

Tips to Manipulate the Partial Dates

Deli Wang, Ossining, NY

Chunxia Lin, InVentiv Health Clinical, Ossining, NY

ABSTRACT

A partial date is simply any date where the date is incomplete, but not wholly missing. More commonly in clinical trials, the day and/or month are missing. In these cases, SAS programmers may be asked to impute a reasonable date or time per client's requirement or statistical purpose. This paper introduces two different imputation logics with the missing day set to the last day of the month. Both methods take the leap years into consideration, and generate same results. However, one method imputes the day depending on the leap year and month, while another not considering any.

INTRODUCTION

Dates are an integral and critical part of the data collected within clinical trials. In reality, however, partial dates are almost inevitable during the clinical data collection. In clinical trials, partial dates are most common in variables where the date is historical information (e.g. prior medication, medical history). This is a general problem because it is possible that the subject does not recall a wholly accurate start date for a medication they have been taking for a number of years, or when a particular medical event occurred during childhood. Generally it is preferable to leave the date as it is if the date is merely to be reported like in SDTM or listings. However, if the date is to be used to calculate the duration of an adverse event or the duration of a drug, then a partial date will prevent the duration being calculated. In this case, SAS programmers may be asked to impute a reasonable date or time per client's requirement or statistical purpose. However, how to decide what is the best approach to impute partial dates and corresponding analytical bias it may cause, using the first or last day of the month, or first or last month of the year, or the first or last study contact day, or the first or last date of dosing is out of our discussion scope. Instead this paper will introduce two imputation logics for the partial dates using the last day of month from technical view.

METHOD A: IMPUTATION NOT CONSIDERING LEAP YEAR

Table1 showed some data where the collected variables are year, month and day in character format. Some dates are complete, some are not. Imputing the partial date with the missing day set to the last day of the month is explored. The idea of method A is that adding 1 day to the last day of the month will always falls in the first day of next month. Hence conversely it is simple to impute the last day of the month, that is, subtracting 1 day from the first day of next month per given month data if the date is incomplete. However, to handle the data properly, SAS programmers need to take care of two steps in the imputation logic: 1. when the month happens to be December, use the logic of subtracting 1 day from the first day of next year; 2. Otherwise, use the first day of next month minus 1 day logic. Or in a simple way, the SAS programmer can use the robust INTNX function to resolve above concerns, that is, use the INTNX function to advance date by 1 month first, then subtract 1 day to get last day of previous month.

```
DATA test;
infile cards missover;
input year $ month $ day $;
cards;
2011 2
2010 3
1959 2
2000 2
1975 11
1981 12
2001 1
2003 6
2001 1
2003 10
2002 8
2006 7
2004 9
2004 5
2007 4
2012 10 28
```

```
2008 9 12
2007 1 9
2005 03
;
RUN;
```

| | year | month | day |
|----|------|-------|-----|
| 1 | 2011 | 2 | |
| 2 | 2010 | 3 | |
| 3 | 1959 | 2 | |
| 4 | 2000 | 2 | |
| 5 | 1975 | 11 | |
| 6 | 1981 | 12 | |
| 7 | 2001 | 1 | |
| 8 | 2003 | 6 | |
| 9 | 2001 | 1 | |
| 10 | 2003 | 10 | |
| 11 | 2002 | 8 | |
| 12 | 2006 | 7 | |
| 13 | 2004 | 9 | |
| 14 | 2004 | 5 | |
| 15 | 2007 | 4 | |
| 16 | 2012 | 10 | 28 |
| 17 | 2008 | 9 | 12 |
| 18 | 2007 | 1 | 9 |
| 19 | 2005 | 03 | |

Table 1. Imaginary Date Data

```
DATA A;
SET test;
length date date_imp $50.;

if day='' then do;
date=catx('-', strip(year),strip(put(input(month, best.),z2.)));
if month="12" then date_imp=put((mdy(1,1,input(year,best.))+1)-1, is8601da.);
else date_imp=put((mdy(input(month,best.))+1,1,input(year,best.))-1, is8601da.);
end;
else do;
date=put(mdy(input(month,best.),input(day,best.),input(year,best.)), is8601da.);
date_imp=date;
end;

RUN;
```

