WORKING WITH SAS® DATE AND TIME FUNCTIONS

Andrew H. Karp
Sierra Information Services, Inc.
San Francisco, California USA

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 that a phenomenon of interest occurs in time
- determining the a time interval which has elapsed between two phenomena (e.g., length of stay between admission and discharge from the hospital)
- conditional operations 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 Software provides a wealth of tools for users who need to work with data collected in the time domain. These tools include **functions** which:

- 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. As with other SAS System formats, a date, time or datetime format displays the values of the variable according to a specified width and form. Use of date, time or datetime formats is essential when creating applications or programs in the SAS System portraying the values of variables collected in time. Otherwise, the user will most likely not be able to "make sense" of the values of the variable itself. Over 30 such formats are supported in Version 6 of the SAS System, and a user can create customized formats for date, time or datetime variables using PROC FORMAT.

SAS date/time **informats** are able to convert raw data into a date, time or datetime variable. They read fields (i.e., variables) in either raw data files or SAS data sets . An example is given below.

Key Concepts

A SAS **date, time or datetime variable** is a special case of a numeric variable. The values of a **date variable** represent the number of days before or after January 1, 1960, and the values of a **time variable** is the number of seconds since midnight. 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.).

A SAS date value is a constant that represents a fixed value. Valid SAS dates are from 1582 A. D. (following adoption of the Gregorian Calendar) to 20,000 A. D.

Creating a SAS Date Variable

Several methods exist to create a date, time or datetime variable either from 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 create 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., 19970425 for April 15, 1997) can be converted to a SAS date variable by using the YYMMDD8. ***informat*** in the Data Step. Using this ***informat*** 'converts' the values of the variable to SAS date values. Other ***informats*** include DDMMYY8. , MMDDYY8. and DATETIME18.

Example: ANZAC Day (Australia) 1997
If a value of a variable raw data set to be read by the SAS System contained a value 25/04/97, the YYMMDD10. Informat can be used to create a SAS date variable by placing the informat in the INPUT statement.

INPUT ANZACDAY YYMMDD10.;

The SAS System will automatically convert the text representation of the raw data variable in to a SAS date variable with a value of 13629, the number of days between 1 January 1960 and 25 April 1997

Example 2: Using the MDY Function
In the preceding example, a variable is present in the form YYMMDD in the raw data. It is not uncommon to encounter situations where separate variables exist for the month, the day, and the year of the observation. In this case a SAS date variable is 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

Example: ANZAC Day (Australia) 1997
ANZAC day was celebrated on 25 April 1997 in Australia. This date can be represented as a SAS date variable by coding:
ANZACDAY = MDY(4,25,1997);
The value of this variable is 13629, the number of days between 1 January 1960 and 25 April 1997

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: 25 April 1997 (ANZAC Day)
Assume that a variable exists in a SAS data set containing the dates of all (Australia) national holidays holidays in 1997. 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 25 April 1997 (ANZAC Day):

```
X = MONTH(HOLIDAY)      /* X = 4 */
X = DAY(HOLIDAY)        /* X = 25 */
X = YEAR(HOLIDAY)       /* X = 1997 */
X = QTR(HOLIDAY)        /* X = 3 */
X = WEEKDAY(HOLIDAY)    /* X = 6 */
```

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);

returns the SAS date

Similarly, the time of admission can be obtained by:

TIME = TIMEPART(ADMIT);

returns the SAS time.

Declaring a SAS Date, Time or Datetime Constant

Special features in Base SAS Software 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 = '04JUL97'D will set the new variable X equal to the number of days between January 1, 1960 and July 4, 1997. Similarly:
Y = '09:00'T sets the new variable Y to the number of seconds between midnight and 9 am.
Z = '04JUL97:12:00'DT sets the value of the variable Z to the number of seconds from January 1, 1960 to noon on July 4, 1997.

The TODAY() Function

This function returns today's date as a SAS date value from your computer's system clock.

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

Date, time, and datetime values and functions are used easily 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 '31DEC1993'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 >= '20DEC1993:16:00'DT;

Compound Statements

Users can combine SAS date, time and datetime statements in a single expression, as may be warranted by the particular analytical 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. For example, if a researcher working on the above-described hospital admissions data set desired to only analyze records where patients were admitted in the third quarter of 1993, she could write:

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

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)) IN(1,7)
        AND
HOUR(TIMEPART(ADMIT)) > 16;
```

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

Treating date, time and datetime variables 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 discern 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 of one of the more than 30 SAS **formats** for date, time or datetime variables. These are documented in the SAS Language: Reference, Version 6 text and are summarized on pages 66 and 67 of the SAS Language and Procedures: Syntax, Version 6 handbook. Among the commonly used formats are:

Format:	**Result:**
MMDDYY8.	07/04/97
DDMMYY8.	04/07/97
WORDDATE18.	July 4, 1997
WEEKDATE29.	Friday, July 4, 1997
MONYY5.	JUL95

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;
```

returns 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.

