PhUSE 2008

Paper TU08

... but there are no ERROR messages in the LOGs

Shafi Chowdhury, Shafi Consultancy Limited, London, U.K.

ABSTRACT

SAS® is such a versatile tool that it can allow users to do pretty much what they want. The only problem with this flexibility is that users can make mistakes and not realize that there is an error in their code. Many people just search for the word ERROR before assuming that all is well. How often do we see “uninitialized”, “repeats of BY” or “W.D format was too small” in the log and ignore them? This paper will look at some of the mistakes people make in their everyday programming which do not result in an ERROR message in the LOG, but may result in incorrect numbers. This paper will also show how we can identify these hidden errors and resolve them.

INTRODUCTION

SAS® is used across the world in many industries for many different tasks. It is used such widely because it is so versatile, it can be easily adapted for the different situations where it is used. The only problem with this versatility is that the user then has to take care that what they do makes sense for their purpose. Something can be considered acceptable in one situation and not in another; the user has to make the decision. This is all well and good provided the user is aware of what they need to look out for, and this is often the weakest point.

Often users just search their LOG files for the word ERROR, and if they are a little more vigilant then they will also search for the word WARNING. However, these are not the only indicators to suggest that there may be errors in the result. These other indicators or “silent errors” can produce incorrect results or produce correct results but highlight data issues without any ERROR or WARNING messages appearing in the LOG. Most of the programs written in the pharmaceutical industry are data driven to a certain extent. Investigators love to challenge us and they find ever more innovative ways to produce incorrect results. This means the programmers have to be on their guard to ensure bad data is identified and dealt with in a pre-specified manner.

Defensive programming is described in Good Programming Practice guides across the industry. This is where known “unwanted” messages are eliminated from the LOG using conditional programming codes and then sending user specified messages to the LOG to highlight bad data. However, there are too many different possibilities as to what might be considered an error, so some of the most common ones from the pharmaceutical industry are presented here.

UNINITIALIZED

This is a common problem. It can be the result of a typo or simply using a variable which has not been defined yet. It means a variable which has not yet been initialised in the data step is being used in a calculation. As SAS does not know the variable, a missing value is assigned to it during the calculation. This assumption of a missing value can lead to further consequences, some of which may not be intended. As a result it is always good practice to initialise variables prior to them being used in calculations. This way, at least the user will know that if a missing value was used then it was missing on purpose and not as a result of an oversight.

Program:

```
DATA un_ini;
  a=12;
  b=10;
  c=a*d;
RUN;
```

Log:

```
NOTE: Variable D 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 5:6
NOTE: The data set WORK.UN_INI has 1 observations and 4 variables.
```

Output:

Obs	A	B	C	D
1	12	10	.	.

In this example it should have been c=a*b, but a typo resulted in c=a*d. The final consequence of this is that where as it should have resulted in c=120, the result now is c=. (missing) and there is an additional variable D which also has a missing value.

REPEATS OF BY

This perhaps is not as common, but it still appears in many LOG files and the users are not aware that its effect could be a problem. When we are merging two datasets often one of them has more than one record and the other has one record per by variable combination. The result is that the dataset with one record is merged to all the records for that by variable combination in the second dataset to create a new dataset. However, if one patient has two records instead of one for some reason, then the result is very different for that patient. Unless the message in the log is recognised, this problem can go unnoticed and lead to errors in things like calculating change from baseline.

