SCSUG Paper

THE SECRET TO SAS DATES

Alison Little, MPP
Texas Health and Human Services Commission

ABSTRACT

How do you know whether the field you are working with which is called “birth date” or “procedure date” etc. is a real SAS date? And how can you work with date fields, whether they are real or fake SAS dates? This paper will help you feel confident identifying what you are using and working with date variables.

HOW SAS AND HUMANS SEE DATES

When is the SCSUG SAS forum coming up? What are you doing Saturday night? When is your brother’s birthday? There are lots of reasons to want to be specific about a particular point in time. These are some examples of dates that make sense to humans:

- January 09, 2017
- 1/9/2017
- 09jan2017
- 20170109

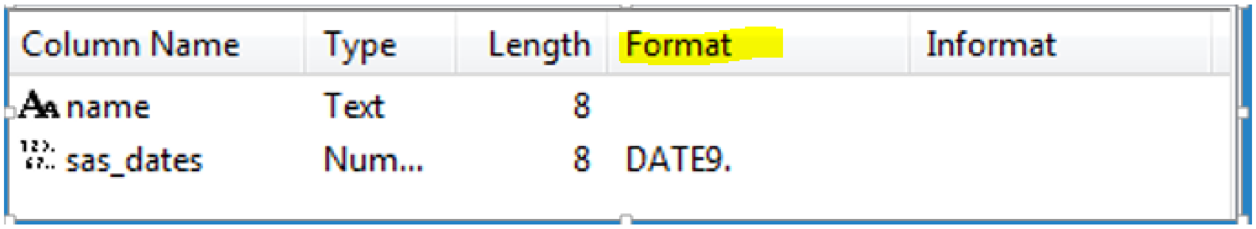
Humans parse these dates by looking at separate values for month, day, and year. In contrast, these are examples of dates that make sense to SAS:

- 20828 (Number of days between Jan. 1, 1960 and Jan. 9, 2017)
- -2502 (Number of days between Jan. 1, 1960 and Feb. 24, 1953)

The secret of working with SAS dates is to think about **both** what SAS is seeing and what you are seeing. And the way to move between these is to understand SAS formats.

In regular life, we say “format” to mean “how something looks.” In SAS, “format” has a more specific meaning – it is something (a costume) that can be applied to a variable. You can use SAS-created formats or user-created formats.

You can check whether your variable has a format applied by right-clicking the table and looking under “view columns” for a view like the following. In the screenshot that follows, the variable “name” has no format applied. The variable “sas_dates” has date9. format applied.



Column Name	Type	Length	Format	Informat
name	Text	8		
sas_dates	Num...	8	DATE9.	

What Formats/Costumes are Available for SAS Dates?

There are many SAS formats available to apply to SAS dates. The following are just a few examples: [1]

Using DATE9. format:	9JAN2017
Using DATETIME13. format:	09JAN17 : 00 : 00
Using MMDDYY8. format:	1/09/17
Using MMDDYY10. format:	1/09/2017
Using YYMMN6. format:	201701
Using WEEKDATE29. format:	Monday, January 9, 2017

Translating From SAS to a Human

Here’s how to translate between SAS and a human.

SAS sees 20828. You can use a statement like:

```
Format birth_date date9.;
```

to get the date into a format you as the human can read.

Translating From a Human To SAS

Here’s how to translate between SAS and a human.

A human has a date in mind - say, January 9, 2017. SAS needs to know how many days it’s been since Jan. 1, 1960, but the human probably doesn’t know (quite a few….?)

You can use a statement like:

```
If my_date eq ‘09jan2017’d then grand_prize=1;[2]
```

This tells SAS “here’s what the date looks like to me…you convert it to what works for you.”

Another way to tell SAS which date you mean is the MDY function. For example:

```
if sas_dates= MDY(1,9,2017) then grand_prize=1;
```

DOES MY “DATE” FIELD HAVE SAS DATES?

Let’s say you receive a new data set that you are getting to know. One of the variables has “date” in the name…but what kind of date is it? It is important to determine this, because whether your variable has SAS dates or not will impact the tools available to you to work with it.

