
Assessing the Computational Accuracy in Statistical
Packages: A Note On the Use of Mathematica

Steve Hunka

Centre for Research in Applied Measurement and Evaluation
University of Alberta

Centre for Research in Applied Measurement and Evaluation
University of Alberta, Edmonton, AB, Canada

Research Report No. CRAME 99-01

August, 1999

Assessing the Computational Accuracy in Statistical Packages: A Note On the Use of Mathematica

Steve Hunka, Professor Emeritus
University of Alberta

The National Institute of Standards and Technology (NIST) in the U.S. has as its objective to "improve the accuracy of statistical software by providing reference to data sets with certified computational results". In support of this goal the Statistical Engineering and Mathematical and Computational Sciences Division of NIST's Information Technology Laboratory provides data sets through their web site which can be used to check the accuracy of algorithms in areas of analysis of variance, linear regression, nonlinear regression, and univariate summary statistics. The web site can be accessed by the address www.nist.gov/itl/div898/strd.

In the process of preparing a couple of papers (McCullough, 1998, 1999) on the accuracy of SPSS, SAS, and S+ statistical packages, Bruce McCullough, a senior economist with the Federal Communications Commission, Washington, D.C., contacted the author with regards to the author's analysis of variance program written using Mathematica. In his testing of the program he found that the program did as well as SPSS and SAS, but not as well as S+, and that for the computationally more difficult data sets provided by NIST, 0 accuracy existed. (For an evaluation of Excel, see McCullough and Berry, in press.) McCullough's 1999 paper reports on the accuracy of SPSS, SAS, and S+ for computations involving univariate statistics, analysis of variance, linear regression, non-linear regression, random number generation, and some common statistical distributions. Instructors who use these packages in laboratory assignments would be well advised to incorporate McCullough's results to demonstrate some of the inaccurate computations produced by statistical packages, and to counter the attitude that "if the computer produced the results it must be correct!". Those researchers working in the area of Item Response Theory and differential item functioning will find McCullough's comparisons of SPSS, SAS, and S+ for non-linear regression particularly useful.

This note reports on how Mathematica's capabilities for maintaining rational computations were used to provide an analysis of variance program that provides sums of squares with a high degree of accuracy, and exceeding the accuracy of certified values provided by NIST.

NIST Anova Data Sets

Data sets provided by NIST are ordered by "their level of difficulty (lower, average, and higher) according to their stiffness - the number of constant leading digits". Data sets have 1, 7, or 13 leading digits, and 21, 201, or 2001 observations per cell. The data sets having a "higher" level of computational difficulty provided by NIST, and which are of interest here, were generated by an earlier research project by Simon and Lesage (Simon & Lesage, 1988, 1989) and simply referenced by NIST as Simon 1 to 9.

The data set Simon 7 contains data for a 1-way anova with 9 groups. There are 21 observations in each cell (N=189) and all the dependent variable data has the form 100000000000.n, where n is a single digit which varies between 2 and 6. Simon 8 contains the same pattern of data but N=1809 with 201 observations per cell, and Simon 9 has N=18009 with 2001 observations per cell. Of course, such data is unlikely to be encountered by most researchers, however, the use of such data provides a test of the quality, i.e., computational accuracy, of an algorithm used to compute the sums of squares.

In order to gain some appreciation of the problem of computational accuracy consider calculating the sum of squares for a set of values of the form 100000000000.n by representing such values as $(100000000000 + 0.n)^2$. Squaring a single datum, and keeping in mind that $(a+b)^2 = (a^2 + 2ab + b^2)$, and letting the integer $a=100000000000$ and $b=0.2$, a^2 is of the order of 10^{24} and b^2 is 0.04. Rough calculations indicate it would take 80 binary bits to just represent a^2 because

$2^{80} = 1.209... \times 10^{24}$. If there were 1000 such observations we would have a sum of the order 10^{27} which would require more than 80 binary bits. In addition to representing the integer portion of our value, we also need to consider how the value 0.04 is to be stored...we shall see later that it takes an infinite sequence of binary digits to precisely represent some real values less than 1. Thus, in this example the number of binary bits required to represent the sum of squares exceeds the usual word length in most computers and requires specialized software and appropriate algorithms to ensure that accuracy is maintained.

NIST uses a special Fortran extended package for their calculations providing for 500 digits during computation and then rounding back to 15 digits. The certified values for Simon 9 are given as follows:

```
SS treatment=1.6008000000000000E+02
SS error=1.8000000000000000E+02
F=2.0010000000000000E+03
```

with corresponding mean squares and R^2 given in the same format.

