Working With SAS® Date and Time Functions

Andrew H. Karp
Sierra Information Services
San Francisco, California
- and -
University of California, Berkeley
Extension Division

Introduction

Many SAS® applications require that operations be performed on data collected in the time domain. Among these types of operations are:

- determining the frequency with a phenomenon of interest occurs in time
- determining the a time interval which has elapsed between two phenomena (e.g., length of stay I between admission and discharge from the hospital)
- operating conditionally on observations in a SAS® data set based on date and/or time values (e.g., select from a SAS® data set only those observations which occurred after a particular point in time).
- aggregation of observations from one time frequency to another (e.g., daily to monthly)
- interpolation (i.e., estimation) of lower frequency observations from data collected at a higher frequency (e.g., estimating monthly values from data collected quarterly)

The SAS® System for Information Delivery provides a wealth of tools for users who need to work with data collected in the time domain. These tools include **functions** which can:

- create a SAS date, time or datetime variable from either raw data or from variables in an existing SAS data set
- determine the interval between two periods
- declare a SAS date or time variable as a constant
- extract 'parts' from a SAS date variable, such as the month, day or week, or year

A second set of tools, SAS® date/time **formats**, modify the *external representation* of a SAS® date or time variable. Over 30 such formats are supported in Version 6 of SAS, and a user can create customized formats for date, time or datetime variables using PROC FORMAT.

Finally, SAS® date/time **informats** are able to convert raw data into a date, time or datetime variable.

Key Concepts

A SAS® **date, time or datetime variable** is a special case of a numeric variable. The values of **date variable** represent the number of days before or after January 1, 1960, and the values of a **time variable** are the number of seconds since midnigh A time variable is independent of date. Similarly, a SAS® **datetime** variable contains both date and time information (e.g., January 20, 1994 at 4:13 p.m.).

Creating a SAS® Date Variable

Several methods exist to create a date, time or datetime variable from either raw data or from numeric variables in an existing SAS® data set. Users should determine how a date or time variable is represented in their data set in order to choose the appropriate method by which to creat the desired date or time variable.

Example 1: Using the YYMMDD8. Informat

A variable in a raw data set which has the format YYMMDD8. (e.g., 19940704 for July 4, 1994) ca be converted to a SAS date variable by using the YYMMDD8. ***informat*** in the Data Step. Using this ***informat*** will 'convert' the values of the variable to SAS date values. Other ***informats*** include DDMMYY8. , MMDDYY8. and DATETIME18.

Example 2: Using the MDY Function

In the preceding example a variable is present in the raw data set which is in the form YYMMDD It is not uncommon to encounter situations whe separate variables exist for the month, the day, a the year of the observation. In this case a SAS d variable can be created by using the ***MDY Function***. The form of this function is:

Newvar = MDY(Monvar, Dayvar, Yearvar)

Where:

Newvar = name of new variable to be created
Monvar = numeric variable representing Month
Dayvar = numeric variable representing Day
Yearvar = numeric variable representing Year

A constant can be substituted for any of the variables in the function. For example, if all of the observations in the data set occurred in 1993, the user can specify:

Newvar = MDY(Monvar, Dayvar, 1993)

and SAS will assume that all observations for which the variable Newvar is to be created occurred in the year 1993. Declaring a constant term of 1 for the day variable is a common method to aggregate daily observations in to monthly observations (see below).

Extracting 'parts' from a SAS Date Variable

Several SAS® functions are available to obtain information about the values of a SAS® date variable. These include:

MONTH	Returns the month
DAY	Returns the day
YEAR	Returns the year
QTR	Returns the quarter
WEEKDAY	Returns the day of the week (1 = Sunday)

Example: July 4, 1994

Assume that a SAS® variable exists in a SAS® data set which contains the dates of all (U.S.) Federal holidays in 1994. One observation is available for each holiday, and the SAS® variable name is **HOLIDAY**. Applying the above SAS® functions to this variable obtains the following results for July 4, 1994 (Independence Day):

X = MONTH(HOLIDAY)	X = 7
X = DAY(HOLIDAY)	X = 4
X = YEAR(HOLIDAY)	X = 1994
X = QTR(HOLIDAY)	X = 3
X = WEEKDAY(HOLIDAY)	X = 2

Extracting 'parts' from a SAS® Time Variable

Users can obtain the values of the 'parts' of a SAS® time variable using the **HOUR, MINUTE and SECOND** functions. These functions are applied in the same fashion as the date functions listed above.

