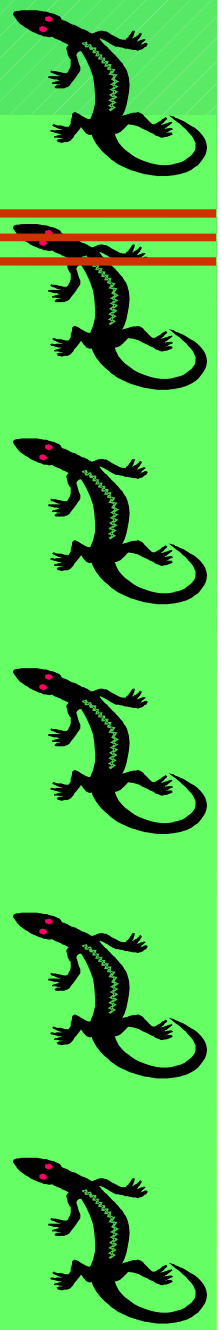


Looking Beneath the Surface of the SASLOG – Digging into the Roots of NOTEs, WARNINGs, and ERRORs

Andrew T. Kuligowski
HSN

Knowing your Audience

New Mexico

Little Rock

Memphis

Texas

Knowing your Audience

HAMILTON ?

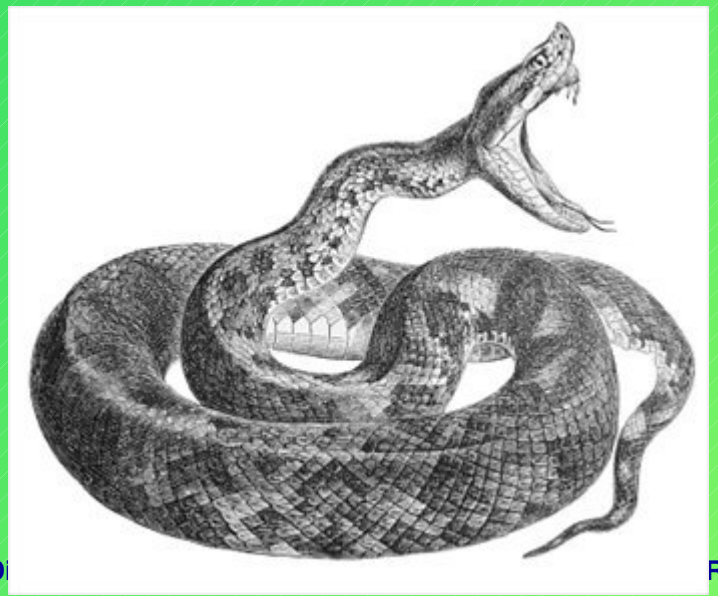
Knowing your Audience

Knowing your Audience

Montréal et Québec?

Looking Beneath the Surface of the SASLOG Introduction

Fleur-de-lis



Fer-de-lance

Looking Beneath the Surface of the SASLOG Introduction

Introduction

MERGE Statement: Repeats of BY Values

INPUT Statement:
Reached past the end of a line.

BEST. Format

Conclusion

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

Show of hands - How many people have ever gotten the following message:

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

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

What do you do?

Look it up in the Error Messages manual?

Maybe eventually … if they ever *release* an Error Messages manual!

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

What do you do?

Check out old SAS Global Forum Proceedings?

No, most articles assume you expected duplicate BY values and want to process them.
What if you *didn't* expect them?

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

We are going to merge two files.

The BREED file contains a list of dog breeds with related information.

