Introduction to the rugarch package.
(Version 1.0)

Alexios Ghalanos

October 12, 2011

Contents

1 Introduction

The pioneering work of Box, Jenkins and Reinsel (1970) in the area of autoregressive moving average models paved the way for related work in the area of volatility modelling with the introduction of ARCH and then GARCH models by Engle (1982) and Bollerslev (1986), respectively. In terms of the statistical framework, these models provide motion dynamics for the dependency in the conditional time variation of the distributional parameters of the mean and variance, in an attempt to capture such phenomena as autocorrelation in returns and squared returns. Extensions to these models have included more sophisticated dynamics such as threshold models to capture the asymmetry in the news impact, as well as distributions other than the normal to account for the skewness and excess kurtosis observed in practice. In a further extension, Hansen (1994) generalized the `GARCH` models to capture time variation in the full density parameters, with the Autoregressive Conditional Density Model[1], relaxing the assumption that the conditional distribution of the standardized innovations is independent of the conditioning information.

The `rugarch` package aims to provide for a comprehensive set of methods for modelling univariate `GARCH` processes, including fitting, filtering, forecasting, simulation as well as diagnostic tools including plots and various tests. Additional methods such as rolling estimation, bootstrap forecasting and simulated parameter density to evaluate model uncertainty provide a rich environment for the modelling of these processes. This document discusses the finer details of the included models and conditional distributions and how they are implemented in the package with numerous examples.

The `rugarch` package forms part of the rgarch project on r-forge `rgarch.r-forge.r-project.org/` which also includes the `rmgarch` package for multivariate `GARCH` models. Previously, both univariate and multivariate models were included in one large package which was split for release to CRAN in August 2011.

The package is provided AS IS, without any implied warranty as to its accuracy or suitability. A lot of time and effort has gone into the development of this package, and it is offered under the GPL-3 license in the spirit of open knowledge sharing and dissemination. If you do use the model in published work DO remember to cite the package and author (type `citation`("rugarch") for the appropriate BibTeX entry) , and if you have used it and found it useful, drop me a note and let me know.

A section on FAQ is included which deals with some often asked questions.

2 Model Specification

This section discusses the key step in the modelling process, namely that of the specification. This is defined via a call to the `ugarchspec` function,

```
> args(ugarchspec)

function (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),
    distribution.model = "norm", start.pars = list(), fixed.pars = list(),
    ...)
NULL
```

[1] This may be included in the package at a future date.

Thus a model, in the `rugarch` package, may be described by the dynamics of the conditional mean and variance, and the distribution to which they belong, which determines any additional parameters. The following sub-sections will outline the background and details of the dynamics and distributions implemented in the package.

2.1 Univariate ARFIMAX Models

The univariate GARCH specification allows to define dynamics for the conditional mean from the general ARFIMAX model with the addition of ARCH-in-mean effects introduced in Engle (1987). The ARFIMAX-ARCH-in-mean specification may be formally defined as,

$$\Phi(L)(1-L)^d(y_t-\mu_t)=\Theta(L)\varepsilon_t, \tag{1}$$

with the left hand side denoting the Fractional AR specification on the demeaned data and the right hand side the MA specification on the residuals. (L) is the lag operator, $(1-L)^d$ the long memory fractional process with $0<d<1$, and equivalent to the Hurst Exponent H - 0.5, and μ_t defined as,

$$\mu_t=\mu+\sum_{i=1}^{m}\delta_i x_{i,t}+\xi\sigma_t^k, \tag{2}$$

where we allow for `m` external regressors x and ARCH-in-mean on either the conditional standard deviation, $k=1$ or conditional variance $k=2$. These options can all be passed via the arguments in the `mean.model` list in the `ugarchspec` function,

- armaOrder (default = (1,1). The order of the ARMA model.)
- include.mean (default = TRUE. Whether the mean is modelled.)
- archm (default = FALSE. The ARCH-in-mean parameter.)
- archpow (default = 1 for standard deviation, else 2 for variance.)
- arfima (default = FALSE. Whether to use fractional differencing.)
- external.regressors (default = NULL. A matrix of external regressors of the same length as the data).

Since the specification allows for both fixed and starting parameters to be passed, it is useful to provide the naming convention for these here,

- AR parameters are 'ar1', 'ar2', ...,
- MA parameters are 'ma1', 'ma2', ...,
- mean parameter is 'mu'
- archm parameter is 'archm'
- the arfima parameter is 'arfima'
- the external regressor parameters are 'mxreg1', 'mxreg2', ...,

Note that estimation of the mean and variance equations in the maximization of the likelihood is carries out jointly in a single step. While it is perfectly possible and consistent to perform a 2-step estimation, the one step approach results in greater efficiency, particularly for smaller datasets.

2.2 Univariate GARCH Models

In GARCH models, the density function is usually written in terms of the location and scale parameters, normalized to give zero mean and unit variance,

$$\alpha_t = (\mu_t, \sigma_t, \omega), \tag{3}$$

where the conditional mean is given by

$$\mu_t = \mu(\theta, x_t) = E(y_t|x_t), \tag{4}$$

and the conditional variance is,

$$\sigma_t^2 = \sigma^2(\theta, x_t) = E((y_t - \mu_t)^2|x_t), \tag{5}$$

with $\omega = \omega(\theta, x_t)$ denoting the remaining parameters of the distribution, perhaps a shape and skew parameter. The conditional mean and variance are used to scale the innovations,

$$z_t(\theta) = \frac{y_t - \mu(\theta, x_t)}{\sigma(\theta, x_t)}, \tag{6}$$

having conditional density which may be written as,

$$g(z|\omega) = \frac{d}{dz}P(z_t < z|\omega), \tag{7}$$

and related to $f(y|\alpha)$ by,

$$f(y_t|\mu_t, \sigma_t^2, \omega) = \frac{1}{\sigma_t}g(z_t|\omega). \tag{8}$$

The `rugarch` package implements a rich set of univariate GARCH models and allows for the inclusion of external regressors in the variance equation as well as the possibility of using variance targeting as in Engle (1996). These options can all be passed via the arguments in the `variance.model` list in the `ugarchspec` function,

- model (default = 'sGARCH' (vanilla GARCH). Valid models are 'iGARCH', 'gjrGARCH', 'eGARCH', 'apARCH' and 'fGARCH').
- garchOrder (default = c(1,1). The order of the GARCH model.)
- submodel (default = NULL. In the case of the 'fGARCH' omnibus model, valid choices are 'GARCH', 'TGARCH', 'GJRGARCH', 'AVGARCH', 'NGARCH', 'NAGARCH', 'APARCH' and 'ALLGARCH')
- external.regressors (default = NULL. A matrix of external regressors of the same length as the data).
- variance.targeting (default = FALSE. Whether to include variance targeting.)

The rest of this section discusses the various flavors of GARCH implemented in the package, while Section 2.3 discusses the distributions implemented and their standardization for use in GARCH processes.

2.2.1 The standard GARCH model ('sGARCH')

The standard GARCH model (Bollerslev (1986)) may be written as:

$$\sigma_t^2 = \left(\omega + \sum_{j=1}^{m} \zeta_j v_{jt}\right) + \sum_{j=1}^{q} \alpha_j \varepsilon_{t-j}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2, \tag{9}$$

with σ_t^2 denoting the conditional variance, ω the intercept and ε_t^2 the residuals from the mean filtration process discussed previously. The GARCH order is defined by (q, p) (ARCH, GARCH), with possibly `m` external regressors v_j which are passed *pre-lagged*. If variance targeting is used, then ω is replaced by,

$$\bar{\sigma}^2 \left(1 - \hat{P}\right) - \sum_{j=1}^{m} \zeta_j \bar{v}_j \tag{10}$$

where $\bar{\sigma}^2$ is the unconditional variance of ε^2 which is consistently estimated by its sample counterpart at every iteration of the solver following the mean equation filtration, and $\bar{v}_j$ represents the sample mean of the j^{th} external regressors in the variance equation (assuming stationarity), and $\hat{P}$ is the persistence and defined below. One of the key features of the observed behavior of financial data which GARCH models capture is volatility clustering which may be quantified in the persistence parameter $\hat{P}$. For the 'sGARCH' model this may be calculated as,

$$\hat{P} = \sum_{j=1}^{q} \alpha_j + \sum_{j=1}^{p} \beta_j. \tag{11}$$

Related to this measure is the 'half-life' (call it $h2l$) defined as the number of days it takes for half of the expected reversion back towards $E\left(\sigma^2\right)$ to occur,

$$h2l = \frac{-\log_e 2}{\log_e \hat{P}}. \tag{12}$$

Finally, the unconditional variance of the model $\hat{\sigma}^2$, and related to its persistence, is,

$$\hat{\sigma}^2 = \frac{\hat{\omega}}{1 - \hat{P}}, \tag{13}$$

where $\hat{\omega}$ is the estimated value of the intercept from the GARCH model. The naming conventions for passing fixed or starting parameters for this model are:

- ARCH(q) parameters are 'alpha1', 'alpha2', ...,
- GARCH(p) parameters are 'beta1', 'beta2', ...,
- variance intercept parameter is 'omega'
- the external regressor parameters are 'vxreg1', 'vxreg2', ...,

2.2.2 The integrated GARCH model ('iGARCH')

The integrated GARCH model (see Engle and Bollerslev (1986)) assumes that the persistence $\hat{P} = 1$, and imposes this during the estimation procedure. Because of unit persistence, none of the other results can be calculated (i.e. unconditional variance, half life etc). The stationarity of the model has been established in the literature, but one should investigate the possibility of omitted structural breaks before adopting the iGARCH as the model of choice. The way the package enforces the sum of the ARCH and GARCH parameters to be 1, is by subtracting $1 - \sum_{i=1}^{q} \alpha_i - \sum_{i>1}^{p} \beta_i$, so that the last beta is never estimated but instead calculated.

2.2.3 The exponential GARCH model

The exponential model of Nelson (1991) is defined as,

$$\log_e\left(\sigma_t^2\right) = \left(\omega + \sum_{j=1}^{m} \zeta_j v_{jt}\right) + \sum_{j=1}^{q} \left(\alpha_j z_{t-j} + \gamma_j \left(|z_{t-j}| - E\left|z_{t-j}\right|\right)\right) + \sum_{j=1}^{p} \beta_j \log_e\left(\sigma_{t-j}^2\right) \quad (14)$$

where the coefficient α_j captures the sign effect and γ_j the size effect. The expected value of the absolute standardized innovation, z_t is,

$$E\left|z_t\right| = \int_{-\infty}^{\infty} |z| f\left(z, 0, 1, ...\right) dz \quad (15)$$

The persistence $\hat{P}$ is given by,

$$\hat{P} = \sum_{j=1}^{p} \beta_j. \quad (16)$$

If variance targeting is used, then ω is replaced by,

$$\log_e\left(\bar{\sigma}^2\right)\left(1 - \hat{P}\right) - \sum_{j=1}^{m} \zeta_j \bar{v}_j \quad (17)$$

The unconditional variance and half life follow from the persistence parameter and are calculated as in Section 2.2.1.