Program:

```
* Create dummy data, patient 1 has two visit 1 records *;
DATA wt_by_vis;
  DO ptno=1 TO 3;
    DO visit=1 TO 3;
      weight=60+MOD(INT((RANUNI(0)*1000)),40);
      OUTPUT;
      IF ptno=1 AND visit=1 THEN DO;
        weight=60+MOD(INT((RANUNI(0)*1000)),40);
        OUTPUT;
      END;
    END;
  END;
RUN;

TITLE3 'dataset 1 - Weight by visit';
PROC PRINT DATA=wt_by_vis;
RUN;
TITLE3;

* Keep all visit 1 records as baseline *;
DATA baseline;
  SET wt_by_vis;
  WHERE visit=1;
RUN;

* Merge baseline to the rest of the data and calculate change from baseline *;
DATA change;
  MERGE wt_by_vis baseline(KEEP=ptno weight RENAME=(weight=baseline));
  BY ptno;
  change=weight-baseline;
RUN;

TITLE3 'dataset 2 - CHANGE from baseline';
PROC PRINT DATA=change;
RUN;
TITLE3;
```

Log:

```
NOTE: The data set WORK.WT_BY_VIS has 10 observations and 3 variables.

NOTE: There were 10 observations read from the data set WORK.WT_BY_VIS.

NOTE: The data set WORK.BASELINE has 4 observations and 3 variables.

NOTE: MERGE statement has more than one data set with repeats of BY values.
NOTE: There were 10 observations read from the data set WORK.WT_BY_VIS.
NOTE: There were 4 observations read from the data set WORK.BASELINE.
```

```
NOTE: The data set WORK.CHANGE has 10 observations and 5 variables.

NOTE: There were 10 observations read from the data set WORK.CHANGE.
```

Output:

```
dataset 1 - Weight by visit

Obs    PTNO    VISIT    WEIGHT
  1       1        1        77
  2       1        1        73
  3       1        2        66
  4       1        3        88
  5       2        1        90
  6       2        2        81
  7       2        3        68
  8       3        1        97
  9       3        2        60
 10       3        3        96

dataset 2 - CHANGE from baseline

Obs    PTNO    VISIT    WEIGHT    BASELINE    CHANGE
  1       1        1        77          77         0
  2       1        1        73          73         0
  3       1        2        66          73        -7
  4       1        3        88          73        15
  5       2        1        90          90         0
  6       2        2        81          90        -9
  7       2        3        68          90       -22
  8       3        1        97          97         0
  9       3        2        60          97       -37
 10       3        3        96          97        -1
```

In this example the programmer used visit 1 as baseline. However, for some reason one patient had two visit 1 observations. So when baseline values were selected this patient ended up with two observations instead of one, and when change from baseline was calculated the results were not as expected. It can be seen that for all other patients the single baseline value is repeated in every observation and change is calculated from that single value. However, the patient with the repeated visit one value only has the second value repeated on all further observations, resulting in change being calculated from two different baseline values. It should also be noted that the order of the visit 1 values are there by chance, and so if the data was sorted differently then the value 77 would have been carried forward instead of 73.

If “repeats of BY values” appears in the LOG, one should check the datasets to see which dataset has duplicate records per BY variable, and then decide what should be done so there is only one observation per by variable. In this example, perhaps there was a data issue and one of them is a repeated measure, and so a rule should be specified that perhaps the second value based on date and time should be used as the baseline. If there is no date and time then the duplicates can still be identified so that one value can be selected manually as the baseline value. However, in most cases it should be possible to add another by variable to identify which observation was recorded first, and a programmed solution as shown below is always best.

```
DATA dups nodups;
  SET baseline;
  BY ptno visit;
  IF NOT(FIRST.ptno AND LAST.ptno) THEN OUTPUT dups;
  IF LAST.ptno THEN OUTPUT nodups;
RUN;

TITLE3 'Duplicate records - only the last observation will be used';
PROC PRINT DATA=dups;
RUN;
```

VARIABLE X WILL BE OVERWRITTEN

This is where two or more datasets are merged and they have variables in common which are not in the BY statement. It results in the values of the second dataset overwriting the values of the first dataset from common variables which are not in the BY statement. This is an error which hopefully does not go unnoticed, but it is difficult to know how often it occurs. The worrying thing here is that it is a message which perhaps is not seen as often as it should be. This is not because it does not occur often, but because the default option used by most users will not print it in the LOG. OPTIONS MSGLEVEL=N is used by most users so that all notes are printed in the LOG. However, the message in the LOG informing the user when variables are being overwritten is only printed when OPTION MSGLEVEL=I is used.

