Practical Guidelines for Testing Statistical Software

Leland Wilkinson
SYSTAT, Inc. 1800 Sherman Ave., Evanston, IL 60201
and
Department of Statistics, Northwestern University

September 1, 1993. Published in P. Dirschedl and R. Ostermann (Eds.), Computational Statistics: Papers Collected on the Occasion of the 25th Conference on Statistical Computing at Schloss Reisensburg. Physica-Verlag, 1994.

Summary

Traditional tests of the accuracy of statistical software have been based on a few limited paradigms for ordinary least squares regression. Test suites based on these criteria served the statistical computing community well when software was limited to a few simple procedures. Recent developments in statistical computing require both more and less sophisticated measures, however. We need tests for a broader variety of procedures and ones which are more likely to reveal incompetent programming. This paper summarizes these issues.

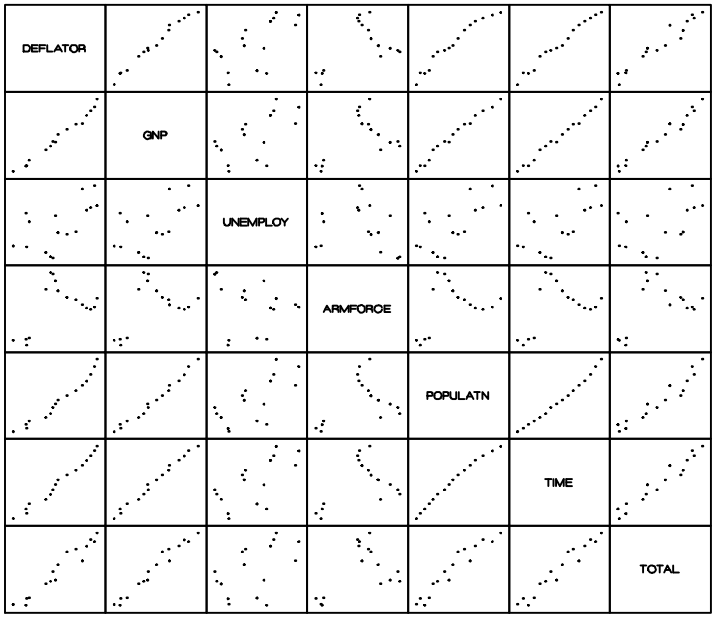
1. Introduction

Over twenty-five years ago, J.W. Longley (1967) submitted a simple regression problem to several widely used computer routines. The data were real economic series taken from government labor statistics, albeit blended in an imponderable linear equation. Longley's 16 case dataset looked innocuous but managed to clobber the most popular routines found in statistical computer libraries of that time. They can still foil some commercial statistics packages today.

Figure 1 shows a scatterplot matrix (SPLOM) of Longley's data. The ill-conditioning is apparent in the high correlations between predictors, particularly DEFLATOR, GNP, POPULATN, and TIME. And the bizarre distributions of ARMFORCE with the other predictors certainly merit further examination before linear modeling. A properly constructed prediction model obviously should not include all these predictors in their raw form. Some have pointed to this ill-conditioning as evidence that the Longley data are artificial and unlikely to be

encountered in practice. One need only talk to the technical support managers in any statistical software company to know that this claim is false, however. For better or worse, social scientists frequently attempt to fit data as ill-conditioned as Longley's and physical scientists often force ill-conditioned data through nonlinear models. Incredibly, it is not uncommon to see intercorrelations of .999 or greater among parameter estimates in published linear and nonlinear models. Software should be able to handle these situations and warn users when ill-conditioning can become a numerical or analytical problem.

Figure 1
Scatterplot Matrix of Longley Data



Several years after Longley, Wampler (1970) formulated a conditioning index for pushing regression routines to their limits. Wampler's paradigm has been used many times since to compare the performance of computer packages, although it measures only one narrow aspect of algorithm performance.

In the light of these developments, the Statistical Computing Section of the American Statistical Association recommended a comprehensive study of the performance of statistical packages (Francis, Heiberger and Velleman, 1975). This project, subsequently modified and published in monograph form (Francis, 1979,

1981) was the first systematic attempt to evaluate the performance of the software used in academics and industry for critical statistical applications.

