Benchmarks and software standards: A case study of GARCH procedures

B.D. McCullough*  
Federal Communications Commission, 445 12th Street SW, Room 2C-134, Washington, DC 20554, USA.  
E-mail: bmccullo@fcc.gov

Charles G. Renfro  
Alphametrics Corporation, P.O. Box 2566, Bala Cynwood, PA 19004-6566, USA.  
E-mail: 74242.2260@compuserve.com

This paper addresses the evaluation of nonlinear methods in econometric software, taking GARCH procedures as a case study. In particular, it analyzes seven widely used packages, utilizing a recently developed benchmark. Four of the packages are found to be unsuitable, in most cases because the developer either does not specifically indicate which of the many possible GARCH models is being estimated, or does not accommodate the most common model specified in the applied literature, or both. A principal finding is that implementation of the GARCH procedure varies so widely that two packages ostensibly doing the same thing actually may be estimating substantively different models. This lack of standardization raises several questions concerning the evaluation of software. These include the issues normally associated with the creation of benchmarks, but also the critical role that software plays, and can play, in the development of modern econometrics.

1. Introduction

It is well-known that two different packages can produce two different solutions to the same estimation problem. Sometimes the discrepancies can be attributed to implementation, as in the treatment of corrections for AR(1) errors [15], which can take several forms including Cochrane-Orcutt, Prais-Winsten and Beach-MacKinnon. At other times no such possible reason for the discrepancies can be identified, which constitutes prima facie evidence that at least one of the packages is inaccurate.

In principle, a benchmark can resolve the question of accuracy: Silk’s [21] use of Calzolari and Panattoni’s [5] FIML benchmark is an example of this. However, benchmarks can be of more use than simply to determine the accuracy of software; they can also assist in setting standard features which econometric software

*Thanks for comments to C. Brooks, C. Cummins, J. Doornik, G. Fiorentini, C. Hallahan, M. Kim, and R. Schoenberg. The views expressed herein are those of the authors and do not necessarily reflect those of their employers.

0747-9662/98/$8.00 © 1998–IOS Press. All rights reserved
should possess, such as defaults and options for econometric procedures. This often-overlooked function of benchmarks has become more important in recent years with the proliferation of sophisticated nonlinear econometric procedures.

Thirty years ago econometric software incorporated techniques which were essentially linear in nature. In such a world, it was possible to focus benchmark evaluation on the issue of algorithmic implementation, as demonstrated by the Longley [14] benchmark. As econometric software began to incorporate nonlinear estimation, the creation of benchmarks became technically more difficult, but there could still be widespread agreement as to how a benchmark model should be specified and how it should be estimated, e.g., Donaldson and Schnabel [8]. Today, however nonlinear procedures are so complex (GMM, ARIMA, GARCH, etc.) that the question of how a benchmark should be specified is no longer clear-cut. For such procedures a benchmark cannot be merely a set of exceedingly accurate estimates. A broader context is required.

For complex econometric procedures, benchmark models should be chosen with two questions in mind, one positive and the other normative: What models can software packages estimate? and What models should they be able to estimate? This positive/normative distinction is the crux of the larger context in which benchmark models need to be considered.

At first sight it might seem that appropriate guidance can be easily obtained. Ostensibly, What models can be estimated? should be evident from the package’s documentation, and What models should a package be able to estimate? is surely just a matter of consulting the econometrics literature. However, in the first case, user manuals tend to be less than comprehensive in their treatment of computational details, especially for the more sophisticated econometric procedures. Sometimes the documentation is even incorrect. In the second case, determining the set of estimatable models actually depends upon the degree of consensus in the literature: if an authoritative survey has been written or the procedure is so well known as to be treated as a matter of course in textbooks, there is likely to be a standard specification; arguably, all packages should be able to estimate such a model – but whether they can is a different question. Even in the case of standard procedures, there are likely to be both subtle and not-so-subtle variations that may have statistical or numerical consequences.