A classic example where this can happen is selecting baselines based on a visit and then merging this new dataset with the original dataset to calculate change. If the programmer forgets to remove the visit variable from the baseline dataset, the baseline visit value will replace the visit value in the original dataset after the merge if the baseline dataset is the second dataset to be specified in the MERGE statement. However, as no messages are sent to the LOG, the user will be unaware of this unless they spot it when checking the output.

Program:

```
OPTIONS MSGLEVEL=N;

DATA wt_by_vis;
  DO ptno=1 TO 3;
    DO visit=1 TO 3;
      weight=60+MOD(INT((RANUNI(0)*1000)),40);
      OUTPUT;
    END;
  END;
RUN;

TITLE3 'Weight by visit';
PROC PRINT DATA=wt_by_vis;
RUN;
TITLE3;

DATA baseline;
  SET wt_by_vis;
  WHERE visit=2;
RUN;

DATA change;
  MERGE wt_by_vis baseline(RENAME=(weight=baseline));
  BY ptno;
  change=weight-baseline;
RUN;

TITLE3 'CHANGE from baseline - VISIT is overwritten for the first observation
of each patient';
TITLE4 'but nothing is mentioned in the LOG';
PROC PRINT DATA=change;
RUN;
TITLE3;
TITLE4;
```

Log:

```
NOTE: The data set WORK.WT_BY_VIS has 9 observations and 3 variables.

NOTE: There were 9 observations read from the data set WORK.WT_BY_VIS.

NOTE: There were 3 observations read from the data set WORK.WT_BY_VIS.

NOTE: The data set WORK.BASELINE has 3 observations and 3 variables.

     DATA change;
       MERGE wt_by_vis baseline(RENAME=(weight=baseline));
       BY ptno;
       change=weight-baseline;
     RUN;

NOTE: There were 9 observations read from the data set WORK.WT_BY_VIS.
NOTE: There were 3 observations read from the data set WORK.BASELINE.
NOTE: The data set WORK.CHANGE has 9 observations and 5 variables.

NOTE: There were 9 observations read from the data set WORK.CHANGE.
```

Output:

Weight by visit

Obs	PTNO	VISIT	WEIGHT
1	1	1	79
2	1	2	76
3	1	3	65
4	2	1	66
5	2	2	60
6	2	3	90
7	3	1	79
8	3	2	92
9	3	3	75

CHANGE from baseline - VISIT is overwritten for the first observation of each patient but nothing is mentioned in the LOG

Obs	PTNO	VISIT	WEIGHT	BASELINE	CHANGE
1	1	**2**	79	76	3
2	1	**2**	76	76	0
3	1	3	65	76	-11
4	2	**2**	66	60	6
5	2	**2**	60	60	0
6	2	3	90	60	30
7	3	**2**	79	92	-13
8	3	**2**	92	92	0
9	3	3	75	92	-17

It can be seen here that Visit=1 of all patients is being overwritten by the baseline visit value from the baseline dataset. So if a summary is performed later on, the results for visit 2 will also include the values of visit 1 without the user being aware of it.

If OPTIONS MSGLEVEL=I is used then although the result will be the same, the following message will appear in the LOG.

```
OPTIONS MSGLEVEL=I;

DATA change;
  MERGE wt_by_vis baseline(RENAME=(weight=baseline));
  BY ptno;
  change=weight-baseline;
RUN;
```

```
INFO: The variable VISIT on data set WORK.WT_BY_VIS will be overwritten by data set
WORK.BASELINE.
NOTE: There were 9 observations read from the data set WORK.WT_BY_VIS.
NOTE: There were 3 observations read from the data set WORK.BASELINE.
NOTE: The data set WORK.CHANGE has 9 observations and 5 variables.
```

