PharmaSUG 2014 - Paper CC30

Times Can Be Tough: Taming DATE, TIME and DATETIME Variables

Sajeet Pavate, PPD, Wilmington, NC

ABSTRACT

Some programmers may not fully understand how the values for Date, Time and Datetime variables are stored and manipulated within SAS especially in relation with each other.
Several previous papers have provided an introduction to how date, time and datetime values work in SAS as well as the different functions and formats that apply to these variables. The aim of this paper is to show common issues that may occur while working with these variables. These issues, for example, occur when missing time component is not considered or when incorrect assumptions are made that can result in invalid or inaccurate calculations.

The paper points to some of these issues with examples from real life scenarios and includes the corrected code to fix such issues. It attempts to educate readers on how these variables are used within SAS and makes them aware and mindful of common pitfalls when working with Date, Time and Datetime variables.

INTRODUCTION

SAS Programmers are generally comfortable working with date values. When Time and Datetime variables are brought into picture, it can become complex especially if the programmer does not work with such variables on a frequent basis. It can lead to issues if one does not fully understand how these variables are used together which results in making wrong assumptions or writing incorrect code. Programmers may incorrectly assume missing time values always to be midnight ’00:00’T for ease in programming even though the specification does not imply any such imputations.

This paper shows some of the pitfalls that could arise when we do calculations using Date, Time and Datetime variables. It provides the corrected code for these scenarios and provides tips on how these issues can be avoided.

Before we start looking at the examples, we need to better understand Date, Time and Datetime values in reference to SAS.

SAS DATE VALUE

is a value that represents the number of days between January 1, 1960, and a specified date. SAS can perform calculations on dates ranging from A.D. 1582 to A.D. 19,900. Dates before January 1, 1960, are negative numbers; dates after are positive numbers.
A date of ‘01JAN1960’D is stored in the variable as a date value = 0
A date of ‘**02**JAN1960’D is stored in the variable as a date value = 1
A date of ‘**31DEC1959**’D is stored in the variable as a date value = -1

SAS TIME VALUE

is a value representing the number of seconds since midnight of the current day. SAS time values are between 0 and 86400.
A Time of ’05:00’T is stored in the variable as a time value = 18000
A Time of **’17**:00’T is stored in the variable as a time value = 61200

SAS DATETIME VALUE

is a value representing the number of seconds between January 1, 1960 and an hour/minute/second within a specified date.
A Datetime of ‘01JAN1960:00:00:00’DT is stored in the variable as a datetime value = 0
A Datetime of ‘01JAN1960:00:00:**15**’DT is stored in the variable as a datetime value = 15
A Datetime of ‘01JAN1960:**05**:00:00’DT is stored in the variable as a datetime value = 18000

Now that we know how values are stored in these variables, let us look at some examples of how datetime values can be populated from numeric date and time values.

POPULATION OF DATETIME VALUES

There are many functions to populate datetime values using numeric date and time variables. Two examples used commonly are given below. The code examples below should be placed in a DATA step.

The variables are defined as follows:
TRTEDT (Last Study Drug Dose Date), TRTETM (Last Study Drug Dose Time), TRTEDTM (Last Study Drug Dose Datetime);
ADT (Assessment Date), ATM (Assessment Time), ADTM (Assessment Datetime);

METHOD #1: TO POPULATE DATETIME VALUE

In this method, the date variable is converted into seconds by multiplying the date value with a factor of '24*60*60'. The time component which is already stored in SAS as seconds is then added to this value.

In the Code Sample 1 below, there is no check for missing time component (TRTETM). In this case, datetime value makes an incorrect assumption of time as midnight '00:00'T. If this is not explicitly specified in the programming specification then this is an incorrect imputation. This datetime variable used in subsequent calculations will produce invalid results.
Note: If a date value is missing, then a SAS Note that Missing values were generated is written to the SAS Log and the datetime variable TRTEDTM will contain a missing value.

```
/*** Incorrect Imputation. Assumes missing time component as '00:00' ***/
TRTEDTM = (TRTEDT*24*60*60) + TRTETM;
```

Code Sample 1. Incorrect Usage

In the corrected line of code shown below in Code Sample 2, we are specifically checking for non-missing date TRTEDT and time TRTETM components and handling it accordingly. If date part or time part is missing for any record, then note that the datetime variable TRTEDTM will also contain a missing value. The SAS Note regarding Missing values will not appear in the SAS log as it is handled by the programming code.

```
/** Correct Usage to create date time variable. Checks specifically for missing date
and time parts ***/
If TRTEDT ne . and TRTETM ne . then TRTEDTM = (TRTEDT*24*60*60) + TRTETM;
```

Code Sample 2. Correct Usage

METHOD #2: TO POPULATE DATETIME VALUE USING SAS FUNCTION

In this method, the 'DHMS' SAS function is used to populate datetime values from numeric date and time variables.

