Statistica & Applicazioni
Vol. III, n. 1, 2005
pp. 37-58

On the accuracy of three statistical softwares

Angelo M. Mineo[§]

Rino Richiusa[§]

Summary: *In this paper we compare the accuracy of three packages that are commonly used for statistical calculations: Excel of Microsoft, version XP Edition 2003, Statistica of Statsoft, version 6, and R, an open-source free software, available on the web, version 1.9.0. To assess the accuracy of each software in different statistical areas, we are going to use benchmarks expressly developed for this aim. The obtained results show a superiority of R in comparison with the other two softwares.*

Keywords: *benchmark, univariate statistics, linear and nonlinear regression, probability distributions.*

1. Introduction

In the era of computers, Statistics has advanced very much in both applied and methodological fields. In the first case, the great development of Statistical softwares makes easy data analysis (especially when we have a lot of data) or makes feasible application of particular methods that would be unusable without a computer (as, for example, resampling methods). In the second case, some Statistical methodologies, such as simulation studies, could not be used without the help of computers. However, we can ask ourselves if statistical software reliability can be taken for granted in any case. This problem depends on the computer finiteness, i.e. on the fact that computers deal with numbers with a finite precision, and on the quality of the implemented algorithms. The present paper compares, for the first time, three widely used statistical softwares: Excel of Microsoft, version XP Edition 2003, Statistica of Statsoft, version 6, and R, an open-source free software, available on the web, version 1.9.0. Although Excel is not a real

[§] Dipartimento di Scienze Statistiche e Matematiche "Silvio Vianelli" – Università degli Studi di Palermo – viale delle Scienze, 90128 PALERMO (e-mail: elio.mineo@dssm.unipa.it).

statistical software, this choice is due to its widespread use among statisticians.

There are other similar papers in the statistical literature. Longley is one of the first researchers that has dealt with this problem; indeed, in 1967 he published an article on the reliability of some software for the estimation of linear regression parameters. Other articles followed (see, for example, Wampler, 1970, or Beaton, Rubin and Barone, 1976). Recently, researchers have dealt with comparisons about accuracy of well known statistical softwares, or softwares used for statistical computation (see, for example, Sawitzki, 1994, Knüsel, 1995, 1998, McCullough, 1998, 1999, McCullough and Vinod, 1999, McCullough and Wilson, 2002).

In the first section of this article we are going shortly to show the kind of errors that might arise when we use a computer, then to compare the three considered softwares by using some batteries of tests on univariate statistics, analysis of variance, linear and nonlinear regression, pseudo-random number generators and on the computation of some quantities of a probability distribution. In order to make this comparison, we are going to use *benchmarks* expressly developed for this aim; the purpose of benchmarking is to assess the quality of the implemented algorithms. With given inputs and correct outputs in hand, inputs are given to the program and outputs are checked versus the correct results.

2. Errors made by a computer

Usually, a user thinks that a computer is extremely safe in speed and in accuracy of the numerical results. Nevertheless great differences exist between the arithmetic of a computer, which is denoted as “artificial”, and the usual one, which is denoted as “natural”. These differences are evident in the final results obtained from the used procedures that show some errors. Some of these errors are inevitable, others not. To a certain extent, all computers produce “incorrect” results, mainly due to two kinds of errors: rounding errors and truncation errors. Both of them are due to two basic characteristics of a computer: its binary number representation and its finite storing capacity.

Indeed, real numbers can not always be represented exactly in a computer. A computer can only store, as a string of zero and one, numbers having a finite amount of digits. Moreover, it is easy to show how a number which has a finite decimal representation could have a no-finite binary representation, so that it could not exactly be represented in a computer. As an example, the decimal number 0.1 has a periodic binary representation: $0.0\,\overline{0011}$; because of the finite computer storage, the decimal 0.1 will be stored as 0.00011001100110011001100110, if the computer uses three bytes

to represent this number; when this number is reconverted in base-10, it becomes 0.09999999403953. In a more general way, let $fl:\nabla\rightarrow\Im$ be the function which yields the floating point binary representation of $x\in\nabla$, then $fl(0.1)$ = 0.09999999403953, while, for example, $fl(10)$ = 10. Generally, neither it is true that $fl(x) = x$, nor that if $x\in\nabla$ then $x\in\Im$.

Let's define the rounding error as the difference:

$$fl(x + y) - (x + y) \tag{1}$$

with $(x, y)\in\nabla$. For two numbers x and y, if it's true that $x\in\nabla$ then $x\in\Im$ and if $y\in\nabla$ then $y\in\Im$, it is not always true that $(x+y)\in\Im$; therefore, we can only say that $fl(x + y)$ is the number in $\Im$ that is closer to the quantity $(x+y)\in\nabla$. The more the number of operations increases, the more the rounding error increases.

In a more formal way, we can define the rounding error as that α satisfying

$$x + y = fl(x + y)\cdot(1+\alpha) \tag{2}$$

The number α is unknown; we can only know the bounds of α. For example, if a computer uses ANSI/IEEE standard 754 for binary floating-point arithmetic (1985) and it stores number using a double precision (eight bytes), it is $|\alpha|<2^{-53}$. If we knew α, we could easily delete the rounding error, simply re-computing the value of the computer by using (2). Nevertheless, it is hardly obtainable for at least two reasons (for more details see Wilkinson, 1965):

- firstly, because of the difficulty that would involve the calculation of possible correlations among the different rounding errors, if we had to do a sequence of operations: for example, if we had to do $xy+z$, then $xy+z = [xy(1+\alpha) + z](1+\beta)$,
- secondly, because the rounding error distributions are not known; i.e. the representable numbers in $\Im$ are not uniformly distributed. In general, the rule, according to which operations involving greater numbers have an higher predisposition to the rounding error, is valid.