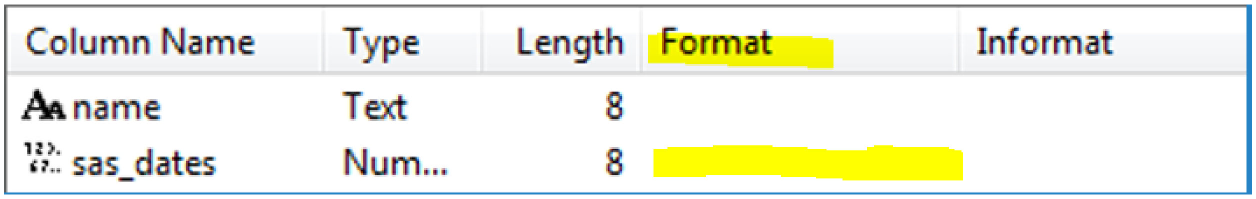
Possibility 1) A numeric variable with real SAS dates

A variable has SAS dates if:

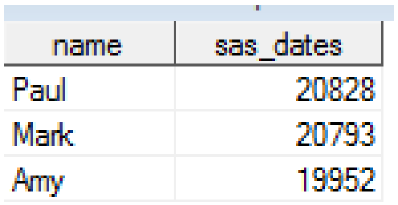
- The variable in your SAS data set has type = number AND
- The values *don’t* make sense to you as dates when you have no date format applied AND

- The values *do* make sense as dates when you do have a date format like yymmdd10. applied AND
- You can try on different date formats and have the appearance of the values change.

For example, in the screenshot below, the variable “sas_dates” has type=numeric. When it does not have a format applied, the values (such as 20828) do not look to a human like days, months, and years.



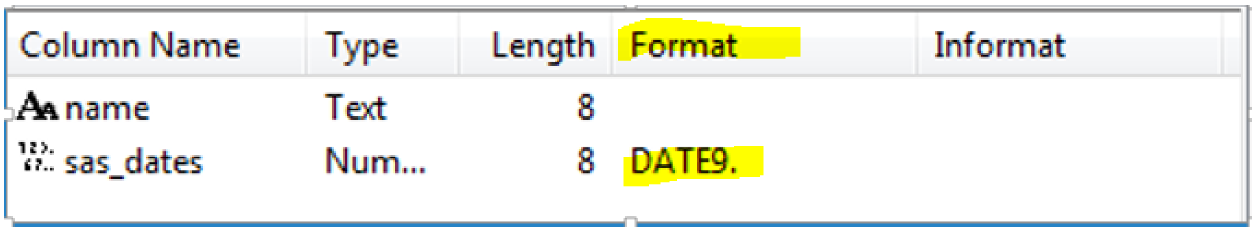
Column Name	Type	Length	Format	Informat
name	Text	8		
sas_dates	Num...	8		



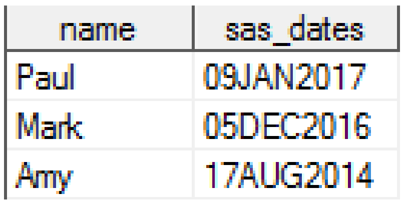
name	sas_dates
Paul	20828
Mark	20793
Amy	19952

Let’s say that we apply a SAS format to the variable using the syntax below:

```
Format sas_dates date9.;
```



Column Name	Type	Length	Format	Informat
name	Text	8		
sas_dates	Num...	8	DATE9.	



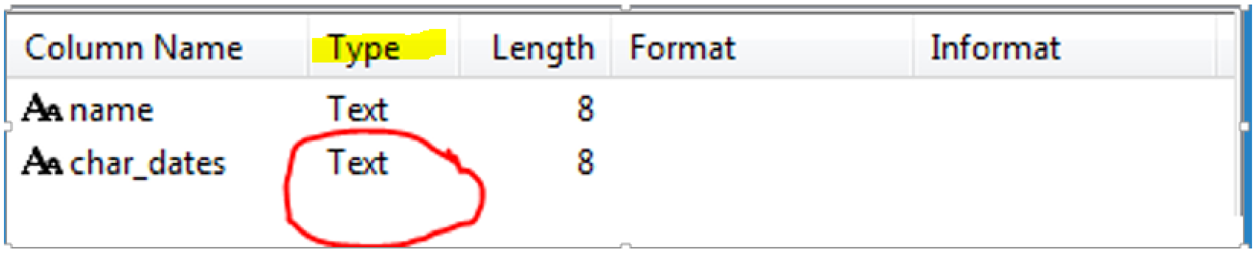
name	sas_dates
Paul	09JAN2017
Mark	05DEC2016
Amy	17AUG2014

