On The Accuracy of GARCH Estimation in R Packages

Chelsey Hill[♣] and **B. D. McCullough**[*♣]

[♣]Department of Decision Sciences & MIS, Drexel University

Submitted: February 4, 2019 • Accepted: October 12, 2019

ABSTRACT: The R software is commonly used in applied finance and generalized autoregressive conditionally heteroskedastic (GARCH) estimation is a staple of applied finance; many papers use R to compute GARCH estimates. While R offers three different packages that compute GARCH estimates, they are not equally accurate. We apply the FCP GARCH benchmark (Fiorentini, Calzolari and Panattoni 1996), proposed by McCullough and Renfro (1999), which uses the Bollerslev and Ghysels (1996) daily returns data, on three R packages: fGarch, rugarch, and tseries.

JEL classification: C22, C58, C87

Keywords: algorithms, benchmark, software accuracy, GARCH

Introduction

Generalized autoregressive conditionally heteroskedastic (GARCH) models are especially popular models in the areas of economics and finance, although their application spans across many subject areas. In addition to their wide-ranging applications, their use in published research remains strong in recent years, as evidenced in Figure 1. Due to the complexity of GARCH, which involves non-linear estimation techniques, modeling is reliant on software implementation.

[*]Corresponding Author. Email: bdmccullough@drexel.edu

There are many commercial and open-source options for software programs that can estimate a GARCH model, including R, SAS, Matlab, Python and Stata. Due to the open-source nature of Python and R, there are 2 and 3 packages, respectively, that can fit a GARCH model. In this research, we specifically focus on the R software package, in which there are three distinct packages in which a univariate GARCH model can be fit: tseries, fGarch and rugarch.

Software choice can vary, but criteria often include popularity. To assess the popularity of the three packages, we consider formal citations, informal citations, download and dependencies. Based on citation data gathered in April 2019, the most frequently cited of the three packages is tseries. The tseries package documentation was published in 2007 and has a total of 211 citations, with 2018 producing the highest number of citations to date. The fGarch package, which also has package documentation citations beginning in 2007, has a total of 37 citations and the newest package, rugrach, has a total of 27 citations since its release in 2012.

Formal citations of software package documentation may be an unreliable measure of use, however, due in part to the consideration of software as technical, rather than scholarly contributions (Soito and Hwang, 2016). As a result, software packages are infrequently cited, if at all, in research. For this reason, a Google Scholar search from 2014-2018 of the package name keywords is depicted in Figure 2, which captures in-text references to the R packages. While fGarch displays largely stable citations over time, rugarch exhibits an increasing trend and tseries prevalence in the literature has also increased in recent years.

Considering the discrepancies between formal and informal citations of the packages in existing literature, we also evaluate the popularity of the packages using two measures based on the comprehensive R archive network: package downloads and dependencies. Using the dlstats package, monthly download metrics were collected for the three packages, as displayed in Figure 3. As shown, the monthly number of tseries downloads far outweighs the downloads of the other two packages. tseries demonstrates an increasing trend in user downloads, although the package reached its peak download numbers in the beginning of 2016. The other two packages have also seen increasing trends with respect to downloads, but have been comparatively stable over time.

Open source software packages do not exist in isolation, and instead have sometimes complex networks of reliance and dependencies within the larger R realm. Figure 4 depicts a comparison of the reverse dependencies, imports and suggestions for the three software packages. Consistent with the citation and download information, tseries is the most popular of the three, following by fGarch and rugarch, respectively.

Based on the prevalence of the tseries package with respect to formal citations, downloads and dependencies, it would appear that the tseries package should be preferred in GARCH

modeling applications using R. Our findings, however, do not support this. While typical criteria for choosing software packages can include popularity, user-friendliness, computational efficiency and the range of features offered, these should be secondary considerations. Instead, accuracy must be the primary consideration. Our findings are consistent with previous research by McCullough and Renfro (2000), which found discrepancies across seven software packages with respect to the fitted GARCH parameters, suggesting that these issues continue to persist.

As discussed in McCullough and Vinod (2003a), in his "GARCH 101" tutorial, Engle (2001) simply ran the GARCH procedure with the options at default and in doing so failed to maximize the likelihood; thus he reported inaccurate GARCH coefficients. More specifically, McCullough and Vinod (2003a) document that Engle (2001) ran the EViews program at default and found a local maximum. Since accuracy was not a priority in Engle (2001), this should be unsurprising. Further details on the practicalities of using software to maximize likelihoods can be found in McCullough and Renfro (2000) and McCullough (2004), as well as McCullough and Vinod (2003b, 2004). To date, there is but one accuracy benchmark for GARCH models, proposed by McCullough and Renfro (1999), based on the work of Fiorentini, Calzolari and Panattoni (1996), which makes use of the Bollerslev and Ghysels (1996) data containing 1,974 observations on the daily percentage nominal returns for the Deutschemark/British pound exchange rate. We utilize this FCP GARCH Benchmark in this research to determine the accuracy of their estimates.

In this paper, we specifically evaluate the three packages with respect to optimization control and accuracy. Any GARCH procedure should have many user-controlled options, at least to control the nonlinear solver and a user should tune these options to obtain the best answer. In our research, we find that despite the tseries package's popularity, the rugarch package offers the greatest flexibility, range of options, consideration of benchmarking and consistency of the three packages, despite its seemingly underwhelming popularity in both citations and practice. Our findings also reinforce the importance of software documentation and benchmarking at the development-level and discernment and understanding at the user-level to promote reproducibility, accuracy, and interpretability of GARCH analysis results using the R software.

In Section 1, we describe the GARCH Model and introduce the R packages we consider, including a description of algorithms and options available to the user. In Section 2, we run them at default to produce some answer, then attempt to vary the options to establish that we have control over the package. Then, we introduce the FCP GARCH benchmark that is applied to the R packages and present our results. Section 3 presents the discussion and conclusions.

1 The GARCH Model and Estimation

The GARCH model is a popular extension of an ARCH model, in which an autoregressive moving average (ARMA) model, rather than autoregressive (AR), is used to model the variance of the time series (Engle, 1982; Bollerslev, 1986). Both ARCH and GARCH methods are commonly employed econometric techniques to handle the presence of volatility clustering. In this paper, we focus on the standard univariate GARCH model, which serves as the foundation for the family of GARCH models, which include the exponential GARCH, or eGARCH (Nelson, 1991), integrated GARCH, or iGARCH (Engle and Bollerslev, 1986), and GJR-GARCH (Glosten, Jaganathan and Runkle 1993) models.

