Paper 1623-2014

Title for SAS® Global Forum 2014 Sample Paper

Jenine Milum, Equifax Inc.

ABSTRACT

No matter how long you've been programming in SAS®, using and manipulating dates still seems to require effort. Learn all about SAS dates, the different ways they can be presented, and how to make them useful. This paper includes excellent examples for dealing with raw input dates, functions to manage dates, and outputting SAS dates into other formats. Included is all the date information you will need: date and time functions, Informats, formats, and arithmetic operations.

WHAT IS A SAS® DATE

A date is unique within SAS programming. It is neither a character value nor a traditional numeric. A SAS date is a special representation of a calendar date.

Unlike dates in many other languages, SAS has a specific numeric value assigned to each day, ever. The starting point for all SAS dates is January 1st, 1960 and is represented as day zero (0). All previous and subsequent dates are represented with numeric values plus (+) or minus (-) from this starting point. The simplicity of this approach is there will never be a point in the past or future that cannot be represented by a number.

Traditional Date		SAS® Date
December 30, 1959	←→	-2
December 31, 1959	←→	-1
January 1, 1960	←→	0
January 2, 1960	←→	1
January 3, 1960	←→	2
------		------
January 23, 1963	←→	1118
March 25, 2014	←→	19807

Table 1. Date assignments

HOW TO TEST A DATE

There will be times you may want to test a date to see what the SAS date represents. The code below verifies several of the dates in Table 1.

```
data _null_;
  today     = date(); **;
  Test_Date = '23jan63'd;
  Day_0     = '01jan1960'd;

  put '    Today : ' today;
  put 'Test Date : ' Test_Date;
  put '    Day 0 : ' Day_0;
run;
```

Log shows:

```
    Today : 19807
Test Date : 1118
    Day 0 : 0

** Note, date() used for this paper is March 25th, 2014.
```

Example 1. Testing Dates

Assigning and setting a date is done by one of the 2 methods above. Either a 2 or 4 digit year is acceptable. The main point is the date is encapsulated by quote marks, immediately by a **d**. This notation allows SAS to know the contents inside the quotes are a date.

INFORMATS - CONVERTING A TEXT DATE TO SAS® DATE

Often the dates we are provided comes to us in the form of a character value from a text field.
There are several ways to let SAS know the value is really a date and allow SAS to convert it into a true SAS date. Utilizing SAS **informats** is the most efficient method to make the conversion.

```
*Raw data as Input;

data Test_Date;
  infile cards;
  input var1 mmddyy10.;

cards;
01/23/1963
;
run;
```

```
*Character data as Input;

data Test_Date;
  var1 = '01/23/1963';
  sas_date = input(var1,mmddyy10.);
run;
```

Example 2. Reading character input

These examples work for all versions of SAS. It takes the information from text and tells SAS the information is a date in a month-month, day-day, year-year format for a length of 10. SAS assumes the slashes (/), or other acceptable separation character, is included in the length of 10.

SAS INFORMATS

Table 2 provides examples of the more common SAS date Informats and the text values they will convert. These informats are valid for all SAS versions.

Traditional Date	Informat
01/23/1963	mmddyy10.
1/23/1963	mmddyy10.
01/23/63	mmddyy8.
1/23/63	mmddyy8.
January 23,1963	*worddate20.*
jan 23, 1963	*worddate12.*
23jan1963	date9.
23jan63	date7.
23-jan-1963	date11.
01-23-63	mmddyy8.
19630123	yymmdd8.

Table 2. Date Informats Sample

The Informats shown in Table 2 worked in v 9.13 and older. For v 9.3 and higher, a handful are no longer available (example: worddatex.). For those that have been retired, and for many of the informats that still exists can be replaces with one Informat: anydtdtew.