Soon after the release of microcomputer statistical packages in the early 1980's, many of the mainframe tests were applied at various times to micro packages (e.g. Simon and Lesage, 1988). The results were in general more alarming than for the mainframe packages. Many developers of microcomputer packages had not participated in the mainframe era of software development or were not trained in numerical analysis and statistics. The microcomputer developers quickly reacted to these reports by modifying their regression algorithms to deal with ill-conditioning. The Longley data soon ran correctly on all but the most amateurish microcomputer packages.

Because most of the accuracy tests focused on regression, the majority of statistical package algorithms continued to be untested by third parties. There were a few exceptions. Wilkinson and Dallal (1977) used a paradigm from Anscombe (1967) to reveal accuracy problems in widely used mainframe statistical package sample moments and discriminant analysis calculations.

Bernhard et al. (1988) assessed the performance of nonparametric procedures in SPSS, SAS, BMDP, and SYSTAT. The results were disappointing. None of the packages handled the tests well, either in more sophisticated areas such as exact probability computations or simple tasks such as dealing with ties and missing data.

Another exception was a booklet, released by Wilkinson (1985), called *Statistics Quiz*. This booklet contained tests for graphics, data management, correlation, tabulation, as well as regression. The central feature of Wilkinson's booklet was a dataset called NASTY, shown in Table 1. The numbers in this dataset were extreme but not unreasonable. Each column was designed to reveal a different type of weakness in computational algorithms. LABEL contains character data. Well designed packages should be able to process character data and label cases intelligently. X contains simple integers. For summary statistics which compute exactly as integers, good packages should return integers (e.g. the mean should print as 5, not 4.999). Other operations which depend on integer values should not be confused by incorrect conversions from integer to floating point values. ZERO contains all zeros, a condition likely to cause zero divide and other singularity errors in badly designed software. MISS contains all missing values. Even packages which handle missing values may fail to deal correctly with a variable all of whose values are missing. BIG is a coefficient of variation problem. The significant variation is in the eighth digit. This means that programs in single precision will be incapable of

analyzing this variable. Even some badly designed double precision programs may fail. It is possible, of course, to construct a similar variable to foil double precision programs, but the practicality of such a test would be questionable. Coefficients of variation in real data seldom are less than 10^{-10} (see Hampel et al., 1986, for a discussion of the prevalence of gross errors and other characteristics of real data). LITTLE presents similar problems to BIG. It is in the dataset to reveal formatting difficulties in output routines. HUGE and TINY are included to reveal formatting problems as well. Finally, ROUND is included to reveal rounding algorithms. The three popular options for displaying numbers are cutting (truncation), rounding up, and rounding even. Cutting is the same as rounding down. Or, if you wish, rounding up is the same as adding .5 and cutting. Cutting ROUND into one digit produces the numbers 0,1,2,3,4,5,6,7,8. Rounding up produces the integers 1,2,3,4,5,6,7,8,9. Rounding even produces 0,2,2,4,4,6,6,8,8.

Rounding even is more appropriate for binary register arithmetic or any accumulation process with finite accumulator because it is unbiased over many calculations. For formatting of output, however, the cutting or round up strategy is less confusing to users. It makes more sense to cut or round up when the numbers displayed will not be further aggregated because this does not introduce granularity where it does not exist. Cutting is used, for example, in the stem-and-leaf diagram (Tukey, 1977). Tukey recommends both cutting and rounding even, depending on the display purpose. In any case, programs should perform these operations consistently. It is not that difficult, when converting binary to decimal, to cut or round incorrectly (e.g. 0,2,3,4,5,6,7,8,9). Negative rounding can also cause problems.

Table 1

NASTY Dataset from Wilkinson (1985)

LABEL	X	ZERO	MISS	BIG	LITTLE	HUGE	TINY	ROUND
ONE	1	0	*	99999991	.99999991	1.0E+12	1.0E-12	0.5
TWO	2	0	*	99999992	.99999992	2.0E+12	2.0E-12	1.5
THREE	3	0	*	99999993	.99999993	3.0E+12	3.0E-12	2.5
FOUR	4	0	*	99999994	.99999994	4.0E+12	4.0E-12	3.5
FIVE	5	0	*	99999995	.99999995	5.0E+12	5.0E-12	4.5
SIX	6	0	*	99999996	.99999996	6.0E+12	6.0E-12	5.5
SEVEN	7	0	*	99999997	.99999997	7.0E+12	7.0E-12	6.5
EIGHT	8	0	*	99999998	.99999998	8.0E+12	8.0E-12	7.5
NINE	9	0	*	99999999	.99999999	9.0E+12	9.0E-12	8.5

