PharmaSUG 2016 - Paper IB05

Compilation of Errors, Warnings and Notes!

Anusuiya Ghanghas Novartis Healthcare Private Limited, Hyderabad, India
Houde Zhang, Novartis Pharmaceuticals, East Hanover, NJ, United States of America
Rajinder Kumar, Novartis Healthcare Private Limited, Hyderabad, India

It isn't making mistakes that's critical; it's correcting them and getting on with the principal task. Donald Rumsfeld

ABSTRACT

Since SAS® has been leading as a promising tool in clinical industry, people become more fancy and positive to understand background play of how SAS works, then errors, warnings and notes are no more to see like technical red flags or loopholes of system, rather they became teachers to beginners or intermediate programmers. It has been proved in many cases that knowing your log helps you to improve basic understanding of SAS than your typical way of learning any tool, it is palpable that most expert SAS programmers admitted that remedial learning is best way of gaining expertise to improve further to write optimized programming, so this paper tries to touch sensible errors, warnings and notes that are really worth of knowing to make your work technically beautiful. Errors, warnings and notes are part and parcel of programming. But for someone who is new to programming these unwanted messages in log can prove a big headache. Sometimes it is seen that more efforts are required in clearing these unwanted messages (error, warning and note) from log than original time required in development of entire code. This paper puts light on some of the most common errors/warnings/notes in log and provides solution for them along with possible reason behind them. When someone who has good experience provides solutions for these messages then programmer thinks, ohh! It's so simple. Obviously experience can't be gained overnight. But for sure examples shown in this paper will be helpful for all the beginner and intermediate programmers for their most of the queries related to unwanted issues faced due to these messages in log.

INTRODUCTION

As Alan Perlis rightly said "There are two ways to write error-free programs; only the third one works". Programmers share wonderful relationship with these unwanted messages in log. However messages in log, provide timely wake-up call to programmers from making silly mistakes. As programming experience grows, confidence level and friendliness with these messages also gets increased. This paper puts light on some basic errors, warnings and notes in log and also tries to provide possible reason or solution.

1.0 TYPES OF MESSAGES

Before jumping to all these lets first try to know what an error, warning and note is. An error in a SAS program is anything which is unexpected by SAS system and prevents from providing right and desired result. Error messages are in most of the cases highly critical and they stop SAS from executing and generate message in red text in log window. Warning is alert bell from SAS to notify what expected is not provided but it still execute the program while generating green texts wherever data/statements/options are not filled in. Notes are descriptor information generated by SAS but it still leaves a choice to SAS user either to edit the current code or keep as it is.

These messages can be divided into two groups as below:

- Syntax/Compile time
- Data/Execution time

Whenever we submit a SAS program then first it is compiled, where compiler checks for syntax correctness. During compilation all the keywords, functions, variable names and their type, initiation etc are checked for correctness. Here we get a message in the form of error/warning/note in log if any typo is there or some syntax errors are present, for example spelling mistake or missing semicolon or referencing of a variable prior to creating it etc. Let's visit all these one by one.

2.0 SYNTAX/COMPILE TIME ERROR

Syntax errors occur when program statements do not conform to the rules of the SAS language. Whenever a SAS program is submitted, it is first get compiled. During compilation, compiler check for syntax and other SAS defined rules. If any non-compliance to syntax or SAS defined rules is found then it provides an error message in SAS Log, which is called compile time error. Here are some examples of syntax errors.

2.1 MISSING SEMICOLON

All SAS programmers learn this as first thing that SAS program is made of SAS statements and each SAS statement should end with semicolon (;). However this semicolon is missed quite frequently by most of the programmers. And the funny thing in this case is that whatever message SAS provides in the LOG, doesn't directly point to absence of semicolon.