Additional Intervals Available in Release 6.07

Four new date and datetime intervals were implemented in Release 6.07 of SAS System Software. They are:

WEEKDAY: counts the number of weekdays between two time values, with the weekend days counted as part of the preceding weekday. By default, Saturday and Sunday are considered "weekends."

TENDAY: counts the number of ten-day intervals between two time values. The default is for the month to be broken in to three periods: a) first through tenth days, b) eleventh through twentieth day, c) twenty-first day through end of the month.

SEMIMONTH: breaks each month in to two periods, starting on the first and sixteenth days of the month.

SEMIYEAR: specifies semiannual periods of six months.

Arithmetic Operation vs. INTCK Function

Important differences exist between how these two methods determine the number of periods of time that 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 28, 1994 and discharged on January 2, 1995. The child is therefore *five days old at discharge*. Subtracting

AGE = '02JAN1995'D - '28DEC1994'D

yields 5, which is the desired result. But, how many *years* old is the child? An acceptable estimated answer is 5/365.25, or .02 *years*. But, using the INTCK function,

AGE=INTCK('YEAR','28DEC1994'D,'02JAN1995'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 take in to account the important differences in results which will occur from using one or the other of these approaches and make sure that the one the apply is appropriate for their particular data processing/analysuis.

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.

Enhancements to the INTNX Function in Release 6.11 of the SAS System.

Prior to Release 6.11 the INTNX function returned a value representing the *beginning* of the interval specified in the function's third argument. For example, the value returned by specifying MONTH as the desired interval was the first day of the month.

Starting with Release 6.11, an optional fourth argument is available for the INTNX function. Users can specify BEGINNING (the default), MIDDLE, or END. Using the MIDDLE argument returns a value representing the period's midpoint and END returns a value representing the end of the period. For example, if a user wanted to advance a date value from January 1, 1997 to June 30, 1997, the END argument would be applied as follows:

NEWVAR = INTNX('Month','01Jan1997'd,5,'END');

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 software 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 be **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 of values of missing values as well as interpolation of higher-frequency observations from observations collected at lower 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.

YEAR 2000 Issues and the SAS System

The so-called "Year 2000 Problem" has raised concerns month both computer systems experts and the general public. The New York Times, for example, recently published an in-depth article on the woes potentially facing industry if the "Y2K" problem is not rectified.

Users of SAS System software can take some measure of comfort in knowing that the product is "Year 2000 Compliant" in that it stores dates and times as integers. No special steps need to

be taken with date, time and datetime values stored in existing SAS Data Sets.

A core issue, however, is how the SAS System references two-digit year values. Starting with Version 6.05, the SAS System includes the **YEARCUTOFF** system option. By default, the value of this option is set to 1900. Unless modified, the result of the YEARCUTOFF option will treat two digit year variables as occurring in the twentieth century. Users who are working with two-digit year variables representing events from other centuries need to adjust the YEARCUTOFF option value accordingly.

One rapid way by which to ensure that SAS date or datetime values refer to the appropriate century is to use a four-digit year value, rather than a two-digit value. For example, if your YEARCUTOFF system option value is set to 1900 (the default), the value of the variable **DATE='01JAN00'D** represents the number of days from January 1, 1900 to January 1, 1960, which is a negative number. If, however, the user set **DATE='01JAN2000'D**, SAS would set the value of the variable DATE to the number of days from January 1, 1960 to January 1, 2000.

As we move closer to the Year 2000, SAS System software users should both a) explore whether changing the YEARCUTOFF system option at their site is appropriate, and, b) explicitly code year values as four, rather than two digits. Using four, rather than two, digits for year values is the quickest way to avoid Year 2000 difficulties with SAS System Software, especially if changing the YEARCUTOFF system option is not appropriate at your site.

Richard D. Langston and Chris Williams of SAS Institute, Inc. presented a paper (see reference below) at SUGI 22 giving details about Year 2000 issues and the SAS System. Among the topics they present is an experimental feature, YEAR2000CHECK, which can search SAS System programs for two-digit year values. The authors stress the both the experimental and limited nature of this feature, which is described at some length in their paper.

Note: SAS and SAS/ETS are the registered trademarks of SAS Institute, Cary, NC, USA

References

Broder, John M., and Laurence Zuckerman, "Computers Are the Future But Remain Unready for It," *The New York Times,* April 8, 1997

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

Langston, Richard D., and Chris Williams, *"The Year 2000: Preparing for the Inevitable,"* Proceedings of the Twenty-Second Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 1997

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

SAS Institute, Inc., SAS® Technical Report P-222, *Changes and Enhancements to Base SAS® Software, Release 6.07,* Cary, NC: SAS Institute, Inc., 1991

SAS Institute, Inc., *SAS ® Software: Changes and Enhancements, Release 6.10,* Cary, NC: SAS Institute, Inc., 1994

SAS Institute, Inc., *SAS ® Software: Changes and Enhancements, Release 6.11,* Cary, NC: SAS Institute, Inc., 1995

The author may be contacted at:

Sierra Information Services, Inc.

1489 Webster Street Suite 1308
San Francisco, CA 94115
415/441-0702
SierraInfo @ AOL.COM
www.SierraInformation.com