One type of rounding error, which is worth being mentioned, is the *cancellation error*. It occurs when two almost equal numbers are subtracted from each other.

Rounding error is not eliminable, but it can be reduced by increasing the number of storing bytes. However, this is not a general rule: for example, Longley (1967) showed that it is not necessarily a consequence that double precision will result in greater accuracy, focusing hence on the details of the algorithm.

Besides the rounding error, there is the *truncation error*. This error depends on software and may be considered an approximation error. The way a software represents a function is by using an expansion in series of this function that necessarily has to be stopped after a finite numbers, let's say *k*, of terms. The difference between the true value of the function and that achieved by summing up the first *k* terms of the series represents the truncation error.

Rounding error and truncation error are strictly correlated to the specific characteristics of computers, and therefore not eliminable; by the way, these types of errors could be amplified, as we have said above, by a bad choice of the algorithm. For example, if we want to sum up the squares of the first 10000 positive integer numbers, we can start from 1 by following an ascending order or we can start from 10000 by following a descending order: it can be shown that in the first case the numerical error produced by a computer is 650 times greater than the error produced in the second case (Dahlquist e Bjorck, 1974).

In general, it is correct to say that such errors do not offset each other, rather they sum up. The final cumulative error may be an increasing function of the number of operations involved in the algorithm. So, it is clear how it is important to choose an algorithm which reduces the number of operations in order to solve a problem.

However, the main purpose of a programmer is to implement procedures that minimize the amount of errors produced by the "artificial" arithmetic; if this aim can be reached with the use of algorithms that need more operations, giving the most "adherent" results to the real solutions, these algorithms are to be preferred, anyway.

Other problems are the overflow and the underflow phenomena. They are much known to computer users so that we shall not deal with them in this paper.

3. The used benchmarks

A "benchmark" is a process to assess the accuracy of algorithms implemented in a software. In this paper the following benchmarks will be used:

- StRD (Statistical Reference Datasets), a benchmark produced by the American NIST (*National Institute for Standards and Technology*), in order to assess the accuracy of the algorithms implemented in four specific statistical areas: univariate statistics, analysis of variance, linear and nonlinear regression;
- Marsaglia's DIEHARD for pseudo-random number generators;

- Knüsel‘s ELV (acronym of ELementare Verteilungen, i.e. elementary distribution) for the accuracy of probability distributions.

A description of the previous three benchmarks follows.

3.1 *StRD*

StRD is a battery of tests ordered by ascending level of difficulty (lower, average and higher). In the NIST’s web site, where it is possible to download the battery of tests, there is this useful information on the meaning of the benchmark results: “These levels are merely provided as rough guidance for the user. Producing correct results on all datasets of higher difficulty does not imply that your software will pass all datasets of average or even lower difficulty. Similarly, producing correct results for all datasets in this collection does not imply that your software will do the same for your particular dataset. It will, however, provide some degree of assurance, in the sense that your package provides correct results for datasets known to yield incorrect results for some software”. Besides datasets, NIST provides the correct results, calling them “certified values”. These values have to be compared with those yielded by the statistical software. These values can be considered “correct”, according to the way they have been computed; indeed, for linear problems, data were read in and represented with 500 decimal digits of accuracy. Simple algorithms, coded in FORTRAN, were used to perform each analysis. The final 500-digit results were rounded to 15 significant digits.

The nonlinear problems were solved by obtaining results with a 128-bit precision, confirmed by at least two different algorithms and software packages using analytic derivatives.

Every time we have looked for the best implementation to maximize the accuracy; in particular, firstly we have looked at the convergence criterion (residual sum of squares or square of the maximum of the parameter differences); secondly, at the tolerance (from a minimum of 1*E*-6 to a maximum of 1*E*-36); thirdly, at the solution method (e.g., the Gauss-Newton method or the optional Levenberg-Marquardt one), and finally at the use of numerical or analytic derivatives. Certified values were obtained by rounding the final solutions to 11 significant digits.

To compare the certified values c with the provided software’s output x, we can define the absolute error and the relative error. The absolute error is the absolute value of the difference between x and c:

$$AE = | x - c | \tag{3}$$

The relative error is the ratio between the absolute error and the absolute value of c:

$$RE = | x - c | / | c | \tag{4}$$

The main characteristics of these two errors are:

- *AE* is strongly dependent on the magnitude of *c*;
- *AE* is misleading unless it is stated what is an error of;
- *RE* is a measure of the number of significant digits of *x* that are correct;
- *RE* has meaning even when *x* is not known and in this case it is given as a percentage value.

In this paper we are going to use the logarithm of the relative error:

$$LRE = -\log_{10}[| x - c | / | c |] \tag{5}$$

Then *LRE* represents the agreement of digits between *x* and *c*. *LRE* is undefined when, hopefully, *x* is exactly equal to *c*, or when *c* is zero; in the first case, *LRE* should be set equal to the number of digits of *c*; in the second case, the logarithm of the absolute error (*LAE*) should be used:

$$LAE = -\log_{10} | x - c | = -\log_{10} | x | \tag{6}$$

For sake of simplicity, no distinction will be made between *LRE* and *LAE*, referring to both as *LRE* and denoting it by using the symbol λ_x, where *x* is the generic computed quantity. Nevertheless, such indexes have some weaknesses. Using *LRE*, three kinds of drawbacks may arise:

- *LRE* is a measure of the number of correct significant digits only when *x* is close to *c*. Therefore, each estimated quantity must be compared to its certified value to be sure that they differ by a factor of less than two; otherwise, we shall simply set $\lambda_x=0$.
- It is possible for *LRE* to exceed the number of digits of *c*; for example, it is possible to have $\lambda_x=11.7$, even though certified value *c* contains only 11 digits. In part, this fact is because computer that uses 64 bits will provide an output with fifteen significant digits, so that a number with 11 digits will be "completed" by adding four zeroes. In such a case λ_x should be set to the number of digits of *c*.
- Any *LRE* value less than one should be set to zero.