The Computational Problem

The problem is to find the deviated sum of squares in a manner which will reduce to a minimum the following sources of computational error:

- 1) inexact binary representation of a real value, e.g., the real value 0.1 requires an infinite sequence of binary digits for an accurate representation
- 2) rounding error, e.g., in squaring a value or in adding a very large number to a very small number there may not be sufficient binary digits in the computer's word length to accommodate the result

NIST correctly notes that to improve computational accuracy the subtraction of the leading constant from all observations in their data sets is a "remedial" measure. Of course, this procedure may not reduce the maximum number of leading digits in a real data set. For a more detailed discussion of errors in numerical computation, see McCullough (1998).

Assessing Accuracy

For the NIST anova data sets, the assessment of accuracy is made on the F statistic by using the following equation:

$$\text{LRE} = -\log_{10}[\text{Abs}[f-F]/F]$$

where f is the program's calculated F value, and F is the NIST certified value. The index LRE = 1.4 when $f=15.333$ and $F=15.9467$. When f equals F , the result is taken to be the number of digits in F .

Approaches to the Problem

The usual computational formula for calculating SS given in many textbooks, of course, will not work because of the very large values encountered when each value is squared and summed over all observations. Squaring observations deviated about the grand mean is an improvement, but does not solve the problem because, for example in Simon 9 one still has to represent in a finite binary word length the integer portion (100000000000) plus the decimal portion, say, 0.1 which is represented by the binary value 0001100 with the underlined portion repeating indefinitely. Since the infinite sequence cannot be represented in a computer with finite word length, a truncation results in the binary value which does not translate to exactly 0.1. This error is

propagated throughout the computations also. One could, of course, subtract the constant 1000000000000 (base 10) from each observation leaving only the single decimal digit to be entered to the program, however, this is not a solution when the leading integers vary and would not provide an adequate test of an algorithm's computational quality.

Another solution to the problem is one that Kyung Bay used in the old DERS and XDER programs at the University of Alberta many years ago. This algorithm keeps a "running" mean and sum of squares as each observation is processed, thus SS is a minimum while handling any datum. (This procedure is useful for reading and processing sequentially each record in a large data set stored on tape or disk.) The algorithm is as follows, where m is the mean, and ss the sum of squares:

```

m=d(1)
ss=0.
Do over j=2,N
  a=(d(j)-m)^2
  ss=ss+a-(a/j)
  m=m+(d(j)-m)/j
end Do

```

A few experiments with this algorithm indicates that although it works quite well, the final accuracy, although quite good, depends upon the order in which the data is input. A plot of SS, for different orderings of the data, as the algorithm proceeds clearly indicates this phenomenon. It will do considerably better than the algorithm SPSS and SAS use. (Users indicate that SPSS Manova is more accurate in calculating sum of squares than the univariate calculations.)

Using Mathematica

Another approach is to increase the precision of Mathematica's calculations by using the operator SetPrecision which allows extra binary digits to be used during calculations. (Mathematica has a host of other special operators for assessing things like the smallest and largest number your system can handle.) Mathematica normally proceeds with double precision in calculations. Using this approach, accuracy exceeding those of SPSS, SAS, and S+ were obtained....but still not to a satisfying level.

A simple solution using Mathematica is suggested by the fact that the NIST group used extended precision. Thus, if the input is given in the format which includes additional zeros following the single decimal digit, e.g., 1000000000000.200000000, Mathematica exceeded 15 digits of accuracy on Simon 9. Adding the extra zeros tells Mathematica that the fractional part 0.2 is exact to the 9th decimal point rather than to a single decimal point. Entry of additional zeros would not be appreciated by those entering the data, and most statistical packages would ignore these trailing zeros anyway. (Mathematica keeps an internal representation of such values with a "marker" identifying the precision.)

The final solution was to note more closely how Mathematica handles integers and real values. Since Mathematica can do an operation like $(a/b) \times (b/c)$ in symbolic form, it also handles integers in exactly the same manner, i.e., cancellation and simplification is automatically carried out. Thus, if all the data is entered in integer form, all calculations proceed in rational form (in the form of a/b where a and b are integers), including SS and MS, although for output one might convert the rational value to real form. The problem then is reduced to (a) identify the nature of the data input by the user, i.e., is it all in integer form, real form, or of mixed type, and (b) if in real form, transform it to rational form. If the data is in mixed form, one could use Mathematica to transform the real values to rational form. In my Mathematica program the latter was not done... I felt the interesting part of the problem had been solved! Mathematica has the operator IntegerQ[] for determining whether a value is an integer or not.