When we do that, the values do look to a human like dates. We could further test what we are looking at by applying different SAS date forms to watch the appearance of values change (e.g. from 09JAN2017 to 01092017).

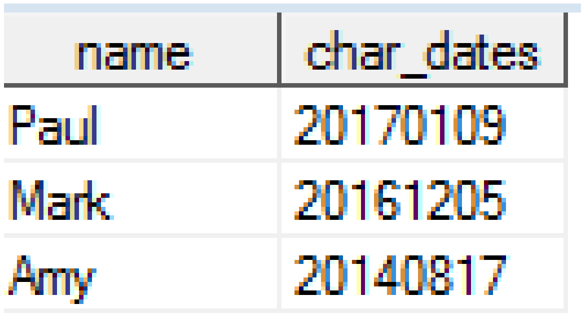
<u>Possibility 2) A character variable</u>

Sometimes you find a variable which is character (type=text). If you have a character variable, it is never a real SAS date! However, it is possible that your character variables meaningfully represent actual points in time.

For example, in the screenshots below, we have a text variable, and when looking at the values, we see that they look like years, months, and days. One reason an analyst might choose a character variable to represent dates is to be able to have a record of values submitted that are not valid dates, such as a handwritten response of “April 32, 2017” along with other values which do have valid dates.



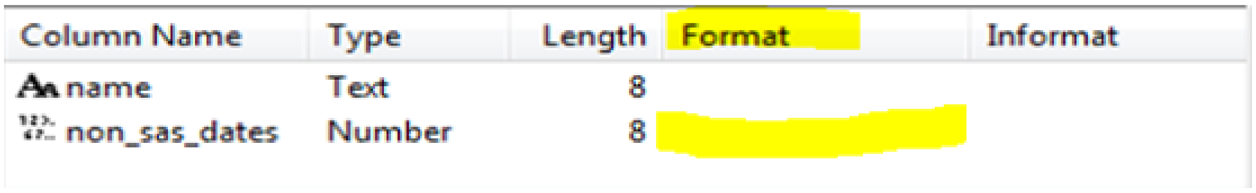
Column Name	Type	Length	Format	Informat
name	Text	8		
char_dates	Text	8		



name	char_dates
Paul	20170109
Mark	20161205
Amy	20140817

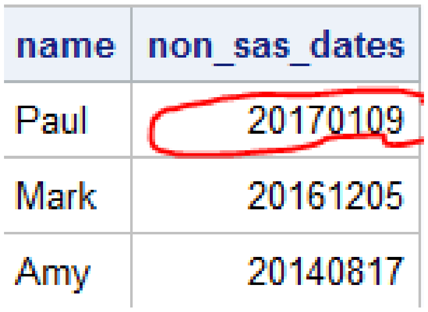
<u>Possibility 3) A numeric variable that does not have real SAS dates</u>

This possibility might require attention to notice. If a numeric variable does not have a date format applied (as in the first screenshot below with yellow highlighting), but the values look like dates to a human anyway, it is not a real SAS date!



Column Name	Type	Length	Format	Informat
name	Text	8		
non_sas_dates	Number	8		

For example, the screenshot below has values that seem to represent a year (2017), month (01), and day (09).



name	non_sas_dates
Paul	20170109
Mark	20161205
Amy	20140817

With numeric variables, SAS can fundamentally only count one thing at a time: a number of days, sticks, coins, ponies, etc. In the example above, the value for Paul is 20,170,109. If you applied a date format like date9. it would apply those to 20,170,109 days since Jan. 1, 1960 – *not* to a date in year 2017.

