International Journal of Forecasting 16 (2000) 349–357
www.elsevier.com/locate/ijforecast

Is it safe to assume that software is accurate?

B.D. McCullough
Federal Communications Commission, 445 12th St. SW, Room 2C-134, Washington, DC 20554, USA

1. Introduction

No.

The application of entry-level reliability tests, e.g., Wilkinson (1985) tests, has uncovered many errors in statistical and econometric software packages (Sawitzki, 1994a,b; McCullough, 2000). The intermediate-level tests proposed by McCullough (1998) also have found many errors (McCullough, 1999a,b; McCullough & Wilson, 1999; Altman & MacDonald, 1999; Vinod, 2000). At the advanced level, very few tests exist, but the analysis of GARCH procedures in McCullough and Renfro (1999) and ARMA procedures in Newbold, Agiakloglou, Miller (1994) cast doubt on the accuracy of some packages. In fact, some packages are so inaccurate that they give incorrect answers to simple textbook problems. Errors have also been found in the statistical distributions of packages (Knüsel, 1995, 1996, 1998); it is not unknown that computer-produced critical values can be less accurate than those found in the appendix of a statistics text. Random number generators (RNGs) also can be bad. Tajima, Ninomiya and Tezuka (1997) analyze a bad RNG that produced incorrect option prices, and MacKinnon (2000) describes how a perfectly good Monte Carlo Study was ruined by a bad RNG.

E-mail address: bmccullo@fcc.gov (B.D. McCullough).

2. Why does it matter?

For the professional forecaster in industry, the answer is obvious: his employer's money, and even his own job depends on the accuracy of his forecasts. Forecasting is parlous enough even with accurate software. It is not uncommon for a professional forecaster to run all his problems on three packages, because he doesn't know which one to trust. Forecasters who do not take this precaution, of course, might not know they have a problem.

For the academician it is a matter of science. The integrity of the cumulated body of knowledge depends upon the integrity of each published piece. The normal self-correcting mechanism of science that weeds out bad results, replication, simply does not apply in the present case. Researchers do make programming and data transcription errors; for this reason alone results should be replicated, even if all software packages were accurate. How many important results in forecasting and economics have been replicated? However, bad software can also lead to bad results, and replication could help uncover bad software, too. Perhaps if more of these bugs were detected and reported, users would be less complacent about valuing user-friendliness more than accuracy when making a software decision.

3. Why is user-friendliness valued more than accuracy?

It is a basic principle of economics that firms will not supply a commodity for which there is no demand. Developers do not supply accuracy because the vast majority of users do not demand it. The veracity of this statement can, for many persons, be verified by introspection: On what criteria do you evaluate software? The answer is probably: ease-of-use and speed. These are also the criteria employed in the vast majority of software reviews. Whether or not the program gives the right answer rarely is a consideration. The reasons for this are varied, but one prime reason is that most users simply have not been trained to appreciate how fragile numerical computations can be. This can best be illuminated by considering the situation in the not-too-distant past: before the advent of desktop computing.

Back then, many things were different. Since computers were scarce, there were not many users. Packages were low-level and required users to possess some programming skills, in addition to some knowledge of statistics. Consequently, a user knew enough not to place blind faith in his own code, let alone someone else's, and computer output was treated with some degree of circumspection and a healthy mistrust. Users could debate the relative merits of single vs. double precision calculation or various methods of matrix inversion, and the concept of user-friendliness had yet to be invented. Users were forced to confront the computational aspects of the problems they wanted to solve, and necessarily developed an appreciation for numerical issues. For example, back then, a user had to program each step of the procedure to perform an AR(1) correction for autocorrelated errors, and had to worry about how to treat the initial observation. Additionally, statistical packages were few in number, and their developers typically had both statisticians and numerical analysts on staff. Accuracy was a primary concern of both users and developers.

Today, many things have changed. Nowadays, with computers on every desktop, the number of users of statistical software has increased exponentially. These new users are unlike the old users, both in training and in temperament. To perform sophisticated statistical analyses, users need possess knowledge of little more than how to point and click a mouse. Steeped in a 'user-friendly' environment and largely devoid of practical programming experience, they cannot understand why a statistical package should be more difficult to use than, say, a spreadsheet program, despite the obvious fact that statistics is much more computationally complex than accounting. These same users are shocked, simply shocked, to discover that computer numbers are not exact, and the phrase 'cumulated rounding error' is wholly foreign to them. What matters to them is not how accurate the package is, but how easy it is to get an answer. After all, how accurate does an estimate have to be if nobody is checking?