2.2.4 The GJR-GARCH model ('gjrGARCH')

The GJR GARCH model of Glosten, Jagannathan and Runkle (1993) models positive and negative shocks on the conditional variance asymmetrically via the use of the indicator function I,

$$\sigma_t^2 = \left(\omega + \sum_{j=1}^{m} \zeta_j v_{jt}\right) + \sum_{j=1}^{q} \left(\alpha_j \varepsilon_{t-j}^2 + \gamma_j I_{t-j} \varepsilon_{t-j}^2\right) + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2, \quad (18)$$

where γ_j now represents the 'leverage' term. The indicator function I takes on value of 1 for $\varepsilon \leq 0$ and 0 otherwise. Because of the presence of the indicator function, the persistence of the model now crucially depends on the asymmetry of the conditional distribution used. The persistence of the model $\hat{P}$ is,

$$\hat{P} = \sum_{j=1}^{q} \alpha_j + \sum_{j=1}^{p} \beta_j + \sum_{j=1}^{q} \gamma_j \kappa, \quad (19)$$

where κ is the expected value of the standardized residuals z_t below zero (effectively the probability of being below zero),

$$\kappa = E\left[I_{t-j} z_{t-j}^2\right] = \int_{-\infty}^{0} f\left(z, 0, 1, ...\right) dz \quad (20)$$

where f is the standardized conditional density with any additional skew and shape parameters $(\ldots)$. In the case of symmetric distributions the value of κ is simply equal to 0.5. The variance targeting, half-life and unconditional variance follow from the persistence parameter and are calculated as in Section 2.2.1. The naming conventions for passing fixed or starting parameters for this model are:

- ARCH(q) parameters are 'alpha1', 'alpha2', ...,
- Leverage(q) parameters are 'gamma1', 'gamma2', ...,
- GARCH(p) parameters are 'beta1', 'beta2', ...,
- variance intercept parameter is 'omega'
- the external regressor parameters are 'vxreg1', 'vxreg2', ...,

Note that the Leverage parameter follows the order of the ARCH parameter.

2.2.5 The asymmetric power ARCH model ('apARCH')

The asymmetric power ARCH model of Ding, Granger and Engle (1993) allows for both leverage and the Taylor effect, named after Taylor (1986) who observed that the sample autocorrelation of absolute returns was usually larger than that of squared returns.

$$\sigma_t^{\delta} = \left(\omega + \sum_{j=1}^{m} \zeta_j v_{jt} \right) + \sum_{j=1}^{q} \alpha_j (|\varepsilon_{t-j}| - \gamma_j \varepsilon_{t-j})^{\delta} + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^{\delta} \tag{21}$$

where $\delta \in \mathbb{R}^+$, being a Box-Cox transformation of σ_t, and γ_j the coefficient in the leverage term. Various submodels arise from this model:

- The simple GARCH model of Bollerslev (1986) when $\delta = 2$ and $\gamma_j = 0$.
- The Absolute Value GARCH (AVGARCH) model of Taylor (1986) and Schwert (1990) when $\delta = 1$ and $\gamma_j = 0$.
- The GJR GARCH (GJRGARCH) model of Glosten, Jagannathan and Runkle (1993) when $\delta = 2$.
- The Threshold GARCH (TGARCH) model of Zakoian (1994) when $\delta = 1$.
- The Nonlinear ARCH model of Higgins and Bera (1992) when $\gamma_j = 0$ and $\beta_j = 0$.
- The Log ARCH model of Geweke (1986) and Pantula (1986) when $\delta \to 0$.

The persistence of the model is given by,

$$\hat{P} = \sum_{j=1}^{p} \beta_j + \sum_{j=1}^{q} \alpha_j \kappa_j \tag{22}$$

where κ_j is the expected value of the standardized residuals z_t under the Box-Cox transformation of the term which includes the leverage coefficient γ_j,

$$\kappa_j = E(|z| - \gamma_j z)^{\delta} = \int_{-\infty}^{\infty} (|z| - \gamma_j z)^{\delta} f(z, 0, 1, ...)\, dz \tag{23}$$

If variance targeting is used, then ω is replaced by,

$$\bar{\sigma}^{\delta} \left(1 - \hat{P}\right) - \sum_{j=1}^{m} \zeta_j \bar{v}_j. \tag{24}$$

Finally, the unconditional variance of the model $\hat{\sigma}^2$ is,

$$\hat{\sigma}^2 = \left(\frac{\hat{\omega}}{1-\hat{P}}\right)^{2/\delta} \tag{25}$$

where $\hat{\omega}$ is the estimated value of the intercept from the GARCH model. The half-life follows from the persistence parameter and is calculated as in Section 2.2.1. The naming conventions for passing fixed or starting parameters for this model are:

- ARCH(q) parameters are 'alpha1', 'alpha2', ...,
- Leverage(q) parameters are 'gamma1', 'gamma2', ...,
- Power parameter is 'delta',
- GARCH(p) parameters are 'beta1', 'beta2', ...,
- variance intercept parameter is 'omega'
- the external regressor parameters are 'vxreg1', 'vxreg2', ...,

In particular, to obtain any of the submodels simply pass the appropriate parameters as fixed.

2.2.6 The family GARCH model ('fGARCH')

The family GARCH model of Hentschel (1995) is another omnibus model which subsumes some of the most popular GARCH models. It is similar to the apARCH model, but more general since it allows the decomposition of the residuals in the conditional variance equation to be driven by different powers for z_t and σ_t and also allowing for both shifts and rotations in the news impact curve, where the shift is the main source of asymmetry for small shocks while rotation drives large shocks.

$$\sigma_t^{\lambda} = \left(\omega + \sum_{j=1}^{m} \zeta_j v_{jt}\right) + \sum_{j=1}^{q} \alpha_j \sigma_{t-1}^{\lambda}(|z_{t-j} - \eta_{2j}| - \eta_{1j}(z_{t-j} - \eta_{2j}))^{\delta} + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^{\lambda} \tag{26}$$

which is a Box-Cox transformation for the conditional standard deviation whose shape is determined by λ, and the parameter δ transforms the absolute value function which it subject to rotations and shifts through the η_{1j} and η_{2j} parameters respectively. Various submodels arise from this model, and are passed to the `ugarchspec` 'variance.model' list via the submodel option,

- The simple GARCH model of Bollerslev (1986) when $\lambda = \delta = 2$ and $\eta_1 j = \eta_2 j = 0$ (submodel = 'GARCH').
- The Absolute Value GARCH (AVGARCH) model of Taylor (1986) and Schwert (1990) when $\lambda = \delta = 1$ and $|\eta_{1j}| \leq 1$ (submodel = 'AVGARCH').
- The GJR GARCH (GJRGARCH) model of Glosten, Jagannathan and Runkle (1993) when $\lambda = \delta = 2$ and $\eta_{2j} = 0$ (submodel = 'GJRGARCH').
- The Threshold GARCH (TGARCH) model of Zakoian (1994) when $\lambda = \delta = 1$, $\eta_{2j} = 0$ and $|\eta_{1j}| \leq 1$ (submodel = 'TGARCH').
- The Nonlinear ARCH model of Higgins and Bera (1992) when $\delta = \lambda$ and $\eta_1 j = \eta_2 j = 0$ (submodel = 'NGARCH').

- The Nonlinear Asymmetric GARCH model of Engle and Ng (1993) when $\delta = \lambda = 2$ and $\eta_1 j = 0$ (submodel = 'NAGARCH').
- The Asymmetric Power ARCH model of Ding, Granger and Engle (1993) when $\delta = \lambda$, $\eta_{2j} = 0$ and $|\eta_{1j}| \leq 1$ (submodel = 'APARCH').
- The Exponential GARCH model of Nelson (1991) when $\delta = 1$, $\lambda = 0$ and $\eta_{2j} = 0$ (not implemented as a submodel of fGARCH).
- The Full fGARCH model of Hentschel (1995) when $\delta = \lambda$ (submodel = 'ALLGARCH').

The persistence of the model is given by,

$$\hat{P} = \sum_{j=1}^{p} \beta_j + \sum_{j=1}^{q} \alpha_j \kappa_j \tag{27}$$

where κ_j is the expected value of the standardized residuals z_t under the Box-Cox transformation of the absolute value asymmetry term,

$$\kappa_j = E(|z_{t-j} - \eta_{2j}| - \eta_{1j}(z_{t-j} - \eta_{2j}))^{\delta} = \int_{-\infty}^{\infty} (|z - \eta_{2j}| - \eta_{1j}(z - \eta_{2j}))^{\delta} f(z, 0, 1, ...)\, dz \tag{28}$$

If variance targeting is used, then ω is replaced by,

$$\bar{\sigma}^{\lambda}\left(1 - \hat{P}\right) - \sum_{j=1}^{m} \zeta_j \bar{v}_j \tag{29}$$

Finally, the unconditional variance of the model $\hat{\sigma}^2$ is,

$$\hat{\sigma}^2 = \left(\frac{\hat{\omega}}{1 - \hat{P}}\right)^{2/\lambda} \tag{30}$$

where $\hat{\omega}$ is the estimated value of the intercept from the GARCH model. The half-life follows from the persistence parameter and is calculated as in Section 2.2.1. The naming conventions for passing fixed or starting parameters for this model are:

- ARCH(q) parameters are 'alpha1', 'alpha2', ...,
- Asymmetry1(q) - rotation - parameters are 'eta11', 'eta12', ...,
- Asymmetry2(q) - shift - parameters are 'eta21', 'eta22', ...,
- Asymmetry Power parameter is 'delta',
- Conditional Sigma Power parameter is 'lambda',
- GARCH(p) parameters are 'beta1', 'beta2', ...,
- variance intercept parameter is 'omega'
- the external regressor parameters are 'vxreg1', 'vxreg2', ...,

2.3 Conditional Distributions

The `rugarch` package supports a range of univariate distributions including the Normal ('norm'), Generalized Error ('ged'), Student ('std') and their skew variants ('snorm', 'sged' and 'sstd') based on the transformations described in Fernandez and Steel (1998) and Ferreira and Steel (2006).[2] Additionally, the Generalized Hyperbolic ('ghyp') and Normal Inverse Gaussian ('nig') distributions are also implemented as is Johnson's reparametrized SU ('jsu') distribution[3] The choice of distribution is entered via the 'distribution.model' option of the `ugarchspec` method. The package also implements a set of functions to work with the parameters of these distributions. These are:

- ddist(distribution = "norm", y, mu = 0, sigma = 1, lambda = -0.5, skew = 1, shape = 5). The density (d*) function.
- pdist(distribution = "norm", q, mu = 0, sigma = 1, lambda = -0.5, skew = 1, shape = 5). The distribution (p*) function.
- qdist(distribution = "norm", p, mu = 0, sigma = 1, lambda = -0.5, skew = 1, shape = 5). The quantile (q*) function.
- rdist(distribution = "norm", n, mu = 0, sigma = 1, lambda = -0.5, skew = 1, shape = 5). The sampling (q*) function.
- fitdist(distribution = "norm", x, control = list()). A function for fitting data using any of the included distributions.
- dskewness(distribution = "norm", skew = 1, shape = 5, lambda = -0.5). The distribution skewness (analytical where possible else by quadrature integration).
- dkurtosis(distribution = "norm", skew = 1, shape = 5, lambda = -0.5). The distribution excess kurtosis (analytical where it exists else by quadrature integration).

This section provides a dry but comprehensive exposition of the required standardization of these distributions for use in GARCH modelling.

The conditional distribution in GARCH processes should be self-decomposable which is a key requirement for any autoregressive type process, while possessing the linear transformation property is required to center $(x_t - \mu_t)$ and scale (ε_t/σ_t) the innovations, after which the modelling is carried out directly using the zero-mean, unit variance, distribution of the standardized variable z_t which is a scaled version of the same conditional distribution of x_t, as described in Equations 6, 7 and 8.

2.3.1 The Normal Distribution

The Normal Distribution is a spherical distribution described completely by it first two moments, the mean and variance. Formally, the random variable x is said to be normally distributed with mean μ and variance σ^2 (both of which may be time varying), with density given by,

$$f(x) = \frac{e^{\frac{-0.5(x-\mu)^2}{\sigma^2}}}{\sigma\sqrt{2\pi}}. \tag{31}$$

[2]These were originally taken from the fBasics package but have been adapted and re-written in C for the likelihood estimation.

[3]From the gamlss package.