The DOGS PER HHLD file contains a list of households with their dogs, 1 record per dog per household.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

BREED	VARIETY	OTHERINF
Bulldog		17834
Dalmatian		49235
Dachshund	Mini	18435
Dachshund	MiniLonghair	75846
Dachshund	MiniWirehair	09431
Dachshund	Std	18098
Dachshund	Std Longhair	75324
Dachshund	Std Wirehair	09389
Ger Sheprd		09622
GoldRetrvr		38292
Husky,Sib		75555
Lab Retrvr		38192

Breed File

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

HHLDID	BREED	VARIETY
0005884	Dalmatian	
0005884	Dalmatian	
0005884	Bulldog	English
0005884	Dachshund	Std Longhair
0005884	Dachshund	Std Longhair
0008824	Ger Sheprd	
0008824	Husky,Sib	
0008824	Lab Retrvr	
0008824	GoldRetrvr	

Dogs per Household File

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
177   DATA DOGDTAIL;
178      MERGE DOGOWNED(IN=IN_OWNED)
179            DOGBREED(IN=IN_BREED);
180      BY   BREED ;
181      IF   IN_OWNED ;
182   RUN;

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

NOTE: The data set WORK.DOGDTAIL has
      13 observations and 7
      variables.
```

MERGE with "repeats of BY values"

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

OK, so what do we do about it?

Ignore it, maybe it'll go away.

Get different data and re-run the example.

Give up, go home, and watch TV.

Try to figure out what caused the situation, and resolve it.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

We of course will go with:

Try to figure out what caused the situation, and resolve it.

(This would be a short presentation if we gave up now – I have to justify this trip!!)

We are going to run PROC MEANS against each of the two datasets, using the BY variables.

This will give us 1 record per unique set of BY values. N= will count the number of records with those values.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
206   PROC MEANS DATA=DOGOWNED
207                NOPRINT  NWAY;
208       CLASS  BREED ;
209       VAR HHLD_ID ;
210       OUTPUT  OUT=SUMOWNED N=CNTOWNED;
211   RUN;

NOTE: The data set WORK.SUMOWNED has
      7 observations and 4 variables.

NOTE: The PROCEDURE MEANS used 0.05 seconds.
```

PROC MEANS - 1st Dataset

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
213   PROC MEANS DATA=DOGBREED
214                NOPRINT  NWAY;
215       CLASS  BREED ;
216       VAR OTHRINFO ;
217       OUTPUT  OUT=SUMBREED  N=;
218   RUN;

NOTE: The data set WORK.SUMBREED has
      7 observations and 4 variables.

NOTE: The PROCEDURE MEANS used 0.05 seconds.
```

PROC MEANS - 2nd Dataset

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

Now, we are going to merge the outputs of the two PROC MEANS steps.

We will only keep the records representing the BY values that have multiple values in each dataset. These are the values that caused the message in the SASLOG.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
580    DATA SUMMERGE (KEEP=BREED CNTBREED   CNTOWNED);
581       MERGE   SUMBREED
582             (RENAME=(_FREQ_=CNTBREED))
583                  SUMOWNED ;
584       BY   BREED ;
585       IF   CNTBREED > 1   AND
586            CNTOWNED > 1;
587 RUN ;

NOTE: The data set WORK.SUMMERGE has
      1 observations and 3 variables.
```

MERGE the PROC MEANS outputs.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

SAS Dataset WORK.SUMMERGE

OBS	BREED	CNTBREED	CNTOWNED
1	Dachshund	6	2

MERGE the PROC MEANS outputs.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

Sometimes, at this point, the problem will become obvious. In that case, there's no reason to keep writing ad hocs - just go fix your routine!

At other times, the solution may not be as apparent. You will need to keep digging ...

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

In our example, we will merge each of our original datasets against the output of the corresponding PROC MEANS.

