the values that are read into a new variable

```
data bc.main;
  infile 'h:\bc\cancerreg.txt';
  input       @1 pnr        10.
              @11 sex        1.
              @12 surname   $15.
              @27 diadate   yymmdd6.;
run;
```

If you never read in data from other sources than SAS datasets, then it is unlikely that you will come in contact with informats.

User-defined formats

User-defined formats and informats can be constructed using PROC FORMAT.

```
proc format;
   value sex    1='Male’
                2='Female';
run;
```

The code above only creates the format, it does not associate it with any variable. Formats can be associated with variables in either data steps or proc steps (see earlier slide) by using the FORMAT statement in a DATA or PROC step.

```
format gender sex.;
```

If we do a PROC PRINT on the data using format SEX. then the result is

```
proc print data=f;
  var gender;
  format gender sex.;
run;
```

Before:		After:	
ID	gender	ID	gender
1	1	1	Male
2	1	2	Male
3	2	3	Female

Any calculations made using a variable in a data step will be based on the raw data (i.e. the format is ignored).

When fitting statistical models, however, the model can be fitted to the formatted value by using options (i.e. formats can be used for grouping/categorisation).

User-defined formats

It is often a wise thing to include the original value in the format label, which will make it easier for you to konw the underlying value

```
proc format;
   value sex    1=“1=Male”
                2=“2=Female”;
run;

proc print data=f;
  var gender;
  format gender sex.;
run;
```

Before:		After:	
ID	gender	ID	gender
1	1	1	1=Male
2	1	2	1=Male
3	2	3	2=Female

User-defined formats useful to group values

If a variable is continuous and we wish to categorise it, then formats can be useful.

```
proc format;
   value agegrp 0-19="0-19"
                20-39="20-39"
                40-high="40+";
run;
proc freq data=e;
  tables age;
  format age agegrp.;
run;
```

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0-19	2	20.00	2	20.00
20-39	3	30.00	5	50.00
40+	5	50.00	10	100.00

User-defined formats useful to group values

```
data e2;
  set e;
  if 0<=age<=19 then agecat=1;
  else if 20<=age<=39 then agecat=2;
  else if 40>=age then agecat=3;
run;
proc format;
   value agecatf 1="1= 0-19"
                 2="2= 20-39"
                 3="3= 40+";
run;
proc freq data=e2;
  tables agecat;
  format agecat agecatf.;
run;
```

agecat	Frequency	Percent
1=0-19	2	20.00
2=20-39	3	30.00
3=40+	5	50.00

SAS Dates

SAS stores date values as the **integer number of days calculated from January 1, 1960**.

SAS Date variables are NUM variables that can be interpreted into dates using Date formats.

Leap years, century, and fourth-century adjustments are made automatically. Leap seconds are ignored, and SAS does not adjust for daylight saving time.

SAS users can convert external data to/from SAS date values by the use of various informats and functions.

Example. Date Formats

3489 is the number of days between 1 Jan 1960 until 21 July 1969.

Stored Value	Format	Displayed Value
3489	DATE9.	21JUL1969
3489	DDMMYY8.	21/07/69
3489	YYMMDD6.	690721
3489	YYMMDD8.	69-07-21
3489	YYMMDD10.	1969-07-21

Reading raw data into SAS Date variables

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

```
data bcdates;
   input date yymmdd6.;
   cards;
310317
681224
651128;
run;
```

Obs	date	Real date:
1	-10517	17 March 1931
2	3280	24 December 1924
3	2158	28 November 1965

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

In a data step (stored permanently to the variable):

```
data bcdates;
   input date yymmdd6.;
   format date yymmdd10.;
   cards;
310317
681224
651128;
run;
```

In a proc step (stored during the PROC only):

```
proc print data=bcdates;
  var date;
  format date yymmdd10.;
run;
```

To print a variable without an assigned permanent format just assign ”no format”:

```
proc print data=bcdates;
  var date;
  format date ;
run;
```

Note: there has to be a space before the ";" in the format statement to remove the format.

Without format:

Obs	date
1	-10517
2	3280
3	2158

With format yymmdd10.