SYSTEMS DATES

There are many situations where a date needs to be dynamic. SAS provides several opportunities to extract a date from the system you are running. Below are a few of the system dates.

```
   data _null_;
     a = date(); **;
     b = today();
     c = "&sysdate"d;
     d = "&sysdate9"d;
     put 'Log shows: ' a b c d;
   run;

Log shows:
19807 19807 19807 19807

** Note, date() used for this paper is March 25th, 2014.
```

Example 3. Using System Dates

As you can see, each returns the same results. NOTE: **&sysdate** and **&sysdate9** return the date the SAS session was started, not necessarily the current date. The system dates are treated as macro variables and require double quotes with the **d** designation. All of these variables now contain valid SAS dates.

DATE FUNCTIONS

Date variables are unique within SAS; therefore have their own set of functions. You may need to extract a specific part of a date for use while maintaining its designation as SAS date value.

As an example; you need to identify the day of the week a specific date occurred. Below are a handful of date functions and the results from utilizing them.

```
data _null_;
  Test_Date = '23jan63'd;
  date1 = day(Test_Date);
  date2 = month(Test_Date);
  date3 = year(Test_Date);
  date4 = qtr(Test_Date);
  date5 = weekday(Test_Date);
put date1 date2 date3 date4 date5;
run;

Log shows:
23 1 1963 1 4
```

Example 4. Using Date Functions

As you can see from the example above, the **day function** returns the day of the month, **month function** returns the month of the year and the **year function** returns the 4 digit year. The **qtr function** returns the quarter of the year where January - March are quarter 1, etc. The **weekday function** returns values 1 through 7 where day 1 is Sunday and so on representing the 7 days of the week.

DATE FUNCTIONS AND DATE ARITHMETIC

Being able to derive additional information from dates is an endeavor requiring the use of additional SAS date functions.

As an example of a few scenarios, one may use date functions to determine:

- time between two intervals
- alter a point in time by a duration of time
- subtract time intervals

```
data _null_;
  date = date(); **;
  Test_Date = '23jan63'd;
  Datedif1 = intck('month',Test_Date,date);     * intck('interval' from, to );
  datedif2 = intnx('month',Test_Date,-1,'e');   * intnx('interval',from,
                                                        increment,<alignment>);
  datedif3 = sum(date,-Test_Date);              * sum(to ,-from );
  datedif4 = datdif(Test_Date,date,'act/act');  * datdif(from,to ,'act/act')or
                                                  '30/360';

  put 'Log shows: ' datedif1 datedif2 datedif3 datedif4 datedif4 mmddyy8.;

run;
```

Results:

```
Log shows: 18689 18689 614 1095 12/31/62


** Note, date() used for this paper is March 25th, 2014;
```

Example 5. Using Date Arithmetic Functions

Referring to Example 5:

- The **intck function** returns the integer number of time "intervals" from one date to another. In this case, with the interval of Month, it is the number of months from March 25th, 2014 to the Test_Date of January 23rd, 1963.

- The **intnx function** advances a date by a given interval and returns a date value. In this case, with the interval of Month, the increment of -1 and the alignment of 'e' returns the last date of the month 1 month prior to the from date. December is the month (-1) from January 23rd, 1963. December 31st is the last day 'e' of that month.

  Creating monthly reporting on a previous month is a good example for using this date function. If today we are in March, I am most likely providing reporting through the end of February. I would want the last day of February to be the date I am working with throughout that program.

- The **sum function**, while very valid with other forms of numeric's, also works with dates. As expected, it returns the number of days between one date interval and another.

- The **datdif function**, in this example using 'act/act' (actual/actual), returns the number of days between one point in time and another point in time. The result is the number of days between March 25th, 2014 to the Test_Date of January 23rd, 1963.

The **intck function** can be a very handy function while working on dates. Looking into this function further, below is a handful of "intervals" and they're returned results.

```
data _null_;
  Date = date();
  Test_Date = '23jan63'd;
  years     = intck('year',Test_Date,date());
  quarters  = intck('qtr',Test_Date,date());
  months    = intck('month',Test_Date,date());
  weeks     = intck('week',Test_Date,date());
  days      = intck('day',Test_Date,date());
  put 'Log shows: ' years quarters months weeks days;
run;

Results:
Log shows: 51 204 614 2670 18689

** Note, date() used for this paper is March 25th, 2014.
```

Example 6. Using Date Arithmetic Advanced Functions

The results from Example 6 shows the # of years, # of quarters, # of months, # of weeks and # of days. This is all accomplished with the use of a single date functions and altering the "interval".

