

Numerical Representation and Rank-Based Statistics: A Cautionary Tale

K. D. Stromberg and T. L. Fox

Abstract

Many rank-based nonparametric statistics such as the Wilcoxon Signed Rank Test or the Wilcoxon Rank Sum Test are computed by ranking the data and assigning any tied values the average rank. Test statistics and p-values are then calculated from the ranked data. A potential problem may arise during the calculation of these statistics since computers can't store some real numbers with infinite precision. Numbers that can't be represented in computer memory exactly are stored with a very small roundoff error. Thus, numbers that appear to be identical in a dataset or on the screen may be stored by the computer with slight differences. By default the SAS system uses the full stored precision of a number to determine whether two values are tied when assigning ranks in both PROCs UNIVARIATE and NPAR1WAY. If a dataset contains tied values among numbers with no exact representation, ranks may be assigned improperly due to roundoff error resulting in an incorrect value of the test statistic and resulting p-value. A simple application of the round function in a datastep prior to calculation of any rank based statistics can remedy this problem.

Introduction:

Many nonparametric statistical methods involve calculation of a test statistic by first ranking the data and then calculating the value of the test statistic and subsequent p-value. Thus, the magnitude of the p-value is based on ranks of the data rather than the actual data values. However, a potential problem may arise during the calculation of these statistics when using a computer since computers cannot always represent numbers

exactly. Therefore, numbers that look identical on the computer screen or in a print out may actually be represented internally by the computer with differences in the 15th or 16th decimal place. Many of the procedures in SAS used to calculate rank-based statistics such as Procs UNIVARIATE, NPAR1WAY, or RANK use the full numerical precision by default when ranking the data. Thus, these procedures may fail to find ties between numbers that the analyst believes are tied when numerical representation errors occur. This results in an incorrect ranking of the data. The influence of this problem on the subsequent p-value depends on the extent of this problem in the dataset. In this paper, we review how computers store numbers and how this can lead to problems when calculating nonparametric test statistics using the Wilcoxon Signed Rank Statistic as an example. Finally, we demonstrate how to correct this problem.

Computer Representation of Real Numbers

Integers can be stored by computers exactly (Press *et al.*, 1992). Thus, the calculation of rank based statistics from integer data poses no problems. However, computers have difficulty storing many non-integer real numbers exactly. Computers store these types of numbers using *floating point* representation. A floating point number is represented in scientific notation as:

$$sM \times B^{e-E}$$

Where s is the sign, M is the *mantissa* or whole number portion, B is the base (2 in the Windows/PC environment), E is the *bias* of the exponent, and e is the exact integer exponent such that $e-E$ determine the actual exponent. On a Windows platform, the bias is 1023 and 8-bytes are reserved for the storage of a single floating point number (see TS-654 and TS-230 for further details). Specifically, 1 bit is used for the sign (s), 11 bits for

the integer exponent E , and 52 bits reserved for the mantissa M . Thus, the byte layout for a 64 bit floating point number in windows looks like:

SEEEEEEE	EEEEMMMM	MMMMMMMM	MMMMMMMM
byte 1	Byte 2	byte 3	Byte 4
MMMMMMMM	MMMMMMMM	MMMMMMMM	MMMMMMMM
byte 5	Byte 6	byte 7	Byte 8

For example, to convert the decimal 35.1 to the floating point representation used by a windows computer first convert the number to a binary scale.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
0	0	1	0	0	0	1	1	0	0	0	1	1	0	0
2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	...		
1	1	0	0	1	1	0	0	1	1	0	0	...		

So the binary representation of 35.1 looks like

100011.000110011001100110011001100... Notice that the integer portion, 35, can be stored exactly (i.e without a remainder). However, the decimal portion 0.1 cannot be represented exactly in a base 2 number system. This is analogous to trying to represent the fraction $1/3$ in the base 10 number system. Next, the decimal is fully normalized or moved over until there is only one digit to the left of it to yield 1.0001100011001100...

The decimal was moved over 5 places so 5 is added to the base to yield the integer exponent value 1028 (i.e. $1023+5=1028$). Converting the exponent portion to binary representation yields 010000000100. Finally, the binary representation is lined up in terms of half bytes with the first bit removed (it is implied since the number is fully normalized) and converted to the hexadecimal scale using the byte layout shown above:

SEEE	EEEE	EEEE	MMMM	MMMM	MMMM	MMMM	MMMM	MMMM	MMMM	MMMM	MMMM	MMMM	MMMM	...
0100	0000	0100	0001	1000	1100	1100	1100	1100	1100	1100	1100	1100	1100	...
4	0	4	1	8	C	C	C	C	C	C	C	C	C	...

yielding 40418CCCCCCCCCD. Note that the final C (1100 in base 2 or 12 in base 10) was rounded up to D in the last half byte indicating that 35.1 cannot be stored exactly and introducing a small roundoff error. Furthermore, this problem can be confounded by the fact that different processors and computer operating systems may represent floating point numbers differently. Computer calculations can also lead to representation errors (for more details and examples see technical support documents TS-654 and TS-230). To view the internal representation of any number stored in a SAS dataset use the HEX16 format.