Following a mean filtration or whitening process, the residuals ε, standardized by σ yield the standard normal density given by,

$$f\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{\sigma} f\left(z\right) = \frac{1}{\sigma}\left(\frac{e^{-0.5z^2}}{\sqrt{2\pi}}\right). \tag{32}$$

To obtain the conditional likelihood of the GARCH process at each point in time (LL_t), the conditional standard deviation σ_t from the GARCH motion dynamics, acts as a scaling factor on the density, so that:

$$LL_t\left(z_t;\sigma_t\right) = \frac{1}{\sigma_t} f\left(z_t\right) \tag{33}$$

which illustrates the importance of the scaling property. Finally, the normal distribution has zero skewness and zero excess kurtosis.

2.3.2 The Student Distribution

The GARCH-Student model was first used described in Bollerslev (1987) as an alternative to the Normal distribution for fitting the standardized innovations. It is described completely by a shape parameter ν, but for standardization we proceed by using its 3 parameter representation as follows:

$$f\left(x\right) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\beta\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{\left(x-\alpha\right)^2}{\beta\nu}\right)^{-\left(\frac{\nu+1}{2}\right)} \tag{34}$$

where α, β, and ν are the location, scale[4] and shape parameters respectively, and Γ is the Gamma function. Similar to the GED distribution described later, this is a unimodal and symmetric distribution where the location parameter α is the mean (and mode) of the distribution while the variance is:

$$Var\left(x\right) = \frac{\beta\nu}{\left(\nu-2\right)}. \tag{35}$$

For the purposes of standardization we require that:

$$\begin{aligned} Var(x) &= \frac{\beta\nu}{\left(\nu-2\right)} = 1 \\ \therefore \beta &= \frac{\nu-2}{\nu} \end{aligned} \tag{36}$$

Substituting $\frac{(\nu-2)}{\nu}$ into 34 we obtain the standardized Student's distribution:

$$f\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{\sigma} f\left(z\right) = \frac{1}{\sigma}\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\left(\nu-2\right)\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{z^2}{\left(\nu-2\right)}\right)^{-\left(\frac{\nu+1}{2}\right)}. \tag{37}$$

In terms of `R`'s standard implementation of the Student density ('dt'), and including a scaling by the standard deviation, this can be represented as:

$$\frac{dt\left(\frac{\varepsilon_t}{\sigma\sqrt{(v-2)/\nu}},\nu\right)}{\sigma\sqrt{\left(v-2\right)/\nu}} \tag{38}$$

The Student distribution has zero skewness and excess kurtosis equal to $6/(\nu-4)$ for $\nu > 4$.

[4] In some representations, mostly Bayesian, this is represented in its inverse form to denote the precision.

2.3.3 The Generalized Error Distribution

The Generalized Error Distribution (GED) is a 3 parameter distribution belonging to the exponential family with conditional density given by,

$$f(x) = \frac{\kappa e^{-0.5\left|\frac{x-\alpha}{\beta}\right|^{\kappa}}}{2^{1+\kappa^{-1}}\beta\Gamma\left(\kappa^{-1}\right)} \tag{39}$$

with α, β and κ representing the location, scale and shape parameters. Since the distribution is symmetric and unimodal the location parameter is also the mode, median and mean of the distribution (i.e. μ). By symmetry, all odd moments beyond the mean are zero. The variance and kurtosis are given by,

$$\begin{aligned} Var(x) &= \beta^2 2^{2/\kappa}\frac{\Gamma\left(3\kappa^{-1}\right)}{\Gamma\left(\kappa^{-1}\right)} \\ Ku(x) &= \frac{\Gamma\left(5\kappa^{-1}\right)\Gamma\left(\kappa^{-1}\right)}{\Gamma\left(3\kappa^{-1}\right)\Gamma\left(3\kappa^{-1}\right)} \end{aligned} \tag{40}$$

As κ decreases the density gets flatter and flatter while in the limit as $\kappa \to \infty$, the distribution tends towards the uniform. Special cases are the Normal when $\kappa = 2$, the Laplace when $\kappa = 1$. Standardization is simple and involves rescaling the density to have unit standard deviation:

$$\begin{aligned} Var(x) &= \beta^2 2^{2/\kappa}\frac{\Gamma\left(3\kappa^{-1}\right)}{\Gamma\left(\kappa^{-1}\right)} = 1 \\ \therefore \beta &= \sqrt{2^{-2/\kappa}\frac{\Gamma\left(\kappa^{-1}\right)}{\Gamma\left(3\kappa^{-1}\right)}} \end{aligned} \tag{41}$$

Finally, substituting into the scaled density of z:

$$f\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{\sigma}f(z) = \frac{1}{\sigma}\frac{\kappa e^{-0.5\left|\sqrt{2^{-2/\kappa}\frac{\Gamma\left(\kappa^{-1}\right)}{\Gamma\left(3\kappa^{-1}\right)}}z\right|^{\kappa}}}{\sqrt{2^{-2/\kappa}\frac{\Gamma\left(\kappa^{-1}\right)}{\Gamma\left(3\kappa^{-1}\right)}}2^{1+\kappa^{-1}}\Gamma\left(\kappa^{-1}\right)} \tag{42}$$

2.3.4 Skewed Distributions by Inverse Scale Factors

Fernandez and Steel (1998) proposed introducing skewness into unimodal and symmetric distributions by introducing inverse scale factors in the positive and negative real half lines. Given a skew parameter, ξ[5], the density of a random variable z can be represented as:

$$f(z|\xi) = \frac{2}{\xi + \xi^{-1}}\left[f(\xi z)H(-z) + f\left(\xi^{-1}z\right)H(z)\right] \tag{43}$$

where $\xi \in \mathbb{R}^+$ and $H(.)$ is the Heaviside function. The absolute moments, required for deriving the central moments, are generated from the following function:

$$M_r = 2\int_0^{\infty} z^r f(z)\,dz. \tag{44}$$

The mean and variance are then defined as:

$$\begin{aligned} E(z) &= M_1\left(\xi - \xi^{-1}\right) \\ Var(z) &= \left(M_2 - M_1^2\right)\left(\xi^2 + \xi^{-2}\right) + 2M_1^2 - M_2 \end{aligned} \tag{45}$$

The Normal, Student and GED distributions have skew variants which have been standardized to zero mean, unit variance by making use of the moment conditions given above.

[5]When $\xi = 1$, the distribution is symmetric.

2.3.5 The Generalized Hyperbolic Distribution and Sub-Families

In distributions where the expected moments are functions of all the parameters, it is not immediately obvious how to perform such a transformation. In the case of the GHYP distribution, because of the existence of location and scale invariant parametrizations and the possibility of expressing the variance in terms of one of those parametrization, namely the (ζ, ρ), the task of standardizing and estimating the density can be broken down to one of estimating those 2 parameters, representing a combination of shape and skewness, followed by a series of transformation steps to demean, scale and then translate the parameters into the $(\alpha, \beta, \delta, \mu)$ parametrization for which standard formulae exist for the likelihood function. The (ξ, χ) parametrization, which is a simple transformation of the (ζ, ρ), could also be used in the first step and then transformed into the latter before proceeding further. The only difference is the kind of 'immediate' inference one can make from the different parametrizations, each providing a different direct insight into the kind of dynamics produced and their place in the overall GHYP family particularly with regards to the limit cases.
The `rugarch` package performs estimation using the (ζ, ρ) parametrization[6], after which a series of steps transform those parameters into the $(\alpha, \beta, \delta, \mu)$ while at the same time including the necessary recursive substitution of parameters in order to standardize the resulting distribution.

Proof 1 *The Standardized Generalized Hyperbolic Distribution. Let ε_t be a r.v. with mean* (0) *and variance* (σ^2) *distributed as $GHYP(\zeta, \rho)$, and let z be a scaled version of the r.v. ε with variance* (1) *and also distributed as $GHYP(\zeta, \rho)$.*[7] *The density $f(.)$ of z can be expressed as*

$$f(\frac{\varepsilon_t}{\sigma}; \zeta, \rho) = \frac{1}{\sigma} f_t(z; \zeta, \rho) = \frac{1}{\sigma} f_t(z; \tilde{\alpha}, \tilde{\beta}, \tilde{\delta}, \tilde{\mu}), \tag{46}$$

where we make use of the $(\alpha, \beta, \delta, \mu)$ parametrization since we can only naturally express the density in that parametrization. The steps to transforming from the (ζ, ρ) to the $(\alpha, \beta, \delta, \mu)$ parametrization, while at the same time standardizing for zero mean and unit variance are given henceforth.
Let

$$\zeta = \delta\sqrt{\alpha^2 - \beta^2} \tag{47}$$

$$\rho = \frac{\beta}{\alpha}, \tag{48}$$

which after some substitution may be also written in terms of α and β as,

$$\alpha = \frac{\zeta}{\delta\sqrt{(1-\rho^2)}}, \tag{49}$$

$$\beta = \alpha\rho. \tag{50}$$

[6] Credit is due to Diethelm Wurtz for his original implementation in the fBasics package of the transformation/standardization function.
[7] The parameters ζ and ρ do not change as a result of being location and scale invariant

For standardization we require that,

$$\begin{aligned} E(X) &= \mu + \frac{\beta\delta}{\sqrt{\alpha^2-\beta^2}}\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)} = \mu + \frac{\beta\delta^2}{\zeta}\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)} = 0 \\ \therefore \mu &= -\frac{\beta\delta^2}{\zeta}\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)} \qquad (51) \\ Var(X) &= \delta^2\left(\frac{K_{\lambda+1}(\zeta)}{\zeta K_\lambda(\zeta)} + \frac{\beta^2}{\alpha^2-\beta^2}\left(\frac{K_{\lambda+2}(\zeta)}{K_\lambda(\zeta)} - \left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^2\right)\right) = 1 \\ \therefore \delta &= \left(\frac{K_{\lambda+1}(\zeta)}{\zeta K_\lambda(\zeta)} + \frac{\beta^2}{\alpha^2-\beta^2}\left(\frac{K_{\lambda+2}(\zeta)}{K_\lambda(\zeta)} - \left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^2\right)\right)^{-0.5} \qquad (52) \end{aligned}$$

Since we can express, $\beta^2/\left(\alpha^2-\beta^2\right)$ *as,*

$$\frac{\beta^2}{\alpha^2-\beta^2} = \frac{\alpha^2\rho^2}{a^2-\alpha^2\rho^2} = \frac{\alpha^2\rho^2}{a^2(1-\rho^2)} = \frac{\rho^2}{(1-\rho^2)}, \qquad (53)$$

then we can re-write the formula for δ *in terms of the estimated parameters* $\hat{\zeta}$ *and* $\hat{\rho}$ *as,*

$$\delta = \left(\frac{K_{\lambda+1}\left(\hat{\zeta}\right)}{\hat{\zeta}K_\lambda\left(\hat{\zeta}\right)} + \frac{\hat{\rho}^2}{(1-\hat{\rho}^2)}\left(\frac{K_{\lambda+2}\left(\hat{\zeta}\right)}{K_\lambda\left(\hat{\zeta}\right)} - \left(\frac{K_{\lambda+1}\left(\hat{\zeta}\right)}{K_\lambda\left(\hat{\zeta}\right)}\right)^2\right)\right)^{-0.5} \qquad (54)$$