This will isolate the particular records that are triggering the "repeats of BY values" condition.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
599   DATA   CHKOWNED ;
600        MERGE   DOGOWNED (IN=IN_BREED)
601                SUMMERGE (IN=IN_MERGE);
602        BY   BREED ;
603        IF   IN_MERGE ;
604   RUN ;

NOTE: The data set WORK.CHKOWNED has
      2 observations and 7 variables.
```

MERGE the summarized data back to the original datasets for further analysis.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
605   DATA   CHKBREED ;
606        MERGE   DOGBREED (IN=IN_BREED)
607                SUMMERGE (IN=IN_MERGE);
608        BY   BREED ;
609        IF   IN_MERGE ;
610   RUN ;

NOTE: The data set WORK.CHKBREED has
      6 observations and 5 variables.
```

MERGE the summarized data back to the original datasets for further analysis.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

SAS Dataset WORK.CHKOWNED

OBS	HHLD_ID	BREED	VARIETY
1	5884	Dachshund	Std Longhair
2	5884	Dachshund	Std Longhair

OBS	BIRTHDAY	GOTCHADY	CNTBREED	CNTOWNED
1	12678	12870	6	2
2	13085	13179	6	2

MERGE the summarized data back to the original datasets for further analysis.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

SAS Dataset WORK.CHKBREED

OBS	BREED	VARIETY	OTHRINFO	CNTBREED	CNTOWNED
1	Dachshund	Mini	1843	6	2
2	Dachshund	MiniLonghair	7584	6	2
3	Dachshund	MiniWirehair	943	6	2
4	Dachshund	Std	1809	6	2
5	Dachshund	Std Longhair	7532	6	2
6	Dachshund	Std Wirehair	938	6	2

MERGE the summarized data back to the original datasets for further analysis.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

At this point, it should be obvious that the problem was caused by the omission of VARIETY from the MERGE. This was not a problem for the 1st dataset, but presented an issue with the 2nd dataset.

Fix the code, rerun it, and everything should be fine now.

Looking Beneath the Surface of the SASLOG MERGE Statement with repeats of BY Values

```
682 DATA DOGDTAIL;
683      MERGE  DOGOWNED (IN=IN_OWNED)
684             DOGBREED (IN=IN_BREED);
685      BY   BREED VARIETY;
686      IF   IN_OWNED ;
687 RUN;

NOTE: The data set WORK.DOGDTAIL has
      9 observations and 6 variables.
```

As you can see, the problematic NOTE is no longer present!

Looking Beneath the Surface of the SASLOG
INPUT Statement reached past end of a line

Show of hands -
How many people have ever gotten the following message:

```
NOTE: SAS went to a new line when
INPUT statement reached past
the end of a line.
```

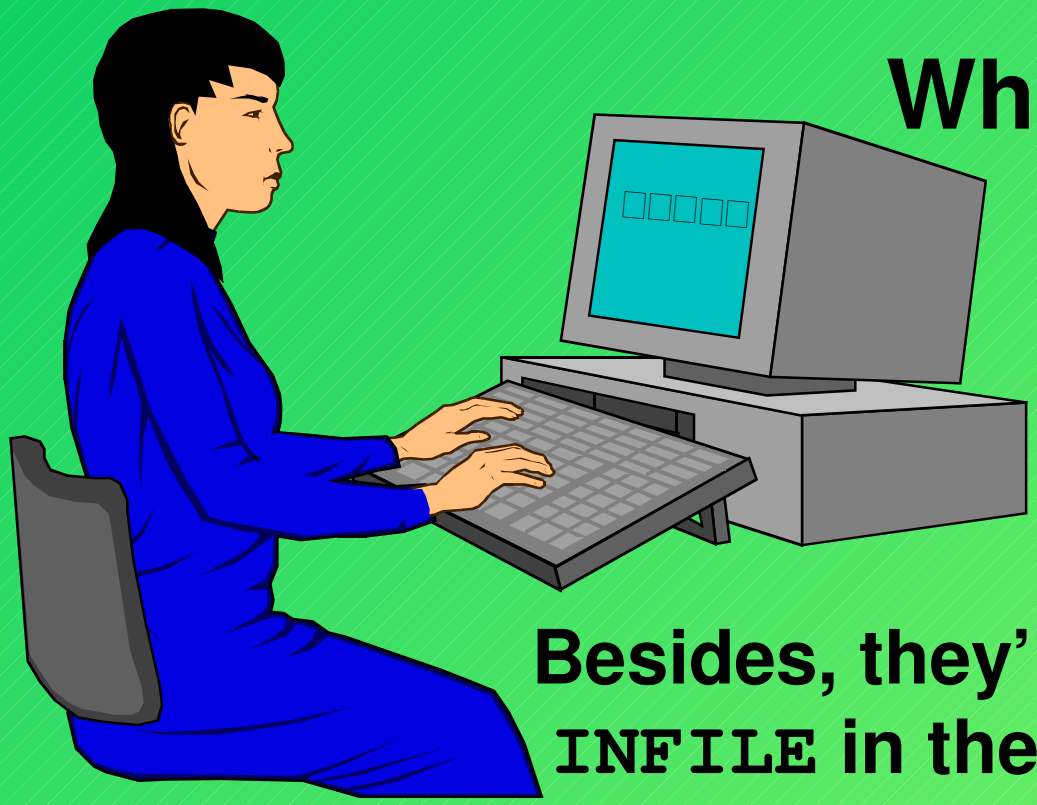
Looking Beneath the Surface of the SASLOG
INPUT Statement reached past end of a line