The solution to this problem is very simple and is probably what most good programming practice guides recommend, only keep variables which are required. In this example the VISIT variable is not required once the baseline has been selected, so by dropping this variable the problem will be solved.

```
DATA change;
  MERGE wt_by_vis baseline(KEEP=ptno weight RENAME=(weight=baseline));
  BY ptno;
  change=weight-baseline;
RUN;
```

```
NOTE: There were 9 observations read from the data set WORK.WT_BY_VIS.
NOTE: There were 3 observations read from the data set WORK.BASELINE.
NOTE: The data set WORK.CHANGE has 9 observations and 5 variables.
```

```
TITLE3 'CHANGE from baseline - only keep required variables';
PROC PRINT DATA=change;
RUN;
```

```
NOTE: There were 9 observations read from the data set WORK.CHANGE.
```

CHANGE from baseline - only keep required variables

Obs	PTNO	VISIT	WEIGHT	BASELINE	CHANGE
1	1	1	79	76	3
2	1	2	76	76	0
3	1	3	65	76	-11
4	2	1	66	60	6
5	2	2	60	60	0
6	2	3	90	60	30
7	3	1	79	92	-13
8	3	2	92	92	0
9	3	3	75	92	-17

The VISIT variable now has the correct values in all observations, and the LOG does not show any messages which suggests there may be issues.

W.D FORMAT WAS TOO SMALL

This is another comment in the LOG which is not too uncommon. It basically means that a format used to print a number is too small, and so SAS ignores the format for that value and fits it as best as possible. At best this just means the loss of a decimal place, but at worse it can lead to an Exx number instead of an actual number appearing in a summary table or a listing. Although these may be picked up in a small summary tables, in large tables and listings where not all values are checked, the errors may go unnoticed.

Program:

```
DATA wd_fmt;
  a=123456.45;
  x=a;
  y=a;
  z=a;
  FORMAT x 5.2 y 6.2 z 8.2;
  LABEL a = "A - no format"
        x = "X - format 5.2"
        y = "Y - format 6.2"
        z = "Z - format 8.2"
        ;
RUN;

TITLE3 "ALL variables have the same value, but the formats affect how they
look";
PROC PRINT DATA=wd_fmt LABEL;
RUN;
TITLE3;
```

Log:

```
  DATA wd_fmt;
    a=123456.45;
    x=a;
    y=a;
    z=a;
    FORMAT x 5.2 y 6.2 z 8.2;
    LABEL a = "A - no format"
          x = "X - format 5.2"
          y = "Y - format 6.2"
          z = "Z - format 8.2"
          ;
  RUN;

NOTE: The data set WORK.WD_FMT has 1 observations and 4 variables.

  TITLE3 "ALL variables have the same value, but the formats affect how they look";

  PROC PRINT DATA=wd_fmt LABEL;
  RUN;

NOTE: There were 1 observations read from the data set WORK.WD_FMT.
NOTE: At least one W.D format was too small for the number to be printed. The decimal may be
shifted by the "BEST" format.
```

Output:

```
                      X -       Y -           Z -
         A - no     format    format       format
Obs      format      5.2       6.2           8.2

1     123456.45    123E3     123456     123456.5
```