Following McCullough and Renfro (1999), the basic GARCH(p, q) model, in which p is the lag parameter and q denotes the ARCH order, is given by

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

where y_t is the dependent variable, x_t is a vector of independent variables and b is a vector of parameters in a linear regression model. ϵ_t represents the stochastic process, ψ_t represents the information set through time t and h_t, the conditional variance is given by

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

where α_0, α_i and β_j are constants and $\alpha_0 > 0$, $\alpha_i \geq 0$, $\quad i = 1, ..., p$, $\quad \beta_j \geq 0$, $\quad j = 1, ..., q$.

This gives rise to the conditional likelihood

$$L_t = \sum_{t=1}^{T} l_t(\theta) \tag{3}$$

$$l_t(\theta) = -\frac{1}{2} \ln h_t - \frac{1}{2} \frac{\epsilon_t^2}{h_t} \tag{4}$$

The mean is denoted μ. This model is only partially specified because starting values must be provided for μ, α_0, α_1, and β_1 and the initialization for h_t and ϵ_t^2 must be defined before the conditional likelihood can be maximized. This initialization is distinct from the starting values that are used to begin a nonlinear estimation and is an important facet of the estimation. A very common initialization is

$$h_t = \epsilon_t^2 = \frac{1}{T} \sum_{s=1}^{T} \epsilon_s^2 = \frac{SS}{T}, \quad t \leq 0 \tag{5}$$

where ϵ_s are estimated residuals from least squares regression. Since the initialized values of h_t and ϵ_t^2 are present in the likelihood function, the estimated model parameters will depend on the initial values, underscoring the importance of this initialization.

Once initialized, the log-likelihood can be maximized. The complexity of the GARCH estimation procedure stems in large part due to the non-linear estimation. Analytical and numerical methods can be used in the estimation, with varying degrees of precision. We define the gradient, $g(\theta)$, and the Hessian, $H(\theta)$, of the likelihood function and Q as the estimate of the covariance matrix.

Fiorentini et al. (1996) presented 5 estimates that can be used for the covariance matrix, which are used in the benchmarking procedures in McCullough and Renfro (1999). The first three methods are gradient methods and the latter two are maximum likelihood methods. Direct use of the Hessian involves setting $Q = -H^{-1}$. Alternatively, by setting $Q = g(\theta)g(\theta)'$, the outer product of the gradient (OPG) can be used to approximate the Hessian, as is typical in the BHHH algorithm. The estimated expected Hessian matrix, the information matrix, in which $Q = E[H^{-1}(\theta)]$, can also be used. The two maximum likelihood estimators are the quasi-maximum likelihood method (QMLE), in which $Q = H^{-1}(gg')H^{-1}$ and the Bollerslev and Wooldridge (BW 1992), where $Q = I^{-1}(gg')I^{-1}$.[1]

1.1 Estimation in R

First, we briefly describe the commands necessary to estimate a GARCH model in each package. Our goal is to get each package to produce a result by running it with the default options. After having produced an answer, we will make sure the options are subject to user control by limiting the number of iterations, which is perhaps the simplest option to demonstrate control. The three packages we consider are tseries (Trapletti and Hornik, 2018), fGarch (Wuertz and Chalabi, 2016), and rugarch (Ghalanos, 2017).

In addition to getting coefficient estimates, we also comment on the standard errors. There are many ways to get standard errors for nonlinear estimation (*e.g.*, BHHH, OPG, etc), and the output should clearly specify which type of standard error is reported. The user should not have to dig through the documentation to find out what kind of standard error is being used. Further, as a matter of usable software, the package should offer more than one method of computing the standard error. Except for rugarch, the documentation for these packages makes no mention of the accuracy of the GARCH estimates they produce.

[1] Since methods were introduced and demonstrated in FCP (1996), more than 20 years ago, we omit the rigorous derivations and direct the interested reader to this literature.

1.1.1 tseries

The tseries package is described as a package for "time series analysis and computational finance", with the ability to estimate both autoregressive moving average (ARMA) and GARCH models (Trapletti and Hornik, 2018). Package documentation indicates that the package is still being maintained by its authors, and was updated in 2018.

For tseries, the GARCH and control commands are as follows:

```
garch(x, order = c(1, 1), series = NULL, control = garch.
control(...), ...)

garch.control(maxiter = 200, trace = TRUE, start = NULL, grad =
c("analytical","numerical"), abstol = max(1e-20,.Machine$double.
eps^2), reltol = max(1e-10, .Machine$double.eps^(2/3)), xtol =
sqrt(.Machine$double.eps), falsetol = 1e2 * .Machine$double.
eps, ...)
```

The garch command is described as fitting the GARCH model by "computing the maximum likelihood estimates of the conditionally normal model". This is the only indication given that the sole conditional distribution that can be utilized is the normal distribution. The package offers either numerical or analytical gradient methods for optimization.

1.1.2 fGarch

The fGarch package is described as a "collection of functions to analyze and model heteroskedastic behavior in financial time series models" and is part of the larger Rmetrics offerings. Based on the change log on CRAN, the package has not been changed since 2013, although the published year on the documentation is listed as 2019.

The garch and control commands for the fGarch package are as follows:

```
garchFit(formula = ~ garch(1, 1), data = dem2gbp, init.rec =
c("mci", "uev"), delta = 2, skew = 1, shape = 4, cond.dist =
c("norm", "snorm", "ged", "sged", "std", "sstd", "snig", "QMLE"),
include.mean = TRUE, include.delta = NULL, include.skew = NULL,
include.shape = NULL, leverage = NULL, trace = TRUE, algorithm =
c("nlminb", "lbfgsb", "nlminb+nm", "lbfgsb+nm"), hessian =
c("ropt", "rcd"), control = list(), title = NULL, description
= NULL, ...)
```

```
garchFitControl(llh = c("filter", "internal", "testing"),
nlminb.eval.max = 2000, nlminb.iter.max = 1500, nlminb.abs.tol =
1.0e-20, nlminb.rel.tol = 1.0e-14, nlminb.x.tol = 1.0e-14,
nlminb.step.min = 2.2e-14, nlminb.scale = 1, nlminb.fscale =
FALSE, nlminb.xscale = FALSE, sqp.mit = 200, sqp.mfv = 500, sqp.met
= 2, sqp.mec = 2, sqp.mer = 1, sqp.mes = 4, sqp.xmax = 1.0e3,
sqp.tolx = 1.0e-16, sqp.tolc = 1.0e-6, sqp.tolg = 1.0e-6, sqp.told
= 1.0e-6, sqp.tols = 1.0e-4, sqp.rpf = 1.0e-4, lbfgsb.REPORT = 10,
lbfgsb.lmm = 20, lbfgsb.pgtol = 1e-14, lbfgsb.factr = 1,
lbfgsb.fnscale = FALSE, lbfgsb.parscale = FALSE, nm.ndeps = 1e-14,
nm.maxit = 10000, nm.abstol = 1e-14, nm.reltol = 1e-14, nm.alpha =
1.0, nm.beta = 0.5, nm.gamma = 2.0, nm.fnscale = FALSE,
nm.parscale = FALSE)
```

