PharmaSUG 2026 - Paper AP-343

Has your SAS being 'MEAN' to your data yet?

Marisol Rivera and Isaac Vazquez; Ephicacy Consulting Group

ABSTRACT

Daily work in the pharmaceutical industry requires extensive data handling and report generation, including the creation of standardized data sets like Study Data Tabulation Model (SDTM) and Analysis Data Model (ADaM) data sets to support protocol and SAP (Statistical Analysis Plan) requirements. During data preparation, statistical programmers routinely apply formats, perform mathematical calculations, and structure data to ensure meaningful and accurate presentation.

At the reporting stage, SAS procedures such as PROC MEANS and PROC UNIVARIATE are commonly used to summarize and analyze data. However, under certain data conditions, these procedures may produce unexpected or inconsistent results. Such behavior can be difficult to detect and may have downstream implications for reporting accuracy and interpretation.

This paper presents a practical example illustrating how seemingly minor data changes, specifically, the presence and handling of a sorting variable, can influence the output of standard SAS summary procedures. The example highlights how small differences in data ordering can lead to variations in results, emphasizing the importance of careful data preparation and validation prior to analysis and reporting.

INTRODUCTION

The example presented in this paper focuses on a specific scenario in which rounding behavior, driven by SAS internal precision handling, affects the results produced by standard summary procedures. All values used in this example are derived from one ADaM data set, where numeric variables are stored with a default precision of two decimal places.

For reporting purposes, PROC MEANS and PROC UNIVARIATE are used to generate standard descriptive statistics commonly required in the pharmaceutical industry, including N, mean, standard deviation, median, minimum, maximum.

This paper is primarily intended for programmers having issues while handling these procedures and not being able to fully validate the result provided by them.

BACKGROUND

ORDER-DEPENDENT ACCUMULATION EFFECTS

Many statistical procedures, including PROC MEANS and PROC UNIVARIATE, compute summary statistics through sequential accumulation of numeric values. In floating-point arithmetic, the order in which numbers are summed can affect the final result due to rounding behavior inherent to binary representation.

When data sets are processed without a consistent and explicit ordering, small numerical differences may arise during collection. Although these differences are typically minimal, they can become noticeable when:

- Large numbers of observations are involved.
- Values contain multiple decimal places.
- Statistical summaries are computed within grouped categories.
- Validation comparisons require exact reproducibility.

Therefore, the ordering of observations prior to statistical computation may indirectly influence results, particularly for statistics derived from cumulative calculations such as the mean or standard deviation.

DISPLAYED PRECISION VERSUS COMPUTATIONAL PRECISION

In clinical reporting workflows, analysis data sets often store values rounded to a defined number of decimal places according to mock shells. However, rounding applied at the display or reporting level does not modify the internal floating-point storage of the variable unless explicitly enforced through data transformation.

Consequently, statistical procedures operate on the stored numeric values rather than their formatted representations. This behavior can create scenarios in which two workflows that apply identical statistical procedures produce slightly different results if preprocessing steps alter the ordering or representation of numeric values.

Understanding this distinction is essential for statistical programmers involved in production and quality control validation, as apparent discrepancies may originate from computational precision effects rather than methodological differences.

For example, even if SAS shows:12.34

Internally it might store something like:12.3399999999998 or 12.3400000000002

When you sum many numbers like this:

- Tiny differences accumulate.
- Order of summation matters
- PROC MEANS and PROC UNIVARIATE may give slightly different results.

REAL-WORLD CASE STUDY

During the Validation process of a summary table, a discrepancy was identified in a single row of a summary table containing several rows of statistics generated using PROC MEANS in production program, while in Validation program the PROC UNIVARIATE was used, trying to replicate identical statistical calculations by category. The difference was observed in only one specific combination of category and parameter, prompting further investigation into the underlying cause of the discrepancy.

After the initial investigation, we determined that both programs, despite being independently programmed, were taking the same approach, sub-setting the data, keeping only the records needed for the calculation, removing missing data, and applying the corresponding procedure. The final presentation of the data was having a difference in one single decimal of a MEAN value out of dozens of calculations that were matching..

DATA SET CONTEXT

We were validating a Summary of Clinical Success by Subgroup (when Clinical success was defined by the reduction in RV/LV ratio from baseline to 48 hours). In this case, the source of the PROC MEANS/UNIVARIATE was the ADEFF (Analysis Data Efficacy) data set which is a data set designed to store efficacy endpoints (primary, secondary, and exploratory) for statistical analysis, which follows the Basic Data Structure (BDS), typically containing one record per subject, per analysis parameter, per analysis visit.