Sawitzki (1993) ran a number of packages through Wilkinson's tests and found serious deficiencies in several widely used statistical programs as well as in the statistical functions of Microsoft EXCEL. The results for the spreadsheet are significant because it is quite possible that more basic statistical calculations are done worldwide in EXCEL than in all statistical packages combined.

Wilkinson also included the simple datasets shown in Table 2. He described them as follows:

> "The following problem was encountered by the National Park Service. There are two files. One (BOATS) has names of boats (NAME), the day the left port (DEPART), and the day they returned home (RETURN). The other file (WEATHER) has the daily temperature (TEMP) for each day (DAY) of the year, numbered from 1 to 365 during the year the boats were out. Now, neither file has the same number of records, of course, but the BOATS file may have multiple records for each boat, since each went on one or more cruises during the year. Your task is to create a file (CRUISE) with a separate record for each boat and the average temperature (AVGTEMP) during its cruise. Warning: the weather for day 5 is missing. Ignore it in computing the average temperature."

Table 2

Data Management Problem from Wilkinson (1985)

BOATS			***WEATHER***	
NAME	**DEPART**	**RETURN**	**DAY**	**TEMP**
Nellie	1	10	1	48
Ajax	4	6	2	40
Queen	3	3	3	45
Ajax	2	3	4	52
			6	44
			7	48
			8	49
			9	50
			10	52
			11	50
			12	49

Surprisingly, few statistical packages have been able to compute this simple problem. This is because their architecture has been based on a spreadsheet model: rows for cases and columns for variables. Those packages which do not include programming languages or sophisticated graphical relational operators cannot handle simple data management problems like this.

Several other investigations of performance outside the regression area have been published. Searle (1979) and Wittkowski (1992) examined analysis of variance output for several packages. They uncovered numerous anomalies, particularly in the choice of error terms for unbalanced and incomplete designs.

2. The contemporary milieu

Two recent dynamics of the rapidly changing computer market have impeded the quality testing of statistical software . The first is the fragmentation of the user community. The second is the lack of interest in accuracy among users.

In the 1970's and early 1980's, the market was governed by a few companies distributing packages on mainframe computers. Several smaller companies existed (see Francis, 1983 for a brief survey), but the main user community was served by a select group of companies. During that time, the principal sources for high quality statistical code were few: Among them were UCLA's Health Sciences Computing

Facility (developers of BMDP), whose code for discriminant analysis, factor analysis, and other procedures found its way into SPSS, SAS, and other packages. Another source was government laboratories, such as Argonne Labs in the U.S. and the Harwell Physics and Numerical Algorithms groups in the UK, which produced certified numerical code. A few commercial organizations, such as IBM and IMSL, also produced certified code. The relative concentration of algorithm sources allowed more rapid monitoring and fixing of bugs and code defects.

In the last decade, published statistical algorithms have proliferated. While well designed algorithms have made their way into quality books and software libraries, there are still many sources which contain flawed code. The more reputable sources have contained warnings about their limitations, although many users have ignored these warnings. Press et al. (1988) for example, have written a widely used text on numerical procedures. They state in their introduction,

> "We call this book Numerical Recipes for several reasons. In one sense, this book is indeed a "cookbook" on numerical computation. However there is an important distinction between a cookbook and a restaurant menu. The latter presents choices among complete dishes in each of which the individual flavors are blended and disguised. The former - and this book - reveals the individual ingredients and explains how they are prepared and combined."

Despite warnings to the contrary, pedagogical code has been adapted for commercial applications with little attention to production issues (underflows, zero-divides, etc.).

The second problem, lack of user interest in accuracy, is more serious than the first. Several major computer trends have contributed to this situation. The proliferation of graphics, user-friendliness, and GUI interfaces have led users to abandon their loyalties to single packages. They expect statistical packages to conform to the metaphors inherent in their word processing, graphics, and spreadsheet software. Microsoft and Apple have come to write the rules for statistical software as much as for the rest of the market. This perspective, coupled with the traditional fear most people have for computers ("Garbage in garbage out"), has led to a laissez-faire attitude about numerical methods. "If the graphs look professional, the numerics probably are too."

