PharmaSUG 2013 - Paper CC38

A Strategy to Ensure Non-Estimable Confidence Intervals Having Equal Lower and Upper Confidence Limits are Displayed Correctly

Claudia Jimenez-Castro, inVentiv Health Clinical, Mexico City, D.F.

ABSTRACT

This paper examines a particular case in which the scientific notation used by SAS® to represent numeric variables resulted in an incorrect representation of non-estimable confidence intervals in an efficacy table.

For these intervals both lower and upper confidence limits were equal. Intervals with both values equal are non-estimable and were coded to be displayed as such. To determine non-estimable values, the lower limit was subtracted from its corresponding upper limit. However, due to the floating point arithmetic used by SAS, subtracting "a" from "a" yielded a result different from zero.

This paper describes the reason behind the error and suggests a strategy to fix the issue.

CASE DESCRIPTION

1. FROM THE IMMUNOLOGICAL POINT OF VIEW

The problem arose with an efficacy table that compares the immune response of two treatment groups. Each group received a different vaccine. The values of this table are mean responses per treatment, response ratios to evaluate differences in treatment, and the respective confidence intervals.

The immune response was measured through an OPA assay. Broadly speaking, this assay consists of putting in contact the blood serum of vaccinated patients, phagocytes (a type of white blood cells) and the pathogen. The values measured in this assay are titers. The greater the value of the titer, the fewer bacteria survived the immune activity of the serum in this assay. In other words, the greater the value of the titer, the bigger the immune response is.

2. FROM THE MATHEMATICAL POINT OF VIEW

The table summarizes the mean response of both groups to several serotypes of the same pathogen, using the geometric means of the titers and their two-sided confidence intervals. In addition, the comparisons of the responses between groups are calculated using geometric mean ratios and 95% two-sided confidence intervals.

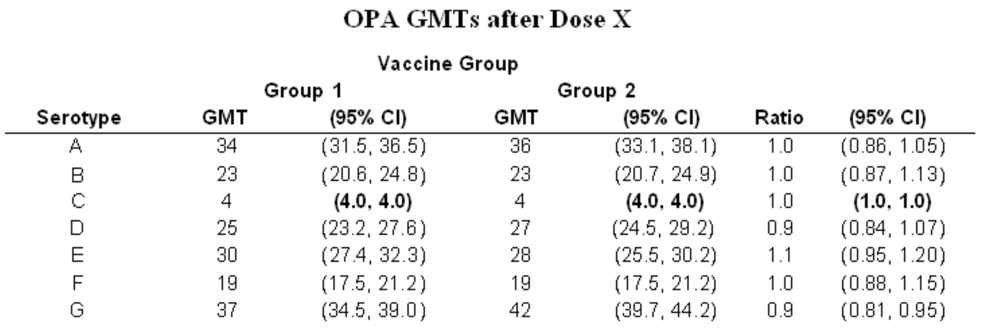
OPA assay results were logarithmically transformed before estimating means and confidence intervals. Therefore, the response ratio is obtained by calculating the difference between groups of the means of log transformed values. Similarly, the confidence interval of the ratio is calculated as the confidence interval of the difference between groups, using log-transformed values. The geometric means of the treatment and the comparison, as well as all confidence intervals are back-transformed to their original units after the calculations.

The hypothesis test used to compare means across treatment groups was evaluated using a t-test. The SUMMARY procedure was used to estimate the population mean and confidence intervals, and the MIXED procedure was used to estimate the mean response ratio and its confidence interval.

THE ISSUE

The issue encountered in this table was the incorrect representation of intervals for the mean and for the mean differences. (See figure 1).

The intervals (4.0, 4.0) and (1.0, 1.0) of serotype C are incorrectly represented here. They should be displayed as "NE" instead because the lower and the upper confidence limits are equal. This is not the case where both levels are displayed with equal values after being rounded to one decimal. They were truly equal because the immune response of all patients of both groups for these serotypes had all equal values, and so there was no way to estimate variability and build a confidence interval.