These recent users can implement an AR(1) correction without knowing: (i) that treatment of the first observation matters; (ii) different packages implement different corrections; and (iii) any given correction is not necessarily implemented correctly (Lovell & Selover, 1994). By their sheer numbers, these new users dominate the demand for statistical software and their desires for speed and user-friendliness more than offset the old users' desire for accuracy. Additionally, producing a statistical package no longer requires a staff that can be expected to contain experts on both statistics and programming. Only a PC and a modicum of programming experience are required, and the number of packages has increased as the cost of producing a package has decreased. Marketing and sales, not accuracy, is the primary concern of many such developers.

Since these new users began computing after the introduction of the PC, they have never even heard of a Hollerith card, let alone used one. These newcomers want bells and whistles, Excel interfaces, and the latest sophisticated techniques. The thought that a program will produce an incorrect answer simply never enters their heads. Many developers, catering to majority preference, have no incentive to provide accurate software. The developer who allocates resources to accuracy instead of the user interface has not, in the past, been rewarded with increased sales. Hopefully, this will change in the not-too-distant future.

Computational details are important, even if often overlooked. There exist many old problems of which any software developer should be aware, and for which solutions are well-known in the statistical computing community. Many developers still are plagued by these old problems, both of a generic statistical computing nature and some problems specific to time series and forecasting. Several examples will be given. There are also newer problems that are not so well-known but for which solutions exist, though the solution may be even less-well-known than the problem. While many developers offer recently-developed sophisticated econometric procedures, very few take the time to ensure that those features incorporate the necessary attention to numerical reliability. This type of problem can distinguish between packages that offer reliable procedures, and those that just offer procedures for the sake of offering procedures.

4. Generic old problems

Consider calculation of the sample variance. The so-called ‘calculator formula’

$$s^2 = \frac{\sum_{i=1}^{n} x_i^2 - (1/n)\left(\sum_{i=1}^{n} x_i\right)^2}{n-1}$$

is often presented as a shortcut in textbooks because it requires fewer calculations than the usual formula. This formula is extremely susceptible to rounding error and is numerically unstable. In fact, it is the worst of the five methods considered by Ling (1974), and is even used as an example of ‘what not to do’ in texts on statistical computing (Thisted, 1988, §2.3.2). This algorithm does not produce an overflow error, but instead (usually) produces ‘answers’ which are seemingly sensible. A fortunate user will discover something amiss, however, when the program returns a negative number (which this algorithm will do for reasonable input values). A simple test (Wilkinson & Dallal, 1977) can reveal whether a package uses the calculator formula. Simply compute the variance of three observations: 90 000 001; 90 000 002; 90 000 003. Assuming division by $n-1$ and not n, if the answer isn’t unity, the package doesn’t use a reliable algorithm. Similarly, the solution of the linear regression problem, $y = X\beta + \epsilon$, is often presented in texts as $b = (X'X)^{-1}X'y$, but b should never be calculated by inverting $X'X$. Yet some packages do this, too.

On the same scale of a negative variance are correlation coefficients greater than unity. Wilkinson (1985), (Test II-D) has a useful test for this. Six of the seven variables are linear transforms of each other, and should produce correlation coefficients of unity. By definition any constant (and the seventh variable, ZERO, is all zeroes) has ‘undefined’ as its correlation with any other variable. Yet many packages cannot pass this test. Two such examples are presented in Table 1.

Both the above problems are easily fixed, and I am aware of several packages that have recently fixed these problems. So take out an old version of your program, install it, and run

Table 1
Results of Test of IID. Lower diagonal of correlation matrix: Only incorrect results are displayed

	X	ZERO	BIG	LITTLE	HUGE	TINY	ROUND
X							
ZERO	0	0					
BIG	1.13	0					
LITTLE	1.01	0	1.14				
HUGE		0	1.13	1.01			
TINY		0	1.13	1.01			
ROUND		0	1.13	1.01			
				Package 'A'			
	X	ZERO	BIG	LITTLE	HUGE	TINY	ROUND
X							
ZERO	0	1					
BIG	1.129	0	1.277				
LITTLE	1.001	0	1.137	1.013			
HUGE		0	1.30	1.001			
TINY		0	1.30	1.001			
ROUND		0	1.30	1.001			
				Package'B'			

these tests. This will give you an idea of whether your developer was paying attention to numerical detail before someone pointed these problems out to him.