In the past, companies identified the statisticians behind them or on their boards, (e.g. Brown, Dixon, Engelman, Hartigan, Hill, Jennrich, and Sampson at BMDP; Gentle and Kennedy at IMSL; Nelder and others at NAG; Goodnight and Sall at

SAS; Ryan at Minitab; Velleman at DataDesk; Polhemus at STATGRAPHICS; Becker, Chambers, Cleveland, Gale, Pregibon, Tukey, and Wilks for *S*). Some newer companies have refused to reveal the names of their developers. Unfortunately, reviewers of software have showed little interest in investigating the backgrounds of the companies and people producing commercial software. They would be surprised to discover that some companies selling widely advertised statistical software have no statisticians on their staff, despite their use of words like "lab" and "institute."

Two changes are needed in the testing of statistical software in order to improve this situation. First of all, we need to understand that selling statistical software is not like selling soft drinks. The reputation of companies is important in all business, but especially so in the area of technical software. We need to find a way to ferret out amateurish or sloppy developers. Second, we need to understand that statistical software quality is not simply a matter of computing regression examples to 15 digits of accuracy. Computing graphs correctly and accurately is just as critical as computing regression equations. We must expand our test suites.

How do we do this? I propose that we take a psychological approach. We need to understand the psychology of the opportunistic programmer. Opportunists motivated by money and fame take short cuts to achieve their goal. They use without modification "public domain" code from books, networks, and old computer tapes and libraries. Opportunists fail to recognize or hear about the well-known lore of the field, such as the spectral failings of the original RANDU uniform random number generator. And opportunists freely imitate other commercial packages by reverse engineering displays and interfaces without understanding their significance.

3. Practical tests

Instead of miscellaneous examples, I will present several maxims for helping one to devise effective tests. These maxims are generalizations and therefore often untrue. Some are, on the surface, mutually contradictory. I am presenting them to stimulate thinking about accuracy, however. The central principle of these maxims is that tests of statistics packages should focus on areas likely to affect users of real datasets. Artificial benchmarks are needed, but it is now time to move away from tiny datasets constructed to reveal specific numerical problems and toward larger and more realistic datasets which represent everyday applications more closely.

The maxims underscore another problem. Frozen standards will not improve performance of living software. Users and reviewers who make up tests unanticipated by developers will be more successful. For a forum on many of these issues, see Eddy et al., (1991).

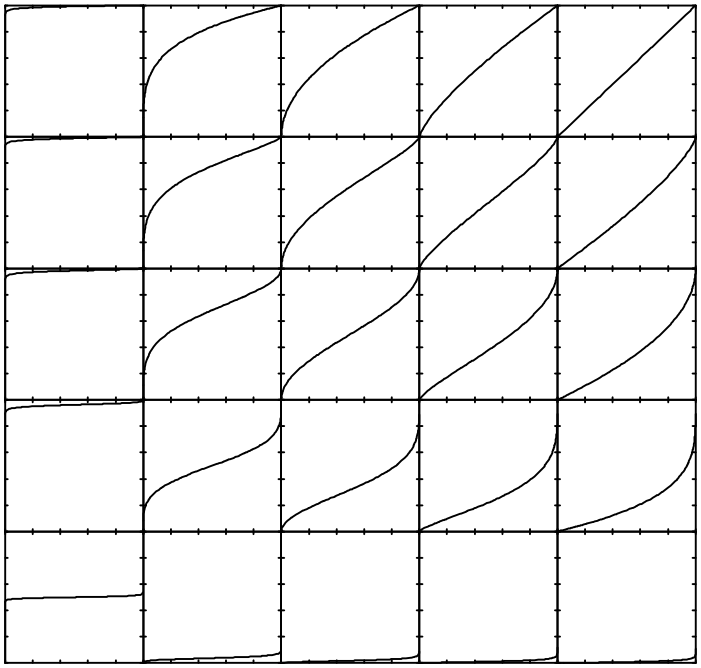
1. **Borrowed code is a ticking bomb**. The most likely place to trip up a poorly written package is in the places where code has been taken from textbooks or public sources. The reasons are obvious. Those who borrow computer code from others usually don't know what it was intended to do and where its limitations are.