Considering the creation of a standardized benchmark, there are several issues that must be resolved. Frequently, it will be necessary to carefully determine, and possibly restrict, the range of candidate estimation options. A person writing a benchmark program might even select a single specification and then estimate that model with benchmark precision. Of course, making this choice imposes the requirement that at least some of the packages tested must be able to estimate this model – otherwise, how might any package’s results be compared to the benchmark? This requirement would seem to restrict the choice to only standard specifications, if any such exist. However, as indicated earlier, for there to be a standard specification essentially implies a well developed literature, which also raises the question, How do we know
packages are reliable during the process of the infancy or adolescence of a procedure? In fact, benchmarks that point to the significant questions are needed at each stage of a procedure’s progressive adoption. In this way, benchmarks can be of more use than simply measuring accuracy: they can also help set standards for software implementations of econometric procedures, which is, incidentally, an important part of replicability [6,10,20].

With these considerations in mind, this paper attempts to benchmark the GARCH procedures in several software packages. Since our focus is software packages and benchmarks in general, we do not identify specific packages by name. To fix ideas, Section Two describes the types of problems encountered when using these packages to conduct default estimation of a simple GARCH model. Section Three then discusses the basics of GARCH estimation, with attention to computational details and sources of numerical error. Section Four presents a benchmark for GARCH estimation due to Fiorentini, Calzolari and Panattoni (1996, hereafter FCP). Section Five applies the benchmark to the several packages. Section Six presents the conclusions.

2. Default estimation

An obvious empirical requirement for a benchmark is an appropriate dataset. Such a dataset should be well-known and readily available. Bollerslev and Ghysel’s (1996, hereafter BG) 1974 observations on the daily percentage nominal returns for the Deutschemark/British pound exchange rate can be used (the “BG data,” which are available at the Journal of Business and Economic Statistics archive, ftp://www.amstat.org).

All packages should be able to estimate the basic GARCH model, a constant with GARCH(1,1) errors. In many papers the model is presented as

$$y_t = \mu + \epsilon_t$$

where $$\epsilon_t | \Phi_{t-1} \sim N(0, h_t)$$ and

$$h_t = \alpha_0 + \alpha_1 \hat{\epsilon}_{t-1}^2 + \beta_1 \hat{h}_{t-1}$$

We note two important features of this model. First, it maximizes not the likelihood, but the conditional likelihood. Second, this is only a partially-specified model, because the elements on which the likelihood is conditioned are not specified: initial values for $$\hat{\epsilon}_0^2$$ and $$\hat{h}_0$$ are not given. In fact, the initialization of the series $$\hat{\epsilon}_t$$ and $$\hat{h}_t$$, though often overlooked, can substantially affect the “solution” produced by the software. Two common methods of initialization are: (1) discarding observations; and (2) estimating the unconditional expectation of the series. Of necessity, all packages implement a default initialization, and so can estimate some GARCH(1,1) model, but which one of the many possibilities often is unclear for not all packages mention what initialization is used – this is a serious omission.
For the above simple model, Table 1 presents default coefficient estimates for the seven packages we have evaluated. There appears to be some agreement, but the need for a benchmark is apparent – the difference cannot be attributed to rounding error. One source of discrepancy is that the packages are not all maximizing the same GARCH function. Different conditional likelihood functions, which arise from different initializations for $\epsilon_0^2$ and $h_0$, lead naturally to different parameter estimates. Another possible source of discrepancy is numerical error, which is discussed in detail in McCullough and Renfro [18].

At this point, it is useful to pose a very general question. Initially, one might think that there is a simple GARCH conditional likelihood well-established in the literature and that all packages could estimate it by invoking the appropriate options. In such a case, it would be possible to attribute discrepancies solely to numerical error. There is such a GARCH conditional likelihood, but many packages cannot estimate it because they do not have options to permit its estimation. Thus the question is raised, What options should GARCH procedures have?

Consider also Table 2, which presents default $t$-statistics on the coefficients. Here there is less agreement, and the need for a benchmark is even more apparent. In addition to the above-mentioned sources of discrepancy, there are two more. First, the differing parameter estimates can lead to different standard error estimates. Second, there are at least five different methods for computing standard errors for GARCH coefficients (which can be computed with varying degrees of accuracy), and packages often do not mention what method is used – another serious omission.