The fGarch package is plagued with many issues, one of which is sloppy documentation. For instance, the description of the `garchFit` command is: "Estimates the parameters of an univariate ARMA-GARCH/APARCH process." While the documentation makes it clear that an APARCH model can be estimated using the function, the primary role of the `garchFit` function is clearly to estimate a GARCH model, and all examples in the documentation using this function are to fit a "pure" GARCH model of order (1,1). Meanwhile, the description of the `garchFitControl` command is: "Estimates the parameters of an univariate GARCH process." The control command does not do this, nor is it likely that the control command was ever intended to do this. The purpose of this function is unclear in the documentation and no illustrative examples are provided. In fact, we will show that the control command does not even do what it is supposed to do.

Unlike the `garch` command in the tseries package, the `garchFit` command offers several options for specifying the conditional distribution and optimization method; however, the documentation does not clearly define what the abbreviations stand for.

1.1.3 rugarch

The rugarch package is the only package of the three considered which provides an instruction guide, known in R as a vignette ("Introduction to the rugarch Package"), in addition to the reference documentation. The documentation itself does address the issue of accuracy, though incompletely and incorrectly.

The rugarch package has a built-in function, `ugarchbench(benchmark = "published")`, which reports the rugarch-estimated parameters versus published benchmarks, which turns out to be the FCP GARCH Benchmark; however, it is up to the user to determine that the

results are accurate. Further, the documentation for the command only cites Brooks (1997), which is a limited review of three software programs using simulated data and only using BHHH for the computation of standard errors. This does not appear to have anything to do with the benchmarking for accuracy; it does a simulation analysis and seems to look at the speed and cost, not accuracy. The documentation should reference the article by McCullough and Renfro (1999).

The `ugarchfit` command requires a specification of the garch model that is effected via the `spec` option with a call to `ugarchspec`. The pair of commands is given below.

```
ugarchfit(spec, data, out.sample = 0, solver = "solnp",
solver.control = list(), fit.control = list(stationarity = 1,
fixed.se = 0, scale = 0, rec.init = 'all'), numderiv.control =
list(grad.eps=1e-4, grad.d=0.0001, grad.zero.tol=
sqrt(.Machine$double.eps/7e-7), hess.eps=1e-4, hess.d=0.1,
hess.zero.tol=sqrt(.Machine$double.eps/7e-7), r=4, v=2),...)

ugarchspec(variance.model = list(model = "sGARCH", garchOrder =
c(1, 1), submodel = NULL, external.regressors = NULL,
variance.targeting = FALSE), mean.model = list(armaOrder = c(1, 1),
include.mean = TRUE, archm = FALSE, archpow = 1, arfima = FALSE,
external.regressors = NULL, archex = FALSE), distribution.model =
"norm", start.pars = list(), fixed.pars = list(), ...)
```

The `ugarchfit` command is described as a "method for fitting a variety of univariate GARCH models". The command offers six possible solvers, and defines each in the "Details" section of the manual. There are many options for specifying the conditional distribution, as outlined in the `ugarchspec` section of the manual. As in the fGarch package, rugarch estimates μ by default.

2 Analysis

Our methodology for evaluating the three available R packages for univariate GARCH modeling involves two major components: (1) examining the ability to control the optimization procedure and (2) assessing the performance against an established GARCH benchmark.

The data used in the analysis is the Bollerslev and Ghysels (1996) data containing 1,974 observations on the daily percentage nominal returns for the Deutschemark/British pound exchange rate. We use the FCP GARCH Benchmark to determine the accuracy of their estimates.

2.1 Optimization Control

2.1.1 tseries

When run at default, in particular executing the command `garch(x)`, trace is on and we can observe that the optimizer performed 27 iterations before declaring false convergence. The false convergence can be eliminated by providing decent starting values. By specifying starting values of 0.01, 0.1, and 0.8 for α_0, α_1 and β_1, respectively, or

```
garch(x, order=c(1,1), control=garch.control(start=c(0.01,0.1,0.8))),
```

relative convergence is indicated.

To demonstrate control over the optimization procedure, we limit the number of iterations by changing the command to

```
garch(x,control=garch.control(maxiter=5)).
```

This successfully limits the number of iterations to 5, so we believe we have control over the optimizer and have obtained a solution.

While we are able to demonstrate control over the optimizer, there are some shortcomings identified. Based on the output obtained when running the command, the results does not include an estimate of μ. The documentation does not describe what type of GARCH model is being fitted, and makes no mention of why no mean parameter is estimated (even though GARCH models typically are fitted with a mean). The estimated parameters exclude μ, and there is no mention in the documentation if it is being estimated and withheld or if there is an alternate assumption underlying the model.

2.1.2 fGarch

Running at default, `garchFit(~ garch(1,1),x)`, we find a solution in 19 iterations. The default algorithm is `nlminb` so it seems natural to limit the number of iterations as follows:

```
garchFit(formula = ~ garch(1, 1), data = x,
control=garchFitControl(nlminb.iter.max=5)),
```

but the number of iterations is still 19 *and* several warnings are issued, all of the variety:

```
In if (.params$control$llh == "internal") { ...  :  the condition has
length > 1 and only the first element will be used.
```

Adding the option `llh=''filter''` to `garchFitControl` removes the warnings, but as one can see, the default for `llh` *is* `filter`, so the source of the error is unclear. Further, the number of iterations is not limited to 5; again we get 19 iterations.