Or use INTNX function to replace above highlighted yellow code. INTNX function increments a date, time, or datetime value by a given time interval, and returns a date, time, or datetime value.

```
DATA A1;
SET test;
length date date_imp $50.;

if day='' then do;
date=catx('-', strip(year),strip(put(input(month, best.),z2.)));
date_imp=put(intnx('month',input(strip(date)||"-01",yymmdd10.),1)-1, is8601da.);
end;
else do;
date=put(mdy(input(month,best.),input(day,best.),input(year,best.)), is8601da.);
date_imp=date;
end;

RUN;
```

| | year | month | day | date | date_imp |
|----|------|-------|-----|------------|------------|
| 1 | 2011 | 2 | | 2011-02 | 2011-02-28 |
| 2 | 2010 | 3 | | 2010-03 | 2010-03-31 |
| 3 | 1959 | 2 | | 1959-02 | 1959-02-28 |
| 4 | 2000 | 2 | | 2000-02 | 2000-02-29 |
| 5 | 1975 | 11 | | 1975-11 | 1975-11-30 |
| 6 | 1981 | 12 | | 1981-12 | 1981-12-31 |
| 7 | 2001 | 1 | | 2001-01 | 2001-01-31 |
| 8 | 2003 | 6 | | 2003-06 | 2003-06-30 |
| 9 | 2001 | 1 | | 2001-01 | 2001-01-31 |
| 10 | 2003 | 10 | | 2003-10 | 2003-10-31 |
| 11 | 2002 | 8 | | 2002-08 | 2002-08-31 |
| 12 | 2006 | 7 | | 2006-07 | 2006-07-31 |
| 13 | 2004 | 9 | | 2004-09 | 2004-09-30 |
| 14 | 2004 | 5 | | 2004-05 | 2004-05-31 |
| 15 | 2007 | 4 | | 2007-04 | 2007-04-30 |
| 16 | 2012 | 10 | 28 | 2012-10-28 | 2012-10-28 |
| 17 | 2008 | 9 | 12 | 2008-09-12 | 2008-09-12 |
| 18 | 2007 | 1 | 9 | 2007-01-09 | 2007-01-09 |
| 19 | 2005 | 03 | | 2005-03 | 2005-03-31 |

Table 2. Dataset A Created by Method A

METHOD B: IMPUTATION CONSIDERING LEAP YEAR

The idea of method B is to impute the missing day as 28th/29th, 30th or 31th depending on the month. If the month happens to be February, use INTCK function to decide whether the event occurred on a leap year. As we know, INTCK function returns the integer count of the number of interval boundaries between two dates, two times, or two datetime values. Hence if the difference between Feb 1st and Mar 1st is 29, then the event occurred on a leap year, and imputes the missing day as 29th, otherwise, impute with 28th. If the month falls in April, June, September, and November, then impute with 30th, otherwise impute with 31th.

```
DATA B;
  SET test;
  length date date_imp $50.;

  if day='' then do;
    date=strip(year)||"-"||strip(put(input(month, best.),z2.));
    if input(month,best.)=2 then do;
      if intck("day", mdy(2,1,input(year,best.)),mdy(3,1,input(year,best.)))=28
        then date_imp=catx('-',year,"02","28");
      else date_imp=catx('-',year,"02","29");
    end;

    else if input(month,best.) in (4,6,9,11) then
      date_imp=catx('-',year,put(input(month, best.),z2.),"30");
    else date_imp=catx('-',year,put(input(month, best.),z2.),"31");
  end;

  else do;
    date=catx('-',year,put(input(month, best.),z2.),put(input(day, best.),z2.));
    date_imp=date;
  end;

RUN;

PROC COMPARE base=A compare=B;
RUN;
```

```

The COMPARE Procedure
Comparison of WORK.A with WORK.B
(Method=EXACT)

Data Set Summary

Dataset          Created          Modified    NVar    NObs
WORK.A          24DEC13:14:23:51  24DEC13:14:23:51    5      19
WORK.B          24DEC13:14:23:51  24DEC13:14:23:51    5      19

Variables Summary

Number of Variables in Common: 5.

Observation Summary

Observation      Base  Compare
First Obs        1      1
Last Obs         19     19

Number of Observations in Common: 19.
Total Number of Observations Read from WORK.A: 19.
Total Number of Observations Read from WORK.B: 19.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 19.

NOTE: No unequal values were found. All values compared are exactly equal.

```

Output 1. PROC COMPARE Output between Method A and Method B

CONCLUSION

In many real world situations date data are often incomplete, with only a month and year known, or just the year. This paper presents two methods to impute the partial dates when required to use the last day of the month. Method B manipulates the incomplete dates per common sense depending on the leap year and month. While method A is against common sense but robust to cover each month case.

REFERENCES

1. SAS/BASE Software: Version 9, SAS® Institute Inc., Cary NC
2. Bowman, Rachel. "Partial Dates; decisions and implications of handling partially missing dates" PhUSE2006 Available at <http://www.lexjansen.com/phuse/2006/po/po11.pdf>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Deli Wang
E-mail: deli.wang@hotmail.com

Name: Chunxia Lin
Enterprise: InVentiv Health Clinical
Work Phone: (914) 9230173
E-mail: lin.trisha@yahoo.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.