Let's have a look at a below example.

```
DATA TEST
SET PRETEST;
LENGTH VAR1 $1. ;
VAR1='Y';
RUN;
```

When this program is submitted for execution then it gives below message in log.

```
NOTE: The data set WORK.TEST has 1 observations and 1 variables.
NOTE: The data set WORK.SET has 1 observations and 1 variables.
NOTE: The data set WORK.PRETEST has 1 observations and 1 variables.
```

If that was not the motive to create 3 datasets, then for avoiding this we can use SAS option DATASTMTCHK=ALLKEYWORDS; which helps in avoiding use of SAS keywords as dataset or variable name. As per SAS rules keywords are reserved and can't be used as dataset and variable name. This option helps in understanding the error in better way.

```
OPTIONS DATASTMTCHK=ALLKEYWORDS;
DATA TEST
SET PRETEST;
LENGTH VAR1 $1. ;
VAR1='Y';
RUN;
```

When above program is submitted then it provides below message in log to inform programmer that SET is used as a dataset name:

```
1753   SET PRETEST;
       ---
       57

ERROR 57-185: SET is not allowed in the DATA statement when option
DATASTMTCHK=ALLKEYWORDS.  Check for a missing semicolon in the DATA statement, or use
DATASTMTCHK=NONE.
```

It can be seen with DATASTMTCHK option error message in the log, points directly to missing semicolon. With slight change in above program with respect to missing semicolon location if it is submitted as below then lets see, how it impacts error message:

```
DATA TEST;
SET PRETEST
LENGTH VAR1 $1. ;
VAR1='Y';
RUN;
```

We get below message in log:

```
1759   LENGTH VAR1 $1. ;
                  ---
                  22
                  200
ERROR: File WORK.LENGTH.DATA does not exist.
ERROR: File WORK.VAR1.DATA does not exist.

ERROR 22-322: Syntax error, expecting one of the following: a name, a quoted string,
(, -, :, ;, CUROBS, END, INDSNAME, KEY, KEYRESET, KEYS, NOBS, OPEN, POINT, _DATA_,
_LAST_, _NULL_.
ERROR 200-322: The symbol is not recognized and will be ignored.
NOTE: The SAS System stopped processing this step because of errors.
```

```
WARNING: The data set WORK.TEST may be incomplete.  When this step was stopped there
were 0 observations and 1 variables.
```

Here error message is more of general, which starts with non-existence of datasets LENGTH and VAR1, and then goes on to provide the list of options which it was expecting.

Other way of avoiding this is to follow proper indentations and making sure each line has only one statement, which ends with a semicolon. Because having multiple statements on same line can make things quite messy and a semicolon can be missed easily, while if proper indentation is applied then its readability gets increased and chances of making minor mistakes like missing semicolon becomes very less. Below piece of code provides a good example of proper indentation.

```
DATA Test;
   SET WORK.PRETEST;
   Length VAR1 $1;
   If VAR>=3 then do;
      If VAR > 5 then VAR1='H';
      Else VAR1='N';
   End;
   Else if VAR ne . then VAR1 = 'L';
RUN;
```

Here proper indentation helps in understanding the flow of program and makes it easier to understand at which point a statement ends.

2.2 TYPO

Most of the coding part is done by human being by simply typing with the help of keyboard, hence chances of some misspelling or typo can't be denied. If it is a SAS keyword which is misspelled then SAS tries to guess it and replace it, but if it is a variable or dataset name then SAS expects that variable or dataset already in place. If it is variable name which is misspelled as another existing variable then it can lead to incorrect result and logical error, if misspelled variable doesn't match to any existing variable then SAS gives related message in the log.

Let's have a look at below examples for more clarity:

```
DATE test;
Var = 1;
Run;
```

When above program is submitted then SAS executes this with below warning message.

```
WARNING : Assuming the symbol DATA was misspelled as date.
```

However when same program having misspelled variable or dataset name as in below example can cause different result:

```
DATA Test;
Set work.tset;
VAR=1;
Run;
```

On submission it provides below error in log:

```
ERROR: File WORK.TSET.DATA does not exist.

NOTE: The SAS System stopped processing this step because of errors.

WARNING: The data set WORK.TEST may be incomplete.  When this step was stopped there
were 0 observations and 1 variables.

WARNING: Data set WORK.TEST was not replaced because this step was stopped.
```

Here input dataset name is misspelled to TSET from TEST, which in WORK library doesn't exist. As a result it stops execution. If by any chance WORK library has TSET dataset then it reads the dataset and generates TEST dataset. This would not be correct one and can lead to further errors, such as non-existence of desired variable in this dataset.