This illustration shows how the original value as presented in A can be presented in tables or listings if the correct format is not used. The only way to know that the correct format is not used is the message in the LOG, and so it is very important that this message is not ignored. The only problem with SAS at the moment is that it does not mention which variable is the cause, and so if many variables are being listed or appear in a dataset, then they all have to be checked to determine which variable has the format issue. This can be done using PROC FREQ of all numeric variables, so that all values are listed and a quick scan should highlight which have problems.

```
  DATA wd_fmt2;
    DO i=1 TO 10;
      a=RANUNI(0)*2000;
      b=RANUNI(0)*100;
      OUTPUT;
    END;
    FORMAT a 3.1 b 4.1;
  RUN;

NOTE: The data set WORK.WD_FMT2 has 10 observations and 3 variables.

  TITLE3 'Data with an format which is too small';
  PROC PRINT DATA=wd_fmt2;
  RUN;

NOTE: There were 10 observations read from the data set WORK.WD_FMT2.
NOTE: At least one W.D format was too small for the number to be printed. The decimal may be
shifted by the "BEST" format.

  TITLE3 'Check which value is incorrect';
  PROC FREQ DATA=wd_fmt2;
    TABLES a b;
  RUN;

NOTE: There were 10 observations read from the data set WORK.WD_FMT2.
NOTE: At least one W.D format was too small for the number to be printed. The decimal may be
shifted by the "BEST" format.

Data with an format which is too small

Obs      I       A        B

  1      1     2E3      94.1
  2      2     2E3       9.9
  3      3     1E3      53.1
  4      4     505      48.0
  5      5     2E3      41.6
  6      6     2E3      68.7
  7      7     2E3      76.2
  8      8     998      17.2
  9      9     2E3       4.4
 10     10     1E3      47.1
```

```
Check which value is incorrect

The FREQ Procedure
                                        Cumulative    Cumulative
  A     Frequency     Percent     Frequency      Percent
-------------------------------------------------------------
505             1       10.00             1        10.00
998             1       10.00             2        20.00
1E3             2       20.00             4        40.00
2E3             6       60.00            10       100.00

                                        Cumulative    Cumulative
   B     Frequency     Percent     Frequency      Percent
-------------------------------------------------------------
 4.4             1       10.00             1        10.00
 9.9             1       10.00             2        20.00
17.2             1       10.00             3        30.00
41.6             1       10.00             4        40.00
47.1             1       10.00             5        50.00
48.0             1       10.00             6        60.00
53.1             1       10.00             7        70.00
68.7             1       10.00             8        80.00
76.2             1       10.00             9        90.00
94.1             1       10.00            10       100.00
```

Here we can see that variable A is the problem, and so changing the format of A will remove the note from the LOG.

```
  DATA wd_fmt2;
    DO i=1 TO 10;
      a=RANUNI(0)*2000;
      b=RANUNI(0)*100;
      OUTPUT;
    END;
    FORMAT a 6.1 b 4.1;
  RUN;

NOTE: The data set WORK.WD_FMT2 has 10 observations and 3 variables.

  TITLE3 'Data with the correct format';
  PROC PRINT DATA=wd_fmt2;
  RUN;

NOTE: There were 10 observations read from the data set WORK.WD_FMT2.

Data with the correct format

Obs     I         A       B

  1     1     918.6     7.2
  2     2     485.1    72.5
  3     3     561.0    62.9
  4     4    1593.8    17.4
  5     5    1966.5     1.0
  6     6    1868.3    32.6
  7     7     878.8    81.5
  8     8    1708.4    63.3
  9     9     669.3    54.2
 10    10    1689.1    26.7
```

CONCLUSION

As demonstrated in this paper, it is very important that programmers use all options open to them to ensure problems are highlighted, and any messages other than the standard NOTE stating how many observations are read in or sent to the output dataset is investigated. Failure to investigate and resolve these issues can have serious consequences on the results those programs produce.

It is also important to note that these issues will not be obvious if the programs are validated simply by checking the code, and so outputs must be checked very carefully in order to spot these issues. Using OPTIONS MSGLEVEL=I is also highly recommended to be used as default because it only sends messages to the LOG when there is a problem, which once resolved will not have any other impact.

CONTACT INFORMATION

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

Author Name	: Shafi Chowdhury
Company	: Shafi Consultancy Limited
Address	: 7 Blossom Way, Uxbridge, Middlesex, UB10 9LL, U.K.
Phone	: +44 770 288 7219
Email	: Shafi@shaficonsultancy.com
Web	: www.shaficonsultancy.com