A single line of code (a pure function) using Mathematica can be used to transform real values to a simple rational form. For Simon 9 where $N=18009$, it takes about 45 seconds on a Mac 7200/120. In rational form each real value datum is represented as the ratio of two integers of high precision. When a datum can be exactly represented as a rational value V , the operator $N[V,1000]$ will give the real value accurate to the 1000th decimal point! Once the data is transformed to rational values, all calculations proceed in rational form, including $(X'X)^{-1}$, $B'X'XB$, $(K'B)' [K'(X'X)^{-1}K]^{-1} K'B$ used in forming SS for contrast K . Since $X'X$ contains integer values, the inverse is in rational form, B is also calculated in rational form, and K is usually in integer form. (For values such as 0.5 in a contrast matrix one would enter 1/2.)

The result for Simon 9, as well as other NIST data sets, and the Mathematica program are available at the site:

www.mathsource.com/cgi-bin/MathSource/Applications/Statistics/0208-943

Readers may download the Mathematica file containing the program and results for various data sets, and the MathReader (for PC or Mac) which permits opening the file for viewing only. (Those who have the Mathematica software system on their computers need not download the MathReader.)

For the Simon 9 data set, the following results are obtained:

Sample Variance: 4251/225100
 SStotal cfm: 8502/25
 SSmodel cfm: 4002/25
 Model R-sqd: 667/1417
 SSerror: 180/1
 (*cfm is read as "corrected for the mean")

Notice that the sample variance is given as $(4251/225100)$ which can be divided out to your heart's desire to get as much precision you want in a real value. $SStotcfm=(8502/25)=340.08$ exactly. The full model $R^2 = (667/1417)$ is exact, but its real approximation $(0.470712773\dots)$ is not. The real values in the Anova Table are exact values with Mathematica suppressing the zeros following the last digit if all are 0. The Anova table provided by the program has the following results:

Source	SS	Df	MS	F
SS _{TOTCFM}	340.08	18008	.0188849	
SS _{MCFM}	160.08	8	20.01	2001
SS _{Error}	180	18000	0.01	

Rather than output of rational values (which can be done in Mathematica) the Anova table shows a conversion to real form. In the table above the value of $SStotcfm$ is exactly 340.0800000.... with the trailing 0s suppressed for table formatting. Similarly, the F value of 2001.00000000.... is exact with trailing 0s suppressed. The MS of .0188849 is a real value approximation to $340.08/180$.

The rational values for Simon 7,8, and 9 were sent via e-mail to the standards group at NIST with the suggestion that their certified values would be more accurate if reported in rational form as provided by Mathematica. They kindly acknowledged the accuracy and noted that it was easier for them to automate testing of many data sets using an extended Fortran package, but would consider Mathematica with any revision of their www site information.

Rational Approximations and Calculations

In making the rational calculations in Mathematica only one line of special code was required to ensure that real values input by the user were presented to the program in rational form:

```
getrn[d]:=Map[Map[Sign[#]*FromDigits[RealDigits[#]&,&#&,d];
```

The operation `RealDigits` takes the real-valued number apart, `FromDigits` is like the inverse operation of `RealDigits` but gives a rational value, and `Sign` maintains the proper sign of the value. (Notice, like APL one tries to avoid Do Loops used in Fortran. One has to think of numerical processing as functional operations.) The `Map` operation is one of the most powerful in Mathematica, as it maps a function over an object (like data or an expression). Following the deposition of the Anova program with the MathSource site, it was found that Mathematica provides an operator to carryout the conversion of any real value to rational form:

```
Rationalize[x,0]
```

This operator is about five times faster than the use of the function `getrn`. Wolfram (1996) notes in *The Mathematica Book* that when the second argument in the operation `Rationalize` is 0 (a tolerance value), Mathematica "yields the best possible rational approximation given the precision of your input" (page 700). `Rationalize[0.7,0]` produces `7/10`, which suggests that multiplying the numerator by 10^n and dividing by 10^n where $n=1$ will produce the rational value `7/10`. Thus any real value of finite length can be converted in this manner providing that sufficient binary digits are available for storage of 10^n . Mathematica can store integers of any length providing your computer has the available memory.

If the value `E` is output in real form, Mathematica will display `2.71828` but the value actually being stored is `2.71828182845904509` (an approximation to `E` using 18 digits). If the latter real value is rationalized we get `353613854/130087267`. If this latter rational value is calculated to 30 decimal digits and compared to `E` calculated to 30 decimal digits, the error is of the order 1.23743×10^{-16} . If `Pi` is calculated to 40 digits, then rationalized we get the rational `265099323460521503743 / 84383735478118508040` which when divided out to 40 digits will show no error. In the latter example 40 digits of accuracy is maintained throughout. However, this rationalized value of `Pi` is not exactly equal to `Pi`.