Obs	date
1	1931-03-17
2	1968-12-24
3	1965-11-28

Converting a CHAR variable into a SAS Date variable

This can be done using the INPUT function.

The following code converts a CHAR variable (birthdate_char) into a SAS Date variable (birthdate).

```
birthdate = INPUT(birthdate_char, yymmdd6.);
```

Note that yymmdd6. is an informat in this statement.

Print without format:

birthdate_char	birthdate
'310317'	-10517
'681224'	3280
'651128'	2158

Print with format (YYMMDD10.):

birthdate_char	birthdate
'310317'	1931-03-17
'681224'	1968-12-24
'651128'	1965-11-28

Extracting the birthdate from PNR

The following code extracts date of birth from PNR (CHAR variable) and writes it out as a SAS date variable.

```
birthdate = INPUT(SUBSTR(pnr,1,6), yymmdd6.);
```

The substr function (substring) reads 6 positions starting from the first position of variable PNR.

Note that yymmdd6. is an informat.

PNR	birthdate
’310317-0367’	-10517
’681224-0873’	3280
’651128-2766’	2158

Converting a NUM variable into a SAS Date variable

To convert a numerical variable into a SAS Date variable, you must first convert the NUM into a CHAR, and then convert the CHAR into a DATE.

The PUT function converts any type of variable into a CHAR variable.

```
birthdate_char = PUT(birthdate_num, 6.0);
```

```
birthdate = INPUT(birthdate_char, yymmdd6.);
```

Note that the 6.0 in the PUT function is a format, it describes the variable we read from.

Note that yymmdd6. in the INPUT function is an informat.

birthdate_num	birthdate_char	birthdate
310317	’310317’	-10517
681224	’681224’	3280
651128	’651128’	2158

SAS Date Functions

Why do we want to use SAS Dates?

Why can't we use CHAR and NUM variables for dates?

Dates are special numerical values, we want to make complicated calculatations on them, such as differences between dates (age, duration).

Dates do not follow the common base 10 (multiples of 10, i.e. 100, 1000 etc.) but use units of 12 (months), 28,29,20,31 (days). One year isn't 10 months, and one month isn't 10 days.

With SAS Date functions we can take into account leap years, the differences in the lengths of months, etc.

Ex. Calculating difference in years between two SAS Dates

To calculate the difference in years between two SAS Dates use the YRDIF function (new in version 8). The result is a NUM variable.

```
duration = YRDIF(startdate,enddate,'actual');
```

Print without format:

startdate	enddate	duration
3489	16034	34.3479

Print with formatted dates:

startdate	enddate	duration
1969-07-21	2003-11-25	34.3479

Ex. Converting separate year, month, day NUM variables into a SAS Date

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

```
birthdate = MDY(month, day, year);
```

Print without format:

year	month	day	birthdate
1969	7	21	3489

Print with format yymmdd10. for birthdate:

year	month	day	birthdate
1969	7	21	1969-07-21

Ex. Using MDY to build dates

You can use the substr function to build your own dates:

```
diagyr=input( substr(diagdate_char, 1,4) , 4.0);
diagmon=input( substr(diagdate_char, 5,2) , 4.0);
diagday=input( subtr(diagdate_char, 7,2) , 4.0);

diagdate = MDY(diagmon, diagday, diagyr);
```

which gives identical result to using

```
diagdate = INPUT(diagdate_char, yymmdd8.);
```

diagdate_char	diagyr	diagmon	diagday	diagdate
”19690721”	1969	7	21	3489

Ex. Using MDY to build dates

Useful if you want to create a date 15 years after diagnosis:

```
diagyr=input( substr(diagdate_char, 1,4) , 4.0) + 15;
diagmon=input( substr(diagdate_char, 5,2) , 4.0);
diagday=input( subtr(diagdate_char, 7,2) , 4.0);

diagdate15 = MDY(diagmon, diagday, diagyr);
```

diagdate_char	diagyr	diagmon	diagday	diagdate15
”19690721”	1984	7	21	8967

Date Functions in SAS

DATDIF(sdate,edate,'actual')	returns the number of days between two dates
DATE()	returns the current date as a SAS date value
DAY(date)	returns the day of the month from a SAS date value
MDY(month,day,year)	returns a SAS date value from month, day, and year NUM values
MONTH(date)	returns the month from a SAS date value