Transforming into the $(\tilde{\alpha},\tilde{\beta},\tilde{\delta},\tilde{\mu})$ *parametrization proceeds by first substituting 54 into 49 and simplifying,*

$$\begin{aligned} \tilde{\alpha} &= \frac{\hat{\zeta}\left(\frac{K_{\lambda+1}(\hat{\zeta})}{\hat{\zeta}K_\lambda(\hat{\zeta})} + \frac{\hat{\rho}^2\left(\frac{K_{\lambda+2}(\hat{\zeta})}{K_\lambda(\hat{\zeta})} - \frac{(K_{\lambda+1}(\hat{\zeta}))^2}{(K_\lambda(\hat{\zeta}))^2}\right)}{(1-\hat{\rho}^2)}\right)^{0.5}}{\sqrt{(1-\hat{\rho}^2)}}, \\ &= \frac{\left(\frac{\hat{\zeta}K_{\lambda+1}(\hat{\zeta})}{K_\lambda(\hat{\zeta})} + \frac{\hat{\zeta}^2\hat{\rho}^2\left(\frac{K_{\lambda+2}(\hat{\zeta})}{K_\lambda(\hat{\zeta})} - \frac{(K_{\lambda+1}(\hat{\zeta}))^2}{(K_\lambda(\hat{\zeta}))^2}\right)}{(1-\hat{\rho}^2)}\right)^{0.5}}{\sqrt{(1-\hat{\rho}^2)}}, \\ &= \left(\frac{\frac{\hat{\zeta}K_{\lambda+1}(\hat{\zeta})}{K_\lambda(\hat{\zeta})}}{(1-\hat{\rho}^2)} + \frac{\hat{\zeta}^2\hat{\rho}^2\left(\frac{K_{\lambda+2}(\hat{\zeta})}{K_{\lambda+1}(\hat{\zeta})}\frac{K_{\lambda+1}(\hat{\zeta})}{K_\lambda(\hat{\zeta})} - \frac{(K_{\lambda+1}(\hat{\zeta}))^2}{(K_\lambda(\hat{\zeta}))^2}\right)}{(1-\hat{\rho}^2)^2}\right)^{0.5}, \\ &= \left(\frac{\frac{\hat{\zeta}K_{\lambda+1}(\hat{\zeta})}{K_\lambda(\hat{\zeta})}}{(1-\hat{\rho}^2)}\left(1 + \frac{\hat{\zeta}\hat{\rho}^2\left(\frac{K_{\lambda+2}(\hat{\zeta})}{K_{\lambda+1}(\hat{\zeta})} - \frac{K_{\lambda+1}(\hat{\zeta})}{K_\lambda(\hat{\zeta})}\right)}{(1-\hat{\rho}^2)}\right)\right)^{0.5}. \qquad (55) \end{aligned}$$

Finally, the rest of the parameters are derived recursively from $\tilde{\alpha}$ *and the previous results,*

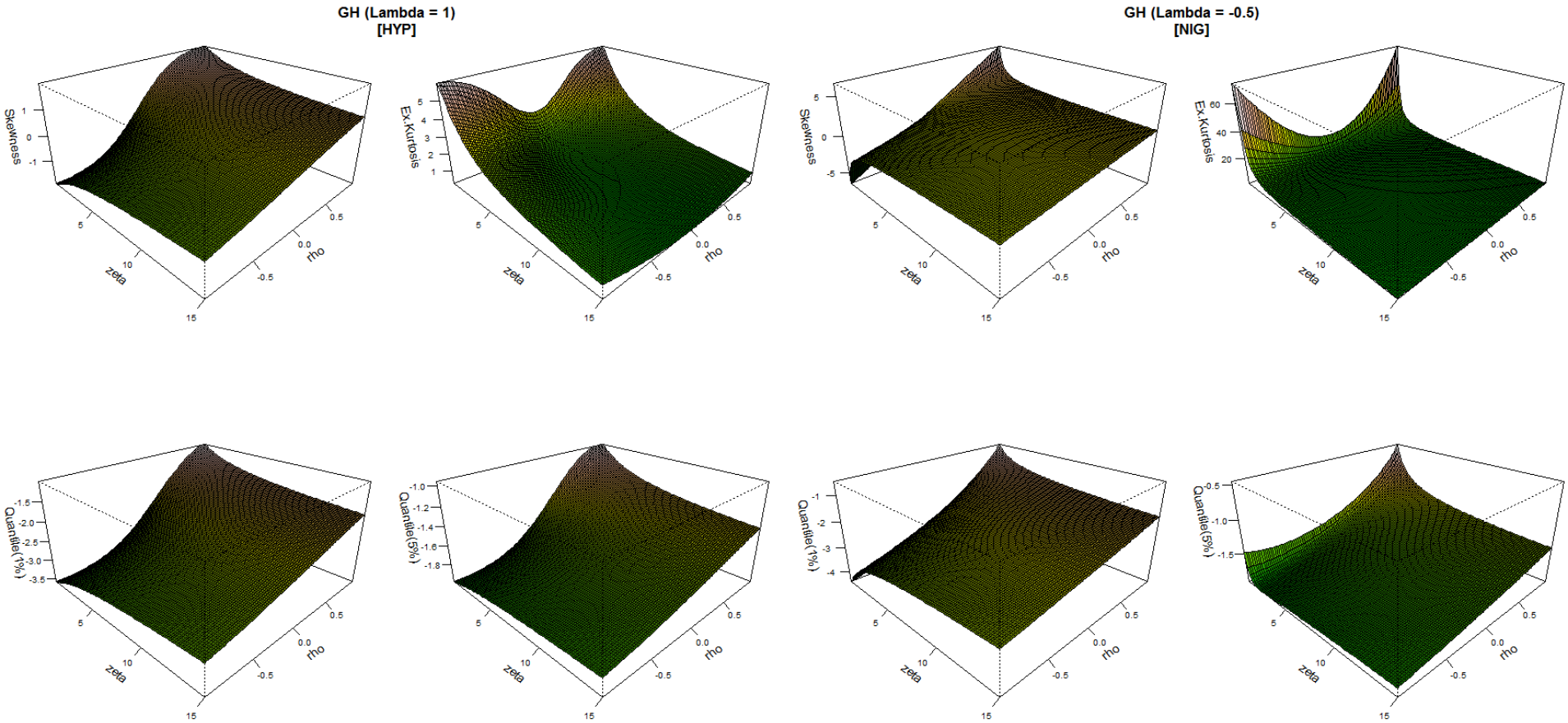
$$\begin{aligned} \tilde{\beta} &= \tilde{\alpha}\hat{\rho}, \qquad (56) \\ \tilde{\delta} &= \frac{\hat{\zeta}}{\tilde{\alpha}\sqrt{1-\hat{\rho}^2}}, \qquad (57) \\ \tilde{\mu} &= \frac{-\tilde{\beta}\tilde{\delta}^2 K_{\lambda+1}\left(\hat{\zeta}\right)}{\hat{\zeta}K_\lambda\left(\hat{\zeta}\right)}. \qquad (58) \end{aligned}$$

For the use of the (ξ, χ) *parametrization in estimation, the additional preliminary steps of converting to the* (ζ, ρ) *are,*

$$\zeta = \frac{1}{\hat{\xi}^2} - 1, \tag{59}$$

$$\rho = \frac{\hat{\chi}}{\hat{\xi}}. \tag{60}$$

Particular care should be exercised when choosing the GH distribution in GARCH models since allowing the GIG λ parameter to vary is quite troublesome in practice and may lead to identification problems since different combinations of the 2 shape (λ, ζ) and 1 skew (ρ) parameters may lead to the same or close likelihood. In addition, large sections of the likelihood surface for some combinations of the distribution parameters is quite flat. Figure 1 shows the skewness, kurtosis and 2 quantiles surfaces for different combinations of the (ρ, ζ) parameters for two popular choices of λ.



(a) $\lambda = 1$(HYP)

(b) $\lambda = -0.5$(NIG)

Figure 1: GH Distribution Skewness, Kurtosis and Quantile Surfaces

2.3.6 Johnson's Reparametrized SU Distribution

The reparametrized Johnson SU distribution, discussed in Rigby (2005), is a four parameter distribution denoted by $JSU(\mu, \sigma, \nu, \tau)$, with mean μ and standard deviation σ for all values of the skew and shape parameters ν and τ respectively. The implementation is taken from the GAMLSS package of Rigby (2005) and the reader is referred there for further details.

3 Fitting

Once a `uGARCHspec` has been defined, the `ugarchfit` method takes the following arguments:

```
> args(ugarchfit)

function (spec, data, out.sample = 0, solver = "solnp", solver.control = list(),
    fit.control = list(stationarity = 1, fixed.se = 0, scale = 0),
    ...)
NULL
```

The out.sample option controls how many data points from the end to keep for out of sample forecasting, while the solver.control and fit.control provide additional options to the fitting routine. Currently, 4 solvers are supported, with the main one being the augmented Lagrange solver solnp of Ye (1987) implemented in R by Ghalanos and Theussl (2011). The main functionality, namely the GARCH dynamics and conditional likelihood calculations are done in C for speed. For reference, there is a benchmark routine called `ugarchbench` which provides a comparison of rugarch against 2 published GARCH models with analytic standard errors, and a small scale comparison with a commercial GARCH implementation. The fitted object is of class `uGARCHfit` which can be passed to a variety of other methods such as show (summary), plot, ugarchsim, ugarchforecast etc. The following example illustrates its use, but the interested reader should consult the documentation on the methods available for the returned class.

```
> spec = ugarchspec()
> data(sp500ret)
> fit = ugarchfit(spec = spec, data = sp500ret, solver.control = list(trace = 0))
> show(fit)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model      : ARFIMA(1,0,1)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.000517    0.000090   5.7620        0
ar1     0.834804    0.058900  14.1732        0
ma1    -0.865358    0.054076 -16.0028        0
omega   0.000001    0.000000   5.2865        0
```

```
alpha1  0.087730    0.007667  11.4428        0
beta1   0.904987    0.008387 107.9058        0

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.000517    0.000101   5.1181 0.000000
ar1     0.834804    0.048413  17.2435 0.000000
ma1    -0.865358    0.044909 -19.2691 0.000000
omega   0.000001    0.000001   2.1924 0.028351
alpha1  0.087730    0.029486   2.9753 0.002927
beta1   0.904987    0.028773  31.4530 0.000000

LogLikelihood : 17901.99

Information Criteria
------------------------------------

Akaike       -6.4805
Bayes        -6.4733
Shibata      -6.4805
Hannan-Quinn -6.4780

Q-Statistics on Standardized Residuals
------------------------------------
      statistic  p-value
Lag10     14.11 0.078859
Lag15     27.78 0.009717
Lag20     31.01 0.028675

H0 : No serial correlation

Q-Statistics on Standardized Squared Residuals
------------------------------------
      statistic p-value
Lag10     3.069  0.9299
Lag15     5.905  0.9495
Lag20     8.424  0.9716

ARCH LM Tests
------------------------------------
             Statistic DoF P-Value
ARCH Lag[2]      1.482   2  0.4765
ARCH Lag[5]      1.738   5  0.8841
ARCH Lag[10]     3.018  10  0.9810

Nyblom stability test
------------------------------------
Joint Statistic:  175.774
Individual Statistics:
mu      0.1978
```

```
ar1     0.2202
ma1     0.1678
omega  21.5725
alpha1  0.1341
beta1   0.1122

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.49 1.68 2.12
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value      prob sig
Sign Bias           0.3309 7.407e-01
Negative Sign Bias  3.0036 2.680e-03 ***
Positive Sign Bias  2.4489 1.436e-02  **
Joint Effect       29.0989 2.135e-06 ***


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     182.8    8.624e-29
2    30     188.2    3.007e-25
3    40     230.3    5.779e-29
4    50     234.0    6.090e-26


Elapsed time : 1.628
```

3.1 Fit Diagnostics

The summary method for the `uGARCHfit` object provides the parameters and their standard errors (and a robust version), together with a variety of tests which can also be called individually.