For example, the FORTRAN code for a clustering tree display routine in one PC statistical package was taken from a textbook. The authors did not realize that the code applied only to ultrametric trees. Their program produced incorrect displays for several types of hierarchical clustering.

How do we reveal borrowed code? One way is to find known bugs. Popular sources of carelessly borrowed code - despite their copyrights - are *ACM Algorithms* and *Applied Statistics* algorithms (published by the Royal Statistical Society). These sources usually publish updates and corrections that are not always seen by those who hastily appropriate actual computer code.

Another place to look is in asymptotic performance of algorithms. Many published algorithms, even reviewed ones, fail to implement comprehensive error trapping. For example, probability distribution routines often have difficulty with extreme parameter values. Try computing beta values for ***B***(.001,.001,.001) and ***B***(.999,.001,.001). Figure 2 shows this distribution in the lower left corner. The cumulative distribution is almost a step function with these parameter values. It is tricky at these boundaries to asymptote properly (to 0 and 1 respectively). For packages lacking cumulative distribution functions, datasets can be constructed to test these distributions in the analytic sections (e.g. ANOVAs or *t* tests)..

Figure 2
Beta Cumulative Distribution Function
Rows and Columns Show Beta Parameter from .01 to .99



Poor asymptotic performance can also be a symptom of borrowing approximate textbook formulas instead of using more accurate series. For example, one package prints probabilities for the null distribution of the correlation coefficient in a handy "statistics calculator." It refuses to calculate a probability for a value larger than .98.

2. **Simple things are dangerous**. This is a corollary of the old mechanic's saying that the most dangerous tool is the one least feared. For example graphs seem to be the easiest part of a statistical package. They are the last place reviewers look for accuracy. Wilkinson's NASTY dataset has been used on numerical and some graphics routines. It should be used to test other applications which seem obvious. Any routines requiring sample moments should be subjected to the same tests used to evaluate moments calculations themselves.

3. **Imitation is neither sincere nor flattering**. Everyone's work depends on the prior contributions of others, but true imitators don't know what they are copying. We should beware of packages which do ordinary things exactly the same

way other packages do. This often means the imitators didn't know what they were doing.

One program assiduously imitated the STATGRAPHICS routines for distribution fitting. Like STATGRAPHICS, the program collapsed sparse categories for small sample sizes. The imitators failed to monitor the degrees of freedom correctly, however, and so produced negative chi-squares for small samples.

The same imitators copied a display called a "spectral plot" out of another statistical package. They didn't realize that the word "spectral" referred to the spectral decomposition in the graph being copied. Instead, they cut 3-D scatterplots into so-called "spectral planes," thus devising a similar looking plot which had nothing to do with a spectral decomposition. After this was pointed out to them, they continued to maintain their dubious use of the word rather than admit their mistake.

Finally, the same group devised a "box plot" that looks exactly like the Tukey schematic plot found in other packages. Instead of medians, hinges, and outer fences, the imitators used means, standard deviations, and ranges, perhaps daunted by the calculations needed to produce the correct box plot. Their box plots need little testing to discover they are invariably symmetric.

4. **Reviewers ignore the obvious**. A corollary of this statement is the magician Randi's observation that psychokinetic performers who bend spoons without touching them can fool Ph.D. physicists, but they can't fool magicians. Well intentioned reviewers, panels, certification committees, and others evaluating statistical software often focus intensively on a tiny part of a package while ignoring obvious tests of performance elsewhere. For example, nonlinear regression programs have been tested extensively using datasets from the numerical function minimization literature, but no reviewer has run the Longley data through nonlinear regression packages. Few nonlinear packages can compute the linear coefficients and their standard errors correctly for this dataset.

Other obvious pitfalls are easy to test. Submit a constant vector to a time series routine which requires first order differencing and observe subsequent calculations. Submit perfectly collinear data to a linear regression routine. Submit fully saturated single degree of freedom models to ANOVA routines. Feed an identity matrix to a principal components or factor analysis program.