Running `garchFit( ~ garch(1,1),x)@fit\$params\$control` to view the control options and defaults, we see that the default maximum iterations is 5000. Running

```
garchFit(formula= ~ garch(1,1), data = x,
control=garchFitControl(nlminb.iter.max=5))@fit\$params\$control
```

shows not only

```
$nlminb.iter.max
[1] 5
```

but also

```
$MIT
[1] 2000
```

even though

```
garchFit(formula= ~ garch(1,1), data = x,
control=garchFitControl(nlminb.iter.max=5))
```

results in 19 iterations.

Thus, the value of 5 is passed to the function but it has no effect on the number of iterations, and there are also several control parameters, including `MIT`, that are not listed in the documentation! This certainly looks like "maximum iterations"; but setting "`MIT = 5`" still results in 19 iterations. It appears, then, that arguments passed from the function call to the optimizer do not have an effect on the modeling, even though the information passed is updated in the model output! Specifically, using the options to specify the maximum number of iterations *does not work*; at least we could not make it work in what we consider to be a reasonable amount of time. Furthermore, Nash (2014) suggests that the R optim port routines that the fGarch package utilizes should be deprecated.

As mentioned, the lack of examples provided in the documentation further contributes to the difficulty of understanding the specifications and gaining control over the function and estimation of the GARCH model, particularly with respect to the `garchControlFit` function. We conclude that without demonstrable control over the options for the optimizer, we will have to execute all runs at default settings for the fGarch package. We also note that the documentation is obviously wrong in many places, which suggests a sloppiness that may well pervade the underlying code, *e.g.*, the `MIT` option mentioned above. However, unlike the tseries package, the default command sets `include.mean=TRUE` and `garchFit` estimates μ by default.

2.1.3 rugarch

The most parsimonious command we could get to run is `ugarchfit(spec = ugarchspec(),x)` but it did not show the number of iterations. According to the manual for `solnp`, the default solver in `ugarchfit`, it should print the coefficients at each iteration, but somehow this feature is suppressed. To get the number of iterations we had to run `ugarchfit(spec = ugarchspec (), x, solver.control = list(trace=1))` from which we see convergence in two iterations. If we run `ugarchfit(spec = ugarchspec(), x, solver.control = list(trace=1, outer.iter=1))` we get a failure to converge message, and if we set `outer.iter=2` we get the same answer as the previous runs. We believe that we have control over the optimizer.

The `ugarchfit` function returns two sets of standard errors. The "robust" standard errors are vaguely described as "based on White (1982)" (Ghalanos, 2017, p. 27), which is hardly specific, since there are many robust methods "based on" White's article. With respect to the default standard errors estimated, the reference manual and vignette do not even hint at the method used to compute the default standard errors. We will have to see if we can deduce whether the default standard errors are based on the gradient, the Hessian or the Information Matrix.

2.2 Accuracy

Since this benchmark was published nearly 20 years ago and it is the only available GARCH benchmark, every package should permit this initialization if only for the purpose of benchmarking the package; any package that does not permit this initialization is not written by a person who wants to benchmark his package. Again, we emphasize that this initialization is distinct from the choice of starting values that are used for parameter estimation.

FCP estimated this model with analytic derivatives, which are more accurate than numerical derivatives and generally lead to more accurate parameter estimates. We cannot expect that packages using numerical derivatives can achieve more than a few digits of accuracy; certainly we do not expect packages with numerical derivatives to achieve full six-digit accuracy.

Table 1 will allow us to answer two questions: How accurate are the estimates? and, if necessary, What kind of standard errors does the package offer?

First and foremost, we have to ask, Can the benchmark model be estimated? This requires that the FCP initialization be used. If so, then two more questions follow: (1) how many accurate digits at default? and (2) how many accurate digits can we get if we tune the optimizer by carefully choosing the options?

To tune the optimizer, we do the following:

1. Select starting values rather than let the software choose them. We first tried using

the default results, but those produced false convergence. We settled on using the first significant digit of the default results: 0.01, 0.1, 0.8.

2. Analytic derivatives are the default, and they are generally better than numerical derivatives, so we kept analytic derivatives.

3. We varied some options, e.g., tolerances, but we did not conduct an exhaustive search to find the combination of options that produces the most accuracy.

2.2.1 tseries

The documentation makes no mention of accuracy and the output also does not identify the type of standard error reported. The garch function offers one solver, with only the vague description that it is a "Quasi-Newton optimizer", which uses a "Hessian approximation computed from the BFGS update". For this reason, especially poor approximations are likely to result in the event of early termination of the iterative estimation and the quasi-Newton approach lacks the safeguards against saddle points inherent in Newton-Raphson (McCullough and Renfro, 1999). Quasi-Newton methods are, however, less sensitive to the initial starting values.

In the documentation for the `summary.garch` command, we find that the covariance matrix is computed by the outer product of the gradient (OPG) method, which is the only method offered by the package. Unfortunately, this is an inferior method that only uses first derivative information, although it uses analytic derivatives. It offers only one type of standard error, and the type is not labeled in the output. Instead, the user has to search the documentation for it.

tseries makes no mention of an initialization. The actual initialization of h_t and ε_t is not discussed at all. Per the tseries documentation, "default initialization is to set the GARCH parameters to slightly positive values and to initialize the intercept such that the unconditional variance of the initial GARCH is equal to the variance of x" (page 10). This actually refers to starting values for parameter estimation, not the initialization for the recursion of the error term. Running at default, `summary(garch(x))` produces results for a three parameter model, although a "false convergence" message is produced.

Tuning the model involved altering the convergence tolerance and starting parameters for the parameter estimation. Tuning produced only a marginal improvement in accuracy, which is limited to 2-3 significant digits, as shown in Table 2. This is exceedingly poor performance for analytic derivatives. We conjecture that some initialization other than the benchmark initialization is being used.

The tuned estimates of the OPG standard errors are shown in Table 3. While the `garch` command in the tseries package has the benefit of producing analytical derivatives and we

are able to prove that we have control over the optimizer, the performance compared to the FCP benchmark is poor. Despite attempts to obtain improvements in performance by tuning, the benefit of tuning was negligible in improving the accuracy of the estimates. In addition, we are forced to make comparisons based on a three parameter model instead of a four parameter model, as is the case of the FCP benchmark.