So unconcerned with accuracy are users, that software developers can ignore with impunity not only verified reports of errors by users, but published articles detailing such errors. Sawitzki (1994a,b) uncovered a variety of errors in several statistical packages, and discussed them with developers in 1991, 1992, and 1993. Sawitzki (1994b, p. 289) noted, that 'The vendor reaction ranged anywhere between co-operative concern and rude comments.' Many of those errors persisted through subsequent versions of the software, and some of those errors remain uncorrected even today, nine years after developers first were informed: an appalling lack of concern for accuracy. Only a similar lack of concern for accuracy by users permits software developers to (not) respond so egregiously to published accounts of unreliability. Of course, some developers are extremely responsive to even unpublished errors, fixing them almost immediately. The problem is that the user who cares about accuracy has no idea whether his developer falls into the former category or the latter.

As yet another example of inattention to numerical detail, some developers tout 'double precision calculation' and 'sophisticated graphics' — but conveniently neglect to mention that the graphics routines are written in single precision. When double precision calculations are handed off to a single precision graphics routine, funny things can happen, such as points being dropped from the graph without warning. Consider Wilkinson (1985) Test IIB, which asks the program to graph BIG against LITTLE. Both these variables are well within range to be handled by a double precision program and, since they are linear transforms of each other, their graph should be a straight line, as shown in the left panel of Fig. 1. The panel on the right shows what can happen when the program claims to be double precision but the

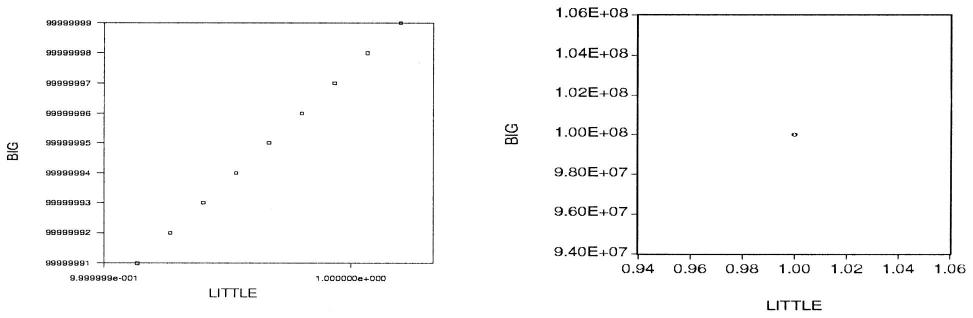

Fig. 1. Wilkinson's Test IIB Results for Two Packages: correct (left) and incorrect (right).

graphics routine is only single precision. For less extreme data, such a program can drop observations without warning, thus giving the user an incorrect graphical representation of the data.

5. Problems for time series and forecasting

The preceding examples are all rather generic, but procedures of crucial importance to time series and forecasting also are similarly plagued by inattention to computational detail. In particular, I mention computation of partial autocorrelation coefficients (of great import for identifying Box–Jenkins models), estimation of ARMA models, and GARCH estimation.

The Yule–Walker equations (YWE) are frequently used to compute partial autocorrelation coefficients for the identification of Box–Jenkins models; often it is the only method offered by a package. Yet is well-known that the Yule–Walker equations are extremely susceptible to numerical error and, in fact, all three editions of Box and Jenkins warn against their use. Priestley (1981, pp. 350–52) lists four methods for computing partial autocorrelation coefficients in decreasing order of numerical reliability, and YWE is last on the list. See also de Hoon et al. (1996), McCullough (1999c), and Tjøstheim and Paulsen (1983). Why do developers insist on using a numerically deficient method, without at least offering stable alternatives? Perhaps for the same reason that many treatises on time series discuss only one method, the YWE, and do not bother to mention its numerical liabilities (e.g. Granger & Newbold, 1986; Hamilton, 1994; Diebold, 1997).

In an important article in this journal, Newbold et al. (1994) (NAM) fit ARMA models to a few datasets using several packages. Not surprisingly, they found that different packages gave different answers to the same problem, especially when forecasting from the fitted model. Sometimes they could trace the discrepancy to algorithmic differences; other times they could not. These latter cases strongly suggest that at least one of the packages is inaccurate. Some forecasters place great emphasis on ease-of-use and the ability to use their forecasting software with other packages, say, for document delivery to clients. These persons might have a tendency to be less concerned with software reliability. However, NAM remark (p. 577) that 'it is difficult to be sanguine about the fact that two researchers on our campus could quite easily produce forecasts differing by as much as four-tenths of a standard error, after fitting the same model to the same data.' Because there is no benchmark for ARMA models, NAM could not determine which of the packages gave the correct answer. The salient point remains: we have no idea which programs can correctly calculate ARMA models or forecast from the fitted ARMA model. Skeptical readers who have not met with failure when attempting to replicate examples from forecasting texts are invited to produce forecasts from the same model/data for two packages using either the Kalman filter or VAR methods. Or with a GARCH model.