In order to consider the various methods, let $g(\theta)$ and $H(\theta)$ be the gradient and Hessian of the conditional likelihood function, respectively, and let $Q$ be an estimator

Table 1  
GARCH estimates

<table>
<thead>
<tr>
<th>package</th>
<th>$\mu$</th>
<th>$\alpha_0$</th>
<th>$\alpha_1$</th>
<th>$\beta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>-0.00540</td>
<td>0.0096</td>
<td>0.142</td>
<td>0.821</td>
</tr>
<tr>
<td>X2</td>
<td>-0.00608</td>
<td>0.0098</td>
<td>0.144</td>
<td>0.818</td>
</tr>
<tr>
<td>X3</td>
<td>-0.00624</td>
<td>0.0108</td>
<td>0.153</td>
<td>0.806</td>
</tr>
<tr>
<td>X4</td>
<td>-0.00619</td>
<td>0.0108</td>
<td>0.152</td>
<td>0.806</td>
</tr>
<tr>
<td>X5</td>
<td>-0.00613</td>
<td>0.0107</td>
<td>0.153</td>
<td>0.806</td>
</tr>
<tr>
<td>X6</td>
<td>-0.00919</td>
<td>0.0098</td>
<td>0.144</td>
<td>0.818</td>
</tr>
<tr>
<td>X7</td>
<td>-0.00619</td>
<td>0.0108</td>
<td>0.153</td>
<td>0.806</td>
</tr>
</tbody>
</table>

Table 2  
GARCH $t$-stats

<table>
<thead>
<tr>
<th>package</th>
<th>$\mu$</th>
<th>$\alpha_0$</th>
<th>$\alpha_1$</th>
<th>$\beta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>-0.64</td>
<td>8.01</td>
<td>11.09</td>
<td>53.83</td>
</tr>
<tr>
<td>X2</td>
<td>-0.72</td>
<td>3.80</td>
<td>5.80</td>
<td>26.34</td>
</tr>
<tr>
<td>X3</td>
<td>-0.74</td>
<td>8.13</td>
<td>10.95</td>
<td>48.64</td>
</tr>
<tr>
<td>X4</td>
<td>-0.74</td>
<td>8.15</td>
<td>10.97</td>
<td>48.61</td>
</tr>
<tr>
<td>X5</td>
<td>-0.73</td>
<td>5.58</td>
<td>7.91</td>
<td>36.96</td>
</tr>
<tr>
<td>X6</td>
<td>-1.08</td>
<td>8.09</td>
<td>10.77</td>
<td>45.90</td>
</tr>
<tr>
<td>X7</td>
<td>-0.67</td>
<td>1.66</td>
<td>2.86</td>
<td>11.12</td>
</tr>
</tbody>
</table>
of the covariance matrix. The choice $Q = g(\theta)g(\theta)'$ produces the outer product of the gradient (OP) estimator which is the basis of the BHHH (Berndt-Hall-Hall-Hausman) method. The BHHH method is the usual Gauss-Newton method applied to maximizing a likelihood rather than minimizing a sum of squares. Direct use of the Hessian (H) such that $Q = -H(\theta)$ produces the Newton-Raphson method. Choosing $Q = E[H(\theta)]$ (the negative of the information matrix (IM)) produces the method of scoring. These are the three usual estimators for nonlinear procedures.

In the context of maximum likelihood estimation, there are two others. The quasi-maximum likelihood (QMLE) estimator is $Q = H^{-1}gg'gH^{-1}$ and the Bollerslev-Wooldridge (BW) estimator is $Q = I^{-1}(g'g)I^{-1}$. The accuracy of these various covariance matrix estimators is affected not only by the accuracy of the coefficient estimates, but also by the method of calculating derivatives: using finite-difference approximations, rather than exactly calculated analytic derivatives, induces numerical error in the estimated standard errors of the coefficients.

It is worth noting that estimation schemes vary in the amount of derivative information used. For example, Newton-Raphson requires explicit calculation of the function value, $L(\theta)$, the gradient, and the Hessian. The BHHH method, by contrast, requires only explicit calculation of the function and the gradient, as do quasi-Newton methods such as Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms. Moreover, the method by which derivatives are calculated matters. In particular, analytic derivatives are more accurate than numerical derivatives and lead to more accurate estimation [12, pp. 285].

An analytic gradient leads to more accurate estimates than a numerical gradient [1, p. 117]. Finite-difference approximation of the gradient should only be used when analytic derivatives are not available, either because there is no closed-form expression for the derivatives or the function is too complex to differentiate. The most accurate way to calculate $H(\theta)$ is by analytic differentiation of an analytic gradient. This is more accurate than a finite-difference approximation based on an analytic gradient which, in turn, is more accurate than a finite-difference approximation based on a numerical gradient.