2.2.2 fGarch

The documentation makes no mention of accuracy. The package offers several solvers, but we cannot even limit the number of iterations for the default solver. The package offers two types of standard errors, one based on the Hessian and another based on QMLE, but the output and manual lack clarity. There are many options available in the package for estimation, but a combination of poor documentation and technical issues limits us to running the package with the default options.

fGarch has an option `init.rec` to specify "a character string indicating the method how to initialize the mean and variance recursion relation" which can take one of two arguments, "`mci`" and "`uev`". However, the documentation makes no mention of what "`mci`" and "`uev`" might mean. The "`mci`" option produces default results, which is to be expected, while the "`uev`" option produces an error message:

```
Error in .garchFit(formula.mean = args$formula.mean, formula.var =
 args$formula.var,  :   Algorithm only supported for mci Recursion
```

It is most unusual to release software with options that are not supported.

The default model using the fGarch package is estimated using the command:

```
summary(garchFit(~garch(1,1),x))
```

which includes the phrase "Std. Errors: based on Hessian". Yet the documentation ("Details" section of the command `garchFit`) discusses QMLE and refers the reader to Davidson and MacKinnon's 2004 text for a discussion of the robust covariance matrix. Via experimentation, we discovered that if we invoke the option to change the conditional distribution from normal to QMLE, then the output contains QMLE standard errors. This was not at all obvious, and should be spelled out clearly in the manual because QMLE standard errors are generally better for GARCH estimation.

We are unable to tune the procedure, since the options do not seem to change anything. We failed to limit the number of iterations already. Here we tried to change the default absolute, relative and x tolerance from their default values of 0, 1E-10, and 1.58E-10, respectively, to 1E-7 and observed no change in the solution. We do not have reason to believe that our changes in defaults have been made, given our lack of control over the optimizer.

As shown in Table 4, the coefficients are estimated with 4-6 digits of accuracy, and the Hessian standard errors are estimated with 2-4 digits of accuracy as shown in Table 5.

With some difficulty, we were able to identify the default standard errors as Hessian standard errors. If QMLE is specified as the conditional distribution instead of the default, the output produces QMLE standard errors, specifically the Eicker-White sandwich estimators. These details should be explicitly and clearly stated, not hidden. These estimates are accurate to 1 or two digits, as shown in Table 6. If we could tune the procedure, we might be able to obtain more accuracy.

```
summary(garchFit(~garch(1,1),x, cond.dist="QMLE"))
```

We are able to change the conditional distribution and thus estimate QMLE standard errors, as suggested in the documentation, rather than the default method, which provides standard errors based on the Hessian. The QMLE standard errors are estimated with 1-3 digits of accuracy.

The accuracy of the estimates found using the `garchFit` command in the fGarch package compared to the FCP benchmark is an improvement over those found using the tseries package. Additionally, all parameters are estimated, including μ. Unfortunately, without control over the optimizer we cannot tune the optimzer to obtain better results.

2.2.3 rugarch

The reference manual does mention accuracy, without asserting that it hits a benchmark, and cites the wrong article. The ugarchbench procedure documentation cites Brooks (1997), which is a review of software for GARCH models, and makes no mention of benchmarking. The package offers several optimization algorithms and two types of standard errors, but is missing more precise information regarding the latter. rugarch also has an option `init.rec`, and the documentation explains clearly that when this is set to "all" it is the FCP initialization. It also happens to be the default choice for this option. According to the rugarch manual, we are able to run the FCP GARCH benchmark on the BG data via its `ugarchbench` function.

The FCP Benchmark model with four parameters is specified and then estimated with the following commands:

```
model=ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm")

 mygarch = ugarchfit(spec = model, data = x)
```

The above is the default model because we have not altered the algorithm or the tolerances, etc. Table 6 shows these default coefficients, which are accurate to 2 or 3 digits. The corresponding standard errors are given in Tables 7 (Hessian) and 8 (robust), which are accurate to 3 and zero digits, respectively. The poor accuracy of the robust standard error makes us suspect that rugarch uses some robust standard error other than QMLE. The author really should specify what method he uses.

Tuning does not lead to better performance and are omitted for this reason. We are surprised that we cannot squeeze an extra digit of accuracy by tuning, but maybe the defaults have been chosen for maximal accuracy.

In the "Details" section of `ugarchfit-methods` it states that "the GARCH optimization routine first calculates a set of feasible starting points which are used to initiate the GARCH recursion". From `ugarchfit-methods` section, we also find that the `rec.init` option determines the type of initialization for the variance recursion. "Valid options are 'all' which uses all the values for the unconditional variance calculation, an integer greater than or equal to 1, denoting the number of data points to use for the calculation, or a positive numeric value less than one, which determines the weighting for use in an exponential smoothing backcast". "all" is the default option, and appears to correspond to the initialization for the FCP benchmark, though it would be nice if this were more explicitly stated in the documentation.

The `ugarchfit` command in the rugarch package has many strengths with respect to the estimation, documentation and accuracy of GARCH models. We are able to demonstrate control over the solver. It seems that the defaults were chosen with the FCP benchmark in mind, resulting in optimal performance; if this is true it should be explicitly stated. However, the "robust" standard error is not the QMLE used by many other packages, and the documentation does not specifically state what type of robust standard error is used. There are some areas of documentation that are lacking, but overall the estimation of the parameters of the GARCH model have high accuracy and transparency with respect to their performance against the benchmark.

2.2.4 Comparing Coefficient Accuracy

Following McCullough and Renfro (1999), we present the log relative error (LRE), to compare the tuned coefficient and standard error estimates across the three packages. The log relative error is computed as

$$LRE = -log_{10}[\mid x - c \mid / \mid c \mid] \tag{6}$$

where x is the estimated value and c is the benchmark value. The LRE measures the number of accurate digits, where larger LRE values indicate a more accurate estimate.

Table 10 presents the *LRE* values for the coefficient estimates of the three packages. As shown, fGarch produces the most accurate coefficient estimates, followed by rugarch and tseries.

The *LRE* for the standard errors are presented in Table 11, organized by the estimate of the covariance matrix used. This table further highlights the poor performance of tseries with respect to accuracy. As shown and expected, the accuracy is highest when directly using the Hessian matrix, which fGarch and rugarch can do.

3 Discussion and Conclusion

Despite more than 20 years passing since the introduction of a benchmarking procedure for univariate GARCH methods, the persistence of the problems identified in McCullough and Renfro (1999) in our current estimation of GARCH models using 3 packages in the same software program emphasizes the non-triviality of the issue. Despite Fiorentini et al. (1996) presenting and examining closed-form options for maximizing the GARCH conditional likelihood, we find that the most popular package of the three considered, tseries, makes use of a method that they find to be inferior.