Extracting the DATE and/or TIME 'parts' of a SAS® Datetime variable

The date and or time elements (or 'parts') of a SAS datetime variable can be extracted using the DATEPART and TIMEPART functions. Suppose a patient is admitted to the hospital at 5:45 p.m. on Monday, July 11, 1994, and a datetime variable (called ADMIT) is present in the hospital's patient database with this information. (This value is the number of seconds between January 1, 1960 and 5:45 p.m. on July 11, 1994). If only the date of admission is desired,

DATE = DATEPART(ADMIT); will return the SAS date

Similarly, the time of admission can be obtained by:

TIME = TIMEPART(ADMIT); will return the SAS time

Declaring a SAS® Date, Time or Datetime Constant

Special features in the BASE SAS ® product allow users to declare a particular date or time as a constant without having to know the number of days from January 1, 1960 and/or the number of seconds since midnight. A date or time constant is declared by enclosing a date or time in single quotes, followed by the letter D and/or T to signify either date or time or DT to signify a datetime variable. For example:

X = '04JUL94'D will set the new variable X equal to the number of days between January 1, 1960 and July 4, 1994. Similarly:
Y = '0900'T sets the new variable Y to the number of seconds between midnight and 9 am.
Z = '04JUL94:12:00'DT sets the value of the variable Z to the number of seconds from January 1, 1960 to noon on July 4, 1994.

The TODAY( ) Function

This function returns today's date as a SAS® date value from your computer's system clock. Note that there are no spaces placed between the Y and the left parenthesis in TODAY and there are no spaces places between the left and right parentheses.

Using SAS ® Date, Time or Datetime Values/ Functions in the Data Step and WHERE Clauses

Date, time, and datetime values and functions are easily used in the Data Step and WHERE Clauses (both as Data Set options and to restrict operation

of a SAS Procedure to a specified subset of cases in a SAS® Data Set).

Subsetting IF statement

Assuming the SAS date variable ADMIT (from the previous hospital admission example) resides on a SAS® data set, a subset of observations for patients admitted after December 31, 1993 can be obtained by specifying:

```
IF ADMIT GT '31DEC93'D;
```

An equivalent programming statement would be:

```
IF YEAR(ADMIT) > 1993 ;
```

If the variable ADMIT were a SAS *datetime* variable, a subset of observations for patients admitted on or after 4 p.m.on December 20, 1993 can be obtained by specifying:

```
IF ADMIT => '20DEC93:16:00'DT;
```

Compound Statements

Users can combine SAS date, time and datetime statements in a single expression, as may be warranted by the particular analytic situation at hand. Appropriate combination of these statements reduces processing time and programming steps.

Compound Subsetting IF Statement: Example 1

Two or more conditions for a SAS date, time or datetime variable can be tested simultaneously. If, for example, a researcher working on the above-described hospital admissions data set who desired to only look at records where patients were admitted in the third quarter of 1993 could write:

```
IF YEAR(ADMIT) = 1993 and
QTR(ADMIT) = 3;
```

This is faster than:

```
IF YEAR(ADMIT) = 1993;
If QTR(ADMIT)  = 1993;
```

The first method processes the records in the data set much more efficiently than the first.