OPA GMTs after Dose X

	Vaccine Group					
	Group 1		Group 2			
Serotype	GMT	(95% CI)	GMT	(95% CI)	Ratio	(95% CI)
A	34	(31.5, 36.5)	36	(33.1, 38.1)	1.0	(0.86, 1.05)
B	23	(20.6, 24.8)	23	(20.7, 24.9)	1.0	(0.87, 1.13)
C	4	**(4.0, 4.0)**	4	**(4.0, 4.0)**	1.0	**(1.0, 1.0)**
D	25	(23.2, 27.6)	27	(24.5, 29.2)	0.9	(0.84, 1.07)
E	30	(27.4, 32.3)	28	(25.5, 30.2)	1.1	(0.95, 1.20)
F	19	(17.5, 21.2)	19	(17.5, 21.2)	1.0	(0.88, 1.15)
G	37	(34.5, 39.0)	42	(39.7, 44.2)	0.9	(0.81, 0.95)

Figure 1. First version of the table

You might wonder why these intervals are non-estimable. Here is a short explanation.

The general form of a confidence interval (Altman et al., 1988; Sullivan, 2012) is

Confidence interval = point estimate $\pm$ Z(standard error of the point estimate)

In this case study, the point estimates are the geometric mean titers and the geometric mean titer ratios displayed in the GMT and Ratio columns of the table, Z is the value from the standard normal distribution for the selected confidence level and the standard error of the point estimate is the standard deviation of the point estimate across hypothetical repeated samples. The standard error of the point estimate is a measure of its accuracy. It expresses the amount by which the population estimate is expected to differ from the value calculated from the sample.

The details of the calculation of the standard errors are beyond the scope of this article. Suffice it to say that the standard error is a term that includes the variance and the sample size. It is directly related to the variance and inversely correlated to the sample size. For a detailed explanation on the construction of confidence intervals, Bethea (1984) is an excellent reference.

In this study case, the data for serotype C are all equal, that is, the sample variance of these data is zero. In consequence, the standard error used in the confidence interval for the means and for the mean difference is also zero.

In these cases, the expressions of the confidence intervals are transformed into

Confidence interval = point estimate $\pm$ 0.

These are not intervals but point estimates. Therefore, in these cases, the confidence interval is not estimable.

When this table was created, this issue was quite an unexpected situation. Neither the code nor the data set from where confidence intervals were calculated gave a hint of the reason behind the incorrect representation of the intervals.

Estimating a mean response and a confidence interval from a group of identical values is not a common event but rather a rare one. However, it is worth knowing what to do in case you ever come across it.

FLOATING-POINT ARITHMETIC: THE REASON BEHIND THE ISSUE

SAS stores numeric variables in floating-point format. This format is a type of scientific notation in which numbers are represented as values between 0 and 1 raised to a power of 2 or 16 (SAS Institute, 2010, pp.38-39). The use of floating-point format for storing and handling numeric variables allows computers to store large numbers and manage a high level of precision in its calculations with high computational speed.

Floating-point format is thus a convenient way to represent and store real numbers in a computer. However, there are some downsides to it. The first shortcoming of floating-point representation is that some numbers cannot be represented exactly. Among these numbers are most decimal fractions (SAS Institute, 2010, p. 42) and numbers that are larger or smaller than those that can be represented (Montgomery, 2008, pp. 7-8). The resulting imprecision can cause problems with computations and comparisons. The second problem of floating-point representation is the erroneous calculation of the result of divisions by zero. The correct result is undefined. However, this result is represented as well defined numbers instead. The third problem consists of invalid operations, which result in NaNs (Not a Number) (Montgomery, 2008, pp. 8).

Gorrell (2008), Go (2008), Montgomery (2008), SAS Institute (2007) and SAS Institute (2010), among other authors, have amply discussed this topic. These authors provide more detailed explanations of the floating-point format and detailed examples of how it can affect calculations.

To overcome the imprecision in the storage and calculation of numerical values caused by floating-point representation, several authors have proposed some strategies such as:

1. The use of integers rather than values with decimals for calculations. This recommendation stems from the fact that integers are not affected by floating-point representation (Go, 2008, p. 3; SAS Institute Inc., 2007, p. 10). The strategy consists of multiplying variables that contain observations with decimals by 10n -where n is the power with which all decimal values are converted to integers-, do the calculation and then, divide results by the same10n to get the original values back.

   I find this approach useful as it certainly avoids the possible imprecision caused by floating-point representation and allows programmers to stay in control of their data. However, it can be cumbersome particularly when several calculations are made in the same programme. Moreover, it can increase the use of disk space if values converted to integers are stored in new variables.

2. A second recommendation is to use functions such as ROUND (Gorrell, 2008, p.60, Go, 2008, p. 4 and SAS Institute, 2007, p.10) and TRUNC (SAS Institute, 2010, p.45) in specific steps of the calculation process where you suspect that floating-point representation can cause an issue.