Numeric Representation and Ranking

Two numbers that appear identical can sometimes be represented internally by the computer with small differences. This phenomenon can arise when computations are involved as outlined in example 1 or when data is read in from multiple sources. In this example two variables with three “identical” observations are read into a dataset. The observations for one variable (x) are read directly into the dataset while the other variable (y) is created through a computation. On the screen or in a report all values of x and y appear identical. However, subtracting the variables indicates that there are small differences in the way the computer is storing each pair of observations.

Example 1:

```
*** CREATING SOME DATA;
data ex1;
  x=0.3; y=3*0.1;      z=x-y; z2=round(x-y,0.0001); output;
  x=0.7; y=7*0.1;      z=x-y; z2=round(x-y,0.0001); output;
  x=0.01; y=0.1*0.1;    z=x-y; z2=round(x-y,0.0001); output;
run;
```

Ex1: Differences in How SAS Stores Floating Point Numbers

Value of x	Value of y	Value of x-y	Value of round(x-y)
0.3	0.3	-5.55E-17	0
0.7	0.7	-1.11E-16	0
0.01	0.01	-1.73E-18	0

Notice that after rounding the difference is zero as expected.

Example 2 demonstrates that SAS uses the full stored precision of numbers to rank the data. In this example, the data from example 1 is rearranged so that all values of *x* and *y* are read into one column containing six observations. In another column, the same six observations are stored after rounding *x* and *y* to the nearest 0.0001. Next, PROC RANK is used to rank the data from each column from 1 to 6.

Example 2:

```

*** REARRANGE THE DATA SO X AND Y ARE IN SAME COLUMN;
*** ROUND X AND Y AND STORE THIS VALUE IN A SECOND COLUMN;
data ex2;
    set ex1;
    origval=x; fixval=round(x,0.0001); output;
    origval=y; fixval=round(y,0.0001); output;
    keep origval fixval;
run;

*** RANK BOTH THE ORIGINAL AND ROUNDED DATA;
proc rank data=ex2 out=ex2_2;
    var origval fixval;
    ranks rorigval rfixval;
run;

```

Ex. 2: Correct and Incorrect Ranking of Floating Point Numbers

Original Value Prior to Rounding	Value After Rounding	Rank of Original Values Prior to Rounding	Rank of Value After Rounding
0.3	0.3	3	3.5
0.3	0.3	4	3.5
0.7	0.7	5	5.5
0.7	0.7	6	5.5
0.01	0.01	1	1.5
0.01	0.01	2	1.5

Note: Proc RANK does NOT identify ties in the original value
After rounding Proc RANK correctly identifies ties in the data.

Notice that since PROC RANK by default uses the full precision of the numbers to assign ranks to the data it fails to find ties in the data and subsequently ranks the data incorrectly. However, after rounding the data to the nearest 0.0001, PROC RANK identifies the ties and ranks the data correctly.

Influence of Numerical Representation on Rank-Based Statistics

In the third example, the Wilcoxon Signed Rank Test computed from Proc UNIVARIATE in SAS will be used to illustrate the effect numeric representation errors can have on rank based nonparametric statistics. Recall that the Wilcoxon Signed Rank Test is a one sample test that tests for a significant shift in the data. In formal statistical terms the null hypothesis tested is:

$$H_0: \theta = 0$$

and asserts that the population distribution from which the data was collected is symmetrically distributed about zero (Hollander and Wolfe, 1999). To compute the test statistic (T^+) any values of zero are first dropped. Next the absolute value of the remaining values are ranked from 1 to N . By convention, values that are tied get the average of the ranks for the tied values. The test statistic is then calculated by summing

the ranks from the data values that were originally greater than zero. The p-value is then computed by comparing the observed value of T^+ to either its exact or asymptotic distribution under the null hypothesis (for the computational details see Hollander and Wolfe, 1999 or the SAS Procedures Manual v.8).