The basic GARCH(p, q) model is given by

$$y_t = x_t'b + \epsilon_t \quad \epsilon_t | \Psi_{t-1} \sim N(0, h_t) \tag{1}$$

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j} \tag{2}$$

Consider the Bollerslev and Ghysels (1996) data, 1974 observations on the daily percentage nominal returns for the Deutschemark/British pound. Imagine trying to fit a constant with a GARCH(1, 1) error to these data. Default estimates of the coefficients (with t-statistics in parentheses) from several packages are presented in Table 2. Though there seems to be some consensus, the estimates do vary sufficiently to give one pause. One reason for this variation is that all packages are not maximizing the same (conditional) likelihood. The model defined by Eqs. (1)–(2) is only partially specified. To complete the specification and define the likelihood, the presample values of ϵ_t^2 and h_t must known for $t \leq 0$. Some packages leave the user to guess how the series ϵ_t^2 and h_t are initialized.

Table 2
GARCH coefficient estimates and t-statistics

Package	μ	α_0	α_1	β_1
X1	−0.00540	0.0096	0.142	0.821
	(−0.64)	(8.01)	(11.09)	(53.83)
X2	−0.00608	0.0098	0.144	0.818
	(−0.72)	(3.80)	(5.80)	(26.34)
X3	−0.00624	0.0108	0.153	0.806
	(−0.74)	(8.13)	(10.95)	(48.64)
X4	−0.00619	0.0108	0.152	0.806
	(−0.74)	(8.15)	(10.97)	(48.61)
X5	−0.00613	0.0107	0.153	0.806
	(−0.73)	(5.58)	(7.91)	(36.96)
X6	−0.00919	0.0098	0.144	0.818
	(−1.08)	(8.09)	(10.77)	(45.90)
X7	−0.00619	0.0108	0.153	0.806
	(−0.67)	(1.66)	(2.86)	(11.12)

The t-statistics are even more troublingly disparate. A small part of this is due to the different coefficient estimates. The large part is due to different methods of computing standard errors. There are at least five ways to compute standard errors for GARCH estimates (e.g., OPG, Information Matrix, Hessian, Bollerslev–Wooldrdige, and Quasi-MLE), and some packages leave the user to guess what method is used.

All these matters are discussed fully in McCullough and Renfro (1999), who present benchmark results for Packages X1–X7 based on the work of Fiorentini, Calzolari and Panattoni (1996) (FCP). Among many other interesting things, FCP provide further evidence that analytic derivatives are more accurate than numerical derivatives. Only a handful of packages implement analytic first derivatives for GARCH, though they are presented in a basic econometrics text (Greene, 1993, §18.5). I am aware of only one package that uses analytic second derivatives for GARCH estimation. Researchers at the University of Reading's Economics Department are preparing for this journal a software review that not only will apply the FCP GARCH benchmark to several packages, but also examine the consistency of volatility forecasting and E-GARCH estimation across packages. GARCH models are only one example of this phenomenon. What about neural networks and genetic algorithms and STAR and TAR and Markov switching models? Which developers offer them only as part of bells and whistles, and which are actively committed to providing accurate software? How is the user to discriminate between two packages when developers provide no tangible evidence of accuracy, and reviewers ignore the issue?

6. What can be done?

Journals and researchers should seriously consider actively encouraging replicable research. Simply adopting 'policies' that authors should make their data and code available for replication purposes is ineffectual (Dewald, Thursby & Anderson, 1986; Anderson & Dewald, 1994), perhaps because the disincentives for authors to comply are too great (Feigenbaum & Levy, 1993).

Until recently, such archiving was costly and rarely done. The advent of the worldwide web has dramatically altered the equation. The cost of maintaining such an archive in the form of a website is small. Data archives are proliferating rapidly, and even some journals maintain them (e.g., *Economic Journal, Journal of Business and Economic Statistics, Journal of Applied Econometrics*). One journal even has a data/code archive (*Macroeconomic Dynamics*). For researchers who maintain the standard of producing replicable research, the cost is also near zero: it is a simple matter to upload code and data to the website. For researchers who do not write clean, clearly commented code and otherwise do not adhere to the standard of replicable research, the cost will be substantial — but it should be. The purpose of replication has always been to maintain high quality research. In the present day, there is an additional benefit: better software.