Let's have a look at similar examples:

```
DATA Test;
Set wrok.test;
Run;
```

On submission this provide below error in log:

```
ERROR: Libref WROK is not assigned.

NOTE: The SAS System stopped processing this step because of errors.
```

Due to typo here library name WORK becomes WROK, SAS tries to search for LIBRARY reference to WROK, which was not assigned as a result it stops execution. If by any chance library WROK is assigned then SAS tries to find TEST dataset in the library. This can provide incorrect result due to wrong reference to the library. This is a semantic error, where syntax wise it is correct but this library is not defined prior hence it is incorrect. Means, elements in SAS statements are correct but are not valid for usage.

```
DATA Test;
Set WORK.TEST;
Var1 = Vra+1;
Run;

NOTE: Variable Vra is uninitialized.

NOTE: Missing values were generated as a result of performing an operation on missing
values. Each place is given by: (Number of times) at (Line):(Column).  1 at 1788:11
```

Here variable VAR is misspelled as VRA, as variable VRA was not created earlier in dataset WORK.TEST, hence SAS gives above message in the LOG and creates numeric variable VRA as missing, and as a result it returns variable VAR1 as missing.

Below are some of the reasons for variable uninitialized message in the log.

- A misspelled variable name.
- Using a variable that has been dropped.
- Using the wrong data set.
- Using a variable before creating it.

Same as variable is uninitialized, for macro variable if it is referenced prior to creating it then it gives below message in the LOG.

```
WARNING: Apparent symbolic reference VRA not resolved.
```

3.0 DATA/EXECUTION ERROR:

Once a program is compiled then submitted for execution. Whatever error comes during execution time of the program are called execution time error. Mostly execution errors are data related. Let's have a look at some of common execution errors.

- Missing value generated.
- Repeats of BY value.
- Numeric value converted to character and vice versa.
- Divide by zero.

3.1 MISSING VALUE GENERATED

This message is generated as a result of performing operation on missing value. Some programmers and organizations allow it to be ignored. But even you are ignoring this message it's good to know reason behind it. Below two examples provides more insight about this.

```
DATA A;
       VAR1=1;
       VAR2=2;
       VAR3=5;
       VAR4=.;
       TEST1=VAR1+VAR2+VAR3+VAR4;
       TEST2=SUM(VAR1, VAR2, VAR3, VAR4);
RUN;
```

Here in LOG programmer will get a message that missing values are generated as a result of performing an operation on missing value. Here value of variable TEST1 is missing, if that is what expected then its fine otherwise if sum of non-missing value was excepted then sum function should be used as is the case with variable TEST2.

Before ignoring this in LOG it's better to know its reason and consequences. Apart from using function other way of avoiding this message in LOG is to check for non-missingness of variables before passing to any operations.

3.2 REPEATS OF BY VALUE

This message is more of logical error then data error. It appears in the log when more than one dataset has multiple records at by value in a merge. Let's see what happens when programmer want to merge below two tables FIRST and LAST:

var	var2	var3
1	1	1
1	2	1
2	1	1

Table 1: FIRST

var	var2	var4
1	1	1
1	2	2
2	1	1

Table 2: LAST

```
DATA final;
merge first(in=a) last(in=b);
by var;
if b;
RUN;
```

When above program is submitted then it provides below message in log:

```
NOTE: MERGE statement has more than one data set with repeats of BY values.
```

Solution: In this case programmer need to check the logic again and if needed can add one or two more variables in by variable statement (in above case adding variable var2 to by variable would help) or if it is known multiple records in the dataset are excepted but unique records at by variable level should be sufficient then duplicate records can be dropped from the particular dataset (here unique records from dataset FIRST at VAR variable level should be sufficient).

3.3 NUMERIC VALUE CONVERTED TO CHARACTER AND VICE VERSA

When a character variable is passed in place of numeric to a SAS function then SAS tries to convert that variable in numeric and uses it as a numeric variable. This happen when programmer mixes both types of variables, then SAS convert data from one type to another and allows program to execute and print the message that value have been converted to the LOG.

```
DATA CONVERSION;
       LENGTH VAR1 VAR2 VAR3 $8;
       VAR1=5;
       VAR2=3; VAR3='3';
       VAR4=VAR1+VAR2+VAR3;
RUN;
NOTE: Character values have been converted to numeric values at the places given by:
(Line):(Column). 1892:20
```

It's good that SAS provides this facility to convert it to numeric variable. But that doesn't mean one should ignore it. It is always better to handle this at programming level so that it doesn't create further problem. Being free from conversion warning your program would run smoothly and will take less time in the execution. For overcoming this better to use PUT or INPUT function for converting numeric to character and character to numeric.