In Code Sample 3 below, note that for missing date (ADT) or time (ATM) component, a SAS Note that Missing values were generated as a result of performing an operation on missing values is written to the SAS Log and the datetime variable ADTM will contain a missing value.

```
/*** Alternative Method to create a datetime variable. This code will create a missing
datetime value if the Date or Time value is missing. ***/
ADTM=DHMS(adt, HOUR(atm), MINUTE(atm), SECOND(atm));
```

Code Sample 3. One option of DHMS

In the Code Sample 4 below, we are specifically checking for missing date ADT and time ATM components. The SAS Note that Missing Values were generated will not be written to the SAS Log. The datetime ADTM variable will contain a missing value if either date or time part has a missing value.

```
/*** This code is specifically checking for non-missing date and time values before
creating the date time values.  ***/
If ADT ne . and ATM ne . then ADTM=DHMS(adt, HOUR(atm), MINUTE(atm), SECOND(atm));
```

Code Sample 4. DHMS option avoids issues in the log

Now that we know how to create datetime values, we can look at some scenarios where incorrect usage of datetime, date and time variables can result in issues and the corresponding corrected code is also provided.

SCENARIO #1

In many studies where the Infusion times are relatively short, the CRF usually only collects the Infusion Start Date (EXSTDTN), Infusion Start Time (IVSTM) and Infusion End Time (IVETM). Note that the Infusion End Date (EXENDTN) is not collected on the CRF as the sites are generally expected to complete the infusion on the same date. In such studies, you need to take into account situations where the infusion was past midnight and therefore the end dates are now one day after the value in the start date variable.

In this scenario #1, we need to pick the last Infusion Date and Time for a subject.
For this example, the records in the CRF when the last two infusions are given on the same date for a subject are shown below in Table 1:

USUBJID	EXSTDTN	IVSTM	IVETM
ABC-101-1001	10MAR2014	09:20	10:20
ABC-101-1001	10MAR2014	23:05	00:05

Table 1. Sample Source Data for a Subject

The date and time variables are all of type numeric in this scenario.
Assuming that the last infusions were given on 10MAR2014, we can tell from the Table 1. above that the last infusion for this subject was given at 23:05 on 10MAR13. Therefore the last infusion date and time for this subject is 11MAR14 (EXENDTN) 00:05 (IVETM)
If the start time IVSTM is ignored in the calculation to pick the last record, it could lead to invalid results.

In the Code Sample 5 below, Start Time is ignored in the sorting. This will place the IVETM=10:20 to be the last record and will be the record that is picked as last infusion date and time since IVETM=10:20 is greater than IVETM=00:05 in terms of seconds stored in IVETM variable.
In this incorrect logic, the end date is selected as '10MAR2014' and the end time is selected as '10:20' which is the invalid value for the last dose date and time.

```
/*** Incorrect code to calculate Last Infusion Date and Time ***/
/*** Sorting the dataset to pick the last record. Start Time IVSTM is ignored in
sorting to pick the latest record.  ***/
proc sort data=exdts;
   by usubjid exstdtn ivetm;
run;

/*** Incorrect record is picked since Start Time IVSTM is not being considered.
  The record with IVETM=10:20 is picked as this time value is larger than IVETM=00:05
in terms of seconds stored in IVETM variable ***/
data exendt;
  set exdts;
  by usubjid exstdtn ivetm;

  if last.usubjid then do;
    output;
  end;
run;
```

Code Sample 5. Incorrect Usage

The corrected code to pick the End Date and Time is given below in Code Sample 6. The start time IVSTM is also included in order to sort the records in the correct chronological order to pick the latest record. Note we need to include another data step in the end to ensure that the end date is incremented to the next day, if the infusion end time passes midnight. In Code Sample 6, the end date (EXENDTN) is selected as '11MAR2014' and the end time (IVETM) is selected as '00:05'.

```
/*** Corrected code to calculate Last Infusion Date and Time ***/
/*** IVSTM is included in the Sorting to pick the last record ***/
proc sort data=exdts;
   by usubjid exstdtn ivstm ivetm;
run;

/*Correct record is picked since Start Time is being considered. */
data exendt;
  set exdts;
  by usubjid exstdtn ivstm ivetm;

  if last.usubjid then do;
    output;
  end;
run;

/* This data step is required to increment the EXSTDTN by 1 day if the Infusion time
passes midnight. */
data exendt(keep=USUBJID EXENDTN IVETM);
  set exendt;

  length EXENDTN 8;
  if ivetm ne . and ivstm>ivetm then do;
    if exstdtn ne . then exendtn = exendtn+1;
  end; else exendtn=exstdtn;
run;
```

Code Sample 6. Correct Usage

SCENARIO #2

For subjects that discontinued early, the specification states to flag a record as AVISIT='End of Termination' (EOT) if the date of the particular record was between last study drug dose datetime and 48 hours (2 days) from last dose datetime (TRTEDTM<=ADTM<=TRTEDTM+48hours).
The specification states that if any time component is missing then use the non-missing date components in the calculation.
This study collects both date and time values in the CRF for this domain as well as in the Exposure domain where last study drug datetime is calculated.