DATE FORMATS

Outputting dates from SAS is yet another ability of SAS to manipulate dates. Most people don't know what today's date is in SAS. People need to be able to visualize what date is represented in one of the normal and acceptable forms. Date formats are used to control the look and results of dates that are currently in SAS form. You can presents dates in data fields, in report titles or labels. There are quite a few date formats. They may be easily located in SAS Help and other SAS Documentation. Below are a few to give you a feel of how to use them.

```
data _null_;
  Test_Date = '23jan1963'd;        * SAS date 1118;
  date1 = put(Test_Date,mmddyy8.);
  date2 = put(Test_Date,worddate15.);
  date3 = put(Test_Date,monyy7.);
  date4 = put(Test_Date,julian5.);
  put date1 date2 date3 date4;
run;

Results:
Log shows: 01/23/63 Jan 23, 1963 JAN1963 63023
```

The formats above clearly control the look of a SAS date. Even though the date being used above is the SAS date 1118, it is output in a form familiar to us all.

WARNING SIGNS OF BAD DATES

There are always times when SAS dates aren't manipulated properly. Signs of such occasions are null values (.) or the occurrence of too many January 1st, 1960 values, meaning the date really returned a zero.

A LISTING OF MOST SAS DATE FUNCTIONS, INFORMATS AND FORMATS – ALL IN ONE PLACE!

DATE FUNCTIONS

DATDIF returns the number of days between two dates **DATE** returns the current date as a SAS date value **DATEJUL** converts a Julian date to a SAS date valueDATEPARTextracts the date from a SAS datetime value **DATETIME** returns the current date and time of day as a SAS datetime value **DAY** returns the day of the month from a SAS date value **DHMS** returns a SAS datetime value from date, hour, minute, and seconds **HMS** returns a SAS time value from hour, minute, and seconds **HOUR** returns the hour from a SAS time or datetime value **INTCK** returns the integer number of time intervals in a given time span **INTNX** advances a date, time, or datetime value by a given interval, and returns a date, time, or datetime value **DATE** returns the Julian date from a SAS date value	**JULDATE7** returns a seven-digit Julian date from a SAS date value **MDY** returns a SAS date value from month, day, and year values **MINUTE** returns the minute from a SAS time or datetime value **MONTH** returns the month from a SAS date value **QTR** returns the quarter of the year from a SAS date value **SECOND** returns the second from a SAS time or datetime value **TIME** returns the current time of dayTIMEPARTextracts a time value from a SAS datetime value **TODAY** returns the current date as a SAS date valueWEEKDAYreturns the day of the week from a SAS date value **WEEKDAY** returns an integer that represents the day of the week, where 1=Sunday, 2=Monday,…, 7=Saturday **YEAR** returns the year from a SAS date value YRDIF returns the difference in years between two dates **YYQ** returns a SAS date value from the year and quarter

DATE INFORMATS

http://support.sas.com/documentation/cdl/en/leforinforref/64790/HTML/default/viewer.htm#p13eggx7qc7l6in1oprinvadzubl.htm

DATEw. day, month abbreviation, and year: 17oct91 ddMONyy **DATETIMEw.d** date and time: ddMONyy:hh:mm:ss 17oct91:14:45:32 **DDMMYYw.** day, month, year: ddmmyy , dd/mm/yy , 17/10/91 dd-mm-yy, or dd mm yy **JULIANw.** year and day of year (Julian dates): yyddd 91290 **MMDDYYw.** month, day, year: mmddyy , mm/dd/yy , 10/17/91 mm-dd-yy, or mm dd yy	**MONYYw.** month abbreviation and year Oct91 **TIMEw.** dhours, minutes, seconds: hh:mm:ss 14:45:32or hours, minutes: hh:mm. **YYMMDDw.** year, month, day: yymmdd , yy/mm/dd , 91/10/17 yy-mm-dd, or yy mm dd **YYQw.** year and quarter of year: yyQq 91Q4