The `inforcriteria` method on a fitted or filtered object returns the Akaike (AIC), Bayesian (BIC), Hannan-Quinn (HQIC) and Shibata (SIC) information criteria to enable model selection by penalizing overfitting at different rates. Formally, they may be defined as:

$$\begin{aligned} AIC &= \frac{-2LL}{N} + \frac{2m}{N} \\ BIC &= \frac{-2LL}{N} + \frac{m\log_e(N)}{N} \\ HQIC &= \frac{-2LL}{N} + \frac{(2m\log_e(\log_e(N)))}{N} \\ SIC &= \frac{-2LL}{N} + \log_e\left(\frac{(N+2m)}{N}\right) \end{aligned} \tag{61}$$

The Q-Statistics are the test statistics from the Box-Pierce test on the Standardized Residuals with $(10, 15, 20)$ lags and d.o.f the number of the AR and MA parameters, and Squared Standardized Residuals with $(10, 15, 20)$ lags and d.o.f the number of ARCH and GARCH parameters (q, p) Looking at the summary report, the high p-values for the Standardized Squared

Residuals indicates that there is little chance of serial correlation at the lags tested. The evidence for the Standardized Residuals is not as convincing but one should consider other factors, particularly when it comes to forecasting models.

The ARCH LM test of Engle (1982) tests the presence of ARCH effects by regressing the squared residuals of a series against its own lags. Since the NULL is of No ARCH effects, a high p-value, as evidenced by the summary indicates that the GARCH model used was adequate to remove any such effects present prior to fitting (i.e. it is a good idea to test the series prior to fitting a GARCH model!).

The `signbias` calculates the Sign Bias Test of Engle and Ng (1993), and is also displayed in the summary. This tests the presence of leverage effects in the standardized residuals (to capture possible misspecification of the GARCH model), by regressing the squared standardized residuals on lagged negative and positive shocks as follows:

$$\hat{z}_t^2 = c_0 + c_1 I_{\hat{z}_{t-1}<0} + c_2 I_{\hat{z}_{t-1}<0}\hat{z}_{t-1} + c_3 I_{\hat{z}_{t-1}\geqslant 0}\hat{z}_{t-1} + u_t \tag{62}$$

where I is the indicator function and $\hat{z}_t$ the estimated standardized residuals from the GARCH process. The Null Hypotheses are $H_0 : c_i = 0$ (for $i = 1, 2, 3$), and that jointly $H_0 : c_1 = c_2 = c_3 = 0$. As can be inferred from the summary of the previous fit, there is significant Negative and Positive reaction to shocks. Using instead a model such as the apARCH would likely alleviate these effects.

The `gof` calculates the chi-squared goodness of fit test, which compares the empirical distribution of the standardized residuals with the theoretical ones from the chosen density. The implementation is based on the test of Palm and Vlaar (1997) which adjusts the tests in the presence on non-i.i.d. observations by reclassifying the standardized residuals not according to their value (as in the standard test), but instead on their magnitude, calculating the probability of observing a value smaller than the standardized residual, which should be identically standard uniform distributed. The function must take 2 arguments, the fitted object as well as the number of bins to classify the values. In the summary to the fit, a choice of $(20, 30, 40, 50)$ bins is used, and from the summary of the previous example it is clear that the Normal distribution does not adequately capture the empirical distribution based on this test.

The `nymblom` test calculates the parameter stability test of Nyblom (1989), as well as the joint test. Critical values against which to compare the results are displayed.

Finally, some informative plots can be drawn either interactively(which = 'ask'), individually (which = 1:12) else all at once (which = 'all') as in Figure 2.

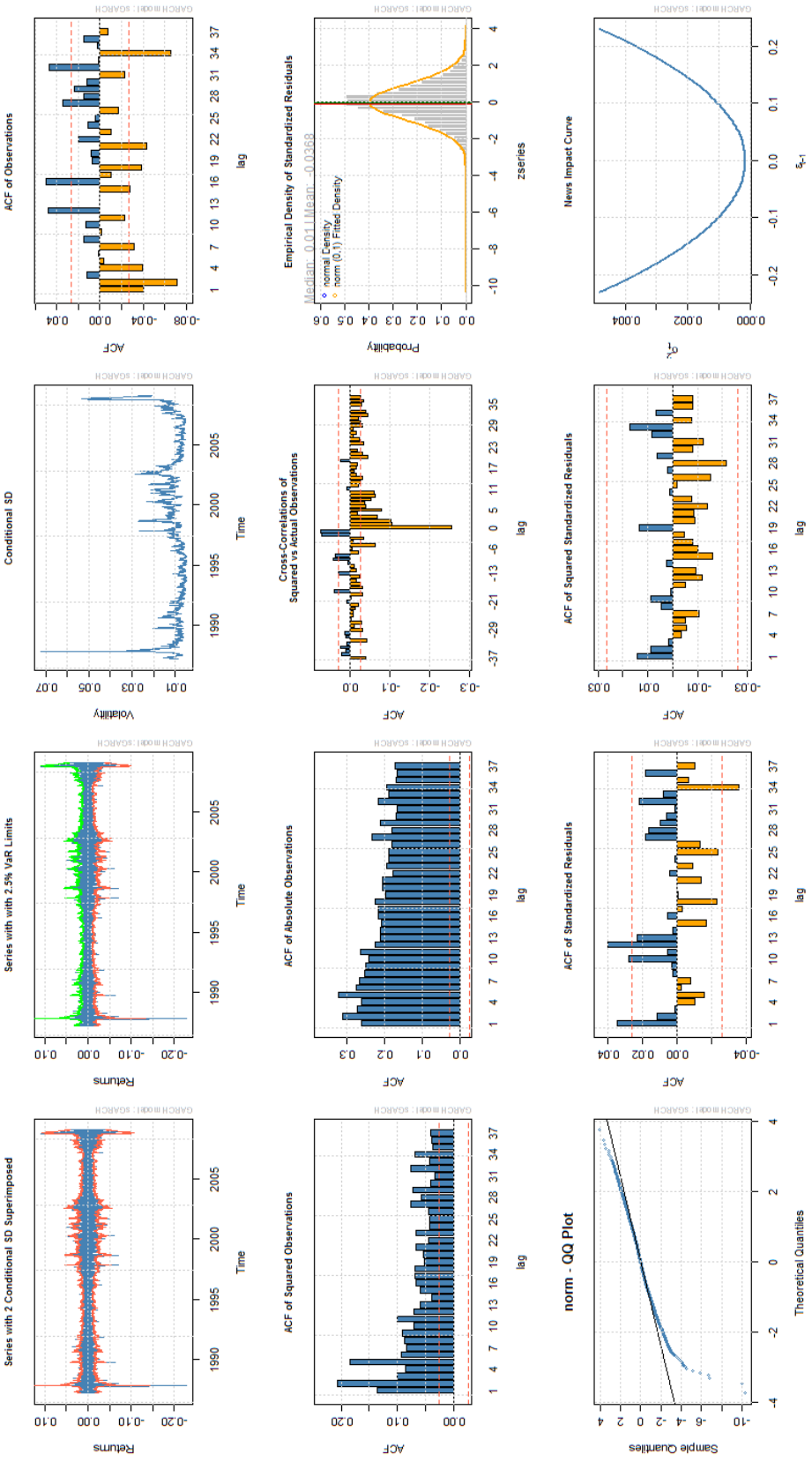


Figure 2: uGARCHfit Plots

4 Filtering

Sometimes it is desirable to simply filter a set of data with a predefined set of parameters. This may for example be the case when new data has arrived and one might not wish to re-fit. The `ugarchfilter` method does exactly that, taking a `uGARCHspec` object with fixed parameters. Setting fixed or starting parameters on the GARCH spec object may be done either through the `ugarchspec` function when it is called (via the fixed.pars and start.pars arguments to the function), else by using the `setfixed<-` and `setstart<-` method on the spec object. The example which follows explains how:

```
> data(sp500ret)
> spec = ugarchspec(variance.model = list(model = "apARCH"), distribution.model = "std")
> setfixed(spec) <- list(mu = 0.01, ma1 = 0.2, ar1 = 0.5, omega = 1e-05,
+     alpha1 = 0.03, beta1 = 0.9, gamma1 = 0.01, delta = 1, shape = 5)
> filt = ugarchfilter(spec = spec, data = sp500ret)
> show(filt)

*----------------------------------*
*          GARCH Model Filter      *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : apARCH(1,1)
Mean Model      : ARFIMA(1,0,1)
Distribution    : std

Filter Parameters
-----------------------------------

mu     1e-02
ar1    5e-01
ma1    2e-01
omega  1e-05
alpha1 3e-02
beta1  9e-01
gamma1 1e-02
delta  1e+00
shape  5e+00

LogLikelihood : 5627.291

Information Criteria
-----------------------------------

Akaike       -2.0345
Bayes        -2.0237
Shibata      -2.0345
Hannan-Quinn -2.0307

Q-Statistics on Standardized Residuals
```

```
--------------------------------------
      statistic p-value
Lag10      1231       0
Lag15      1240       0
Lag20      1248       0

H0 : No serial correlation

Q-Statistics on Standardized Squared Residuals
--------------------------------------
      statistic p-value
Lag10     179.2       0
Lag15     185.3       0
Lag20     189.8       0

ARCH LM Tests
--------------------------------------
             Statistic DoF P-Value
ARCH Lag[2]      171.5   2       0
ARCH Lag[5]      176.4   5       0
ARCH Lag[10]     177.9  10       0


Sign Bias Test
--------------------------------------
                   t-value      prob sig
Sign Bias            8.299 1.303e-16 ***
Negative Sign Bias   5.695 1.297e-08 ***
Positive Sign Bias   0.572 5.673e-01
Joint Effect        95.584 1.383e-20 ***


Adjusted Pearson Goodness-of-Fit Test:
--------------------------------------
  group statistic p-value(g-1)
1    20     27973            0
2    30     39420            0
3    40     49738            0
4    50     58614            0
```

The returned object is of class `uGARCHfilter` and shares many of the methods as the `uGARCHfit` class. Additional arguments to the function are explained in the documentation.

5 Forecasting and the GARCH Bootstrap

There are 2 types of forecasts available with the package. A rolling method, whereby consecutive 1-ahead forecasts are created based on the out.sample option set in the fitting routine, and an unconditional method for n>1 ahead forecasts. It is also possible to combine the 2 creating a rather complicated object. Additionally, it is possible to estimate the forecast density by means of the `ugarchboot` method which implements the strategy described in Pascual, Romo and Ruiz

(2006). This method also includes the option to include parameter uncertainty via a monte carlo method. An example illustrates together with Figure 3 created using the plot command on the resulting `uGARCHboot` object:

```
> data(sp500ret)
> spec = ugarchspec(variance.model = list(model = "eGARCH", garchOrder = c(1,
+     1)), mean.model = list(armaOrder = c(1, 1), arfima = FALSE),
+     distribution.model = "std")
> fit = ugarchfit(spec = spec, data = sp500ret, out.sample = 0,
+     solver = "solnp", solver.control = list(trace = 0))
> bootpred = ugarchboot(fit, method = "Partial", n.ahead = 120,
+     n.bootpred = 2000)
> show(bootpred)

*----------------------------------*
*     GARCH Bootstrap Forecast     *
*----------------------------------*
Model : eGARCH
n.ahead : 120
Bootstrap method:  partial

Series (summary):
          min      q.25      mean     q.75      max forecast
t+1  -0.16970 -0.012573  0.000160 0.014647 0.087117 0.001266
t+2  -0.14045 -0.012198  0.000820 0.015917 0.088455 0.001043
t+3  -0.15926 -0.013462  0.000291 0.014136 0.097728 0.000880
t+4  -0.11009 -0.012629  0.000991 0.014653 0.088158 0.000761
t+5  -0.26634 -0.013391 -0.000698 0.013250 0.123779 0.000674
t+6  -0.24480 -0.012854 -0.000012 0.013587 0.089862 0.000610
t+7  -0.13669 -0.011373  0.001100 0.015331 0.097236 0.000564
t+8  -0.38366 -0.013117 -0.000036 0.014572 0.084560 0.000530
t+9  -0.26574 -0.012405  0.000074 0.014094 0.088518 0.000505
t+10 -0.21571 -0.011858  0.000652 0.013448 0.127877 0.000486
.....................

Sigma (summary):
          min    q0.25     mean    q0.75      max forecast
t+1  0.024673 0.024673 0.024673 0.024673 0.024673 0.024673
t+2  0.023374 0.023534 0.024438 0.024645 0.044920 0.024368
t+3  0.022176 0.022738 0.024186 0.025007 0.049329 0.024071
t+4  0.021142 0.022094 0.023938 0.025026 0.047645 0.023781
t+5  0.020084 0.021567 0.023656 0.024912 0.051322 0.023498
t+6  0.019257 0.021188 0.023476 0.024977 0.065176 0.023221
t+7  0.018390 0.020751 0.023252 0.024853 0.065914 0.022951
t+8  0.017720 0.020401 0.023005 0.024788 0.062841 0.022688
t+9  0.016862 0.020194 0.022880 0.024554 0.093453 0.022430
t+10 0.016061 0.019880 0.022697 0.024362 0.088260 0.022178
.....................
```