If you try to use "human-to-SAS-date-speak" with a variable that is not a SAS date, like'17mar2016'd " or MDY(3,17,2016), then SAS will not know what you are talking about. SAS can only use syntax like that if you are actually working with numeric variables that have the actual number of days since Jan. 1, 1960 to represent your dates.

FIRST TASK

Now we'll compare how we would handle a work task - how to choose one year of data based on dates of service – after you determine what kind of variable you're working with.

<u>Possibility 1) A numeric variable with real SAS dates</u>

If you have SAS dates, you can perform your task by using either of the previously-discussed ways of translating from a human to SAS in order to define a range of dates.

Here's one way:

```
data two; set one;

if '01sep2015'd le date_of_service le '31aug2016'd ; run;
```

Here's another way:

```
data two; set one;

If MDY(9,1,2015) le date_of_service le MDY(8,31,2016) ; run;
```

<u>Possibility 2: A a character variable</u>

Your task is to define a range of dates, but it's not workable to do that with values of a character variable (which, as far as SAS is concerned, have meanings as disconnected as "apple", "shoe", and "tree.") In order to pick a range of dates, you'll need to convert the values of the character variable to values of a numeric variable.

```
newvar=input(left(char_10_fake_date),yymmdd10.);
```
[3]

It's important to use an informat (in the example above, yymmdd10.) which describes how SAS should read in the values. If there are values of the character variable that cannot translate to valid dates using that informat (e.g., if you had a value "9999/99/99" for the character variable char_10_fake_date), the numeric variable newvar will be given missing values.

Possibility 3) A numeric variable that does not have real SAS dates

If you have a numeric variable that contains meaningful information about months, days, and years, but it is not a real SAS date, one option would be to convert it to SAS dates using syntax somewhat similar to the syntax above. This is a safe option that would work for a wide range of tasks (add a number of days to a date, use INTCK or INTX functions, calculate an age from two dates, etc.), including the current one of looking for a range of dates.

The following syntax changes fake numeric dates to SAS dates, although it's important to check the values in newvar after converting. As with the syntax on the preceding page, the informat must match the way that the values of the numeric fake date are arranged (for example, yymmdd10. would be appropriate for values like 2015/01/09 but not for values like 01/09/2015). Any values of num_fake_date that cannot be read in as real dates using the informat below will have missing values for the numeric variable newvar.

```
Newvar=input(left(put(num_fake_date, best.)),yymmdd10.);[4]
```

There is an alternative approach to working with numeric fake dates, which can only work for a limited number of tasks. The crucial aspect to making this work is to think through both what you see and what SAS sees.

For example, with a numeric fake date with values like 20150901, SAS sees these values as numbers (20,150,901 ponies) and not as month=09, days=01, and year=2015. However, in the example below, you can do something that works for you as a human (choosing observations that are between two dates) and for SAS (which is doing work based on an assumption that we're talking about 20 million ponies).

Please note that this only works because all of the dates between September 1, 2015 and August 31, 2016 have fake-date values with numbers which correspond to the numbers in the range (20150901, 20150902, 20150903, etc.). If the fake dates had been in a format with months leading instead of years, it would not be possible to choose the range you want without converting to real SAS dates.

```
data two;

set one;

if 20150901 le date_of_service le 20160831 then output two;

run;
```

One thing to keep in mind is that if there are values of the fake date variable with values such as "20159999," with 99 representing missing values, observations with these values *would* be selected in the range above, but would be converted to missing if you tried to convert them to real SAS date values.

SECOND TASK

A second task which requires different tools depending on the type of variable is to choose all clients with a specific date of birth - in this case, May 8, 2017.

Possibility 1) A numeric variable with real SAS dates

Working with real SAS dates is straightforward, as long as you use the ways of translating between SAS and humans that have been previously discussed.