Some packages offer misleading information on this point, suggesting that the difference between numerical and analytic derivatives is only speed or a negligible loss of accuracy. Anonymous quotes from two user manuals demonstrate this. “[Numerical derivatives] provide sufficient accuracy to obtain appropriate solutions to the problems. However, it is relatively slow compared to analytic derivatives.” “Analytic [derivatives] will, in general, be slightly more accurate than numeric derivatives, if the calculation of analytic derivatives has been carefully optimized by hand to remove common subexpressions.” Both these statements are true only in specific cases, and should not be taken as general rules.

---

1It is true that, with proper programming, numerical derivatives can be practically as good as analytic derivatives. However, this proper programming entails more than the simple forward difference or even central difference approximations commonly found in statistical and econometric software. See Quandt [19, pp. 731–735].
The general rule is that if the gradient is analytic, then the Hessian may be calculated numerically from the gradient with no loss of accuracy. This is supported by Monte Carlo evidence [8]. However, this is only a general rule and it does not apply to GARCH estimation. For example, Bollerslev, Engle and Nelson [4, § 2.2] observe that numerical second derivatives for GARCH models tend to be unstable. For this reason, they advocate the use of methods which avoid calculation of second derivatives, such as BHHH.

Tables 1 and 2 obviously demonstrate the need for careful evaluation, especially since documentation sometimes is unclear and even incorrect. We will give specific examples to support these statements.

3. The basic GARCH model

The GARCH model has many extensions, but the basic GARCH($p$, $q$) model is given by

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

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

The conditional likelihood function for this model can be written as

$$L_t(\theta) = \sum_{t=1}^{T} l_t(\theta)$$  

$$l_t(\theta) = -\frac{1}{2} \ln h_t - \frac{1}{2} \frac{\epsilon_t^2}{h_t}$$

As given, the model is only partially specified and therefore not estimatable. In order to fully specify the model, the initialization of the series $h_t$ and $\epsilon_t^2$ must be defined. This is a particularly important implementation issue because, as can be seen from (4), these initial values enter the conditional likelihood function. We note that this is a separate issue from the determination of starting values for an iterative, nonlinear estimation procedure. Lumsdaine (1996, lemma 6) shows that dependence on the initialization is asymptotically negligible. Perhaps for this reason, many authors do not even discuss the initialization in their GARCH papers, though this is tantamount to not identifying the conditional likelihood function on which reported estimation results are based. To be explicit, ceteris paribus, different initializations lead to different parameter estimates because different conditional likelihoods are being maximized. It is intuitively obvious that reported results do depend on the initialization, since in finite samples the initialization is not necessarily negligible and can even be substantial, as shown by Diebold and Schuermann [7]. The initialization also is an important consideration when benchmarking accuracy is the objective.
Many choices for this initialization have been proposed in the literature. One such initialization is

\[ h_t = \epsilon_t^2 = \frac{1}{T} \sum_{s=1}^{T} \epsilon_s^2 = \frac{SS}{T}, \quad t \leq 0 \]  

An initial vector of residuals for \( \epsilon_s \) can be computed from a preliminary OLS regression. Equations (1–5) constitute FCP’s “Model I” (they consider three models). It is not unreasonable to suggest that every GARCH procedure should be able to estimate Model I for two reasons. First, a minimal degree of flexibility is thereby insured. Second, the existence of the FCP GARCH benchmark allows users to assess the accuracy of their packages. The primary reason a package might not be able to estimate this model is because the developer has decided not to allow the user to control the initialization. Such packages are of questionable value, since the user cannot specify the GARCH likelihood to be maximized. Some packages have options for various initializations, but this is still unsatisfactory. The researcher might well need an initialization for which the developer has not provided an option – thus, the need for flexibility is apparent.

We note that Model I does not consider an important case that occurs frequently when higher-order GARCH models are estimated: the imposition of stationarity conditions (which involves estimation with nonlinear constraints, and is technically much more difficult than unconstrained estimation). The implementation of analytic derivatives then becomes problematic, and estimation may well have to proceed with numerical derivatives. The same may hold for GARCH procedures that permit missing observations – analytic derivatives become problematic and numerical derivatives may be preferred for computational reasons. Thus, some loss of accuracy may be the price paid for estimating more complicated models. However, in the case of higher-order models, such models most often are used for specification testing, and the final form frequently is a GARCH model for which analytic derivatives are known.