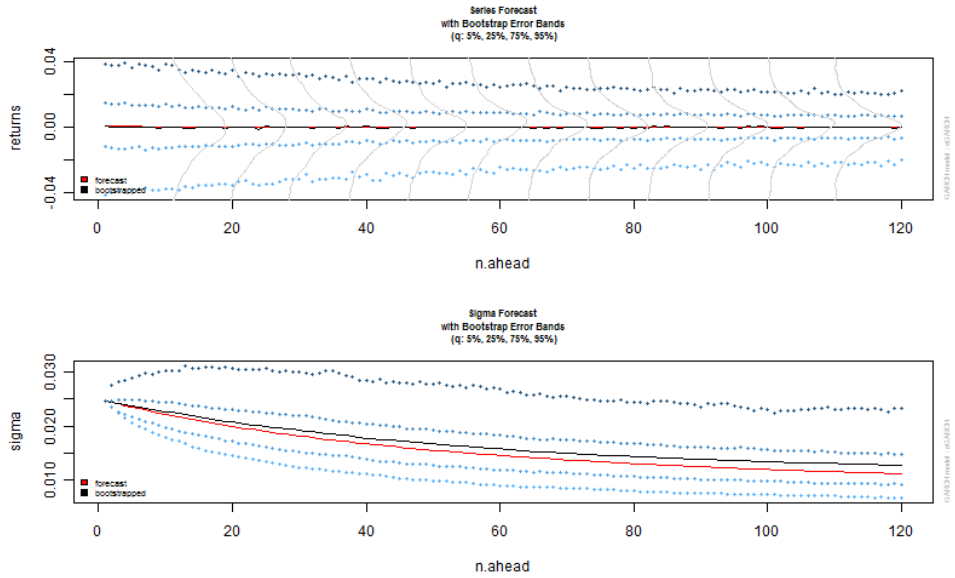


Figure 3: GARCH Bootstrap Forecast Plots

6 Simulation

Simulation may be carried out either directly on a fitted object (`ugarchsim`) else on a GARCH spec with fixed parameters (`ugarchpath`). The `ugarchsim` method takes the following arguments:

```
> args(ugarchsim)

function (fit, n.sim = 1000, n.start = 0, m.sim = 1, startMethod = c("unconditional",
    "sample"), presigma = NA, prereturns = NA, preresiduals = NA,
    rseed = NA, custom.dist = list(name = NA, distfit = NA),
    mexsimdata = NULL, vexsimdata = NULL, ...)
NULL
```

where the n.sim indicates the length of the simulation while m.sim the number of independent simulations. For reasons of speed, when n.sim is large relative to m.sim, the simulation code is executed in C, while for large m.sim a special purpose C++ code (using Rcpp and RcppArmadillo) is used which was found to lead to significant speed increase. Key to replicating results is the rseed argument which is used to pass a user seed to initialize the random number generator, else one will be assigned by the program. In any case, the returned object, of class `uGARCHsim` (or `uGARCHpath`) contains a slot with the seed(s) used.

7 Rolling Estimation

The `ugarchroll` method allows to perform a rolling estimation and forecasting of a model/dataset combination, optionally returning the VaR at specified levels. More importantly, it returns the distributional forecast parameters necessary to calculate any required measure on the forecasted density. The following example illustrates the use of the method where use is also made of the parallel option and run on 10 cores. Figure 4 is generated by calling the plot function on the returned `uGARCHroll` object. Additional methods, and more importantly extractor functions can be found in the documentation.

```
> data(sp500ret)
> spec = ugarchspec(variance.model = list(model = "eGARCH"), distribution.model = "jsu")
> roll = ugarchroll(spec, data = sp500ret, n.ahead = 1, forecast.length = 500,
+     refit.every = 25, refit.window = "recursive", parallel = TRUE,
+     parallel.control = list(pkg = "snowfall", cores = 10), solver = "solnp",
+     solver.control = list(tol = 1e-05, delta = 1e-06, trace = 0),
+     calculate.VaR = TRUE, VaR.alpha = c(0.01, 0.05))

> report(roll, type = "VaR", n.ahead = 1, VaR.alpha = 0.01, conf.level = 0.95)

VaR Backtest Report
==========================================
Model:                  eGARCH-jsu
Backtest Length:        500
Data:                   SP500RET

==========================================
alpha:                  1%
Expected Exceed:        5
Actual VaR Exceed:      9
Actual %:               1.8%

Unconditional Coverage (Kupiec)
Null-Hypothesis:        Correct Exceedances
LR.uc Statistic:        2.613
LR.uc Critical:         3.841
LR.uc p-value:          0.106
Reject Null:            NO

Conditional Coverage (Christoffersen)
Null-Hypothesis:        Correct Exceedances &
                        Independence of Failures
LR.cc Statistic:        2.943
LR.cc Critical:         5.991
LR.cc p-value:          0.23
Reject Null:            NO

> report(roll, type = "fpm")

GARCH Roll Mean Forecast Performance Measures
--------------------------------------------
Model : eGARCH
no.refits : 20
n.ahead   : 1
n.rolls   : 500

    n.ahead.1
MSE 4.080e-04
MAE 1.295e-02
DAC 5.500e-01
N   5.000e+02
```

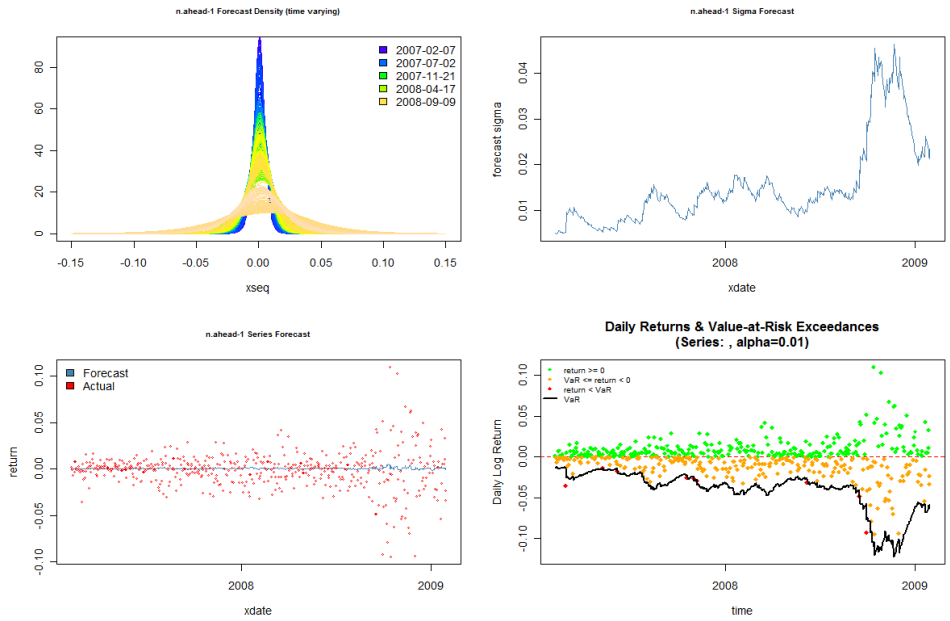


Figure 4: GARCH Rolling Forecast Plots

8 Simulated Parameter Distribution and RMSE

It is sometimes instructive to be able to investigate the underlying density of the estimated parameters under different models. The `ugarchdistribution` method performs a monte carlo experiment by simulating and fitting a model multiple times and for different 'window' sizes. This allows to obtain some insight on the consistency of the parameter estimates as the data window increases by looking at the rate of decrease of the Root Mean Squared Error and whether we have $\sqrt{N}$ consistency. This is a computationally expensive exercise and as such should only be undertaken in the presence of ample computing power and RAM. As in other functions, parallel functionality is enabled if available. The example which follows illustrates an instance of this test on one model and one set of parameters. Figures 5 and 6 complete this example.

```
> spec = ugarchspec(variance.model = list(model = "gjrGARCH"),
+     distribution.model = "ged")
> print(persistence(pars = unlist(list(mu = 0.001, ar1 = 0.4, ma1 = -0.1,
+     omega = 1e-06, alpha1 = 0.05, beta1 = 0.9, gamma1 = 0.05,
+     shape = 1.5)), distribution = "ged", model = "gjrGARCH"))

persistence
      0.975

> setfixed(spec) <- list(mu = 0.001, ar1 = 0.4, ma1 = -0.1, omega = 1e-06,
+     alpha1 = 0.05, beta1 = 0.9, gamma1 = 0.05, shape = 1.5)
> dist = ugarchdistribution(fitORspec = spec, n.sim = 2000, n.start = 1,
+     m.sim = 100, recursive = TRUE, recursive.length = 6000, recursive.window = 1000,
+     rseed = 1066, solver = "solnp", solver.control = list(trace = 0),
+     parallel = TRUE, parallel.control = list(pkg = "snowfall",
+         cores = 20))
> show(dist)
```

```
*-----------------------------------*
*    GARCH Parameter Distribution    *
*-----------------------------------*
Model : gjrGARCH
No. Paths (m.sim) : 100
Length of Paths (n.sim) : 2000
Recursive : TRUE
Recursive Length : 6000
Recursive Window : 1000

Coefficients: True vs Simulation Mean (Window-n)
                    mu     ar1       ma1      omega   alpha1   beta1   gamma1
true-coef   0.00100000 0.40000 -0.100000 1.0000e-06 0.050000 0.90000 0.050000
window-2000 0.00097122 0.39691 -0.097773 1.0672e-06 0.046603 0.90054 0.051852
window-3000 0.00101426 0.38922 -0.089512 1.0209e-06 0.047097 0.90072 0.052936
window-4000 0.00099796 0.39372 -0.095054 1.0219e-06 0.049798 0.90056 0.047732
window-5000 0.00099337 0.40143 -0.102950 1.0081e-06 0.048943 0.90087 0.049678
window-6000 0.00098468 0.39561 -0.096174 1.0102e-06 0.049052 0.90009 0.050879
             shape
true-coef   1.5000
window-2000 1.4944
window-3000 1.4960
window-4000 1.4929
window-5000 1.4899
window-6000 1.4902
```