In this paper, we have examined the three R packages that offer GARCH estimation, tseries, fGarch and rugarch, and concluded that rugarch is the package we would use for GARCH estimation, although it is not without flaws. rugarch, while its documentation leaves something to be desired, offers several optimizers, two different types of standard errors, can hit the benchmark, offers forecasting capabilities and can be tuned. However, its "robust standard error" is of an unspecified variety, and does not hit any of the five benchmarked standard errors; we don't know what kind of robust standard error the package employs.

The second package that hits the benchmark, fGarch, has poor documentation, poor control of options and we cannot tune the optimizer to improve performance. While it offers two different types of standard errors and forecasting ability, these glaring deficiencies should warrant skepticism from users.

Most importantly, perhaps, we find that the most popular GARCH modeling package, tseries, has several weaknesses that make it a particularly poor choice for econometrics practitioners. tseries does not estimate a model with a mean, has only one optimizer, one type of standard error (based on the inferior OPG method that fails to make use of second derivative information), cannot produce out-sample forecasts and cannot hit the FCP GARCH benchmark.

References

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroscedasticity. *Journal of Econometrics*, 31:307–327.

Bollerslev, T. and Ghysels, E. (1996). Periodic Autoregressive Conditional Heteroscedasticity. *Journal of Business and Economic Statistics*, 14:139–151.

Brooks, C. (1997). GARCH Modelling in Finance: A Review of the Software Options. *Economic Journal*, 107(443):1271–1276.

Engle, R. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987–1008.

Engle, R. (2001). GARCH 101: The Use of ARCH/GARCH Models in Applied Econometrics. *Journal of Economic Perspectives*, 15(4):157–168.

Engle, R. and Bollerslev, T. (1986). Modelling the Persistence of Conditional Variances. *Econometric Reviews*, 5(1):1–50.

Fiorentini, G., Calzolari, G., and Panattoni, L. (1996). Analytic Derivatives and the Computation of GARCH Estimates. *Journal of Applied Econometrics*, 11:399–417.

Ghalanos, A. (2017). Introduction to the rugarch package. (Version 1.3-8). https://cran.r-project.org/web/packages/rugarch/.

Glosten, L. R., Jagannathan, R., and Runkle, D. E. (1993). On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *Journal of Finance*, 48(5):1779–1801.

McCullough, B. D. (2004). Some Details of Nonlinear Estimation. In Altman, M., Gill, J., and McDonald, M. P., editors, *Numerical Issues in Statistical Computing for the Social Sciences*, pages 199–218. John Wiley & Sons, New York.

McCullough, B. D. and Renfro, C. G. (1999). Benchmarks and Software Standards: A Case Study of GARCH Procedures. *Journal of Economic and Social Measurement*, 25(2):59–71.

McCullough, B. D. and Renfro, C. G. (2000). Some Numerical Aspects of Nonlinear Estimation. *Journal of Economic and Social Measurement*, 26(1):63–77.

McCullough, B. D. and Vinod, H. D. (2003a). Comment: Econometrics and Software. *Journal of Economic Perspectives*, 17(1):223–224.

McCullough, B. D. and Vinod, H. D. (2003b). Verifying the Solution from a Nonlinear Solver: A Case Study. *American Economic Review*, 93(3):873–892.

McCullough, B. D. and Vinod, H. D. (2004). Verifying the Solution from a Nonlinear Solver: A Case Study: Reply. *American Economic Review*, 94(1):400–403.

Nash, J. C. (2014). On Best Practice Optimization Methods in R. *Journal of Statistical Software*, 60(2):1–14.

Nelson, D. B. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica*, 59(2):347–370.

Soito, L. and Hwang, L. J. (2016). Citations for Software: Providing Identification, Access and Recognition for Research Software. *International Journal of Digital Curation*, 11(2):48–63.

Trapletti, A. and Hornik, K. (2018). tseries: Time Series Analysis and Computational Finance. R package version 0.10-42. https://cran.r-project.org/package=tseries.

Wuertz, D. and Chalabi, Y. (2016). fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling. R package version 3010.82.1. https://cran.r-project.org/package=fGarch.