What do you do?

Hire a consultant?

No, we handle this ourselves.

Looking Beneath the Surface of the SASLOG
INPUT Statement reached past end of a line



What do you do?

Hire a consultant?

Besides, they'll just look up `INFILE` in the manual, and use options `MISSOVER`, `TRUNCOVER`, or `STOPOVER` as a workaround.

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

MISSOVER and TRUNCOVER will prevent SAS from reading the next line, but will continue processing.

STOPOVER will force an error condition and stop reading the dataset.

Neither explain why your data are not what you expected.

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

Attendance File

```
DATE    CLASS     REG  ABSNT
022105  Physics    12     2
022105  Botany     15     7
022105  Geology    16     9
022105  Anatomy     8     1
022105  Zoology    10     0

ID#  OF  STUDENTS  IN  ATTENDANCE
27  29  33  34  37  41  42  43  44  45
 6   7   9  28  35  36  40  51
13  29  30  31  39  45  46
10  12  22  25  32  47  49
 1   3   7   8   9  12  18  19  22  23
```

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

```
15 DATA  ATTEND;
16    ARRAY ATNDID (25)
17        ATNDID01-ATNDID25 ;
18    INFILE  'C:\ATTEND.DAT';
19    INPUT   @  1  DATE     MMDDYY8.
20            @ 10  CLASS     $CHAR8.
21            @ 18  REGIST          2.
22            @ 21  ABSENT          2. @ ;
23    pt = 24 ;
24    DO CNT = 1 TO REGIST;
25        INPUT @ pt ATNDID(CNT) 2. @ ;
26        pt = pt + 3 ;
27    END ;
28 RUN;
```

SASLOG - INPUT Statement reached past end of line

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

```
NOTE: 5 records were read from the infile
      'C:\ATTEND.DAT'.
The minimum record length was 43.
The maximum record length was 52.

NOTE: SAS went to a new line when INPUT statement
      statement reached past the end of a line.

NOTE: The data set WORK.ATTEND has
      3 observations and 31 variables.
NOTE: The DATA statement used 0.98 seconds.
```

SASLOG - INPUT Statement reached past end of line

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

OK, so what do we do about it?

A) Ignore it, and go get a snack. Where's the nearest Tim's?

B) Snack?? – Heck, let's go to the local tavern instead!!

C) Note some oddities in the SASLOG – for example, it told us that it read 3 lines even though we provided 5 lines worth of data!

Looking Beneath the Surface of the SASLOG
INPUT Statement reached past end of a line

We of course will go with:

C) Note some oddities in the SASLOG …

(Although the "Tim's" and "tavern" options sound tempting …)

How come SAS only read 3 lines, when we passed it 5?

Let us use the LENGTH= option to find out how long each line is while we read it in.