DATE FORMATS

http://support.sas.com/documentation/cdl/en/leforinforref/64790/HTML/default/viewer.htm#p0z62k899n6a7wn1r5in6q5253v1.htm

DATEw. day,month abbreviation,year: ddMONyy 17oct91 **DAYw**. day of month 17 **DDMMYYw**. day,month,year: dd/mm/yy 17/10/91 **DOWNAMEw**. name of day of the week Thursday **JULDAYw**. day of year 290 **JULIANw**. year and day of year: yyddd 91290 **MMDDYYw**. month, day, year: mm/dd/yy 10/17/91 **MMYYw.** month and year: mmMyy 10M1991 **MMYYCw.** month and year: mm:yy 10:1991 **MMYYDw**. month and year: mm-yy 10-1991 **MMYYPw.** month and year: mm.yy 10.1991 **MMYYSw**. month and year: mm/yy 10/1991 **MMYYNw**. month and year: mmyy 101991 **MONNAMEw.** name of month October **MONTHw**. month of year 10 **MONYYw.** month abbreviation and year: OCT91 **MONyyQTRw.** quarter of year 4 **QTRw.** quarter of year 4	**QTRRw**. quarter in Roman numerals IV **WEEKDATEw.** day-of-week, month-name dd,yy Thursday, October 17, 1991 **WEEKDATXw.** day-of-week, dd month-name Yy Thursday, 17 October 1991 **WEEKDAYw**. day of week 5 **WORDDATEw**. month-name dd, yy October 17, 1991 **WORDDATXw.** dd month-name yy 17 October 1991 **YEARw.** year 1991 **YYMMw**. year and month: yyMmm 1991M10 **YYMMCw.** year and month: yy:mm 1991:10 **YYMMDw**. year and month: yy- mm 1991-10 **YYMMPw**. year and month: yy.mm 1991.10 **YEARw**. year 1991 **YYMMw.** year and month: yyMmm 1991M10 **YYMMCw.** year and month: yy:mm 1991:10 **YYMMDw.** year and month: yy- mm 1991-10 **YYMMPw.** year and month: yy.mm 1991.10 **YYMMSw**. year and month: yy/mm 1991/10 **YYMMNw**. year and month: yymm 199110 **YYMONw**. year and month abbreviation: 1991OCT

YYMMDDw. year, month, day: yy/mm/dd 91/10/17 **YYQw**. year and quarter: yyQq 91Q4 **YYQCw**. year and quarter: yy:q 1991:4 **YYQDw**. year and quarter: yy-q 1991-4 **YYQPw**. year and quarter: yy.q 1991.4 **YYQSw**. year and quarter: yy/q 1991/4 **YYQNw**. year and quarter: yyq 19914	**YQRw**. year and quarter in Roman 1991 **QIV** numerals: yyQrr **YYQRCw**. year and quarter in Roman 1991:IV numerals: yy:rr **YYQRDw**. year and quarter in Roman 1991-IV numerals: yy-rr **YYQRPw**. year and quarter in Roman 1991.IV numerals: yy.rr **YYQRSw**. year and quarter in Roman 1991/IV numerals: yy/rr **YYQRNw.** year and quarter in Roman 1991IV numerals: yyrr

CONCLUSION

Presented here are the most common tools used while working with dates in SAS programming. Quite frequently, there is more than one valid method. They are part of nearly all programming efforts in one capacity or another.

REFERENCES

SAS 9.4 Help and Documentation for Informats and Formats; *How to Read, Write and Manipulate SAS Dates,* presented at SESUG Nov. 2007 by this author.

ACKNOWLEDGMENTS

A big “Thank you” to several friends that helped me prepare this paper and presentation.

CONTACT INFORMATION

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

Name: Jenine Milum
Organization: Equifax INC
City, State: Atlanta, GA
Phone: 770-842-6898
Email: JenineMi@yahoo.com
sascommunity.org: http://www.sascommunity.org/wiki/User:Jeninemilum