Figure 1: Google Scholar Citations Time Series

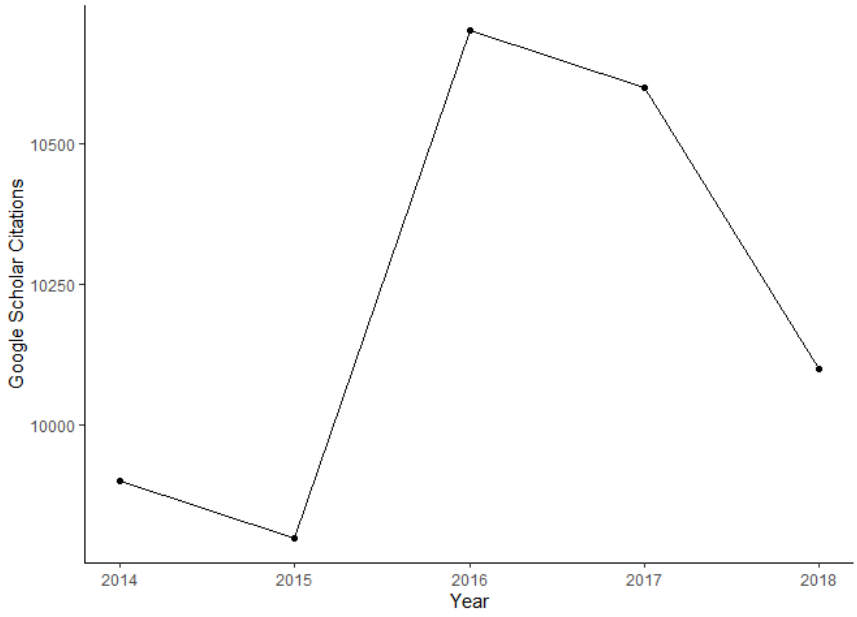

Figure 2: GARCH R Packages Google Scholar Search Result Time Series

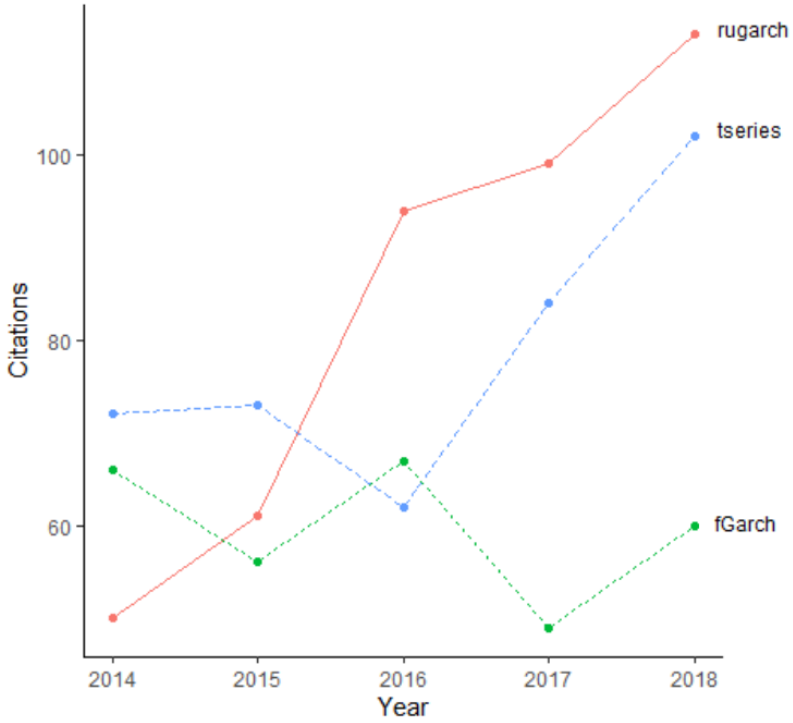

Figure 3: Package Downloads Time Series

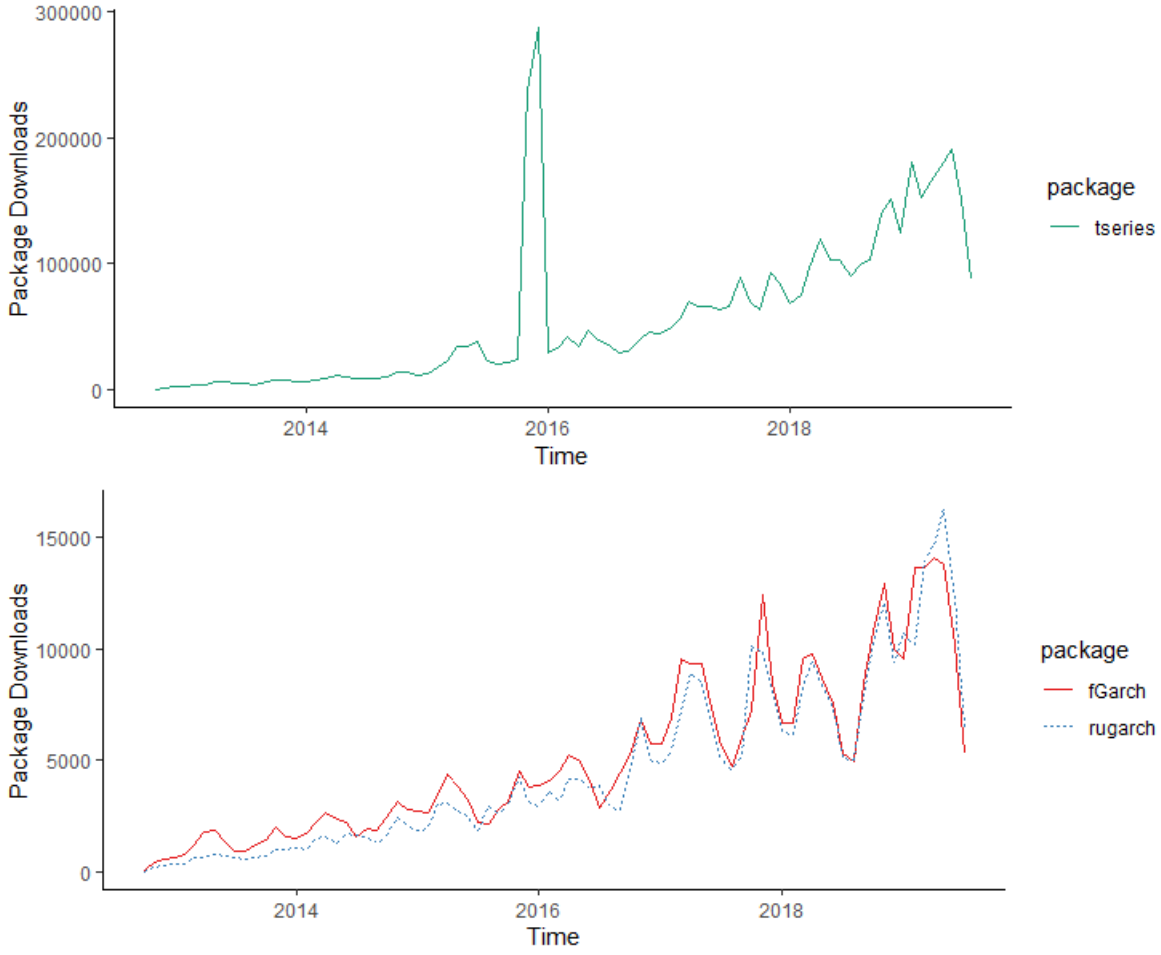

Figure 4: Reverse Dependencies, Imports and Suggestions across GARCH packages in R

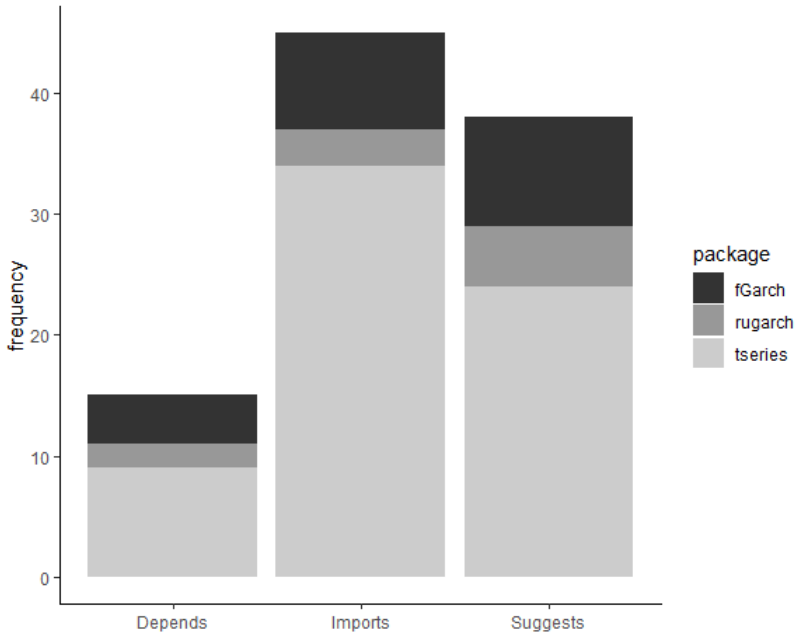

Figure 5: Benchmark data from Bollerslev and Ghysels (1996)