In literature, it is pointed out how important is that the *LRE* should be at least 9 for linear procedures. For nonlinear procedures 4 or 5 digits of accuracy are enough, in order to consider acceptable the result. This does not mean, for example, that users computing the arithmetic mean need nine digits of accuracy. On the contrary, the idea is that a program, which can not

deliver nine digits of accuracy for this kind of problems, is likely to deliver inaccurate results, when it has to carry out more difficult problems.

3.2 *DIEHARD*

The random number generation procedures have a more and more important role in statistical studies based on the computer intensive use. Obviously, also in this field, there are limits that any computer can never overcome. As everybody knows, it is impossible that a computer generates "pure" random numbers. Really, they are pseudo-random numbers, i.e. they are numbers, provided by a deterministic algorithm, that look like random numbers. So, if the generator works well, numbers appear to be random. Ripley (1990) describes the characteristic which a good random number generator has to own. To assess the goodness of random number generators of a uniform distribution we shall use Marsaglia's DIEHARD. It is a battery of tests, written in C and downloadable from the web (*http://stat.fsu.edu/pub/diehard/diehard.zip*). DIEHARD includes statistical tests for the randomness of the generated numbers that, in the best of the cases, should result statistically not significant. Marsaglia, in DIEHARD documentation, suggests to use every test of the 15 implemented, because there are many possible "way out" from randomness. These fifteen tests try to grab all of them.

3.3 *ELV*

Statistical software has replaced the old tables for the computation of the principal quantities of a probability distribution. Using a computer makes it easy and fast, but also in this case we have to ask ourselves if the computed critical values or percentiles are reliable.

The first step in assessing probability distributions is obtaining a program to compute exact values: we chose Knüsel's ELV program. Output from this program is referred to as "exact" percentiles or critical values. Output from the used software is referred to as "estimated values".

It is impossible to test every possible combination of inputs for a probability distribution (we should change critical values, percentiles and parameters). So we have to draw a sample of critical values, p-values and parameters. We follow the Knüsel's testing strategy, that for continuous distributions considers these values:

- submit the following sequence of percentiles to the inverse probability function of the packages taken into consideration:

{0.0001; 0.001; 0.01; 0.1; 0.2; …; 0.9; 0.99; 0.999; 0.9999}

to obtain the respective critical values. The most serious thing would be seeing algorithms that fail in the tails of the distribution.

- submit the following smallest critical values to the package's probability distribution procedure:

$$\{1.00E\text{-}05;\ 1.00E\text{-}06;\ 3.00E\text{-}07;\ 2.00E\text{-}07\}$$

to obtain the respective percentiles. The most serious thing would be seeing algorithms that fail in the tails of the distribution.

- choose appropriate values for the distribution parameters.

On the contrary, with discrete distributions we do not have exact values of the inverse probability function. So we have to submit critical values, to compute the respective percentiles and to compare them to the ELV's exact ones. In this case, it is very difficult to fix a correct strategy: we have to proceed with attempts by choosing central and extreme values of *x*.

The relative error *RE* (4) is used to measure the accuracy of probabilities. Essentially, a good result is obtained when $RE < 1.00E\text{-}6$, i.e. when the first six significant digits agree with the first six significant digits of the exact percentile. A greater relative error means a not much reliable software. According to these criteria, we have tested the following probability distributions: Poisson, Binomial, Hypergeometric, Standard Normal, Chi-Squared, Student's t, Snedecor's F.

4. Results

Results of the three considered packages are introduced, divided for different statistical areas.

4.1 *Univariate Statistics*

The StRD's univariate summary statistics suite has nine datasets; six of these ones are of lower level of difficulty, two are of average level of difficulty, and one is of higher level of difficulty (see table 1).

NIST provides certified values for sampling means, standard deviations and the first-order autocorrelation coefficient.

As we said above, in this case it is desirable to have all *LRE* equal or greater than 9, that is the first 9 significant digits of the estimated values agree with the digits of the certified values. If this does not happen, then the implemented algorithm probably is not the best one. The computed *LRE*'s are shown in table 2.

Looking at the results, we can say that, for EXCEL, the computation of the mean is accurate, but not for the standard deviation. Although the EXCEL

documentation does not provide any information about its algorithm, we can argue that the program does not use the well known formula:

$$S = \sqrt{\frac{\sum (x_i - M)^2}{n-1}} \tag{7}$$

Indeed, an algorithm based on (7), implemented in any computer with 64 bits storage capacity, will return, for data set NumAcc4, an higher result than 8.25. Most probably, the algorithm used by EXCEL is based on the "shortcut formula" for the variance. However, this formula is not the best way to calculate the variance, except for hand calculation with few small observations: surely, it is not suitable for a computer program which may handle many large observations. In statistical literature the "shortcut formula" is considered the worst algorithm for its numerical instability (McCullough, 1998), even if Ling (1974) deals with this issue and his conclusion is that this formula is not the worst in absolute, depending on the distribution of the data.