4. The benchmark

It is well-known that the GARCH likelihood is a complicated and highly nonlinear function \([4,9]\), which is an implicit argument for the use of numerical derivatives. However, at least for the standard GARCH(1,1) model, analytic first derivatives are easily derived \([13, \S 18.5]\). In a recent paper, Fiorentini, Calzolari and Panattoni \([11]\) paved the way for substantially more accurate GARCH estimation by providing readily useable, closed-form expressions for the first- and second-derivatives of the

\footnote{It is unclear whether Bollerslev \([2]\) held \( SS \) fixed at the value provided by the preliminary regression, or whether \( SS \) was updated with current parameter estimates on each iteration. This demonstrates the need for a benchmark model at every stage of a procedure’s development.}
GARCH conditional likelihood. For their Monte Carlo study, they wrote FORTRAN code to estimate three types of GARCH models with analytic derivatives. Their Monte Carlo study reached three conclusions which are relevant for present purposes:

- For methods which use an analytic gradient and a Hessian, numerical calculation of the Hessian is inferior to analytic calculation of the Hessian. Specifically, numerical second derivatives tend to be unstable.
- The OP estimator of the standard error is inferior to methods based on second derivative information, especially the QML and BW estimates.
- Faster convergence is achieved by using one algorithm to start and another to finish than by using one algorithm throughout the iterative process. Such use of two different algorithms often is referred to as a “hybrid algorithm” or a “mixed method.”

Most importantly, though, they provide code for estimating a benchmark model. The FCP code, applied to the BG data, constitutes a benchmark. Model I, which corresponds roughly to Bollerslev’s [2] seminal formulation, is estimated:

\[ y_t = \mu + \epsilon_t \quad \text{where} \quad \epsilon_t | \Phi_{t-1} \sim N(0, h_t) \quad \text{and} \quad h_t = \alpha_0 + \alpha_1 \hat{\epsilon}_t^2 + \beta_1 h_{t-1}. \]

We need six starting values for: \( \mu, \alpha_0, \alpha_1, \beta_1, \hat{\epsilon}_0^2 \) and \( \hat{h}_0 \). Two natural choices are: \( \mu = -0.016427 \), the sample mean of \( y \); and \( \alpha_0 = 0.221130 \) the sample variance of \( y \). The values \( \alpha_1 = 0.35 \) and \( \beta_1 = 0.50 \) were chosen after examining several possibilities, all of which led to the same solution. The least squares regression of \( y_t \) on a constant yields a sum of squared errors which, divided by the sample size, produces the starting value for both \( \hat{\epsilon}_0^2 \) and \( \hat{h}_0 \), in accordance with Eq. 5. The FCP code uses \( ||\hat{\theta}^k - \hat{\theta}^{k-1}|| < \epsilon ||\hat{\theta}^{k-1}|| \) as a stopping rule with convergence tolerance \( \epsilon = 10^{-9} \). Successful convergence is tested by examining whether the squared norm of the gradient is less than \( \epsilon \), i.e., the convergence criterion is \( ||g(\hat{\theta})||^2 < \epsilon \). Results to six significant digits are presented in Table 3. We note that BG estimated the same model on the same data, and obtained the following results (QMLE standard errors in parentheses): \( \mu = -0.006 (0.009) \), \( \omega = 0.264 (0.075) \), \( \alpha_1 = 0.153 (0.054) \), and \( \beta_1 = 0.806 (0.073) \) where, in terms of the original model, \( \omega = \alpha_0 / (1 - \alpha_1 - \beta_1) \). These results are consistent with the FCP benchmark.

### 5. Applying the benchmark

Relying on defaults does not permit the packages to be compared, since no two of the packages use the same defaults. This is not surprising since defaults can be

---

3 Again, we note that Bollerslev might have held \( \hat{\epsilon}_0^2 \) and \( h_0 \) fixed, but this is unclear.

4 We would prefer to report more digits using the methodology employed by NIST for the nonlinear least squares problems in the StRD: use two different implementations of two algorithms in quadruple precision and report the common digits.
Table 3
FCP GARCH Benchmark

