Paper 9541-2016

Cleaning up your SAS® log: Note Messages

Jennifer Srivastava, Quintiles Transnational Corporation, Durham, NC

ABSTRACT

As a SAS programmer, you probably spend some of your time reading and possibly creating specifications. Your job also includes writing and testing SAS code to produce the final product, whether it is SDTM datasets, ADaM datasets or statistical outputs such as tables, listings or figures. You reach the point where you have completed the initial programming, removed all obvious errors and warnings from your SAS log and checked your outputs for accuracy. You are almost done with your programming task, but one important step remains.

It is considered best practice to check your SAS log for any questionable messages generated by the SAS system. In addition to messages that begin with the words WARNING or ERROR, there are also messages that begin with the words NOTE or INFO. This paper will focus on five different types of NOTE messages that commonly appear in the SAS log and will present ways to remove these messages from your log.

INTRODUCTION

The SAS log is a record of what happens when you run your SAS program and is an essential tool for debugging code. The log includes program statements, as well as messages generated by SAS. These messages can begin with the words WARNING, ERROR, NOTE or INFO. This paper will explore in detail some of the reasons that cause various NOTE messages to appear in the SAS log, and will give suggestions on how to remove these messages from the log, if this is desired.

By default, note messages will be printed in the SAS log. In order to have all log messages starting with NOTE: suppressed, use the following option:

```
options nonotes;
```

To return to the default use the following option:

```
options notes;
```

NOTE MESSAGES

Some notes are very informative, such as those that tell which version of SAS is being used or those that indicate the dataset name, the number of observations and the number of variables within the dataset. Examples of these and a few other useful notes are listed below:

```
NOTE: SAS (r) Proprietary Software 9.4 (TS1M1)

NOTE: The data set WORK.LAB_NEW has 10 observations and 8
      variables.

NOTE: Libref IN1 was successfully assigned as follows:
      Engine:        V9
      Physical Name: S:\SAS\PAPER_2016\TESTDATA


NOTE: Compressing data set WORK.LB0 decreased size by 63.34
      percent.
```

```
      Compressed is 7626 pages; un-compressed would require
      20800 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           6.99 seconds
      cpu time            2.66 seconds

NOTE: 0 observations with duplicate key values were
      deleted.
```

Other notes may be indicative of some problem within your code. Depending on your company's standards, the best practice may be to avoid having certain notes appear in your SAS log. Below are examples of five different types of note messages that you probably want to avoid and some suggestions on how to deal with them.

EXAMPLE 1 : MERGE STATEMENT HAS MORE THAN ONE DATA SET WITH REPEATS OF BY VALUES

This note should never show up in your SAS log and if it does, you may get unexpected and undesirable results. The code below shows a case where the note is being generated because the datasets are at the SUBJECT and VISIT level, instead of just the SUBJECT level. Once VISIT is added to the merge, the note in question goes away.

```
246       data dog;
247        merge dog
248              new_data;
249        by subjid;
250       run;

INFO: The variable VISIT on data set WORK.DOG will be overwritten by data set WORK.NEW_DATA.
NOTE: MERGE statement has more than one data set with repeats of BY values.
NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: There were 90 observations read from the data set WORK.NEW_DATA.
NOTE: The data set WORK.DOG has 90 observations and 10 variables.
NOTE: Compressing data set WORK.DOG decreased size by 0.00 percent.
      Compressed is 3 pages; un-compressed would require 3 pages.
```

```
256         data dog;
257          merge dog
258                new_data;
259          by subjid visit;
260         run;

NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: There were 90 observations read from the data set WORK.NEW_DATA.
NOTE: The data set WORK.DOG has 90 observations and 10 variables.
NOTE: Compressing data set WORK.DOG decreased size by 0.00 percent.
      Compressed is 3 pages; un-compressed would require 3 pages.
NOTE: DATA statement used (Total process time):
      real time             0.00 seconds
      cpu time              0.00 seconds
```

EXAMPLE 2 : NUMERIC CONVERTED TO CHARACTER OR CHARACTER CONVERTED TO NUMERIC

Below is some code showing notes referring to character values being converted to numeric or numeric values being converted to character. This type of note can be removed by using the correct variable type.

```
211        data dog;
212         length blflag $1;
213         set dog;
214         if subjid='30' then weight=20;
215         if visit=1 then blflag=1;
216        run;

NOTE: Character values have been converted to numeric values at the places given by: (Line):(Column).
      214:12   215:5
NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
      215:25
NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.DOG has 90 observations and 9 variables.
```

Taking a look at the dataset contents, it becomes obvious that SUBJID is a numeric variable and that VISIT is a character variable.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
9	BIRTHDT	Num	8	DATE9.	DATE9.	BIRTHDT
1	BLFLAG	Char	1			
4	BREED	Char	20	$20.	$20.	Breed
6	GENDER	Char	1	$1.	$1.	Gender
3	NAME	Char	9	$9.	$9.	Name
2	SUBJID	Num	8			Subject ID
8	VISIT	Char	1			Visit number
5	VISITDT	Num	8	DATE9.	DATE9.	Visit date
7	WEIGHT	Num	8			Weight

Once the code is adjusted accordingly, the notes in question no longer appear in the log as shown below.

```
221         data dog;
222          length blflag $1;
223          set dog;
224          if subjid=30 then weight=20;
225          if visit='1' then blflag='1';
226         run;

NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.DOG has 90 observations and 9 variables.
```