For converting from character to numeric:

```
New_var = input(old_var,informat.);
```

For converting from numeric to character:

```
New_var = put(old_var,format.);
```

3.4 INFINITE LOOP AND DIVIDED BY ZERO

This also is more of logical error. Sometimes programmer wonders, syntax and logic wise its program is fine and even when tried on a small set of data, it works perfectly. But when full data is passed, it keeps on running and takes too much time for execution. In most cases the execution is aborted. The possible reason can be program being in infinite loop or divided by zero.

Infinite loop example:

```
J=2; i = 10;
DO WHILE (i>5);
i=i-2;
i=i+j;
output;
END;
```

Solution: Before performing such operation it's better to check for such thing which helps in avoiding this kind of scenario. Here as the condition in DO WHILE loop would always be true. Hence this loop keeps on executing endlessly. This is a logical error. This should be corrected before submitting the program for execution. May be value of J should be less than 2.

Divide by Zero: At first instance below code looks fine but when submitted it gives "`Division by zero detected`" message in log:

```
J=2; i = 12;
DO WHILE (i>j);
i=i-j-1;
k = j/i;
output;
END;
```

In such cases it is always better to check denominator not being zero, as shown below:

```
if i ne 0 then k = j/i;
```

Which would ensure it is not resulting in divide by zero.

4.0 QUICK TIPS

Never say, "oops." Always say, "Ah, interesting." As seen above it is always a good practice to have a clean LOG. Below are some tips which could help in creating a good program.

- Write program in a neat, clean and organized way with proper indentation.
- Use comments where ever required.
- Develop program piece by piece in place of writing entire program in one go.
- User created error and warning messages in LOG can be helpful.
- Adding message in the LOG (with PUT statement) can help in getting attention of the programmer.

SAS follows top to bottom, left to right approach for reading, compiling and executing the program. Same should be followed for resolving error, warning and notes from the LOG. As initial messages can be reason for later ones also.

CONCLUSION

Having been gone through this paper, we definitely have some basic things to take home away.

- Being a beginner or intermediate programmer, it explains how important to learning out of mistakes via Log, other way called remedial learning.
- It helps how SAS works in background and how those messages could be understood to correct your mistakes.
- There may not be few things much dangerous as we discussed may not impact results but leaves you in a state of satisfaction being technical beauty.

REFERENCES

- Delwiche, Lora D. and Susan J. Slaughter. *Errors, Errors, Warnings, and Notes (Oh My) A Practical Guide to Debugging SAS® Programs*, Davis, CA.
- Roger Staum, *To Err is Human; to Debug, Divine,* SAS Institute, New York, NY.
- SAS Institute Inc. (2009). *SAS onlineDOC®, Version 9.3,* Chapter 8, Error Processing and Debugging; pp 105-123, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The authors take this opportunity to thank Muralikrishna Chakravarthula manager-lead statistical analyst at Novartis Healthcare Private Limited and Sudarshan Reddy, Senior Statistical Programmer at inVentiv Health Clinical, whose support and guidance encouraged us to write this paper.

DISCLAIMER

The contents of this paper are the work of the authors and do not necessarily represent the opinions, recommendations or practices of Novartis Healthcare Private Limited.

CONTACT INFORMATION

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

Name: Anusuiya Ghanghas
Enterprise: Novartis Healthcare Private Limited
City, Country ZIP: Hyderabad, India -500032
Phone: +91 - 8421484652
E-mail: this.anusuiya@gmail.com

Name: Houde Zhang
Enterprise: Novartis Pharmaceuticals
City, Country ZIP: East Hanover, NJ, USA
E-mail: onmyway2007@yahoo.com

Name: Rajinder Kumar
Enterprise: Novartis Healthcare Private Limited
City, Country ZIP: Hyderabad, India -500032
Phone: +91 - 9545988828
E-mail: Rajinder_sihag@yahoo.co.in