Moreover, *LRE* values for the autocorrelation coefficients are too low.

Based on these considerations, EXCEL's performance for the univariate summary statistics can be judged inadequate.

Table 1. *StRD's datasets for univariate summary statistics.*

Name	Level	N. of Observations	Sources
PiDigits	Lower	5000	Observed
Lottery	Lower	218	Observed
Lew	Lower	200	Observed
Mavro	Lower	50	Observed
Michelso	Lower	100	Observed
NumAcc1	Lower	3	Generated
NumAcc2	Average	1001	Generated
NumAcc3	Average	1001	Generated
NumAcc4	Higher	1001	Generated

For Statistica 6, we have preferred solving problems by choosing an option for extended precision, so that the numbers of significant digits for these calculations increased. *LRE*'s are shown in table 2.

In Statistica 6 the computation of sample mean and standard deviation is nearly perfect. On the other hand, the performance of the first-order autocorrelation coefficient algorithm is swinging. Statistica 6, like EXCEL, does not return the coefficient for the data set NumAcc1, because its

algorithm only works when observations are more than four. Finally, Statistica 6 refuses to compute autocorrelation coefficients for the datasets Mavro, NumAcc3 and NumAcc4.

Table 2. *LRE values, indicated as λ, of means (M), standard deviations (S) and autocorrelation coefficients (r) computed on the considered data sets for the three softwares (NS=no solution).*

Data set	EXCEL			STATISTICA			R		
	λ_M	λ_S	λ_r	λ_M	λ_S	λ_r	λ_M	λ_S	λ_r
PiDigits	15	15	3.95	14.63	15	13.78	15	15	15
Lottery	15	15	2.06	15	15	15	15	15	15
Lew	15	15	2.60	15	15	14.84	15	15	14.79
Mavro	15	13.12	1.76	15	15	NS	15	13.12	13.75
Michelso	15	13.83	3.59	15	15	12.16	15	13.84	13.40
NumAcc1	15	15	NS	15	15	NS	15	15	15
NumAcc2	14.03	11.57	3.30	15	15	13.74	15	15	15
NumAcc3	15	9.46	3.30	15	15	NS	15	9.46	11.24
NumAcc4	14	8.25	3.30	15	15	NS	15	8.25	9

In R, sample mean algorithm works greatly, autocorrelation coefficient algorithm is quite accurate. Standard deviation algorithm could be better. Anyway, the latter seems to be a little better procedure than the EXCEL's one (all the *LRE*'s have the same values except for dataset NumAcc2; here R is more accurate than EXCEL).

4.2 *One way Analysis of Variance*

StRD benchmark has eleven datasets with different levels of difficulty: lower (four datasets), average (four datasets) and higher (three datasets). The levels of difficulty are considered according to Simon e Lesage (1989). Indeed, in this statistical area precision problems arise when there are very high values and when they are similar to each other. For this reason, lower level tests have only one constant beginning digit, except for the dataset SiRstv, which has three constant beginning digits; average level tests have 7 constant beginning digits; higher level tests have 13 constant beginning digits (see table 3).

NIST provides certified values for sums of squares, degrees of freedom, mean squares and *F*-statistics. We computed the *LRE*s for the *F*-statistics, because most of the certified values are used in calculating it; this means that if they are wrong for the *F*-statistics, they are wrong, too.

In EXCEL it is possible to note as, by increasing test difficulty, *LRE* decreases quickly (the last dataset *LRE* is zero). However, EXCEL only failed higher difficulty tests; surely, the algorithm can be improved.

Table 3. *StRD datasets for ANOVA and LRE values, indicated as λ, for the F-statistic returned by the three considered software (c is the constant beginning digits, n is number of replicates per cell and k is number of treatments).*

Data set	**c**	**n**	**k**	**Source**	λ_F **EXCEL**	λ_F **Statistica**	λ_F **R**
SiRstv (l)	3	5	5	Observed	13.06	12.00	14.73
SmLs01 (l)	1	21	9	Generated	15	14.50	15
SmLs02 (l)	1	201	9	Generated	14.24	15	15
SmLs03 (l)	1	2001	9	Generated	12.33	14.54	15
AtmWtAg (a)	7	24	2	Observed	10.16	9.25	9.18
SmLs04 (a)	7	21	9	Generated	10.43	8.75	10.41
SmLs05 (a)	7	201	9	Generated	10.21	8.80	10.87
SmLs06 (a)	7	2001	9	Generated	10.19	8.11	9.94
SmLs07 (h)	13	21	9	Generated	4.08	3.16	4.43
SmLs08 (h)	13	201	9	Generated	1.76	1.90	4.19
SmLs09 (h)	13	2001	9	Generated	0	0	4.17

Statistica 6 has the same trend. The program gives *LRE* slightly lower than the EXCEL's ones, though they are acceptable, but for higher difficulty datasets.

In R, the *aov()* command has been used. The same decreasing trend exists here. R's relative errors would draw a curve which, however, would be above the other two curves. Then R results are the best ones among the three sequences. R algorithm is the most accurate. Also the *lm()* command was used, but it yielded similar results.

4.3 *Multiple linear regression*

StRD multiple linear regression suite has eleven datasets with the levels of difficulty showed in table 4. The first group is made of problems very simple to solve, sometimes the second group has given negative R^2 in literature, while the problems in the last group are difficult because they have multicollinear data. This suite includes some "historical" datasets as the Longley data (1967) and several NIST datasets developed by Wampler (1970).