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

```
51 DATA  LINELONG;
52   INFILE   'C:\ATTEND.DAT'
53              MISSOVER   LENGTH=LNSZ;
54   INPUT   @   1 DATE         MMDDYY8.
55           @ 10 CLASS          $CHAR8.
56           @ 18 REGISTCT             2.
57           @ 21 ABSENTCT             2. @ ;
58   LINESIZE = LNSZ;
59   STDNTCNT = (LINESIZE - 23 + 1)/3;
60 RUN;
```

SASLOG – Determining Line Size

Note that we will also determine the Student Count: Total Line Size – 23 chars (key info) + 1 char (compensate for not blank padding last student ID) divided by 3 (because each student ID is 3 chars long, including blank)

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

DATE	CLASS	REG	ABS	LNSZ	SCNT
022105	Physics	12	2	52	10
022105	Botany	15	7	46	8
022105	Geology	16	9	43	7
022105	Anatomy	8	1	46	7
022105	Zoology	10	0	52	10

↑ ↑

Attendance File after analysis

Note that only 1 class (Zoology) has the same number of "*REG*istered" vs. "*S*tudent *CNT*".

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

DATE	CLASS	REG	ABS	LNSZ	SCNT
022105	Physics	12	2	52	10
022105	Botany	15	7	46	8
022105	Geology	16	9	43	7
022105	Anatomy	8	1	46	7
022105	Zoology	10	0	52	10

Attendance File after analysis

Note that only 1 class (Zoology) has the same number of “REGistered” vs “Student CNT”.

Also note that the value of ABSent is ZERO – 0 for this record (and *only* for this record.)

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

The answer - We neglected to include Absentees in our algorithm. We should not expect to find an ID for every registered student. Instead:

```
Attending Students =
    Registered - Absentee
```

(If we were not sure of this, we could have modified our ad hoc to add this calculation and compare it to the SCNT variable.)

Looking Beneath the Surface of the SASLOG
INPUT Statement reached past end of a line

```
579 DATA  ATTEND;
580   ARRAY ATNDID (25)
581         ATNDID01-ATNDID25 ;
582   INFILE  'C:\ATTEND.DAT';
583   INPUT  @  1  DATE   MMDDYY8.
584          @ 10  CLASS   $CHAR8.
585          @ 18  REGIST        2.
586          @ 21  ABSENT        2. @ ;
587   STDNTCNT = REGIST - ABSENT ;
588   pt = 24 ;
589   DO CNT = 1 TO STDNTCNT ;
590      INPUT @ pt ATNDID(CNT) 2. @ ;
591      pt = pt + 3 ;
592   END ;
593 RUN;
```

RESOLVED

SASLOG - INPUT Statement reached past end of line

Looking Beneath the Surface of the SASLOG
INPUT Statement reached past end of a line

```
NOTE: 5 records were read from the infile
      'C:\ATTEND.DAT'.
      The minimum record length was 43.
      The maximum record length was 52.

NOTE: The data set WORK.ATTEND has
      5 observations and 32 variables.

NOTE: The DATA statement used 0.7 seconds.
```

RESOLVED

SASLOG - INPUT Statement reached past end of line

The message is gone!
And we have all 5 observations!

Looking Beneath the Surface of the SASLOG INPUT Statement reached past end of a line

OK, so what if the example wasn't quite as contrived?

Check other assumptions. (I took a shortcut by assuming the problem occurred while we were reading in our array.)

Try an alternate tool. (For example, I considered using the `$VARYING.` informat in this example.)