Here's one way:

```
data two; set one;

if dob='08may2017'd then output two; run;
```

Here's another way:

```
data two; set one;

if dob=MDY(5,8,2017) then output two; run;
```

Possibility 2: A character variable

If you know exactly how the dates are formatted, it would be possible to identify a specific date by searching for the text string. For example, you could use a statement:

```
if char_dob="5/8/2017" then output two;
```

Alternatively, you could convert the character variable to SAS dates.

```
Newvar=input(left(char_10_fake_date_1),yymmdd10.);
```
[5]

Possibility 3) A numeric variable that does not have real SAS dates

One way to handle a numeric variable with fake SAS dates is to change it to real SAS dates. This is a good go-to solution which would work for a wide range of tasks (add a number of days to a date, use INTCK or INTX functions, calculate an age from two dates, etc.). Syntax to do this is:

```
Newvar=input(left(put(num_fake_date,best.)),yymmdd8.);
```
[6]

Another way to handle it, which would work only for certain tasks, is to look for the exact number that looks to humans like May 8, 2017. Because the values for this variable are numeric, do not use quotation marks.

```
data two;

set one;

if dob=20170508 then output two;  /*SAS still thinks we're talking about 20 million ponies*/

run;
```

A FEW MORE THINGS TO THINK ABOUT

Variables that Give Month and Year

You can read in strings from a text file like “201611” with yymmn6. informat and they will turn into SAS dates.

That means that SAS turns each value into a # of days since Jan. 1, 1960. The strings like 201611 had no day information, but when you created the SAS dates, they defaulted to day 1 of the month. You can change to date9. format to see them differently.

So if you need to select a range based on month of service, you treat these just like the real SAS dates that they are:

```
if ’01sep2015’d le date_of_service le ’31aug2016’d;
```

How SAS thinks about Missing Values with Numeric Data

Task: You need to choose all rows that have a value below some number (including a SAS date), like:

1. dates of service before May 5, 2017, or
2. all clients under age 18 at date of service.

SAS *thinks of missing values as a very small negative number.*

(like -99999999999999999999999999999999999999999)

So **don’t do this!!!**

```
if date_of_service < ‘01may2017’d then group = ‘early group’;

if age < 18 then agegroup = ‘children’;
```

When you choose numeric values less than some value, **rows with missing values for those variables will be included**, which is generally not what you want!!! (You don’t want to say that all patients with missing age are children, for example).

Better ways to choose values less than some value of numeric data are:[6]

Set a lower end of the range:

```
data two;

set example;

if ‘01may1800’d < date_of_service < ‘01may2017’d then output group = ‘early group’;

run;
```

Or say less than and also not missing:

```
data two;

set example;
```

```
    if (date_of_service < ‘01may2017’d) and (date_of_service is not missing) then group = ‘early
    group’;

    run;
```

CONCLUSION

Dates that make sense to humans include separate values for months, days and years, whereas dates that make sense to SAS are a number of days since January 1, 1960.

A variable has SAS dates if:

- The variable in your SAS data set has type = number AND
- The values *don’t* make sense to you as dates when you have no date format applied AND
- The values *do* make sense as dates when you do have a date format like yymmdd10. applied AND
- You can try on different date formats and have the appearance of the values change.

A variable with “date” in the name has SAS dates if it meets the criteria above, or otherwise might be a character variable or a numeric variable with fake dates. The tools you use to work with your “date” variable vary depending on which of these possibilities is the case, and include converting values to SAS dates if they are not already.

REFERENCES

1 *The Little SAS Book* “Selected Standard Formats” and http://support.sas.com/documentation/cdl/en/etsug/63939/HTML/default/viewer.htm#etsug_intervals_sect011.htm.
2 For more, see “The Essentials of SAS Dates and Times” by Derek Morgan. Paper 1334-2015. Available: https://support.sas.com/resources/papers/proceedings15/1334-2015.pdf.
3 SAS documentation at http://support.sas.com/kb/24/591.html and Stephanie Easterday, Texas Health and Human Services Commission.
4 Stephanie Easterday, Texas Health and Human Services Commission.
5 Ibid SAS documentation at http://support.sas.com/kb/24/591.html and Stephanie Easterday, Texas Health and Human Services Commission.
6 Ibid Stephanie Easterday, Texas Health and Human Services Commission.
7 Ron Cody discusses this in multiple writings, including “Data Cleaning 101,” The Institute for Digital Research and Education, UCLA. Available: https://stats.idre.ucla.edu/wp-content/uploads/2016/02/ss123.pdf.