<table>
<thead>
<tr>
<th>coefficient</th>
<th>H</th>
<th>OP</th>
<th>QMLE</th>
<th>IM</th>
<th>BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu )</td>
<td>-6190.41E-2</td>
<td>.846212E-2</td>
<td>.843359E-2</td>
<td>.918935E-2</td>
<td>.837628E-2</td>
</tr>
<tr>
<td>( \sigma_0 )</td>
<td>.107613E-1</td>
<td>.285271E-2</td>
<td>.132298E-2</td>
<td>.649319E-2</td>
<td>.192881E-2</td>
</tr>
<tr>
<td>( \alpha_1 )</td>
<td>.153134E-0</td>
<td>.265228E-1</td>
<td>.139737E-1</td>
<td>.535317E-1</td>
<td>.194012E-1</td>
</tr>
<tr>
<td>( \beta_1 )</td>
<td>.805974E-0</td>
<td>.335527E-1</td>
<td>.165604E-1</td>
<td>.724614E-1</td>
<td>.218399E-1</td>
</tr>
</tbody>
</table>

Note: What is surprising is that four of the seven packages could not estimate Model I, because they do not provide the necessary options. This is particularly noteworthy in the case of Package X1, the documentation for which makes the claim that the user can supply all the necessary starting values. We subsequently telephoned the developer and received confirmation that the documentation is incorrect: the user can supply starting values for the coefficients, but cannot choose the initialization.

As has been discussed, all four of these packages are of little use for serious research.

The remaining three packages (X4, X5 and X7) can be evaluated using the FCP Benchmark. The points of evaluation are the accuracy of the coefficients and the accuracy of their standard errors. We could simply present the estimates for each package and let the reader compare them to Table 3. Ambitious readers would then count the number of accurate digits, perhaps noting them in the margin for later comparison.

There is an easier way to present the same information. A useful way to measure the number of digits of accuracy is to use the log relative error

\[
LRE = -\log_{10}(\frac{|x - c|}{|c|})
\]

where \( x \) is the estimated value and \( c \) is the benchmark value. The LRE is a measure of the number of accurate digits only if the benchmark value and the estimated value differ by a factor of less than two, so that at least the first digits agree. In two circumstances the LRE is reported as zero: if the first digits do not agree, or if the computed LRE is less than unity. Each of the three packages is used to estimate Model I, and the LREs for the coefficients and standard errors are computed. Results are presented in Table 4.

It is clear that Package X7, which uses Newton-Raphson with analytic gradient and analytic Hessian, hits the benchmark precisely. Package X5, which uses a quasi-Newton method with an analytic gradient, performs reasonably well. Package X4, which uses Newton-Raphson with an analytic gradient with a numerical Hessian, achieves less success than the other packages. However, when provided with a circulating draft of this paper, the developer decided to upgrade the GARCH procedure, and now reports the results given under X4a – a significant improvement.

In fact, other developers also have followed the example of package X4 and retooled their GARCH procedures to accommodate the FCP benchmark after seeing this paper. This does not imply that their previous procedures were “wrong,” for there are many
Table 4
Accurate Digits on the FCP GARCH Benchmark, Various Packages
\( y_t = \mu + \epsilon_t, \epsilon_t|\Phi_{t-1} \sim N(0, h_t), h_t = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 h_{t-1} \)

<table>
<thead>
<tr>
<th>package</th>
<th>parameter</th>
<th>coefficient</th>
<th>H</th>
<th>OP</th>
<th>QMLE</th>
<th>IM</th>
<th>BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>X4</td>
<td>( \mu )</td>
<td>3.6</td>
<td>2.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_0 )</td>
<td>2.3</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_1 )</td>
<td>2.4</td>
<td>2.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \beta_1 )</td>
<td>3.6</td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X5</td>
<td>( \mu )</td>
<td>3.2</td>
<td>4.0</td>
<td>4.3</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_0 )</td>
<td>3.4</td>
<td>3.3</td>
<td>3.4</td>
<td>3.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_1 )</td>
<td>4.1</td>
<td>3.8</td>
<td>3.9</td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \beta_1 )</td>
<td>4.5</td>
<td>3.6</td>
<td>3.7</td>
<td>4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X7</td>
<td>( \mu )</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_0 )</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_1 )</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \beta_1 )</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X4a</td>
<td>( \mu )</td>
<td>2.6</td>
<td>3.4</td>
<td>4.3</td>
<td>2.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_0 )</td>
<td>4.2</td>
<td>4.2</td>
<td>4.3</td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \alpha_1 )</td>
<td>4.4</td>
<td>4.4</td>
<td>4.3</td>
<td>4.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \beta_1 )</td>
<td>5.0</td>
<td>4.4</td>
<td>4.5</td>
<td>4.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"correct" specifications. However, it does serve as compelling evidence that a well-chosen benchmark can serve as a useful means for standardizing software. This, in turn, has two important implications: the developer can appeal to the existing benchmark as a demonstration of reliability; and it enhances the replicability of economic research.