5. **Easy now, hard later**. Incredibly, most purchasers of expensive statistical software decide to buy on an impulse. Either they are not spending their own money, or they are desperate. Reviewers, similarly, are most impressed by features

discovered in the first few minutes of using a package. Researchers at Apple Computer have found that the first five minutes of exposure to a software package determines whether it will be bought and/or used.

The problem with this instant gratification is that software which shines in the first five minutes can tarnish when needed for more complex problems later. Teitel (1986) organized a symposium on how statistical packages handle data management. He extended an open invitation to statistical software developers to solve his test examples. Only BMDP, PRODAS, P-STAT, SAS, SPSS, and SYSTAT successfully completed the problems. Some highly regarded packages did not even enter his contest because they could not handle the data structures. Wilkinson's simple data problem in Table 2 still cannot be handled by most commercial packages. An easy test would be to ask the developer to show how this problem can be solved. Presumably, developers should have the simplest solution, if it exists at all in their package.

Serious data management capabilities are not the exclusive province of command based packages. There is nothing in graphic-user-interface systems which prevents their doing powerful data manipulation. Nevertheless, most menued packages have avoided this area because of the programming difficulties.

6. **The results matter more than the algorithm**. Statistical package developers cannot always be trusted to document their algorithms correctly. There are many reasons for this. Often algorithms are more complex than a simple set of formulas will show. And sometimes developers deliberately conceal important details in order to keep them from competitors. Sometimes documentation departments are staffed by non-programmers or non-statisticians. Finally, companies can lie to impress customers. One company stated that a particular algorithm had been used in its software. This algorithm appeared in another package which it had imitated, but did not exist in the imitation.

Numerical analysts often favor a particular algorithm because of its theoretical performance on certain problems, while they ignore other areas. While error analysis, pioneered by Wilkinson (1965), can be extremely valuable for standardized comparisons, it does not solve all the performance assessment issues. Some reviewers have compounded this mistake by favoring one package over another after comparing documentation only. Even superior algorithms can be coded improperly.

The only sure comparison is on real data examples which are designed to exploit known weaknesses of algorithms. This way, incorrect choices of machine constants and iterative parameters will be exposed.

7. **No one knows everything**. A corollary of this maxim is that no one is interested in everything. A package which passes one test with flying colors may be substandard elsewhere. Companies and individuals tend to specialize. Eminent time-series statisticians, for example, may know little about experimental design. This is why one cannot generalize about the performance of a package from simple tests of regression accuracy.

One developer, wishing to implement probabilities for the Kolmogorov-Smirnov test, apparently stored a set of tabulated values from a textbook. Whenever values outside the table - *in any direction* - are input, the program reports the result as not significant (*n.s.*).

8. **Accuracy is not important**. What matters is how an algorithm handles *inaccuracy*, not accuracy itself. The principal point of Velleman and Francis (1975) and Beaton et al. (1976) is that computing results to many digits on artificial test datasets is not useful in itself. The measurement errors in real datasets usually far exceed the size of the errors produced by computational problems. More important is the performance of a routine across a wide set of real datasets, both large and small. For any algorithm, it is possible to construct a short artificial dataset to exploit its weaknesses. Tests like those in Wilkinson and Dallal's and Longley's papers, on the other hand, are ones that any well-designed package should be able to handle easily.

An extreme case of numerical silliness has appeared recently in the marketing literature of one statistics company. The correlation routines of this company's statistics package were found to be seriously inaccurate on the Wilkinson and Dallal (1977) tests. In response, the company produced a revised version which uses extended precision arithmetic. It now includes in its advertising an example containing variables whose coefficient of variation is less than 10^{-14}. The major statistics packages yield the same Pearson correlation on this dataset to within four digits. The company apparently thinks the remaining difference is of practical importance. Those who make up such examples should be challenged to provide real data that have a similar coefficient of variation and whose measurement error is known to be less than the variation recorded.

Computing accurate estimates on a trick dataset is no guarantee that a package is accurate. Like performance benchmarks - Whetstones, Dhrystones, etc. - accuracy benchmarks can vary widely and depend on complex assumptions. Furthermore, using extended precision with a bad algorithm will not always correct accuracy problems.

Computing accurate estimates for the Longley data is simple. Packages which cannot do this should not be tolerated. Even worse, however, is a package which computes artificial examples correctly but fails to detect mathematical collinearity. Such a package can produce completely false results.