We will assign AVISIT as per the algorithm specified in Scenario # 2 making use of the datetime variables created as shown in Method #1 and 2 above.

Let us assume the values of the variables as shown in Table 2.

USUBJID	TRTEDT	TRTETM	TRTEDTM	ADT	ATM	ADTM
ABC-101-1001	10JAN14	20:00	10JAN14:20:00	12JAN14	.	.

Table 2. Sample Source Data for a Subject

As per the algorithm the date parts should still be used to calculate AVISIT when time parts are missing. In this example, this record qualifies to be flagged as AVISIT=End of Treatment if we consider only date parts as time part is missing. Here, (TRTEDT<=ADT<=TRTEDT+2) condition is satisfied. However, in Code Sample 7, the programmer may not be aware that the datetime values are also missing when time parts are missing. Therefore, the date parts are not considered and hence AVISIT is not assigned as End of Treatment for this record.

```
 /*** Only non-missing datetime records are considered. If only time parts are
missing, then this code will not consider non-missing date parts.
(2*24*60*60) is to add 48 hours (in seconds) ***/
if adtm ne . and trtedtm ne . then do;
  if trtedtm <= adtm <= trtedtm + (2*24*60*60) then avisit='End of Treatment';
end;
```

Code Sample 7. Incorrect Usage

Let us now assume the values of the variables as follows.

USUBJID	TRTEDT	TRTETM	TRTEDTM	ADT	ATM	ADTM
ABC-101-1001	10JAN14	20:00	10JAN14:20:00	12JAN14	22:00	12JAN14:22:00

Table 3. Sample Source Data for a Subject

In Code Sample 8, AVISIT is incorrectly assigned as End of Treatment for records that do not satisfy the criteria.
If we have a non-missing datetime TRTEDTM and non-missing datetime ADTM but the datetime difference is not within 48 hours, then the first condition is not satisfied. However, due to the logic sequence placed in the code below, the condition in the ELSE statement is also checked instead of exiting at that point. AVISIT is incorrectly populated if the condition involving only date parts is satisfied.
Here the condition ADTM (12JAN14:20:00) <= TRTEDTM+48 hours (12JAN14:10:00) is not satisfied and therefore this record should not be flagged as AVISIT=End of Treatment. Due to incorrect logic below in Code Sample 8, when the ELSE condition for only DATE parts is checked; ADT (10JAN14) <= TRTEDT+2 (12JAN14) condition is satisfied. This causes the record to be incorrectly flagged as AVISIT=End of Treatment although it does not satisfy the condition in the programming specifications.

```
/*** Incorrect Logic. The Datetime parts are checked in the IF condition. The DATE
parts are checked in the ELSE condition but they should be checked only when the time
components are missing ***/
if adtm ne . and trtedtm ne . and trtedtm <= adtm <= trtedtm + (2*24*60*60) then
           avisit='End of Treatment';
else if adt ne . and trtedt ne . and trtedt <= adt <= trtedt + 2 then
           avisit='End of Treatment';
```

Code Sample 8. Incorrect Usage

The following lines of code in Code Sample 9 fix both the issues above. This is the correct usage to check for the condition of TRTEDTM+48 hours (2 days).
The first IF condition checks if both Datetime values are present. Only if either of the datetime values are missing, the ELSE condition to consider non-missing DATE parts is checked.
If both datetime variables are non-missing but do not satisfy the first condition of 'TRTEDTM <= ADTM <= TRTEDTM + (2*24*60*60)' then AVISIT is not assigned End of Treatment and the code for non-missing date parts is skipped.

```
/*** Correct Usage for Scenario #2. This resolves both issues in Code Sample 7 and 8.
The sequence of logic is accurate as per the specification ***/
if adtm ne . and trtedtm ne . then do;
  if trtedtm <= adtm <= trtedtm + (2*24*60*60) then avisit='End of Treatment';
end; else if adt ne . and trtedt ne . then do;
  if trtedt <= adt <= trtedt + 2 then avisit='End of Treatment';
end;
```

Code Sample 9. Correct Usage

CONCLUSION

It can sometimes be frustrating when we are working with Date, Time and Datetime variables if we do not fully understand how these variables work in SAS. This paper attempts to increase the knowledge of readers using these variables by giving examples where wrong assumptions or lack of understanding of these variables can lead to invalid output.

As programmers we can successfully tame the Date, Time and Datetime variables if we are mindful of utilizing the correct logic and understand the intricacies of sorting and missing values for these variables.

REFERENCES

SAS® 9.2 Documentation on SAS Support website. Available at http://support.sas.com/documentation/onlinedoc/bookshelf/92/.

ACKNOWLEDGMENTS

The author would like to thank Ashwini Bapat and Jhelum Naik for their encouragement and valuable suggestions. The author would also like to thank PPD colleagues for their support.

DISCLAIMER

The content of this paper are the works of the author and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

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

Sajeet Pavate
PPD
929 North Front Street
Wilmington, NC 28401
Work Phone: +1 910 558 8622
E-mail: sajeet.pavate@ppdi.com