The most efficient means for uncovering bugs in software is to use two (or more!) different packages to solve the same problem. If they give different answers, a potential error in at least one of the packages has been uncovered. Researchers have no incentive to do this.[1] Using two or more packages to solve the same problem will only occur on a large scale if journals actively support replication through data/code archives. Ambitious assistant professors scouring these archives looking for errors doubtless will uncover numerous software bugs that otherwise would have gone undetected.

7. What else can be done?

When users demand accuracy, software developers will supply it. Users will demand accuracy only when they begin to realize the extent of inaccuracies in statistical and econometric software used for forecasting (McCullough & Vinod, 1999). Recently, some packages have begun including 'accuracy' in their advertisements, in addition to the usual suspects (speed and user-friendliness); this could heighten users' awareness: does their package claim to be accurate? If not, why not?

The onus is on developers to demonstrate affirmatively that their programs are reliable. For example, STATA, LIMDEP and TSP all have benchmark results on their respective homepages. While it is too much to expect that a developer can provide such evidence for each and every procedure, at the very least the developer should make use of existing benchmarks (Wilkinson, 1985; McCullough, 1998). In the event that a developer does not meet this minimal standard, it would be helpful if a

[1]Suppose I have an important result to be placed in a top journal but, concerned as I am with 'scientific integrity', I doublecheck my results by running them on another package and get a different answer. Will the journal publish both answers? How am I to resolve the discrepancy? I have no incentive to use a second package and thus, no incentive to uncover bugs in software.

reviewer would apply these benchmarks. However, benchmarking is time-consuming and tedious work. Should a reviewer not be able to spare the time, he should at least note that the developer has failed to provide evidence that the package satisfies existing benchmarks. For example, several packages offer genetic algorithms and neural networks. I know that benchmarks exist for these problems, but I have yet to see a single developer or reviewer report the results of such tests.

Software reviews can play a large role in educating users about desirable features of software packages. In addition to checking to see whether the developer provides evidence of numerical reliability, the reviewer can also assess whether the documentation adequately describes the procedures. STATA, for example, provides copious documentation, including algorithms, for nearly every command. This approach leads to a four volume set of manuals, but even a single volume manual can pay admirable attention to numerical matters, as in the case of the package Ox (Doornik, 1998), which even includes a chapter on numerical accuracy. In the case of GARCH procedures, for example: does the documentation describe how pre-sample values of ϵ_t^2 and h_t are treated? For general nonlinear estimation, if the package uses numerical derivatives, does the documentation describe how the derivatives are calculated (forward or central difference; how is the differencing interval chosen)? Is an algorithm provided for the random number generator (RNG) and is its period given, along with statistical tests for randomness that the generator passes?

To assist in providing users and developers proper incentives, software reviews in this journal will make an effort to include such important topics as benchmark results and an assessment of the package's documentation with respect to numerical details. An effort will also be made to find reviewers who are willing to apply benchmarks. This journal will also be especially interested in publishing comparative reviews showing two or more packages giving different answers to the same problem.

Accuracy is important, and has been too long neglected.

Acknowledgements

Thanks for comments to J. Duvall, C. Chatfield, C. Brooks, C. Bonham, S. Thiruvadanthal and T. Martin. The views expressed are those of the author and not necessarily those of the Commission.

References

Altman, M., MacDonald, M. (1999). The robustness of statistical abstractions: A look 'Under the hood' of statistical models and software, Harvard-MIT Data Center, manuscript.

Anderson, R. G., & Dewald, W. G. (1994). Scientific standards in applied economics a decade after the journal of money. *Credit and Banking Project, Fed. Res. Bank St. Louis Rev. 76*, 79–83.

Bollerslev, T., & Ghysels, E. (1996). Periodic autoregressive conditional heteroscedasticity. *Journal of Business and Economic Statistics 14*, 139–151.

de Hoon, et al. (1996). Why Yule–Walker equations should not be used for autoregressive modelling. *Annals of Nuclear Energy 23*, 1219–1228.

Dewald, W. G., Thursby, J., & Anderson, R. G. (1986). Replication in empirical economics: The Journal of Money. *Credit and Banking Project, American Economic Review 76*, 587–603.

Diebold, F. X. (1997). *Elements of Forecasting in Business, Economics, Government and Finance*, South-Western College Publishing, Cincinatti.