3. Conclusion

It would be nice to have a test suite of problems for certifying statistical packages. Unfortunately, certification cannot keep up with new developments in computer science and statistics. A more promising approach is for users to construct specialized examples in various applications areas. This was the approach of Wilkinson and Dallal (1977) and, more recently, Wittkowski (1992). News of these tests spreads quickly and causes developers to pay attention.

More importantly, let's develop process models for statistical analysis which include real world datasets and typical applications. To assess regression performance, for example, we need to develop tests which reveal how much effort it takes to perform transformations, refit models, recognize outliers, stratify by subgroups, produce professional quality reports and graphs, and manage many large datasets.

The time of statisticians in white lab coats writing certified code has passed, if indeed it ever existed. Users need to discard myths of this sort and be especially wary of companies which use a cloud of mathematical and statistical jargon to disguise incompetence. If users don't investigate or care about the quality of statistical software, they will get they packages they deserve.

Note: MaryAnn Hill and Laszlo Engelman contributed valuable suggestions.

References

Anscombe, F.J. (1967). Topics in the investigation of linear relations fitted by the method of least squares. *Journal of the Royal Statistical Society, Series B, 29*, 1-52.

Beaton, A. E., Rubin, D.B., and Barone, J.L. (1976). The acceptability of regression solutions: Another look at computational accuracy. *Journal of the American Statistical Association, 71*, 158-168.

Bernhard, G., Alle, M., Herbold, M., and Meyers, W. (1988). Investigation on the reliability of some elementary nonparametric methods in statistical analysis systems. *Statistical Software Newsletter, 14*, 19-26.

Eddy, W.F., Howe, S.E., Ryan, B.F., Teitel, R.F., and Young, F. (1991). *The future of statistical software: Proceedings of a forum*. Washington, DC: National Academy Press.

Francis, I., Heiberger, R.M., and Velleman, P.F. (1975). Criteria and considerations in the evaluation of statistical program packages. *The American Statistician, 29*, 52-56.

Francis, I. (1979). *A comparative review of statistical software.* International Association for Statistical Computing, Voorburg, Netherlands.

Francis, I. (1981). *Statistical software: A comparative review*. North-Holland, New York.

Francis, I. (1983). A survey of statistical software. *Computational Statistics and Data Analysis, 1*, 17-27.

Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., and Stahel, W.A. (1986). *Robust statistics: The approach based on influence functions*. New York: John Wiley & Sons, Inc.

Longley, J.W. (1967). An appraisal of least-squares for the electronic computer from the point of view of the user. *Journal of the American Statistical Association, 62*, 819-841.

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. (1988). *Numerical Recipes*. Cambridge: Cambridge University Press.

Sawitzki, G. (1993). Numerical reliability of data analysis systems. Paper presented at the 25th meeting, SIG Computational Statistics of the International Biometrical Society, Reisensburg, Germany.

Searle, S.R. (1979). Annotated computer output for analysis of variance of unequal-subclass-numbers data. *The American Statistician, 33*, 222.

Simon, D., and Lesage, J.P. (1988). Benchmarking numerical accuracy of statistical algorithms. *Computational Statistics & Data Analysis, 7*, 197-209.

Teitel, R.F. (1986). Benchmarking vendor packages. In T.J. Boardman (ed.), *Computer Science and Statistics: Proceedings of the 18th Symposium on the Interface*. American Statistical Association, 193-229.

Velleman, P.F. and Francis, I. (1975). Measuring statistical accuracy of regression programs. In J.W. Frane (ed.), *Computer Science and Statistics: Proceedings of the 8th Symposium on the Interface*. UCLA Health Sciences Computing Facility, 122-127.

Wampler, R.H. (1970). A report on the accuracy of some widely used least squares computer programs. *Journal of the Americal Statistical Association, 65*, 549-565.

Wilkinson, J.H. (1965). *The Algebraic Eigenvalue Problem*. Oxford: Clarendon Press.

Wilkinson, L. (1985). *Statistics Quiz*. Evanston, IL: SYSTAT, Inc.

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

Wittkowski, K.M. (1992). Statistical analysis of unbalanced and incomplete designs - experiences with BMDP and SAS. *Computational Statistics & Data Analysis, 14*, 119-124.