EXAMPLE 3: VARIABLE IS UNINITIALIZED

When you see a note in your log referring to an uninitialized variable, this means that the variable does not exist. Often times this is due to a mistake in the variable name that can be easily rectified. If the variable is not created yet, because the spec is not final or because of pending raw data, you might consider leaving the uninitialized message in the production program log, as a reminder that the work is not complete yet. Below is an example of this. The program specifies VISITS, but the variable name is VISIT.

```
294         * Get the baseline weight from visit 1 *;
295         data baseline(rename=(weight=wt_bl));
296          set dog;
297          if visits='1';
298         run;

NOTE: Variable visits is uninitialized.
NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.BASELINE has 0 observations and 12 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


299
300
301         * Get the baseline weight from visit 1 *;
302         data baseline(rename=(weight=wt_bl));
303          set dog;
304          if visit='1';
305         run;

NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.BASELINE has 30 observations and 11 variables.
NOTE: Compressing data set WORK.BASELINE decreased size by 0.00 percent.
      Compressed is 2 pages; un-compressed would require 2 pages.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

EXAMPLE 4: MISSING VALUES WERE GENERATED AS A RESULT OF PERFORMING AN OPERATION ON MISSING VALUES

One note that can usually be easily avoided is the note that indicates that missing values are generated as a result of performing an operation on missing values. Below is code where the message appears in the log because there are a few missing birthdates in the dataset and therefore age is missing for those records.

```
246         data age;
247          set dog;
248          tday=today();
249          age=round((tday-birthdt)/365,0.1);
250         run;

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).
      3 at 249:6    3 at 249:17    3 at 249:26
NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.AGE has 90 observations and 11 variables.
```

Once records with missing birthdates are excluded from the calculations, the note no longer shows up in the log. Age will still be missing for those records. The dataset does not change, but the log is cleaner.

```
252         data age;
253          set dog;
254          tday=today();
255          if not missing(birthdt) then age=round((tday-birthdt)/365,0.1);
256         run;

NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.AGE has 90 observations and 11 variables.
NOTE: Compressing data set WORK.AGE decreased size by 0.00 percent.
      Compressed is 2 pages; un-compressed would require 2 pages.
```

EXAMPLE 5: INVALID DATA VALUES

Below is some code showing invalid date values. Once records with missing birthdates are excluded from the calculations, the note no longer shows up in the log. Age will still be missing for those records. The dataset does not change, but the log is cleaner.

```
331        data birthdt;
332         set dog;
333         birthdtn=input(birthdtc,date9.);
334        run;


NOTE: Invalid argument to function INPUT at line 333 column 11.
blflag=1 SUBJID=1 VISITDT=01APR2011 WEIGHT=20 VISIT=1 NAME=Wishbone BREED=Jack Russell Terrior BIRTHDT=04FEB2000 GENDER=M size=8 new_weight=19.8
birthdtc=UNUNK2000 birthdtn=. _ERROR_=1 _N_=1
NOTE: Invalid argument to function INPUT at line 333 column 11.
blflag=  SUBJID=1 VISITDT=01JUL2011 WEIGHT=19 VISIT=2 NAME=Wishbone BREED=Jack Russell Terrior BIRTHDT=04FEB2000 GENDER=M size=8 new_weight=18.8
birthdtc=UNUNK2000 birthdtn=. _ERROR_=1 _N_=2
NOTE: Invalid argument to function INPUT at line 333 column 11.
blflag=  SUBJID=1 VISITDT=01JUL2011 WEIGHT=19 VISIT=3 NAME=Wishbone BREED=Jack Russell Terrior BIRTHDT=04FEB2000 GENDER=M size=8 new_weight=18.8
birthdtc=UNUNK2000 birthdtn=. _ERROR_=1 _N_=3
NOTE: Mathematical operations could not be performed at the following places. The results of the operations have been set to missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      3 at 333:11
NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.BIRTHDT has 90 observations and 13 variables.
NOTE: Compressing data set WORK.BIRTHDT increased size by 50.00 percent.
      Compressed is 3 pages; un-compressed would require 2 pages.
```

```
343        data birthdt;
344         set dog;
345
346         /* Impute missing day and month values based on imputation rules defined for the study */
347         if not missing(birthdtc) then do;
348
349          /* if day is missing, make day=01 */
350          if substr(birthdtc,1,2)='UN'  then birthdtc='01' || strip(substr(birthdtc,3,7));
351          /* if month is missing, make month=JAN */
352          if substr(birthdtc,3,3)='UNK' then birthdtc=strip(substr(birthdtc,1,2)) || 'JAN' || strip(substr(birthdtc,6,4));
353
354          format birthdtn date9.;
355          birthdtn=input(birthdtc,date9.);
356         end;
357        run;

NOTE: There were 90 observations read from the data set WORK.DOG.
NOTE: The data set WORK.BIRTHDT has 90 observations and 13 variables.
NOTE: Compressing data set WORK.BIRTHDT increased size by 50.00 percent.
      Compressed is 3 pages; un-compressed would require 2 pages.
```

CONCLUSION

Hopefully the information provided in this paper will give you some insight into why certain Note messages are appearing in your SAS log and also give you some ideas how to remove them. Having a SAS log that is clean and free of extra messages will help the programmer produce a higher quality final product, whether it is an SDTM dataset, ADaM dataset or a statistical output such as a table, listing or figure.

CONTACT INFORMATION

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

Jennifer Srivastava
Quintiles Transnational Corporation
919-749-8567
Jennifer.srivastava@quintiles.com