Doornik, J. A. (1998). *Object-Oriented Matrix Programming using Ox 2.0*, Timberlake Consultants Ltd, London and Oxford.

Feigenbaum, S., & Levy, D. (1993). The market for (ir)reproducible econometrics. *Social Epistemology 7*, 215–232.

Fiorentini, G., Calzolari, G., & Panattoni, L. (1996). Analytic derivatives and the computation of GARCH estimates. *Journal of Applied Econometrics 11*, 399–417.

Granger, C. W. J., & Newbold, P. (1986). *Forecasting Economic Time Series*, 2nd edition, Academic Press, Orlando.

Greene, W. H. (1993). *Econometric Analysis*, 2nd edition, MacMillan, New York.

Hamilton, J. D. (1994). *Time Series Analysis*, Princeton University Press, Princeton.

Knüsel, L. (1995). On the accuracy of the statistical distributions in GAUSS. *Computational Statistics and Data Analysis 20*, 699–702.

Knüsel, L. (1996). Telegrams. *Computational Statistics and Data Analysis 21*, 116.

Knüsel, L. (1998). On the accuracy of statistical distributions in Microsoft Excel. *Computational Statistics and Data Analysis 26*, 375–377.

Ling, R. (1974). Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association 69*, 859–866.

Lovell, M., & Selover, D. (1994). Econometric software accidents. *The Economic Journal 104*, 713–726.

MacKinnon, J.G. (2000). Computing numerical distribution functions in econometrics, in: Pollard, A., Mewhart, D., Weaver, D. (Eds.), *High Performance Computing Systems and Applications*, Dordrecht: Kluwer (forthcoming).

McCullough, B. D. (1998). Assessing the reliability of statistical software: Part I. *The American Statistician 52*, 358–366.

McCullough, B. D. (1999a). Assessing the reliability of statistical software: Part II. *The American Statistician 53*, 149–159.

McCullough, B. D. (1999b). Econometric software reliability: E-Views, LIMDEP, SHAZAM, and TSP. *Journal of Applied Econometrics 14*(2), 191–202.

McCullough, B. D. (1999c). Algorithms for (partial) autocorrelation coefficients. *Journal of Economic and Social Measurement 24*, 265–278.

McCullough, B. D. (2000). Experience with StRD: Application and Interpretation. In: Berk, K., & Pourahmadi, M. (Eds.), *Proceedings of the 31st Symposium on the Interface: Models, Prediction and Computing*, Fairfax, VA: Interface Foundation of North America, pp. 16–21.

McCullough, B. D., & Renfro, C. G. (1999). *Benchmarks and software standards: A case study of GARCH Procedures, Journal of Economic and Social Measurement 25*, 59–71.

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. (1999). On the accuracy of statistical procedures in Microsoft EXCEL 97. *Computational Statistics and Data Analysis 31*, 27–37.

Newbold, P., Agiakloglou, C., & Miller, J. (1994). Adventures with ARIMA software. *International Journal of Forecasting 10*, 573–581.

Priestley, M. B. (1981). *Spectral Analysis and Time Series*, Academic Press, London.

Sawitzki, G. (1994a). Testing numerical reliability of data analysis systems. *Computational Statistics and Data Analysis 18*, 269–286.

Sawitzki, G. (1994b). Report on the numerical reliability of data analysis systems. *Computational Statistics and Data Analysis (SSN) 18*, 289–301.

Tajima, A., Ninomiya, S., & Tezuka, S. (1997). On the Anomaly of Ran1() in Monte Carlo pricing of financial derivatives. In: Charnes, J., Morrice, D., Brunner, D., & Swaine, J. (Eds.), *Proceedings of the 1996 Winter Simulation Conference*, IEEE, Piscataway, NJ.

Thisted, R. A. (1988). Elements of Statistical Computing, Chapman and Hall, New York.

Tjøstheim, D., & Paulsen, J. (1983). Bias of some commonly-used time series estimates. *Biometrika 70*, 389–399.

Vinod, H. D. (2000). Review of GAUSS for Windows, Including its Numerical Accuracy, *Journal of Applied Econometrics*, to appear.

Wilkinson, L. (1985). Statistics Quiz, SYSTAT, Inc, Evanston, IL, available at http://www.tspintl.com/benchmarks.

Wilkinson, L., & Dallal, G. (1977). Accuracy of sample moments calculations among widely used statistical programs. *The American Statistician 31*, 128–131.