If a developer should choose not to offer Model I, he should: (1) describe the fully specified GARCH model being estimated; and (2) provide an independent benchmark (with source code) so that users can verify the implicitly asserted accuracy of the procedure.

6. Conclusions

During the past twenty-five to thirty years that forms the period during which econometric software has become widely available and commonly used by economists, there has been considerable innovation in both econometric techniques and the implementation of those techniques in software. In general, the focus of software evaluation, mainly in the form of reviews, has been upon this demonstration of new ideas. Much less attention has been paid to issues such as the numerical accuracy of programs or normative considerations, such as what features an econometric software package should offer its users.

However, one of the implications of the results that we have presented is that even in the case of a relatively standard specification, a large variation in numerical results can be observed. Although it is not possible to draw general inferences from this case,
two points should be made. First, that the results we have presented were obtained from an entirely arbitrary choice: we did not perform a series of tests of a variety of econometric procedures; we have made only one set of tests and this paper describes our findings. Second, as elsewhere noted by McCullough [17] and Renfro [20], this paper represents an unusual contribution: during the whole of the past 30 years, there has been surprisingly little testing done by economists of the numerical accuracy of econometric packages or of the suitability of particular packages for the application to which they have been put. Thus we cannot rule out the possibility that a series of benchmark tests, considering a variety of procedures, could expose a wide range of problems. For this reason we encourage others to join in the numerical and more general evaluation of econometric software procedures.

In presenting our results, we have been concerned not simply to offer a benchmark, but to try to demonstrate how the design of a software package can make a difference in terms of the usability of the software. As discussed, particularly in the case of nonlinear techniques, it is not sufficient simply to present a single set of results as definitive. Software users need to understand that nonlinear techniques raise the possibility of local, rather than global convergence, as well as a series of potential problems relating to the specifics of the algorithmic implementation. We have discussed the distinction between analytic derivatives and finite approximations, and have noted that this raises numerous complexities, one of the reasons being that the structure of the problem in a particular case cannot necessarily be determined in advance. It is possible to assert that, in general, analytic derivatives are to be preferred on the basic of numeric/analytic considerations; however, someone – the user in many cases – must take the responsibility for specifying these. Nonlinear estimation packages cannot always be presented to users as “canned” procedures; this fact makes it important to begin to consider the design and development of packages as a broadly defined set of characteristics.

In the introduction, we established two questions as critical to the development of useful benchmarks: What specifications can a tested package estimate? and What specifications should a package be able to estimate? The issue of which specifications can be estimated is central to the establishment of any benchmark: obviously, benchmarks are most useful to the degree that they establish a standard for existing packages. However, the issue of which specifications a package should be able to estimate is much more centrally normative and transcends the design of software packages. It is a general rule that users of software packages are dependent upon those packages to determine the range of estimatable models. As a consequence there is a higher degree of symbiosis between the development of econometric theory and the development of software than has heretofore generally been recognized. This paper represents an initial attempt to begin to better define this interaction between benchmarks, software standards, and econometric theory using the GARCH model as a case study.
7. Computational details

All packages were run on a 166 Mhz Pentium under Windows 95. The FCP code was compiled using Lahey Fortran 90, with two switches invoked: -dble, to extend all real and complex numbers to double precision; and -o0, to suppress all optimization. The FCP code was altered in the following way. In both VSGARCMX.FOR and GSGARCOV.FOR there is a function VALUNC which is amended twice. First, the function is declared as REAL*8 so “FUNCTION VALUNC(IDYNAM, C, ...)” becomes “REAL*8 FUNCTION VALUNC(IDYNAM, C, ...)”. Second, the value of LOG(SQRT(2*PI)) is given only to six digits, which makes LogL accurate to only that many digits. Therefore, about four lines before the end of VALUNC the constant 0.9189385 is changed to 0.9189385332046725D0.

References