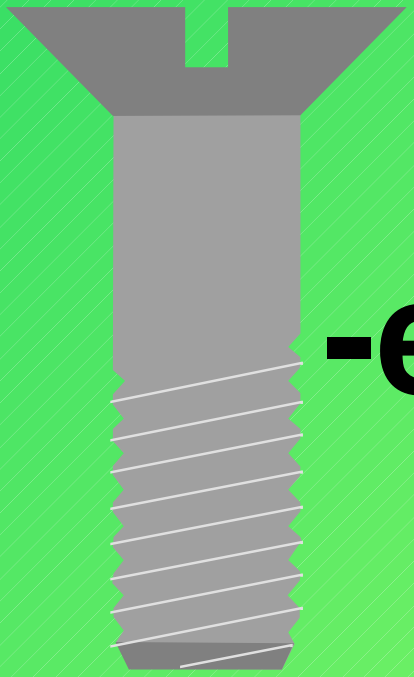
Looking Beneath the Surface of the SASLOG
BEST. format

Show of hands -
How many people have ever gotten the following message:

```
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.
```

Looking Beneath the Surface of the SASLOG BEST. format

What do you do?

We're  **-ed**

NO! Don't give up hope!
There's always a way around a problem!

Looking Beneath the Surface of the SASLOG
BEST. format

SAS ensures that the simple matter of a number exceeding its anticipated length will not cause the entire run to abort.

Sometimes, this is an acceptable situation.

However, most folks would prefer that *THEY* control their output formats, rather than letting SAS override their choices.

Looking Beneath the Surface of the SASLOG BEST. format

In many cases, the offender(s) can be quickly spotted, simply by glancing at the offending output.

In other cases (a few isolated offenders in a large report, for example), the human eye is not enough to do the job – not without help.

Looking Beneath the Surface of the SASLOG BEST. format

```
DATA  FORMAT42;
   INFILE CARDS; DATALINES;
   INPUT   @   1   ACTUAL   $CHAR5.
           @   1   FMT4_2          5.;
   FILE   LOG ;
   PUT @   1   ACTUAL=   $CHAR5.
PUTLOG @ 15   FMT4_2=          4.2 ;
CARDS; DATALINES;
7.499
14.49
768.1
1997
4858.
54632
;
```

Looking Beneath the Surface of the SASLOG BEST. format

```
629  DATA  FORMAT42;
630       INFILE CARDS;
631       INPUT   @   1   ACTUAL $CHAR5.
632               @   1   FMT4_2        5.;
633       FILE   LOG ;
634       PUT @   1   ACTUAL= $CHAR5.
635           @ 15   FMT4_2=        4.2 ;
636  CARDS;
ACTUAL=7.499   FMT4_2=7.50
ACTUAL=14.49   FMT4_2=14.5
ACTUAL=768.1   FMT4_2=768
ACTUAL=1997    FMT4_2=1997
ACTUAL=4858.   FMT4_2=4858
ACTUAL=54632   FMT4_2=55E3
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.
```

Looking Beneath the Surface of the SASLOG BEST. format

OK, so what do we do about it?

A) It's obvious! There are only 6 numbers on the whole report!

B) Leave it on a subordinate's desk with "???" written on a Post-it Note.

C) Try to figure out which number(s) caused the situation to occur.

Looking Beneath the Surface of the SASLOG

BEST. format

A) It's obvious …
is true, but we need to illustrate the point.

B) "delegate it"
is only a good answer if you have someone you can delegate to. (Keep in mind, this is not intended to be a Management seminar!)

Let's go with:

C) Try to figure out which number(s) caused the situation to occur.

Looking Beneath the Surface of the SASLOG BEST. format

```
645 DATA   _NULL_ ;
646    SET   FORMAT42;
647    C_FMT4_2 = PUT(FMT4_2,   Z4.2);
648    WHERE_PT = INDEX(C_FMT4_2, '.');
649    WHERE_E  = INDEX(C_FMT4_2, 'E');
650    IF   WHERE_PT ^= 2   THEN
651       ERRNOTE1 = 'DECIMAL';
652    IF   WHERE_E  ^= 0   THEN
653       ERRNOTE2 = 'EXPONENTIAL';
654    FILE   LOG ;
655    PUT @   1   ACTUAL      $CHAR5.
656        @   7   C_FMT4_2    $CHAR4.
657        @ 13    ERRNOTE1    $CHAR10.
658        @ 24    ERRNOTE2    $CHAR12.;
659 RUN ;
```

SASLOG – Trying to find the numbers that are not printed with our selected formats.