NIST provides certified values for estimates of regression coefficients, coefficient standard deviations, R^2 and the usual table of analysis of variance for linear regression. Obviously, if the former quantities were wrong then also the latter would be wrong. For example, if coefficients are accurate up to 10 digits, residuals could not have a higher accuracy. Then firstly, we have to compute *LRE*s of these quantities and afterwards, only if they were accurate up to 15 digits, we would assess the accuracy of the other statistics.

Table 4. *StRD Data set for the multiple linear regression.*

Name	**Level**	**Model**	**N. parameters**	**N. observations**	**Source**
Norris	Lower	Linear	2	36	Observed
Pontius	Lower	Quadratic	3	40	Observed
NoInt1	Average	Linear	1	11	Generated
NoInt2	Average	Linear	1	3	Generated
Filip	Higher	Polynomial	11	82	Observed
Longley	Higher	Multilinear	7	16	Observed
Wampler1	Higher	Polynomial	6	21	Generated
Wampler2	Higher	Polynomial	6	21	Generated
Wampler3	Higher	Polynomial	6	21	Generated
Wampler4	Higher	Polynomial	6	21	Generated
Wampler5	Higher	Polynomial	6	21	Generated

To summarize the results, we have used the "weakest link of a chain" principle, i.e. we have used the minimum of coefficient *LRE*s and the minimum of standard error *LRE*s. Results are shown in table 5.

It is important to point out that variables too correlated yield inaccurate results. This is due also to the accumulation of rounding errors that produces an output which is not reliable at all. Therefore, it is important for the program to check the collinearity of variables and, if this is very high, not to provide estimates. For example, the dataset Filip is a nearly singular problem. Nevertheless, EXCEL does not take into account data collinearity, by giving estimates which only have an average of 7 digits of accuracy. For this reason, although other results are quite good, EXCEL linear regression routine can be considered inadequate.

On the contrary, Statistica 6 always checks the high collinearity and warns users on it. Only by decreasing the tolerance from 1*E*-4 (default) to 1*E*-6, it has been possible for the package to provide results. Anyway, the program refuses to give a solution for the standard deviation estimates of the problem Wampler1. With other datasets, the package is not able to yield 9 digits of

accuracy, as required to linear procedures. It is true that numerical accuracy of Statistica 6 is worse than EXCEL's one, but it is also true that program is worth for warning users of presence of high collinearity. The latter is a result far more important and appreciable than the numerical accuracy of EXCEL in this area.

Table 5. *LRE values, indicated with λ, for the worst regression coefficients (b) and the worst standard deviations (S) computed by the three considered softwares (NS=no solution due to multicollinearity).*

Data set	EXCEL		STATISTICA		R	
	λ_b	λ_S	λ_b	λ_S	λ_b	λ_S
Norris (b)	12.04	14.07	12.15	10.06	12.25	12.69
Pontius (b)	12	12.02	11.70	8.87	12.27	11.67
NoInt1 (m)	15	15	14.72	13.09	14.72	11.68
NoInt2 (m)	15	14.64	15	13.71	15	11.99
Filip (a)	7.19	7.22	NS	NS	NS	NS
Longley (a)	13.45	14.80	11.49	13.06	13.32	11.79
Wampler1 (a)	15	10.42	6.45	NS	9.33	10.08
Wampler2 (a)	13.39	15	9.60	7.04	12.95	14.85
Wampler3 (a)	10.06	11.40	6.57	10.11	9.57	10.11
Wampler4 (a)	8.13	11.83	6.61	10.26	8.28	11.46
Wampler5 (a)	6.13	12.04	6.24	10.26	6.26	11.46

Using the *lm()* command, R provides estimates of coefficients and of their standard deviations which have the same level of precision of the estimates obtained with the other two programs. About collinearity, the package supplies no message for the five Wampler's problem; it is not so for the dataset Filip: because of the matrix singularity, R does not provide an estimate of the last regression coefficient.

4.4 *Nonlinear regression*

Nonlinear least square regression problems are intrinsically harder than the other ones, and it is generally possible to find a dataset that will defeat even the most robust code. So, evaluation of nonlinear least square software should also include a measure of the reliability of the code, i.e. whether the code correctly recognizes when it has (or has not) found a correct solution. According to McCullogh and Wilson (1998), "for any package it may be too much to expect that the solver can find a solution for each problem, but it is not too much to expect that the solver can figure out when it has not reached

a solution". According to this logic, a software that recognizes when it can find a solution should be appraised much more than another one that gives always a solution, but with no digit of accuracy. For a given problem nonlinear least square solvers are particularly sensitive to starting values; using StRD benchmark, NIST provides three starting values per each problem:

- Start I is relatively far from the final solution;
- Start II is relatively close to the final solution;
- Start III is the actual certified solution.

In general, having a good starting point is the main problem of a statistician who wants to solve a nonlinear regression problem, but finding this is not easy. So, to be sure of the quality of the implemented procedures, we need to check how they work by choosing Start I as starting point. Start III values are only used to assess that software does not work in a bad way, without recognizing that it has reached the minimum. Finally, Start II values have to be used only when Start I values have provided no solution or a solution "far" from the certified one. Other factors affect the efficacy of an implemented algorithm: among them, the convergence criterion, the selected convergence tolerance, the method of solution, the possibility to compute analytic or numerical derivatives, and so on. Finally, a user can combine these four options, and every time a different solution will arise. Again quoting McCullough (1999): "The solution to a nonlinear least squares estimation problem obtained from using default options rarely is as good as that obtained from using some other settings. For every package there does seem to be some preferred combination of options which consistently yields better results than the default options."