TIME()	returns the current time of day
YEAR(date)	returns the year from a SAS date value
YRDIF(sdate,edate,*basis*)	returns the difference in years between two SAS dates

The *basis* in the YRDIF function determines what number of days SAS should use for each month and year. The *basis* can have any of the four values:

30/360 = 30 days each month, 360 days each year - Alias **'360'**

ACT/ACT = Actual days each month, Actual days each year
- Alias **'ACTUAL'**

ACT/360 = Actual days each month, 360 day each year - No Alias

ACT/365 = Actual days each month, 365 day each year - No Alias

Example. Date Functions

Function	Stored value X	With date format DATE.
X=date();	16034	25NOV2003
X=mdy(11,25,2003);	16034	25NOV2003
X=day(3489);	21	-
X=month(3489);	7	-
X=year(3489);	1969	-
X=yrdif(3489, 16034,’actual’)	34.3479	-

Calculating AGE in exact years

To calculate exact age, i.e. age that takes leap years into account, we simply use the YRDIF function for SAS Dates.

We combine it with the INT function, which returns the integer part of the result from the YRDIF function.

```
age = INT(YRDIF(birthdate, enddate, 'actual'));
```

Print with formatted dates:

birthdate	enddate	age
1969-07-21	2003-11-25	34

Calculating AGE in exact years, alternative method SAS version 6

This SAS code was written by Billy Kreuter, who posted it to the SAS-L mailing list several years ago. Billy authored an article in SAS Communications (4th quarter 1998) which discusses this issue in greater detail.

The following code calculates age in completed years from the variables *birth* and *somedate*.

```
age = FLOOR((INTCK('month',birth,somedate)-
            (DAY(somedate) < DAY(birth))) / 12);
```

The approach is to first calculate the number of completed months between the two dates and then divide by 12 and round down to get the number of completed years.

The following code could be used to calculate the number of completed months between the dates birth and somedate.

```
months = INTCK('month',birth,somedate) - (DAY(somedate)
           < DAY(birth));
```

The first part of the code uses the intck function to calculate the number of times a 'month boundary' (e.g from January to February) is crossed between the two dates. Crossing a 'month boundary' does not necessarily mean that a completed month has elapsed so a correction needs to be made when the end date (somedate) is less than the start date (birth).

To convert completed months to completed years one uses

```
years = FLOOR(months/12);
```

The floor function simply rounds a real number down to the nearest integer, for example floor(4.93)=4.

Calculate age at diagnosis from PNR and diagnosis date

It is often necessary to calculate age at diagnosis from variables representing the personal identification number and the date of diagnosis (stored as a character variable).

The first step is to create SAS date variables representing the birth date and diagnosis date.

- The date of diagnosis is stored in CHAR variable DXDAT, which is converted to a SAS date variable called DIAGDATE using the INPUT function.

- Construct the SAS date variable called BIRTHDATE by first extracting the date from the PNR using the SUBSTR function.

- Age at diagnosis is calculated as the number of completed years between the two dates.

The raw data file **rawdata.sas7bdat** contains two observations only:

pnr	dxdat
196511289999	990622
193404199999	590420

```
data bcdiag;
   set rawdata;

/* convert the character variable to a SAS date var */
   diagdate = INPUT(dxdat, yymmdd6.);

/* extract the birthdate from PNR */
   birthdate = INPUT(SUBSTR(pnr,1,8), yymmdd8.);

/* calculate AGE at diagnosis */
   agediag = INT(YRDIF(birthdate, diagdate, 'actual'));

   format diagdate  yymmdd10.
          birthdate yymmdd10.;
run;
```

Print the result:

```
proc print data=bcdiag;
   title 'Calculating age at diagnosis';
   var pnr dxdat birthdate diagdate agediag;
   format birthdate diagdate ; * to get without format;
run;
```

Without the format on birthdate and diagdate:

PNR	DXDAT	BIRTHDATE	DIAGDATE	AGEDIAG
196511289999	990622	2158	14417	33
193404199999	590420	-9388	-256	25

With format YYMMDD10. on birthdate and diagdate:

PNR	DXDAT	BIRTHDATE	DIAGDATE	AGEDIAG
196511289999	990622	1965-11-28	1999-06-22	33
193404199999	590420	1934-04-19	1959-04-20	25

Two-digit years, YEARCUTOFF=option

SAS date informats, formats, and functions all accept two-digit years as well as four-digit years.