   The ROUND and TRUNC functions reduce numbers' length. In doing so, these functions facilitate comparisons of values in which small decimal differences might have introduced due to floating-point representation. However, the reverse is also true. These functions can eliminate small but real differences in decimal values as a consequence of the reduction in the number of significant figures in decimal numbers.

   Moreover, if these functions are applied to the working data set directly, instead of creating a derived data set from it for the sole use of the comparison, and if variables are overwritten, the values are permanently modified from that point onwards. This can lead to incorrect calculations down the process. Therefore, it is advisable to create new variables for rounded values.

3. Perform "fuzzed comparisons". This method is related to the previous point because it is also about rounding. The method has two variants. The first one consists of creating code to specify that if the values to be compared are close enough, they should be considered equal (Montgomery, 2008, pp. 17-18). The second variant is the use of SAS functions such as FUZZ or other functions that contain implicit fuzzing (ceil, floor, etc.) (Go, 2008, p. 6) with the same purpose. According to SAS Institute Inc (2011) "the FUZZ function returns the nearest integer value if the argument is within 1E-12 of the integer (that is, if the absolute difference between the integer and argument is less than 1E-12). Otherwise, the argument is returned". The process of fuzzing comparisons does not require that new variables be created and does not modify the data set. I see these two points as great advantages of this method. The strategy described in this paper is a fuzzed comparison.

THE FIX

The initial code to create confidence intervals was

```
 data x;
    set y;
/*(the following is for non-estimable intervals)*/
    if lowerlimit= . or upperlimit = . or (lowerlimit = upperlimit) then ci =  "NE";
/*(the following is for estimable intervals)*/
else if lowerlimit ^= upperlimit then ci1 =
"("||compress(put(exp(lowerlimit),8.1))||',
'||compress(put(exp(upperlimit),8.1))||')';
run;
```

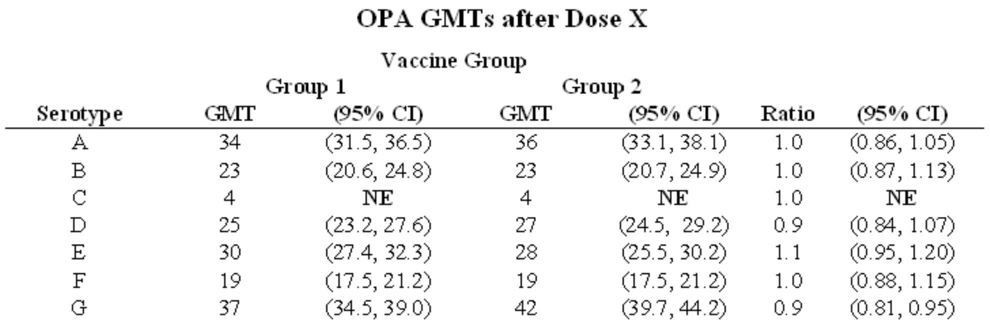
In words, to calculate confidence intervals, the lower limit was subtracted from its corresponding upper limit. If the lower and the upper limits were different from each other, the interval should be displayed as (lower limit, upper limit). If both values were equal, the interval should be displayed as "NE". This code created the output showed above in figure 1. Confidence intervals of serotype C are not correct.

The solution was found assuming that SAS was storing the values of the upper and the lower limits as different by a tiny amount. Under this scenario, the code was modified to fuzz the comparison and to make the limits of the confidence intervals be equal whenever minimal differences existed.

The code was modified as follows:

```
data x;
   set y;
/*( the following is for non-estimable intervals)*/
   if lowerlimit= . or upperlimit = . or (lowerlimit = upperlimit)
   or (abs(upper - lower) le 0.00000000001) then ci =  "NE";
/*( the following is for estimable intervals)*/
else if lowerlimit ^= upperlimit then ci1 =
"("||compress(put(exp(lowerlimit),8.1))||',
'||compress(put(exp(upperlimit),8.1))||')';
run;
```

The new code ignored any difference smaller or equal to 0.00000000001 between the lower and the upper limit. After adding this term, the value of the confidence interval was displayed correctly as "NE", as shown in figure 2.