In fact, a preferred combination of options for "each software" exists only if we are dealing with the same problem. If a specific setting is the best combination for a dataset, the same setting will not give the most accurate result for another non linear regression problem. Every time, the user has to be able to find the "right combination" suitable for that nonlinear regression problem.

Since a unique preferred combination, i.e. uniformly valid for every benchmark, does not exist and since settings have to be uniform in every dataset, in order to compare results correctly, we have chosen to use the default options. The non existence of a combination of options that is globally more efficient is the reason why the default one was preferred. In this case, the presence of a larger number of options offered by a package command has been valued positively. The main characteristics of the benchmarks are shown in table 6. For sake of brevity, the used model for each dataset is omitted.

NIST provides certified values to 11 digits for coefficient estimates, residual sum of squares, residual standard deviation and degrees of freedom.

Here only the first two *LRE* values will be shown; possible errors in the other quantities will derive from errors on the parameter estimates and on the standard deviations of the coefficients.

Table 6. *StRD Dataset for the nonlinear regression.*

Name	**Level**	**N. parameters**	**N. observations**	**Source**
Misra1a	Lower	2	14	Observed
Chwirut2	Lower	3	54	Observed
Chwirut1	Lower	3	214	Observed
Lanczos3	Lower	6	24	Generated
Gauss1	Lower	8	250	Generated
Gauss2	Lower	8	250	Generated
DanWood	Lower	2	6	Observed
Misra1b	Lower	2	14	Observed
Kirby2	Average	5	151	Observed
Hahn1	Average	7	236	Observed
Nelson	Average	3	128	Observed
MGH17	Average	5	33	Generated
Lanczos1	Average	6	24	Generated
Lanczos2	Average	6	24	Generated
Gauss3	Average	8	250	Generated
Misra1c	Average	2	14	Observed
Misra1d	Average	2	14	Observed
Roszman1	Average	4	25	Observed
ENSO	Average	9	168	Observed
MGH09	Higher	4	11	Generated
Thurber	Higher	7	37	Observed
BoxBOD	Higher	2	6	Observed
Rat42	Higher	3	9	Observed
MGH10	Higher	3	16	Generated
Eckerle4	Higher	3	35	Observed
Rat43	Higher	4	15	Observed
Bennett5	Higher	3	154	Observed

EXCEL does not have a routine to solve nonlinear problem. The purpose of benchmarking is to assess algorithms. Carrying out a nonlinear analysis in

EXCEL means to calculate by hand first and second derivates. It is not useful to assess the algorithms implemented by the Microsoft's program for this area. The computation of derivates would have been acceptable if EXCEL had an algorithm based on the Gauss-Newton method or an algorithm for matrix inversion. In EXCEL no-one of these "supports" exists, so we have decided not to do benchmarking analysis in EXCEL.

On the contrary, Statistica 6 has its own routine for the estimation of nonlinear models. The obtained *LRE*s, selected by means of the weakest link principle, are shown in table 7.

Because of the variety of the obtained outputs, it is not possible to draw a general conclusion. Moreover, it is possible to summarize the results with the following considerations:

- The real failures of Statistica 6 are relatively few. The software is not able to reach the four digits of accuracy, required in this type of analysis, for five datasets: Misra1c, MGH09, Thurber, Rat43 and Bennett5.
- The program returns the required four digits of accuracy in 7 of the 27 proposed tests.
- With 7 datasets, Misra1a, Kirby2, Hahn1, Misra1d, Roszman1, Misra1b and Eckerle4, Statistica 6 yields a warning message for possible wrong results. Despite of it, in 5 cases coefficient estimates are highly accurate, even reaching 10 digits of accuracy (dataset Misra1a). It is not so for the standard deviation estimates, whose *LRE*s are all zero. A specification is necessary, though. The 7 considered datasets with Lanczos1 and Misra1c have *LRE*s zero because Statistica 6 yields estimates of standard deviations equal to zero. On the contrary, all certified values are not zero, although their values are very low (an average of 1*E*-6). Then, although no digit agrees with certified values (*LRE* = 0), computed values are not so bad. Indeed, in these cases the first non zero digit is the sixth one, on an average.
- With the datasets Gauss2, Nelson, Lanczos2, Gauss3, ENSO, MGH10 and Bennett5 Statistica 6 provides outputs whose *LRE*s are all zero, both by using Start I and Start II values. No problem occurs when using Start III values.
- Finally, for the dataset MGH17 the program refuses to give results.

It is worth pointing out another positive characteristic of Statistica 6: it has been necessary to use Start II values only three times; all the other times the provided estimates could be obtained by specifying "bad" starting values. Besides, there is a great number of options that user can choose.

Concerning R, in order to obtain estimates, *nlm()*, *nls()* and *optim()* commands have been used. The better outputs (shown in table 7) were obtained by using the *nls()* function. Then the lowest *LRE* has been taken,

according to the weakest link principle. It is useful to distinguish two cases, dealing with estimates of standard deviations and coefficients separately:

- The algorithm to estimate standard deviations always provides accurate results, except for datasets Misra1a and Rat42.