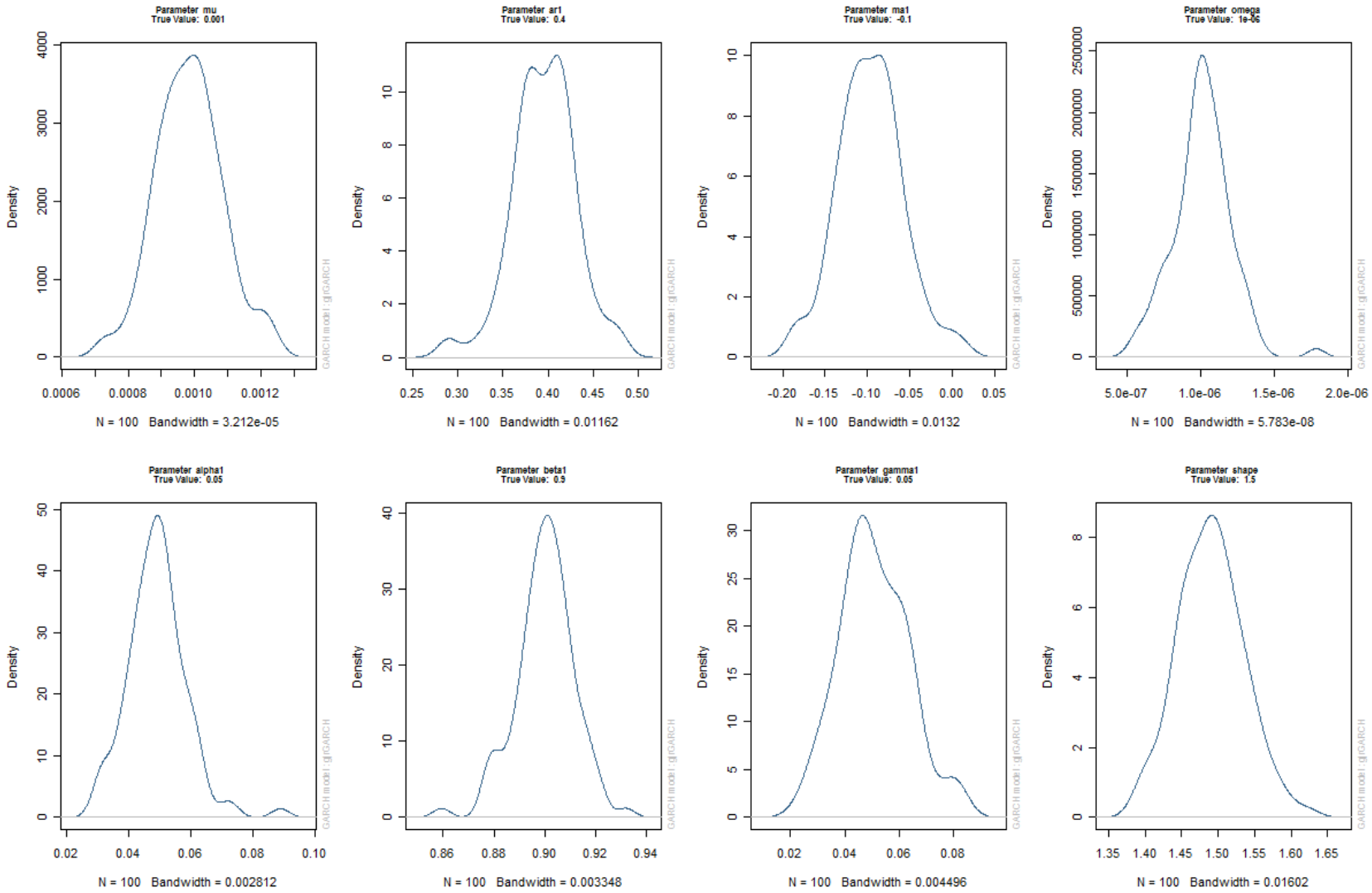


Figure 5: Simulated Parameter Density

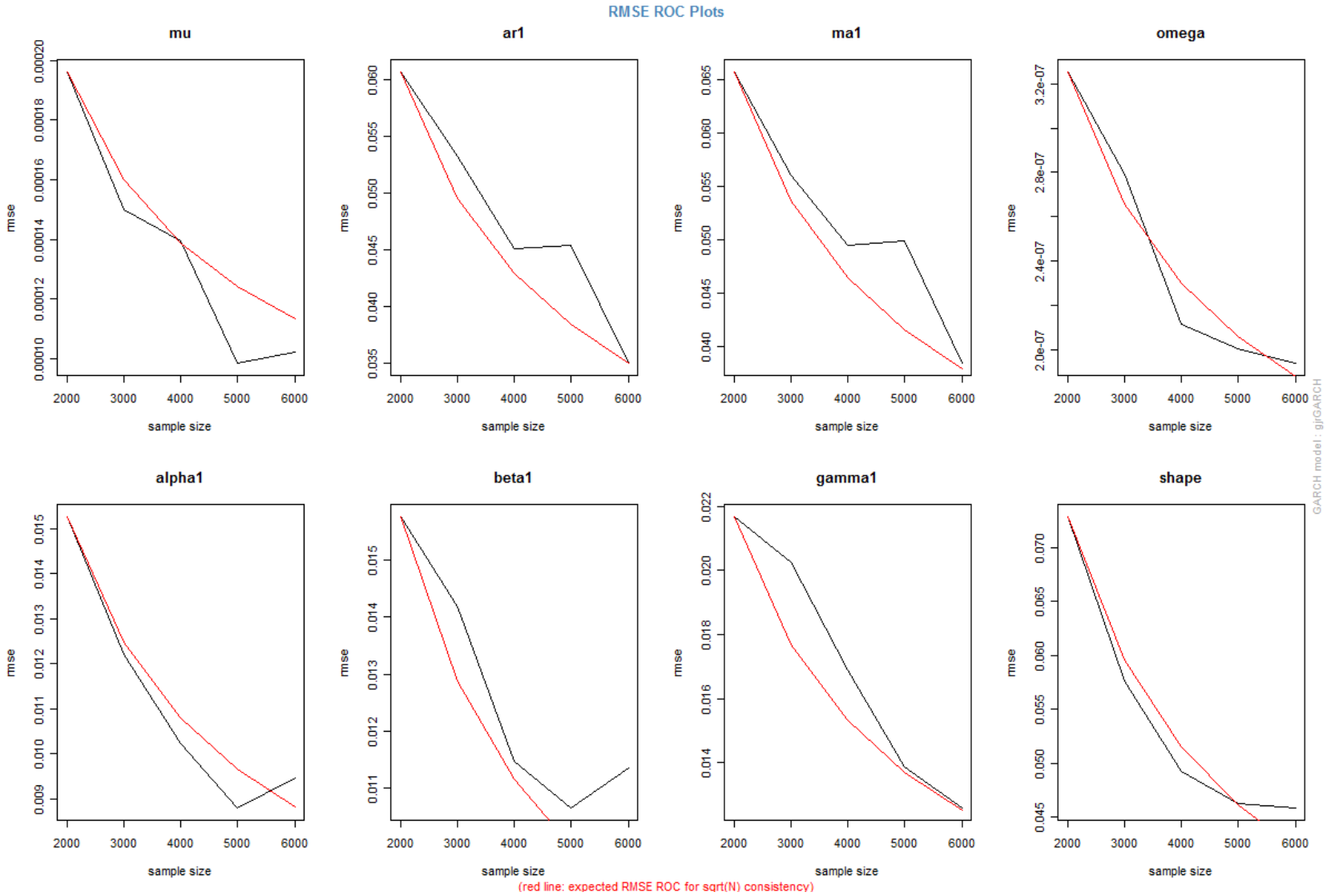
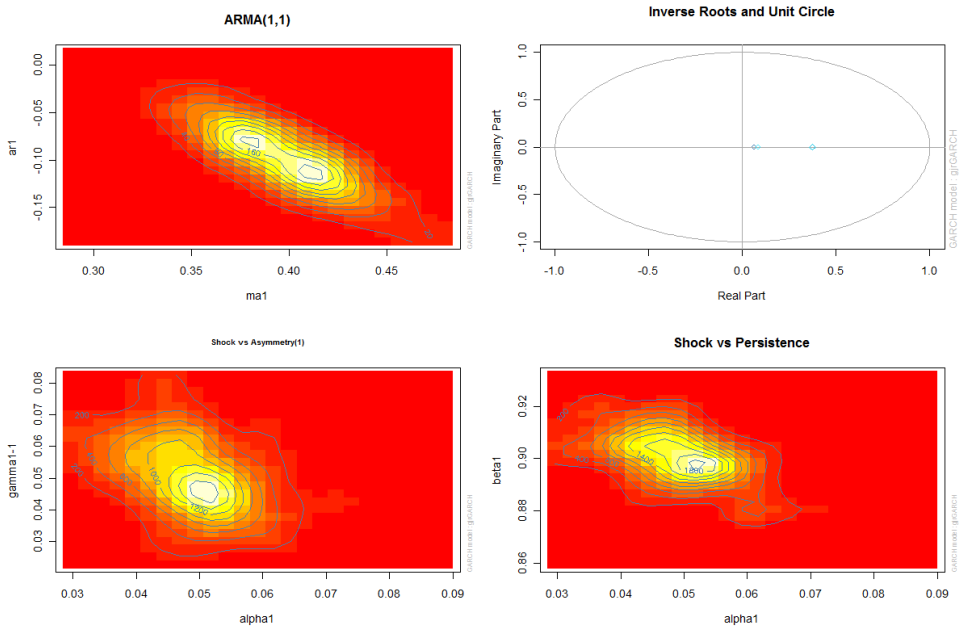
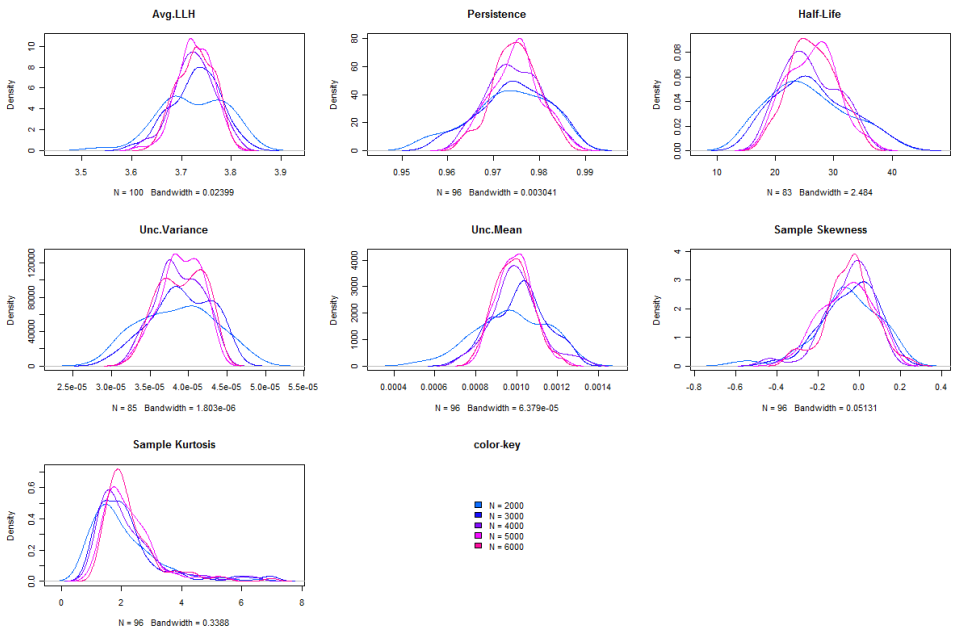


Figure 6: RMSE Rate of Change



(a) Bivariate Parameter Plots



(b) GARCH Stat Plots

Figure 7: GARCH Simulated Parameters Density

9 The ARFIMAX Model with constant variance

The `rugarch` package implements an additional set of methods and classes, mirroring those of the GARCH specification, for modelling ARFIMAX processes with constant variance via Maximum Likelihood. With the exception of plots, the functionality is very similar to that covered so far for GARCH methods. The main functions are `arfimaspec`, `arfimafit`, `arfimaforecast`, `arfimasim`, `arfimapath`, `arfimadistirbution` and `arfimaroll`. The usual extractor, inference and summary methods are replicated for all the ARFIMA classes and the user should consult the documentation for further details.