OPA GMTs after Dose X

	Vaccine Group					
	Group 1		Group 2			
Serotype	GMT	(95% CI)	GMT	(95% CI)	Ratio	(95% CI)
A	34	(31.5, 36.5)	36	(33.1, 38.1)	1.0	(0.86, 1.05)
B	23	(20.6, 24.8)	23	(20.7, 24.9)	1.0	(0.87, 1.13)
C	4	NE	4	NE	1.0	NE
D	25	(23.2, 27.6)	27	(24.5, 29.2)	0.9	(0.84, 1.07)
E	30	(27.4, 32.3)	28	(25.5, 30.2)	1.1	(0.95, 1.20)
F	19	(17.5, 21.2)	19	(17.5, 21.2)	1.0	(0.88, 1.15)
G	37	(34.5, 39.0)	42	(39.7, 44.2)	0.9	(0.81, 0.95)

Figure 2. Final version of the table

The strategy used in this case study is one of several that could have been used. It was decided empirically but analyzing it retrospectively, I consider that it is a good option. By using absolute values, it accounted for both situations where mean 1 < mean 2 or mean 2 > mean 1. Also, since values were recognized as different only when the lower and upper limit were different by a value larger than 0.00000000001, the strategy adequately ignored minimal differences in the lower and upper limits of all confidence intervals of serotype C. Lastly, this strategy did not involve the creation of additional variables, nor did it replace all values with new ones. Therefore, the rest of the program was not affected by the additional code.

RECOMMENDATION

If you suspect that you might be going through a situation like the one described in this paper, you can start by examining the data set from where you are calculating the confidence interval limits to check if all values are equal. This can be done by sorting values and checking if the first value is equal to the last one. If they are, you can apply the solution here described or any other that you find useful.

CONCLUSION

This paper analyzed a particular case in which the floating-point representation that SAS uses to store numerical values caused a numeric precision and representation issue. This issue consisted of a small inaccuracy in the calculation of the lower and upper limits of confidence intervals calculated from a group of observations all with the same value. This led to the incorrect representation of a non-estimable confidence interval in an efficacy table.

The interval was incorrectly represented because the upper and the lower limits were calculated as slightly different values, despite being equal.

The solution, a fuzzed comparison as described in SAS Institute Inc. (2007), consisted of adding a bit of code that made both limits of the interval equal despite having a tiny numerical difference.

REFERENCES

Altman DG, Gardner MJ. (1988) Calculating confidence intervals for regression and correlation. *BMJ*. 296:1238–1242.

Bethea, R. M. (1984). *Statistical Methods for Engineers and Scientists*. New York, NY: Marcel Dekker.

Go, I. C. (2008) “Rounding in SAS ®: Preventing Numeric Representation Problems”. *Proceedings of the 17th Annual Southeast SAS Users Group Conference.* http://analytics.ncsu.edu/sesug/2008/PO-082.pdf.

Gorrell, P. (2008). Numeric length in SAS®: a case study in decision making. *Pharmaceutical Programming*, 1(1), 56-64.

Montgomery, N. (2008). Floating Point error – what, why and how to!!. *Proceedings of the Pharmaceutical Users Software Exchange 2008.* http://www.phusewiki.org/docs/2008/PAPERS/CS08.pdf

SAS Institute Inc. (2007). TS-654: *Numeric Precision 101*. Cary, NC: SAS Institute Inc. Retrieved from http://support.sas.com/techsup/technote/ts654.pdf

SAS Institute Inc. (2010). *SAS® 9.2 Language Reference: Concepts.* (2nd ed.). Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2011). *SAS ® 9.2 Language Reference: Dictionary*. (4th ed.). Cary, NC: SAS Institute Inc. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245897.htm

Sullivan, L. (2012, September). *Confidence Intervals.* Retrieved from http://sph.bu.edu/otlt/MPH-Modules/BS/BS704_Confidence_Intervals/BS704_Confidence_Intervals_print.html

ACKNOWLEDGMENTS

I would like to thank James Trammel for opening the discussion about the floating-point representation, which made me research into this topic and ultimately develop this article.

Thanks to Diane Voran for including me in the study she led, where the case study presented here took place, and for her patience during the process of finding the solution

Also, thanks to Manuel Sandoval, for guiding me though the process of finding a pragmatic solution to the situation presented, and thanks to Ana Mendez, Eric Pedraza, Nancy Brucken and Manuel Sandoval again, for your helpful feedback.

CONTACT INFORMATION

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

Name: Claudia Jimenez-Castro
Enterprise: inVentiv Health Clinical
Address: Insurgentes Sur 716, 4th Floor, Del Valle
City, State ZIP: Mexico, D.F., Mexico City, CP 03100.
Work Phone: (52 722) 319 2648
E-mail: claudia.jimenez@inventivhealth.com