Table 7. *LREs values, indicated as λ, for the worst non linear regression coefficients (b) and the worst standard deviations (S) for the two softwares taken into consideration (NV=no valid solution).*

Data set	STATISTICA			R		
	Start	λ_b	λ_S	Start	λ_b	λ_S
Misra1a (l)	I	10.32	0	II	6.77	0
Chwirut2 (l)	I	6.22	5.26	I	4.87	5.29
Chwirut1 (l)	I	5.84	4.81	II	5.94	6.31
Lanczos3 (l)	I	4.63	4.34	I	NV	NV
Gauss1 (l)	I	6.92	4.79	I	6.92	6.78
Gauss2 (l)	I	0	0	I	6.45	6.25
DanWood (l)	I	7.98	5.96	II	8.04	7.73
Misra1b (l)	I	2.20	0	II	6.64	6.06
Kirby2 (a)	I	5.29	0	II	6.47	6.78
Hahn1 (a)	I	5.92	0	I	5.84	6.73
Nelson (a)	I	0	0	I	5.33	5.36
MGH17 (a)	I	NV	NV	I	5.76	5.19
Lanczos1 (a)	I	10.56	0	I	NV	NV
Lanczos2 (a)	I	0	0	I	NV	NV
Gauss3 (a)	I	0	0	I	6.45	5.92
Misra1c (a)	II	2.17	0	II	7.98	5.90
Misra1d (a)	I	8.55	0	II	6.77	6.64
Roszman1 (a)	I	5.46	0	II	5.50	6.02
ENSO (a)	I	0	0	I	4.01	5.17
MGH09 (h)	I	1.88	2.16	II	4.80	4.91
Thurber (h)	I	1.83	1.74	II	NV	NV
BoxBOD (h)	II	4.45	3.73	I	5.80	5.75
Rat42 (h)	I	5.58	4.01	I	7.13	0
MGH10 (h)	I	0	0	I	6.72	5.91
Eckerle4 (h)	I	0	0	II	7.16	7.04
Rat43 (h)	I	2.93	2.37	II	5.70	5.80
Bennett5 (h)	II	2.66	2.55	I	5.79	5.25

- The algorithm to estimate regression coefficients works optimally. Start II values have only been used when Start I values have provided warning messages, because of singularity of the gradient matrix.

In spite of these good results, the *nls()* function is not able to find a solution for the three Lanczos and Thurber datasets, even by using Start III values. With the Thurber dataset R provides a message which warns users about the presence of a singular gradient matrix; with the three Lanczos datasets the program provides a message explaining its difficulties in convergence.

Because of these messages and the great variety of options too, R is the program to be preferred.

4.5 *Random number generators for a uniform distribution*

As we said above, to test the goodness of random number generators, Marsaglia's DIEHARD has been used.

All the three packages passed DIEHARD tests. So it is clear how the reliability of statistical software in this area is greatly increased since 1996, the year when DIEHARD was released. In spite of that, it is appropriate to think that the actual software development needs uploading this battery of tests.

Anyway, using this battery of tests a particular aspect arises. In EXCEL, 320000 random numbers were generated. Although the period of the algorithm was not reached, i.e. a particular sequence of numbers is not repeated, we have found some random numbers repeated many times. In particular, it was possible to count five generated numbers, i.e. 0.126010925626392, 0.0507827997680593, 0.623126926475823, 0.906704916531877 and 0.975402081362346, twenty-four times in the generated sequence. Besides, EXCEL generated ten 0 and thirteen 1.

Statistica did it, too. The issue here was more serious: it was possible to count the number 0.50007629627369 twenty-six times; there were fourteen 0 and fifteen 1.

On the contrary, in R, numbers seemed to be repeated only very few times. When generating more than one million numbers, it was unusual to count three times the same numbers. There was no number repeated four times. Finally, one 1 and no 0 have been counted. Hence, it is right to argue that pseudo-random number generator implemented in R is the best one among the three considered.

4.6 *Probability distribution*

As we said above, to obtain exact values of the considered distributions, Knüsel's ELV has been chosen.

Somebody can argue from the results obtained for the Poisson distribution that EXCEL apparently gives good outputs for central *x* values, while extreme values are not that good; indeed EXCEL provides zeroes too easily when *x* is very small, so that the relative errors are equal to 1. As far as Statistica 6 is concerned, the values of $P\{X < x\}$ seem to be correct, providing relative errors lower than 1*E*-6, also considering those probability values which EXCEL failed to compute. In R, the used algorithm seems to be correct. Every relative error is greatly lower than 1*E*-6 and even lower than the relative errors computed in Statistica 6.

For the binomial distribution, the EXCEL computed values of $P\{X<x\}$ seem to be correct. EXCEL has little accuracy in working with great values (relative errors are nearly 1*E*-2). Probabilities computed by Statistica 6 seem to be correct. All relative errors are lower than 1*E*-6, also considering those probability values which EXCEL failed to compute. As far as R is concerned, the computed values of $P\{X < x\}$ seem to be correct. Relative errors have the same size as the Statistica's ones.

For the hypergeometric distribution EXCEL performance is not that good. There are great relative errors for great values of *x*. In Statistica 6 it is impossible to compute percentiles or critical values with this kind of distribution. Performance of R is good.

For the standard normal distribution EXCEL results are good: the $P\{X < x\}$ values and the percentiles seem to be good. Algorithms of Statistica and R seem to be good, too (there are relative errors lower than 1*E*-6).

As far as Chi-Squared distribution is concerned, EXCEL fails only in one strange case: with a 0.2 probability value and 1000 degrees of freedom it gives no output. Statistica's algorithm shows irregularities. Although no problem exists when using inverse function with central values of *p*, Statistica's routine does not work with extreme probabilities, neither the program provide a warning message. It is impossible to obtain a result with this probability values $p = (1E\text{-}6; 3E\text{-}7; 2E\text{-}7)$, whatever degree of freedom is chosen. The algorithm of R seems to be correct, with relative errors lower than 1E-6.

For Student's t distribution, the most relevant inaccuracy of the three softwares is found in the central zone of the distribution. Table 8 shows relative errors of EXCEL, Statistica and R for values of *p* close to 0.5. As it is easy to note, no software reaches an acceptable accuracy level: all the relative errors are greater than 1*E*-6.