Now consider an inversion of a simple matrix first with the elements given as real values, and then given as rational values. Let `A` be the matrix in real value form.

$$A = \begin{bmatrix} 1.0 & 0.3 \\ 0.3 & 1.0 \end{bmatrix}$$

Inverting this matrix using Mathematica we get the output results `{{1.0989, -0.32967}, {-0.32967, 1.0989}}` where each vector delimited by `{` and `}` represent row 1 and row 2 of the inverse. If we ask Mathematica to show us the full internal representation of element (1,1) we get `1.09890109890109899`` with the symbol forward slanted single quotation mark (```) identifying that the value is a machine-precision number. If we ask Mathematica to tell us the precision of this value the result is 16.

Now consider the inversion of the matrix `A` in which the elements are given in rational form. Let `B` be the matrix in rational form. (The first element can be considered the rational `1/1`.)

$$B = \begin{bmatrix} 1 & 3/10 \\ 3/10 & 1 \end{bmatrix}$$

Mathematica will give the inverse of `B` as `{{100/91, -30/91},{-30/91, 100/91}}`. The element in position (1,1) is `100/91` a rational value. If we ask Mathematica to give us the precision of `100/91`, the result is `Infinity`. If we ask Mathematica to give us a real value for `100/91` to a maximum length of 1000 digits we get `1.098901` with the underlined portion repeating, i.e. `1.098901 098901098901 099` in which the last digit has been rounded from an 8 to a 9. Thus, strictly

speaking the matrix A is not equal to the matrix B , and the inverse of A is not equal to the inverse of B . The inverse of B has infinite accuracy; the inverse of A does not. Thus, when Mathematica maintains rational values throughout its calculations, the results have infinite precision. As illustrated earlier, for computations involving real values of infinite length, such as E and π , and when these constants do not cancel out, rational values cannot be maintained throughout the computations except in symbolic form, e.g., 2π , πE .

In the above discussion some simple examples have been used to illustrate the problems associated with computational accuracy. The topic is not a simple one, however, as it belongs to the domain of numerical analysis and number theory ... specialized branches of mathematics.

Considering that Mathematica can do numerical, symbolic (e.g., calculus), and graphical processing, it is truly an outstanding software system. It is not, however, a data analysis package! As with writing any software, one needs to have the appropriate set of operators (the computer language) and to devise appropriate and efficient algorithms.

The Reporting Problem: An Opinion

Over the last decade the media has given extensive coverage to research studies which have employed statistical analyses. Frequently, these studies are reported by professionals having expertise in areas other than statistics, or by special interest groups who wish to impress the importance of their cause upon the public. Media reporters tend to report results as if they were "the absolute truth", allowing the professional competence of the researcher to automatically justify statistical competence, and never question the accuracy of the programs used, the validity of the numerical models, the reliability of the data, or the appropriateness of their assumptions. It has also been noted that some researchers have technicians skilled in setting up program packages to run their data analyses, and thus may miss critical warning or error messages...if even given by the statistical package. Furthermore, some researchers are overly influenced in assessing the importance of their results by concentrating on the "P-value" or "odds-ratio", ignoring the importance of the degree of relationship between the dependent and independent variables. When these factors are added to the problems of experimental design, and to the fact that we frequently do not have a perfectly linear "ruler" with which to make our measurements, there is little wonder why the public can be confused and misled by the results of statistical studies.

References

- McCullough, B., (1998), Assessing the Reliability of Statistical Software: Part I, The American Statistician, November 1998, Vol. 52, No. 4, p359-366.
- McCullough, B., (1999), Assessing the Reliability of Statistical Software: Part II, The American Statistician, May, 1999, Vol. 53, No. 2, p149-159.
- McCullough, B. & Wilson, B., (in press) On the accuracy of statistical procedures in Excel, Computational Statistics & Data Analysis.
- Simon, S.D., & Lesage, J.P., (1988) Benchmarking Numerical Accuracy of Statistical Algorithms, Computational Statistics and Data Analysis, 7, p197-209.
- Simon, S.D., & Lesage, J.P., (1989) Assessing the Accuracy of ANOVA Calculations in Statistical Software, Computational Statistics and Data Analysis, 8, p325-332.
- Wolfram, S. (1996), The Mathematica Book, Wolfram Media Inc., Champaign, Ill.