Compound Subsetting IF Statement: Example 2

As with the previous example, placing all subsetting conditions in a single programming step enhances program performance. If the variable ADMIT was a datetime variable, the following compound statement would select only those observations where the patient was admitted on a Saturday or Sunday after 4:00 pm during the second quarter of 1993:

```
IF YEAR(DATEPART(ADMIT)) = 1993
      AND
QTR(DATEPART(ADMIT)) = 2
      AND
WEEKDAY(DATEPART(ADMIT)) => 6
      AND
HOUR(TIMEPART(ADMIT)) > 4;
```

Notice that the above statement operates from 'largest' to 'smallest' time interval : year, quarter, weekday, hour. This arrangement reduces the amount of time each record in the data set will be read. For example, all admissions in 1992 and earlier will be immediately 'discarded' without having their values of quarter, weekday and hour tested.

Compound Statements Using the MDY Function

The MONTH, YEAR and DAY functions can be used 'within' the MDY function to quickly create variables for use in subsequent programming or analysis actions. A common requirement is to aggregate daily observations to monthly observations (e.g., hospital discharges by month). If a SAS® date variable called **DIS** represents the date on which a patient was discharged, a new variable called **DISMTH** might be created using the following SAS® statement:

```
DISMTH=
MDY(MONTH(DIS)),1,(YEAR(DIS));
```

This statement creates a new SAS® date variable representing the number of days between January 1, 1960 and the **first day of the month** in which the patient was discharged and might be subsequently used by PROC SUMMARY or PROC MEANS in a BY or CLASS statement.

External Representations of SAS® Date, Time and Datetime Variables

Characterizing of date, time and datetime variable as numeric variables makes it easier for the SAS® System to operate on them than if they were character variables. This, however, makes in nearly impossible for a (human) end-user to discer the values of these variables and to represent them in a meaningful way in reports or other output.

This problem is easily solved by appropriate use o one of the more than 30 SAS® **formats** for date, time or datetime variables. These are documente in the SAS® Language: Reference, Version 6 text and are summarized on pages 66 and 67 of the SAS® Language and Procedures: Syntax, Version handbook. Among the commonly used formats are:

Format:	Result:
MMDDYY8.	07/04/94
DDMMYY8.	04/07/94
WORDDATE18.	July 4, 1994
WEEKDATE29.	Monday, July 4, 1994
MONYY5.	JUL94

Calculating Time Intervals

A common application of SAS® System date and time capabilities is to determine how long a period has elapsed between two points in time. This can be accomplished by one of two methods:

- arithmetic operation (usually subtraction and/or division) between two SAS date, time or datetime variables or between a SAS date, time, or datetime variable and a constant term
- use of the INTCK function

Arithmetic Operation

The number of days which have elapsed between two points in time is easily determined by subtracting the value of one SAS® date variable from another, or by subtracting a SAS® date variable from a SAS date constant (or vice versa, as may be appropriate). The result can then be divided by an appropriate constant to obtain the desired number of time periods between the two values. A common requirement is to determine how many years have elapsed between two time periods:

YEARS = (date2 - date1)/365.25;

will return the number of units of 365.25 which have occurred between the two date variables. This is a commonly accepted practice to determine the number of years occurring between two points in time. Similarly, 30.4 is frequently used as the denominator to convert the number of days to the number of months.

INTCK Function

A popular and powerful SAS® function, INTCK, is available to determine the number of *time periods* which have been *crossed* between two SAS® date, time or datetime variables. The form of this function is:

INTCK(*'interval',from,to*)

Where:

'interval' = character constant or variable name representing the time period of interest enclosed in single quotes

from = SAS® date, time or datetime value identifying the start of a time span

to = SAS® date, time or datetime value identifying the end of a time span

This function will return the number of time periods which have occurred (i.e., have been *crossed*) between the values of the *from* and *to* variables.