Finally, results of tests on F distribution obtained from algorithms of EXCEL and R seem to be accurate: every relative error is lower than 1*E*-6. The procedure of Statistica for the computation of percentiles seems to be correct; on the contrary, the procedure for the computation of lower tail

probabilities seems to be inaccurate, because it provides $P\{X < x\} = 0$ too easily.

Table 8. *Relative errors for some values of the t-Student distribution.*

x	n	Relative errors		
		EXCEL	**Statistica**	**R**
0.499999	1000	0.001846666	0.0063515	0.0033327
0.499999	100	0.000163256	0.0007253	0.0001961
0.499999	50	0.000187442	9.23118*E*-05	0.0001115
0.499999	10	4.43714*E*-05	2.28981*E*-05	4.6693*E*-06

5. Conclusions

By using the methodology outlined in several papers, some procedures of EXCEL, Statistica and R have been assessed. The three considered softwares are not perfect in all the three areas of interest: estimation, random number generation and probability distribution.

Generally, all packages have supplied accurate sample means and standard deviations. In particular, the algorithm of R for the sample mean computation and the one of Statistica for the standard deviation computation are accurate. The algorithm for the calculation of the first-order autocorrelation coefficient, in spite of its goodness in R, has been not so reliable for the other two packages. In the one-way analysis of variance area a decreasing trend has been noted as constant leading digits have been growing. This is not a bad software warn, but it is due to the limits of storage of the computer. Into the goodness of the three procedures, R algorithm seems better than the other ones: its output values are better, both using *aov()* or *lm()* function.

Surely, linear regression procedures have been the most benchmarked software routines for many years. The effects of these continuous controls have affected positively the results of the three software algorithms. EXCEL, Statistica and R have returned sufficient accurate estimates. It has only been noted that EXCEL has not provided warning messages to reveal the presence of ill-conditioned design matrices.

Nonlinear regression results have not been as good as the linear regression ones. Statistica is not quite accurate, more accurate is R; both provide a great number of options that a user can set. It has not been possible to test EXCEL because of the inexistency of any procedure for nonlinear estimation problems.

EXCEL, Statistica and R have passed all the tests on random number generators included in the DIEHARD benchmark, though EXCEL and Statistica have generated some numbers that are repeated several times.

Finally, lacks have been found in the probability distribution area, though R seems to be more accurate than the other packages in almost all the probability distributions taken into consideration.

In conclusion, performance of R is better than that of the other two softwares. This could be a surprising result because people usually think that a free software is worse than a commercial one. In fact, R is a well developed software and its growth involves many statistical and computational researchers all around the World (for more details see the URL site: http://www.r-project.org). However, it is necessary, in order to complete this comparison, to consider the performance of R in comparison with other common statistical software as SAS or SPSS, and this is what we are going to do in a following paper.

Acknowledgements

The authors would like to thank a referee for helpful suggestions.
Research developed with grants of the University of Palermo.

References

ANSI/IEEE (1985), IEEE Standard 754 for Binary Floating-Point Arithmetic.

Beaton A., Rubin B., Barone J. (1976). The Acceptability of Regression Solution: Another look at Computational Accuracy. *JASA,* **71**, 158-168.

Dahlquist J.E., Bjorck, A. (1974). *Numerical Methods.* Prentice-Hall, New York.

Knüsel L. (1995). On the Accuracy of Statistical Distributions in GAUSS. *Computational Statistics and Data Analysis*, **20**, 699-702.

Knüsel L. (1998). On the Accuracy of some Statistical Distributions in Microsoft Excel 97. *Computational Statistics and Data Analysis*, **26**, 375-377.

Ling R.F. (1974). Comparison of Several Algorithms for Computing Sample Means and Variance. *JASA*, **69**, 859-866.

Longley J.W. (1967). An Appraisal of Least Square Programs for the Electronic Computer from the Point of View of the User. *JASA*, **62**, 819-841.

McCullough B.D. (1998). Assessing the Reliability of Statistical Software: Part I. *The American Statistician*, **52**, 358-366.

McCullough B.D. (1999). Assessing the Reliability of Statistical Software: Part II. *The American Statistician*, **53**, 149-159.

McCullough B.D., Vinod H.D (1999). The Numerical Reliability of Econometric Software. *Journal of Economic Literature*, **37**, 633-665.

McCullough B.D., Wilson B. (1998). On the Accuracy of statistical procedures in Microsoft Excel 97. *Computational Statistics and Data Analysis*, **31**, 27-37.

McCullough B.D., Wilson B. (2002). On the Accuracy of statistical procedures in Microsoft Excel 2000 and Excel XP. *Computational Statistics and Data Analysis*, **40**, 713-721.

Ripley B.D. (1990). Thoughts on Pseudorandom Number Generation. *Journal of Computational and Applied Mathematics*, **31**, 153-163.

Sawitzki G. (1994). Testing Numerical Reliability of Data Analysis Systems.

Computational Statistics and Data Analysis, **18**, 269-286.

Simon S., Lesage J.P. (1998). Benchmarking Numerical Accuracy of Statistical Algorithms. *Computational Statistics and Data Analysis*, **7**, 197-209.

Wampler R.H. (1970). A Report on the Accuracy of Some Widely Used Least Squares Computer Programs. *JASA*, **65**, 549-565.

Wilkinson J.H. (1974). *Rounding errors in algebraic processes.* Englewood Cliffs, Prentice-Hall, New Jersey.