For context reference purposes, the variables used in the proc means were:

- Parameter variable: param=’ RV/LV ratio’
- Category variable: grp=’ Age Group’
- Analysis variable: Analysis Value (AVAL)
- Analysis Visit: AVISITN=1
- Format to display in tables: Two decimals

INITIAL DISCREPANCY DETECTION

During the PROC COMPARE step of the Validation process, we identified an apparently minor discrepancy, just a single decimal difference within one subgroup of a table spanning approximately 30 pages. Despite its small magnitude, the inconsistency raised immediate concern due to the complexity of the table, which included multiple subgroups, visits, and derived variables. At first glance, the issue did not appear to stem from a logical or programming error. However, given the regulatory context and the importance of reproducibility, even the smallest discrepancy warranted a thorough investigation, prompting a deep-dive debugging effort between the production and validation programmers.

Table 1**Error! Reference source not found.** shows the PROC COMPARE results with the only difference we found across the 30 pages.

Value Comparison Results for Variables

PRNTORD	STATORD	Base Value COL_1	Compare Value COL_1
1.1	2	1.51 (0.303)	1.52 (0.303)

Table 1 Original Proc Compare

RESOLVING THE DIFFERENCE PROCESS

What followed was an intensive effort to reconcile the results. We carefully reviewed and compared the production and validation workflows, held multiple discussions, and systematically evaluated every step of the process. Each assumption was challenged, and every line of code was scrutinized. After exhausting the more obvious possibilities (including changing the original Validation approach of a PROC UNIVARIATE to be also a PROC MEANS that production was using) we discovered a small but critical detail, one that initially appeared unrelated to any of these procedures themselves. The root cause was not within the summary step, but rather in the preceding **PROC SORT**.

HYPOTHESIS TESTING

As the investigation progressed, we identified the only tangible difference between the production and validation programs: the inclusion of the variable **AVAL** (used in the VAR statement of PROC MEANS and PROC UNIVARIATE) in the prior **PROC SOR**T. While this variation seemed minor, it became the focal point of our hypothesis. This finding highlighted an important and often overlooked lesson: even seemingly insignificant differences in data preparation steps, it can lead to unexpected and non-intuitive discrepancies in results.

In the following pages you will see the exact code we used to resolve this mismatch.

EXAMPLES

PRODUCTION CODE

As you can see the AVAL variable was included in the proc sort before the PROC MEANS.

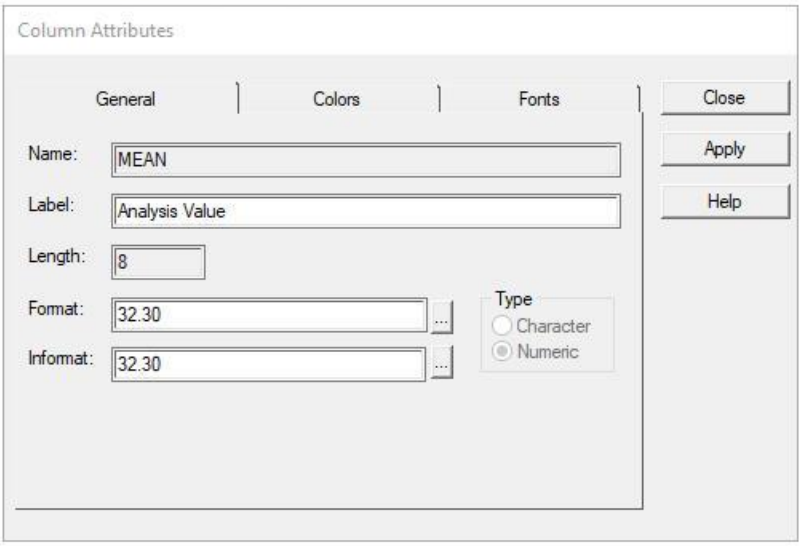
```
proc sort data= ADEFF (where=( grp=' Age Group' and AVISITN=1)) out=outds
    by grp param avisitn aval;
run;

 proc means data= outds  noprint ;
    by grp param avisitn;
      var aval;
         output out=outds2  n=n mean=mean std=std median=median min=min max=max;
 run ;
```

Program 1. Production Code

OUTPUT DATA SET FROM PRODUCTION CODE

For illustration purposes, we are showing the variable MEAN with the highest decimals possible on the MEAN variable:

N	MEAN	STD	MEDIAN	MIN	MAX
46	1.514999999999990000000000000000	0.3031446299	1.52	0.88	2.21



Data Set 1. Production Data Set

FORMATTING

After production side applied the required format (two decimals as per mock shells), using a data step with a 'PUT' statement on a data step as follows:

meansd=compress(put(mean,12.2))|| ' (' || compress(put(std, 12.3))|| ')' ;.