If the dates in your external data sources contain four-digit years, then the SAS System will accept and display those four-digit years without any difficulty as long as you choose the appropriate informat and format (YYMMDD10.).

If dates in your external data sources or SAS program statements contain two-digit years, you can specify the century prefix assigned to them by using the YEARCUTOFF= system option.

The YEARCUTOFF= option specifies the first year of the 100-year span that is used to determine the century of a two-digit year.

If the YEARCUTOFF is set to 1900 then that implies that all two-digit years are assumed to be in the 1900's.

If you are working with a study where the last date of follow-up is 1992, but some individuals in your study were born in the late 1800's, you may wish to set the YEARCUTOFF option to 1893.

This would lead to SAS interpreting values for year between 93 and 99 as being in the 1800's.

YEARCUTOFF=	**Interpretation of years 00-99:**
1900	**00-99 ↔ 1900-1999**
1893	**93-99 ↔ 1893-1899**
	00-92 ↔ 1900-1992
1920	**20-99 ↔ 1920-1999**
	00-19 ↔ 2000-2019

By default

YEARCUTOFF = 1900 in Version 6

YEARCUTOFF = 1920 in Version 8, Version 9

To change the default YEARCUTOFF value use the global statement (written outside DATA steps or PROC steps) OPTIONS.

```
options yearcutoff = 1893;
```

Let's consider an example of reading in dates with both two-digit and four-digit years. Note that in this example the YEARCUTOFF= option has been set to 1920.

```
options yearcutoff=1920;

data schedule;
   input @1 rawdata    $8.
         @1 date       yymmdd8.;
   format date yymmdd10.;
   cards;
651128
19651128
18230314
19131225
131225
run;
```

OBS	RAWDATA	DATE
1	651128	1965-11-28
2	19651128	1965-11-28
3	18230314	1823-03-14
4	19131225	1913-12-25
5	131225	2013-12-25

Note that the dates in observations 1 and 2 are the same (a two digit date of 65 defaults to 1965).

But the dates in observations 4 and 5 are different (a two-digit date of 13 defaults to 2013).

Note: variable DATE has been given format yymmdd10.

Dates in the Data Standard at MEB

MEB's Data Standard includes standards for date variables.

Standardised names, values, missing data and formats for date variables are given (these general standards apply to all software including SAS).

Table 3. Standard for date variables

Variable	Variable description	Type	Format/ Value description
….date	Date	Date/Char	YYYY-MM-DD
....yr	Year	Date/Char	YYYY
…mon	Month	Date/Char	MM
…day	Day	Date/Char	DD
…wk	Week		Number
yrmon	Year and month	Date/Char	YYYY-MM

From Data Standard at MEB

Error codes for dates

When dates are partly missing or incorrect (i.e. 2001-02-30) there is a need for corrections or imputations. The corrections should be done in a new derived variable (e.g. new variable XDIAGDATE may be derived from the raw data variable DIAGDATE).

Together with the new derived variable, information about the errors should be documented with error codes describing the problem and the solution (i.e. error id, description and solution), see Table 6. One example of the usage is shown in Table 7.

From Data Standard at MEB

Table 8. Error codes for dates

DIAGDATE_ERROR	DESCRIPTION	SOLUTION
0	Valid date	None
1	Day missing/stated 00	Day set to 15
2	Month and day missing/stated 00	Month and day set to 0701
3	Last day of month not correct	Day set to last correct
4	Incorrect date	Set value to null (empty)

From Data Standard at MEB

Table 9. Example of the use of error codes in the database

ID	DIAGDATE	DIAGDATE_ERROR	XDIAGDATE
1	1997-03-25	0	1997-03-25
2	1995-07	1	1995-07-15
3	1969	2	1969-07-01
4	2003-02-30	3	2003-02-28
5	5344-23-69	4	

Here title

Here text