Paper 336-2013

We All Have Bad Dates Once in a While...

Randall Deaton, BlueCross BlueShield of Tennessee;
Patrick M Kelly, BlueCross BlueShield of Tennessee

ABSTRACT

Dates in a corporate data arena can be a dangerous liaison. The strain of translating corporate date types to SAS® date types can be tricky to navigate, let alone bringing a third-party date type into the mix. Adding third wheel to your SAS dates can create a comedy of errors. An experienced SAS programmer with knowledge of the SAS macros and a few clever programming tricks can more easily resolve your SAS Dates from a sticky situation to an orderly affair.

INTRODUCTION

One of the things that those of us that work with data have to deal with on a regular basis is data from an outside resource. In the health insurance world this can be from Centers for Medicare and Medicaid Services (CMS), an outside vendor, or complementary data from other insurance companies. In all of the data the one constant is inconsistency.

Much of this date field data comes into our warehouse in a myriad of ways. Sometimes this data is processed as numeric, sometimes as character, and never in the same format. Date fields are the first challenge data analysts must overcome before we can analyze the information and gleam information contained within the data.

FIRST THE DREADED BLIND DATE

The format that has caused much consternation, within our organization, is formatted as a numeric field in the YYYYMMDD format. In the following example we have recreated this format with the following DATA step and output of the code in Figure 1. Setup data output.

```
data work.dates;
input date;
cards;
20130529
19980321
20090111
20011108
20130202
19970921
20110718
run;
```

	date
1	20130529
2	19980321
3	20090111
4	20011108
5	20130202
6	19970921
7	20110718

Figure 1. Setup data output

Attempting to turn this directly into a date field with a PUT statement you get to enjoy the following error in your log.

```
put(date,anydtdte10.) as date1 = (ERROR: Date value out of range)
```

Since the PUT statement does not like the data as is. Using the SUBSTRING function we can reorder the date to order the numbers in a more palatable format.

1. Isolate the 2 digits for the month `(substr(put(date,8.),5,2))`

2. Isolate the 2 digits for the day `(substr(put(date,8.),7,2))`
3. Isolate the 4 digits for the year `(substr(put(date,8.),1,4)`
4. Recombine the three elements with the CATS function.

```
(cats(substr(put(date,8.),5,2),substr(put(date,8.),7,2),substr(put(date,8.),1,4
)) as date2)
```

Now we end up with a character field that is in the order of MMDDYYYY as in Figure 2. Modified date output.

	date	date2
1	19970921	19970921
2	19980321	19980321
3	20011108	20011108
4	20090111	20090111
5	20110718	20110718
6	20130202	20130202
7	20130529	20130529

Figure 2. Modified date output

Next we can turn this date value into a numeric SAS date field by using the INPUT & ANYDTDTE functions as in Figure 3. Converted number SAS date.

```
(input(cats(substr(put(date,8.),5,2),substr(put(date,8.),7,2),
substr(put(date,8.),1,4)),anydtdte.)as date3)
```

	date	date2	date3
1	19970921	19970921	13778
2	19980321	19980321	13959
3	20011108	20011108	15287
4	20090111	20090111	17908
5	20110718	20110718	18826
6	20130202	20130202	19391
7	20130529	20130529	19507

Figure 3. Converted number SAS date

Finally just apply a date format to the field so that it becomes a date field that can be included in a required format for some reports outside of SAS as seen in Figure 4. Final converted date.

```
input(cats(substr(put(date,8.),5,2),substr(put(date,8.),7,2),substr(put(date,8.
),1,4)),anydtdte.) format date9. as date4
```

	date	date2	date3	date4
1	19970921	19970921	13778	21SEP1997
2	19980321	19980321	13959	21MAR1998
3	20011108	20011108	15287	08NOV2001
4	20090111	20090111	17908	11JAN2009
5	20110718	20110718	18826	18JUL2011
6	20130202	20130202	19391	02FEB2013
7	20130529	20130529	19507	29MAY2013

Figure 4. Final converted date

No one likes to drag bad dates out, but the process can be combined into a more efficient single step. Next is the example to just to get this bad date over with, and see the success in Figure 5. Converted in one step.

```
proc sql;
create table work.dates2 as
select distinct
```

```
*,
input(cats(substr(put(date,8.),5,2),substr(put(date,8.),7,2),substr(put(date,8.
),1,4)),anydtdte.) format date9. as better_date
from work.dates
;
quit;
```

	date	better_date
1	19970921	21SEP1997
2	19980321	21MAR1998
3	20011108	08NOV2001
4	20090111	11JAN2009
5	20110718	18JUL2011
6	20130202	02FEB2013
7	20130529	29MAY2013

Figure 5. Converted in one step

AUTOMATE THOSE DATES

Now that we have successfully navigated a bad date, we can set up a macro for speed dating. Now that we have manipulated our dates in a more useable format we can leverage the INTNX function to create some simple macros to assist in setting automatic macro date parameters for your report.

Health insurance reporting is often reporting on data for the last full 12 months or maybe the last full quarter. Using INTNX we can quickly calculate that date span.

If we want to create a report based on the past twelve full months from today we can quickly create a macro for both the start and end date of that period so that the next time we create this report we don't have to manually change our program.
First using a null DATA step we create a macro variable that sets today's date.

```
data _null_;
call symput('today',"'"||put(today(),
date9.)||"'d");
run;
```

Next we create macros for the start and end date of our last twelve full months, based off the today macro variable.

```
data _null_;
call symputx('start',PUT(INTNX('month',&today.,-12,'BEG'),date9.));
call symputx('end',PUT(INTNX('month',&today.,-1,'END'),date9.));
run;

%put ** &start. ** &end. ** ;
```

Using the PUT function lets us see the date in our log so we know what span the report ran on..

If we want to create BEG and END macro variables based on the previous quarter. Just change the INTNX option from 'month' to 'qtr'.

```
(data _null_;
call symputx('start1',PUT(INTNX('qtr',&today.,-1,'BEG'),date9.));
call symputx('end1',PUT(INTNX('qtr',&today.,-1,'END'),date9.));
run;

%put ** &start1. ** &end1. ** ;)
```

CONCLUSION

Without a few bad dates you will never know when you have a good date. With some clever string manipulation and INTNX macros an industrious SAS programmer can turn those bad dates around and build robust processes to ensure success even in the most awkward of dates.

CONTACT INFORMATION

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

Name: Randall Deaton
Enterprise: BlueCross BlueShield of Tennessee
Address: 1 Cameron Hill Circle
City, State ZIP: Chattanooga, TN 37402
E-mail: Randall_deaton@bcbst.com

Name: Patrick M Kelly
Enterprise: BlueCross BlueShield of Tennessee
Address: 1 Cameron Hill Circle
City, State ZIP: Chattanooga, TN 37402
E-mail: patrick_kelly@bcbst.com