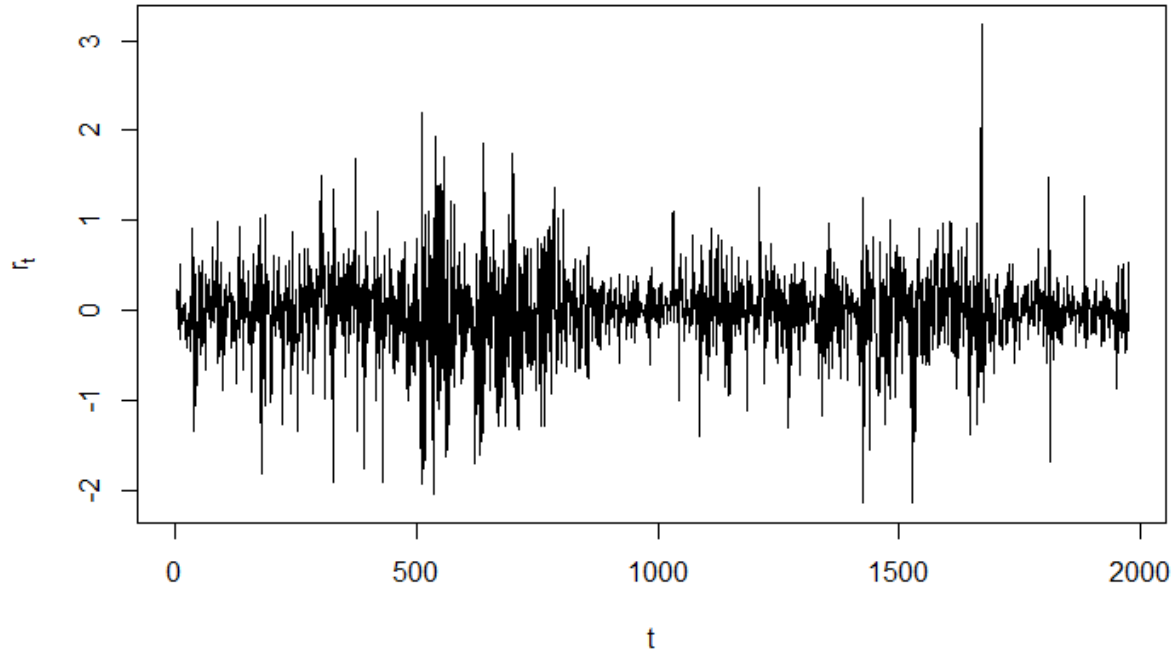

Table 1: FCP GARCH Benchmark

	coefficient	standard error				
		-H	OP	QMLE	IM	BW
μ	-.619041E-2	.846212E-2	.843359E-2	.918935E-2	.837628E-2	.873092E-2
α_0	.107613E-1	.285271E-2	.132298E-2	.649319E-2	.192881E-2	.312364E-2
α_1	.153134E-0	.265228E-1	.139737E-1	.535317E-1	.194012E-1	.273219E-1
β	.805974E-0	.335527E-1	.165604E-1	.724614E-1	.218399E-1	.301509E-1

Table 2: Coefficients for tseries

	μ	α_0	α_1	β_1
benchmark	0.00619041	0.0107613	0.153134	0.805974
default	NA	0.0107843	0.154074	0.805295
tuned	NA	0.0107825	0.154059	0.805317

Table 3: OPG standard errors for tseries

	μ	α_0	α_1	β_1
benchmark	0.00843359	0.00132298	0.0139737	0.0165604
tuned	NA	0.00128802	0.0138210	0.0159639

Table 4: Default (Hessian) standard errors for fGarch, inaccurate digits underlined

	μ	α_0	α_1	β_1
benchmark	0.00846212	0.00285271	0.0265228	0.0335527
default	0.00846200	0.00283751	0.0264217	0.0333813

Table 5: QMLE standard errors for fGarch, inaccurate digits underlined

	μ	α_0	α_1	β_1
benchmark	0.00918935	0.00649319	0.0535317	0.0724614
QMLE	0.00918577	0.00642400	0.0530562	0.0716837

Table 6: Coefficients for rugarch, inaccurate digits underlined

	μ	α_0	α_1	β_1
benchmark	.00619041	.0107613	.153134	.805974
default	.0061<u>8499</u>	.010760<u>2</u>	.15340<u>7</u>	.80588<u>0</u>

Table 7: Hessian standard errors for rugarch, inaccurate digits underlined

	μ	α_0	α_1	β_1
benchmark	.00619041	.0107613	.153134	.805974
default	.0061<u>8499</u>	.010760<u>2</u>	.15340<u>7</u>	.80588<u>0</u>

Table 8: Robust standard errors for rugarch, inaccurate digits underlined

	μ	α_0	α_1	β_1
benchmark	0.00918935	0.00649319	0.0535317	0.0724614
default robust	0.009<u>01680</u>	0.0064<u>9841</u>	0.0<u>493895</u>	0.0<u>691624</u>

Table 9: Coefficients for fGarch

	μ	α_0	α_1	β_1
benchmark	.00619041	.0107613	.153134	.805974
default	.0061903<u>1</u>	.010761<u>4</u>	.153134	.805974

Table 10: LRE for Coefficient Estimates

	tseries	fGarch	rugarch
μ	NA	4.8	3.1
α_0	2.7	5.0	4.0
α_1	2.2	6.0	2.7
β_1	3.1	6.0	3.9

Table 11: *LRE* for Standard Error Estimates

	H		OP	QMLE	
	fGarch	rugarch	tseries	fGarch	rugarch
μ	4.8	4.2	NA	3.4	1.7
α_0	2.3	4.0	1.6	2.0	3.1
α_1	2.4	2.7	2.0	2.1	1.1
β_1	2.3	3.4	1.4	2.0	1.3