FINAL RESULT IN PRODUCTION SIDE

MEANSD
1.51 (0.303)

Data Set 2. Formatted Production Data Set

VALIDATION CODE

In this case Validation Code used PROC UNIVARIATE and it did not have the AVAL variable in the proc sort before the PROC UNIVARIATE.

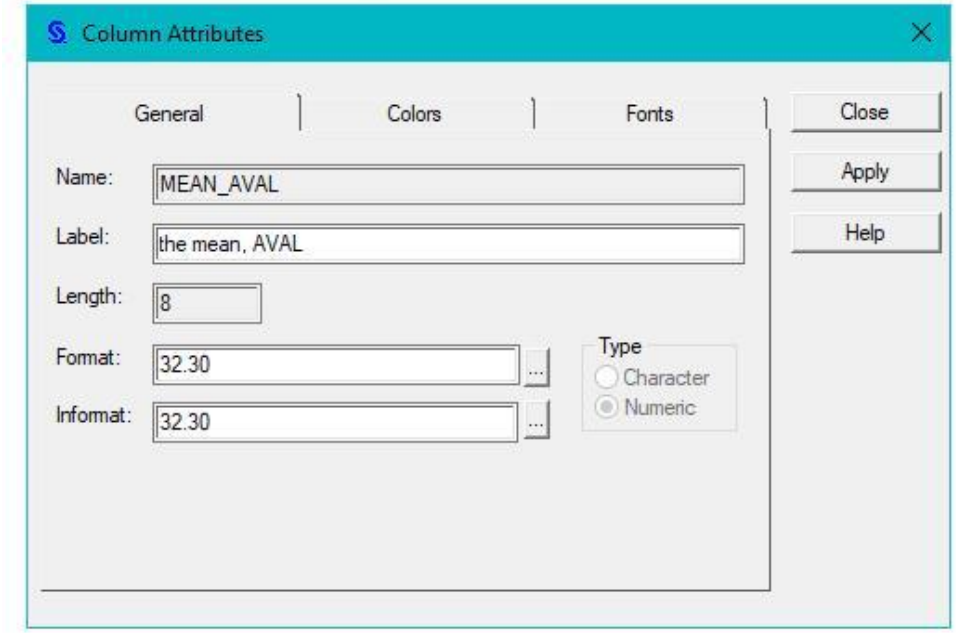
```
proc sort data=adeff;
    by grp param avisitn /*(NO aval variable was included in the sort)*/;
run;
proc univariate data=adeff;
   by grp param avisitn;
     var aval;
        output out=stats mean=mean_aval n=n_aval min=min_aval max=max_aval std=st_aval;
run;
```

Program 2. Validation Code

OUTPUT DATA SET FROM VALIDATION CODE

Similarly to Data Set 1. Production Data Set, we are showing the variable MEAN_AVAL with the highest decimals possible in SAS.

AVISITN	N_AVAL	MEAN_AVAL	ST_AVAL	MAX_AVAL	MIN_AVAL
1	46	1.515000000000000000000000000000	0.3031446299	2.21	0.88



Data Set 3. Validation Code

FORMATTING: (USING A SIMILAR APPROACH THAN IN PRODUCTION SIDE)

mean_fmt= strip(put(mean,8.2))|| ' (' || strip(put(std, 8.3))|| ')' ;

FINAL RESULT IN VALIDATION SIDE:

MEANSD
1.52 (0.303)

Data Set 4. Formatted Validation Data Set

RESOLUTION

After discovering the root of the problem, we just added the variable AVAL to the proc sort previous to the PROC UNIVARIATE in the validation program:

UPDATED VALIDATION CODE

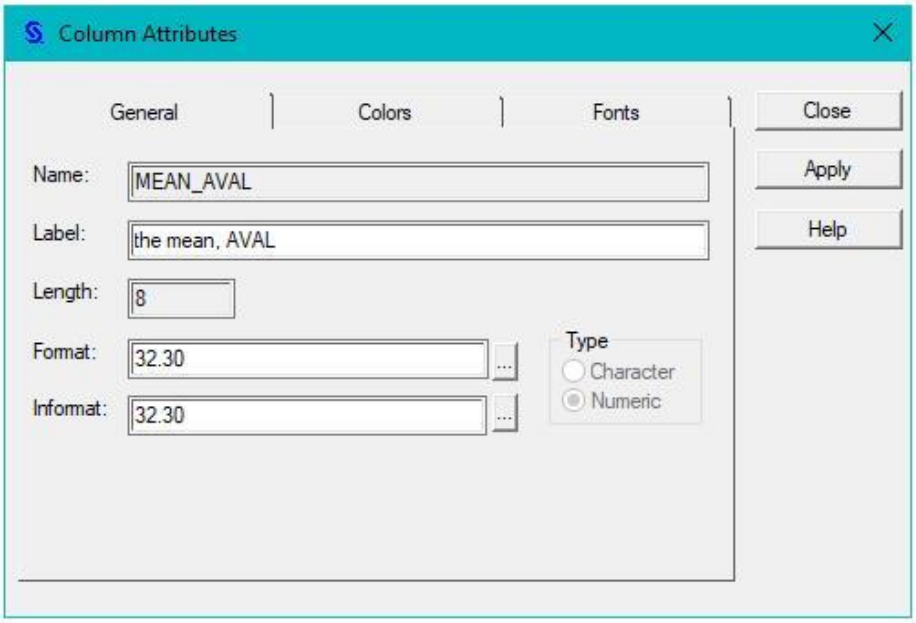
We made sure to add this time the AVAL variable to the PROC SORT before the PROC UNIVARIATE:

```
proc sort data=adeff;
    by grp param avisitn  aval;
run;
proc univariate data=adeff;
   by grp param avisitn;
     var aval;
        output out=stats mean=mean_aval n=n_aval min=min_aval max=max_aval std=st_aval;
run;
```

Program 3. Updated Validation Code

NEW VALIDATION OUTPUT DATA SET

AVISITN	N_AVAL	MEAN_AVAL	ST_AVAL	MAX_AVAL	MIN_AVAL
1	46	1.514999999999990000000000000000	0.30314463	2.21	0.88



Data Set 5. Updated Validation Data Set

As you can see, now this is the same value gotten in Data Set 1. Production Data Set, using this approach, we were able to match the result from the Data Set 2. Formatted Production Data Set and finally we were able to get a clean PROC COMPARE results, in opposite of the Table 1 Original Proc Compare.

UPDATED CLEAN COMPARE:

```
MQCCOMPARE: COMPARE RESULTS FOR prod vs. qc

The COMPARE Procedure
Comparison of WORK.PROD with WORK.QC
(Method=EXACT)

Data Set Summary

Dataset              Created          Modified  NVar    NObs

WORK.PROD   20MAR26:12:49:09  20MAR26:12:49:09     4     345
WORK.QC     20MAR26:12:49:09  20MAR26:12:49:09     4     345


Variables Summary

Number of Variables in Common: 4.
Number of ID Variables: 2.


Observation Summary

Observation      Base  Compare  ID

First Obs           1        1  PRNTORD=1 STATORD=-1
Last  Obs         345      345  PRNTORD=15.4 STATORD=5

Number of Observations in Common: 345.
Total Number of Observations Read from WORK.PROD: 345.
Total Number of Observations Read from WORK.QC: 345.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 345.

NOTE: No unequal values were found. All values compared are exactly equal.
```

Table 2. Clean Compare Results

KEY TAKEAWAY

This experience reinforces a critical principle in statistical programming: reproducibility depends not only on the analytical procedures, but also on the consistency of upstream data handling. Subtle differences in sorting or data structure can influence downstream results in ways that are not immediately apparent. Ensuring alignment in all preprocessing steps including PROC SORT, is essential to achieving reliable and consistent outputs, particularly in high regulatory environments.

RECOMMENDED BEST PRACTICES

- Always include analysis variable in SORT.
- Avoid implicit ordering.
- Control precision.
- Use reproducible macros in both production and validation programs.

CONCLUSION

The investigation demonstrated that a small discrepancy between production and Validation results was caused by SAS internal precision handling during statistical calculations. The difference was triggered by the ordering of the input data and was observed when the variable used in the statistical calculations was not included in the sorting logic prior to invoking PROC MEANS or PROC UNIVARIATE.

By explicitly including the analysis variable in the PROC SORT step before calling the summary procedure, the discrepancy was resolved and consistent results were obtained between production and Validation outputs. While this behavior may not manifest all data sets, it can occur under specific data conditions and may be difficult to detect during routine Validation.

This example highlights the importance of careful data ordering and awareness of numeric precision effects when generating summary statistics. Incorporating appropriate sorting practices can help prevent subtle mismatches and reduce unnecessary investigation during clinical reporting and validation.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:

Marisol Rivera
kmxmarisol@gmail.com

Isaac Vazquez
kmxisaac@gmail.com