Depending on the nature of the data, numerical representation errors can lead to erroneous values of both the test statistic and p-value. This can happen if a dataset contains values that are supposed to be tied yet are represented by the computer differently. As seen in example 2, this will result in incorrect ranking of the data and hence an erroneous p-value. An extreme example of this is demonstrated in example 3. In this example two variables (x and y) with what appear to be “identical” values are created using two different methods of computation. (However, this could represent data coming from two different sources or computer platforms.) The test of interest in this example is whether the difference ($x - y$) is significantly different from zero at the $\alpha = 0.05$ level. For instance, x and y could represent the value of some laboratory test before and after an intervention. In this case the analyst may want to determine if there is a significant “shift” or change in the laboratory value after intervention.

Example 3

```
*** MAKE 22 OBSERVATIONS OF X DATA;
data xdat;
  retain ct 0;
  do i=1 to 3 by 0.1;
    ct=ct+1;
    x=i;
    keep ct x;
    output;
  end;
  ct=22;
  x=3.1;
  output;
run;
```

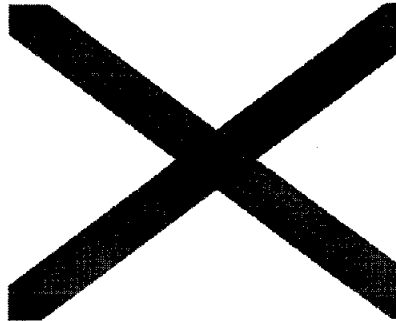
```

*** MAKE 22 OBSERVATIONS OF Y DATA;
data ydat;
    retain ct 0;
    do i=10 to 30;
        ct=ct+1;
        y=i*0.1;
        keep ct y;
        output;
    end;
    ct=22;
    y=3.2;
    output;
run;

*** MERGE THE DATA;
data ex_3;
    merge xdat ydat;
    by ct;
run;

```

A look at the data clearly suggests that all but one observation are identical between x and y suggesting that the Wilcoxon Signed Rank Test should fail to reject the null hypothesis.



However, failing to take into account that both x and y are represented differently by the computer can yield unexpected results as shown below.

```

*** CALCULATE THE DIFFERENCE BETWEEN X AND Y;
*** ROUND THE DIFFERENCE;
data ex_3;
    set ex_3;
    diffval=x-y;
    diff_round=round(diffval,0.00001);
run;

```

```

*** OBTAIN THE WILCOXON SIGNED-RANK TEST STATISTIC FOR
    UNROUNDED DIFFERENCE;
proc univariate data=ex_3 noprint;
    var diffval ;
    output out=nofix signrank=Tstat probs=pval;
run;

*** OBTAIN THE WILCOXON SIGNED-RANK TEST STATISTIC FOR
    ROUNDED DIFFERENCE;
proc univariate data=ex_3 noprint;
    var diff_round;
    output out=fixed signrank=Tstat probs=pval;
run;

```

Example 3

Effect of Numerical Representation on Signed-Rank Statistic

Difference Variable	Value of Test Statistic	P-value
Not Rounded	76.00	0.0010
Rounded	-0.50	1.0000

When the difference is not rounded prior to calculating the Wilcoxon Signed Rank Statistic PROC UNIVARIATE fails to identify some of the ties in the dataset and hence ranks the data incorrectly. This results in a large value of the test statistic and a highly significant p-value ($T^+ = 76.00$, $P = 0.0010$) leading to an incorrect rejection of the null hypothesis. However, when the difference is rounded to the nearest 0.0001 prior to computing the value of the test statistic PROC UNIVARIATE identifies all of the ties in the dataset and ranks the data appropriately. This results in the correct value of the test statistic, p-value ($T^+ = -0.50$, $P = 1.000$), and failure to reject the null hypothesis. Notice how much the p-value and interpretation of the statistical result changed simply due to numerical representation errors.

Conclusions

Although the examples presented in this paper are contrived and represent extreme cases (i.e. p-values ranging from 0.0010 to 1.000 due only to numerical

representation errors) this phenomenon can occur to a lesser extent in practice; particularly when data come from multiple sources [authors' personal experience]. Thus, we recommend that analysts and statisticians make sure they fully understand their data prior to performing any rank-based nonparametric statistics. Furthermore, if the data are continuous or data values have been manipulated prior to analysis in a data step we suggest that the data be rounded prior to analysis to avoid potential erroneous results.

References

- TS-230 "Dealing with Numeric Representation Error in SAS Applications." 1992.
<http://support.sas.com/techsup/technote/ts230.html>.
- TS-654 "Numeric Precision 101." 2001.
<http://support.sas.com/techsup/technote/ts654.html>.
- Hollander, M. and Wolfe, D. 1999. Nonparametric Statistical Methods. 2nd ed. John Wiley and Sons, Inc. New York.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 1992. Numerical Recipes in C: The Art of Scientific Computing. 2nd ed. Cambridge University Press. New York.