Arithmetic Operation vs. INTCK Function

Important differences exist between how these two methods determine the number of periods of time which have elapsed between two points in time. These differences can be demonstrated as follows. Suppose a child is born (and therefore 'admitted' to the hospital) on December 26, 1993 and discharged on January 2, 1994. The child is therefore eight days old at discharge. Subtracting

AGE = '02JAN94'D - '26DEC93'D

yields 8, which is the desired result. But, how many *years* old is the child? An acceptable estimated answer is 8/365.25, or .022 *years*. But, using the INTCK function

AGE = INTCK('YEAR','26DEC93'D,'02JAN94'D)

returns 1 as the result. Why? Because the INTCK function counts the number of time *intervals* which have been *crossed* between the *from* and *to* expressions arguments of the function. Since YEAR was the desired *interval*, and January 1 is 'enclosed' between the *from* and *to* expressions, the child's age is given by the result of the INTCK function to be 1, rather than 8 days.

Users should be aware of the important differences in results which will occur from using one or the other of these approaches and use them accordingly.

The INTNX Function

Also useful is the INTNX function, which *creates* a SAS ® date, time or datetime value that is a given number of time intervals from a starting value. The form of this function is:

INTNX('interval',from,to)

Where *interval, from* and *to* have the same meanings and definitions as for the INTCK function described earlier. For example, suppose a hospital wanted to send a postcard to the parents of newborns three months after the child is born reminding them to schedule a follow-up visit. Using:

MAILDATE = INTNX('month',BDATE,3);

Where BDATE is a date variable representing date of birth, the INTNX function will return values of MAILDATE which are the SAS date values for

the first day of the month which is three months past the child's birthday. Thus, the INTNX and INTCK functions operate in a similar fashion by counting the date *boundaries* which are *crossed.*

Alternatively, if the reminder postcard is to be generated *90 days* after the child is born, a statement such as:
MAILDATE = BDATE + 90;
will set the value of the variable MAILDATE equal to the child's birthday plus 90 days. This result will therefore be the number of days from January 1, 1960 to the child's birthday plus an additional 90 days. This result differs from that obtained by using the INTNX function, which would return the SAS date value for the *first day of the month* three months after the child was born.

Aggregating and Interpolating Data

Data collected in the time domain can be aggregated from a lower to a higher period (e.g., monthly to yearly) using BASE SAS® procedures such as MEANS, SUMMARY, or UNIVARIATE. The appropriate date, time or datetime variable(s) are used in either the CLASS or BY statements (or both, if appropriate).

Some analyses require that a time series by **interpolated** by estimating from higher-period observations to lower-period observations. Estimates of monthly values of a time series containing quarterly observations, may, for example, be desired.

Another key issue in aggregating and/or interpolating data collected in the time domain is treatment of missing observations. Some statistical techniques (e.g., Auto-Regressive Integrated Moving Average, or ARIMA) for analyzing time series data are hampered by the presence of missing values; in other situations an analyst may want to estimate values for missing observations before performing aggregation and/or interpolation operations. In any event, substantial user intervention may be required in the Data Step to substitute 'appropriate' values for missing observations.

A new procedure in the SAS/ETS® (Econometrics and Time Series) module, PROC EXPAND, is, however, uniquely suited to perform both aggregation **and** interpolation of data collected in the time domain, as well as estimation of the values of missing observations in the data set. Appropriate use of PROC EXPAND can provide users with rapid and statistically valid estimation o values of missing values as well as interpolation of lower-frequency observations from observations collected at higher frequencies. PROC EXPAND's data manipulation options also make its use preferable to the Data Step, especially when several operations (interpolate values of missing observations **and** aggregate from one period to another) are desired on the same series.

Note: SAS is the registered trademark of SAS Institute, Cary, NC, USA

References

SAS Institute Inc., *SAS Language: Reference, Version 6, First Edition* Cary, NC: SAS Institute, Inc., 1990

SAS Institute Inc., *SAS Language and Procedures: Usage, Version 6, First Edition,* Cary, NC: SAS Institute, Inc., 1989

SAS Institute, Inc., *SAS/ETS ® User's Guide, Version 6, Second Edition,* Cary, NC: SAS Institute, Inc., 1993

DiIorio, Frank C., *SAS® Applications Programming: A Gentle Introduction,* Belmont, California: Duxbu Press, 1991