10 Miscellaneous Functions

There are a number of plain R functions exported in the package the most important of which are the `WeekDayDummy` (creates a dummy, day of the week variable given a set of dates), `ForwardDates` (to generate A POSIXct vector of future dates), `BerkowitzLR` which implements the density forecast test of Berkowitz (2001), and the `DACTest` function which implements the Directional Accuracy tests of Anatolyev and Gerko (2005) and Pesaran and Timmermann (1992). The unconditional and conditional VaR exceedances tests are currently not exported but may be called via rugarch:::.VaRreport (and the associated plot via rugarch:::.VaRplot).

11 FAQs and Guidelines

This section provides for answers to some Frequently Asked Questions (Q) as well as Guidelines (G) for the use of the `rugarch` package.

Q: Does the package support parallel computation?

Yes. Most functions for which parallel functionality was deemed beneficial include the additional options of 'parallel' and 'parallel.control'. For Unix based systems the multicore package of Urbanek (2009) provides for a very efficient parallel apply on multiple cores. For Windows and all systems, the snowfall package of Knaus (2010) allows for socket based parallel apply which does not include shared memory of objects and can therefore be very expensive. This is definitely not as efficient as multicore, but the only option currently on windows systems. Some extra thought is required when using the snowfall parallel functionality as there is a trade-off to consider between the number of cores committed via socket, the overhead for setting up each new socket and the number of parallel iterations. As an example consider the following:

```
> library(rugarch)

rugarch (version 1.0) initialized.

> data(dji30ret)
> spec = ugarchspec()
> mspec = multispec(replicate(spec, n = 30))
> print(system.time(multifit(multispec = mspec, data = dji30ret[,
+     1:30], parallel = TRUE, parallel.control = list(pkg = "snowfall",
+     cores = 20))))

   user  system elapsed
   1.08    1.88   24.00
```

```
> print(system.time(multifit(multispec = mspec, data = dji30ret[,
+     1:30], parallel = TRUE, parallel.control = list(pkg = "snowfall",
+     cores = 5))))

   user  system elapsed
   0.34    0.23   15.78

> print(system.time(multifit(multispec = mspec, data = dji30ret[,
+     1:30], solver.control = list(trace = 0), parallel = FALSE,
+     parallel.control = list(pkg = "snowfall", cores = 10))))

   user  system elapsed
  49.15    0.00   49.15
```

It is clear that the overhead for using 20 cores is too high for the size of this problem, whereas using 5 cores does lead to a doubling of performance versus the non-parallel version. Also note that prior to using the snowfall package, this should be manually loaded by the user, since strange behavior has been observed when the package tries to load the snowfall itself.

Q: My model does not converge, what can I do?

There are several avenues to consider here. The package offers 4 different solvers, namely 'solnp', 'gosolnp', 'nlminb' and 'L-BGFS-U' (from optim). Each solver has its own merits, and control parameters which may, and should be passed, via the solver.control list in the fitting routines, depending on your particular data. For problems where neither 'solnp' nor 'nlminb' seem to work, try the 'gosolnp' solver which does a search of the parameter space based on a truncated normal distribution for the parameters and then initializes multiple restarts of the 'solnp' solver based on the best identified candidates. The numbers of randomly generated parameters (n.sim) and solver restarts (n.restarts) can be passed via the solver.control list. Additionally, in the fit.control list of the fitting routines, the option to perform scaling of the data prior to fitting usually helps, although it is not available under some setups. Finally, consider the amount of data you are using for modelling GARCH processes, which leads to another FAQ below.

Q: How much data should I use to model GARCH processes with confidence?
The distribution of the parameters varies by model, and is left to the reader to consult relevant literature on this. However, using 100 data points to try and fit a model is unlikely to be a sound approach as you are unlikely to get very efficient parameter estimates. The `rugarch` package does provide a method (`ugarchdistribution`) for simulating from a pre-specified model, data of different sizes, fitting the model to the data, and inferring the distribution of the parameters as well as the RMSE rate of change as the data length increases. This is a very computationally expensive way to examine the distribution of the parameters (but the only way in the non-Bayesian world), and as such should be used with care and in the presence of ample computing power.

Q: Where can one find more examples?

The package has a folder called 'rugarch.tests' which contains many tests which I use for debugging and checking. The files in the folder should be 'sourced' by the user, and the 'runtests.R' file contains some wrapper functions which describe what each test does, and optionally runs chosen tests. The output will be a combination of text files (.txt) and figures (either .eps or .png) in an output directory which the user can define in the arguments to the wrapper function

'rugarch.runtests'. It is quite instructive to read and understand what each test is doing prior to running it...

References

Alexander, C. (2001). Orthogonal GARCH. In *Mastering risk, 2*, pp.˜21–38.

Anatolyev, S. and Gerko, A. (2005). A trading approach to testing for predictability. In *Journal of Business and Economic Statistics, 23(4)*, pp.˜455–461.

de Athayde, G.M. and Flores Jr, R.G. (2002). On Certain Geometric Aspects of Portfolio Optimisation with Higher Moments. In *mimeo*.

Berkowitz, J. (2001). Testing density forecasts with applications to risk management. In *Journal of Business and Economic Statistics 19(4)*, pp.˜465–474.

Blaesild, P. (1981). The two-dimensional hyperbolic distribution and related distributions, with an application to Johannsen's bean data. In *Biometrika, 68(1)*, pp.˜251–263.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. In *Journal of Econometrics, 31(3)*, pp.˜307–327.

Bollerslev, T. (1987). A Conditionally Heteroskedastic Time Series Model for Speculative Prices and Rates of Return. In *Review of Economics and Statistics, 69*, pp.˜542–547

Box, G. and Cox, D.R. (1964). An analysis of transformations. In *Journal of the Royal Statistical Society, Series B (Methodological)*, pp.˜211–252.

Box, G. and Jenkins, G.M. and Reinsel, G.C. Time Series Analysis: Forecasting and Control. In *Holden-Day*.

Broda, S.A. and Paolella, M.S. (2009). CHICAGO: A Fast and Accurate Method for Portfolio Risk Calculation. In *Journal of Financial Econometrics, 7(14)*, pp.˜412–436.

Chen, Y. and Härdle, W. and Spokoiny, V. (2007). Portfolio value at risk based on independent component analysis. In *Journal of Computational and Applied Mathematics, 205(1)*, pp.˜594–607.

Ding, Z. and Granger, C.W.J. and Engle, R.F. (1993). A long memory property of stock market returns and a new model. In *Journal of Empirical Finance, 1(1)*, pp.˜83–106.

Engle, R.F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. In *Econometrica, 50(4)*, pp.˜987–1007.

Engle, R.F. and Bollerslev, T. (1986). Modeling the Persistence of Conditional Variances. In *Econometric Reviews, 5*, pp.˜1–50.

Engle, R.F. and Ng, V.K. (1993). Measuring and testing the impact of new on volatility. In *Journal of Finance, 48*, pp.˜1749–1778.

Engle, R.F. and Lilien, D.M. and Robbins, R.P. (1987) Estimating Time Varying Risk Premia in the Term Structure: The ARCH-M Model. In *Econometrica, 55(2)*, pp.˜391?–407.

Engle, R. and Mezrich, J. (1996) GARCH for Groups. In *RISK, (9)*, pp.˜36–40.

Fernandez, C. and Steel, M.F. (1998). On Bayesian Modeling of Fat Tails and Skewness. In *Journal of the American Statistical Association, 93(441)*, pp.˜359–371.

Ferreira, J.T. and Steel, M.F. (2006). A constructive representation of univariate skewed distributions. In *Journal of the American Statistical Association, 101(474)*, pp.˜823–829.

Ghalanos, A. and Theussl, S. (2011) Rsolnp: General Non-linear Optimization Using Augmented Lagrange Multiplier Method. In *manual*.

Glosten, L.R. and Jagannathan, R. and Runkle, D.E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. In *Journal of Finance, 48(5)*, pp.˜1779–1801.

Geweke, J. (1986). Modeling the Persistence of Conditional Variances: A Comment. In *Econometric Reviews, (5)*, pp.˜57–61.

Hansen, B.E. (1990). Lagrange multiplier tests for parameter instability in non-linear models. In *mimeo*.

Hansen, B.E. (1994). Autoregressive conditional density estimation. In *International Economic Review*, pp.˜705–730.

Hentschel, L. (1995). All in the family Nesting symmetric and asymmetric GARCH models. In *Journal of Financial Economics, 39(1)*, pp.˜71–104.

Higgins, M. L. and Bera, A. K. (1992). A Class of Nonlinear ARCH Models. In *International Economic Review, 33*, pp.˜137–158.

Knaus, J. (2010). snowfall: Easier cluster computing (based on snow). *Manual*.

Mandelbrot, B. (1963). The variation of certain speculative prices. In *Journal of business, 36(4)*, pp.394–419.

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

Nelson, D.B (1989). Testing for the Constancy of Parameters Over Time. In *Journal of the American Statistical Association, 84(405)*, pp.223–230.

Palm, F.C. and Vlaar, P.J.G. (1997). Simple diagnostic procedures for modeling financial time series. In *Allgemeines statistisches Archiv: Organ der Deutschen Statistischen Gesellschaft, 81*, pp.˜85–101.

Pantula, S. G. (1986). Comment: Modelling the persistence of conditional variances. In *Econometric Reviews, (5)*, pp.˜71–73.

Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized Additive Models for Location, Scale and Shape. In *Journal of the Royal Statistical Society. Series C (Applied Statistics), 54(3)*, pp.˜507–554.

Pascual, L. and Romo, J. and Ruiz, E. (2004). Bootstrap predictive inference for ARIMA processes. In *Journal of Time Series Analysis, 25(4)*, pp.˜449–465.

Pascual, L. and Romo, J. and Ruiz, E. (2006). Bootstrap prediction for returns and volatilities in GARCH models. In *Computational Statistics and Data Analysis, 50(9)*, pp.˜2293–2312.

Pesaran, M.H. and Timmermann, A. (1992). A simple nonparametric test of predictive performance. In *Journal of Business and Economic Statistics, 10(4)*, pp.˜461–465.

van der Weide, R. (2002). GO-GARCH: a multivariate generalized orthogonal GARCH model. In *Journal of Applied Econometrics, 17(5)*, pp.˜549–564.

van der Weide, R. (2004). Wake me up before you GO-GARCH. In *Computing in Economics and Finance (2004)*.

van der Weide, R. and Boswijk, P. (2008). Method of moments estimation of GO-GARCH models. In *mimeo*.

Zhang, K. and Chan, L. (2009). Efficient factor GARCH models and factor-DCC models. In *Quantitative Finance, 9(1)*, pp.˜71–91.

Schmidt, R. and Hrycej, T. and Stützle, E. (2006). Multivariate distribution models with generalized hyperbolic margins. In *Computational statistics and data analysis, 50(8)*, pp.˜2065–2096.

Schwert, W (1990). Stock volatility and the crash of '87. In *Review of financial Studies (3)*, pp.˜77–102.

Taylor, S. (1986). Modelling Financial Time Series. *Wiley*, New York.

Urbanek, S. (2009). multicore: Parallel processing of R code on machines with multiple cores or CPUs. *Manual*.

Ye, Y. (1987). Interior Algorithms for Linear, Quadratic, and Linearly Constrained Non-Linear Programming. In *PhD Thesis, Department of ESS, Stanford University*.

Zakoian, J-M. (1994). Threshold Heteroskedasticity Models. In *Journal of Economic Dynamics and Control (15)*, pp.˜931–955.