Looking Beneath the Surface of the SASLOG BEST. format

Note the following lines:

```
C_FMT4_2 = PUT(FMT4_2,  Z4.2);
```

Use the PUT function to store the formatted value in a character variable.

```
WHERE_PT = INDEX(C_FMT4_2, '.');
```

Parse that character variable to determine where the decimal is being inserted.

```
WHERE_E  = INDEX(C_FMT4_2, 'E');
```

Parse that character variable to determine if an "E" is present, indicating exponential notation.

A simple IF test can then show us if any of our numbers fit either condition.

Looking Beneath the Surface of the SASLOG BEST. format

```
7.499 7.50
14.49 14.5   DECIMAL
768.1 0768   DECIMAL
 1997 1997   DECIMAL
4858. 4858   DECIMAL
54632 55E3   DECIMAL    EXPONENTIAL

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.
NOTE: The DATA statement used 0.28
      seconds.
```

Continuing the SASLOG, we can see that MOST of the numbers fail to allow enough space for 2 significant decimal places.

One of them also prints Scientific Notation.

Looking Beneath the Surface of the SASLOG BEST. format

We have verified that most of the numbers on the report are too wide for our original format. We need to redesign our report to allow for more digits to the left of the decimal place.

(or we need to re-run with a different set of numbers.)

Looking Beneath the Surface of the SASLOG

LOST CARD

Unabashed cross-promotion

In Search of the LOST CARD

Andrew T. Kuligowski

SUGI 30 Proceedings

Looking Beneath the Surface of the SASLOG Conclusion

A) Always check your SASLOG!

B) Know your data before you code, but don't be afraid to learn more about your data while you are coding.

C) Sometimes the solution to a coding problem is to leave your original code alone.

Looking Beneath the Surface of the SASLOG Conclusion

The ad hocs in this example may or may not be of use to you in your future endeavors.

Hopefully, the *concept* of knowing your data, and writing your own ad hocs to further understand it will be *very* useful to you for the rest of your career.

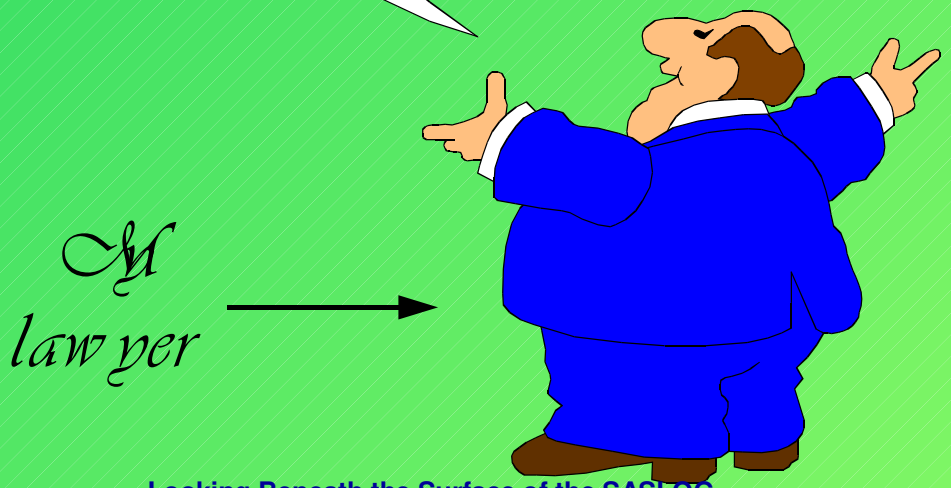
Looking Beneath the Surface of the SASLOG Conclusion

The author can be contacted at:

KuligowskiConference@gmail.com

Looking Beneath the Surface of the SASLOG

SAS is a registered trade-mark or trademark of SAS Institute, Inc. in the USA and other countries. (R) indicates USA registration.



My lawyer