Computing with R
Teaching Computational Finance

Modelling Univariate GARCH Processes

finGARCH - User Guide

Diethelm Würtz

Institut für Theoretische Physik
ETH Zürich

August 6, 2004

1 Introduction

In particular, [we] analyzes[d] seven widely used [GARCH] packages, utilizing a recently developed benchmark. Four of the packages are found to be unsuitable, in most cases because the developer either does not specifically indicate which of the many possible GARCH models is being estimated, or does not accommodate the most common model specified in the applied literature, or both. A principal finding is that implementation of the GARCH procedure varies so widely that two packages ostensibly doing the same thing actually may be estimating substantively different models.
McCullough and Renfro 1999

GARCH, *Generalized Autoregressive Conditional Heteroskedastic*, models have become important in the analysis of time series data, particularly in financial applications when the goal is to analyze and forecast volatility. For this purpose, we describe a R-package for simulating, estimating and forecasting various univariate GARCH-type time series models in the conditional variance and an ARMA specification in the conditional mean. We present a numerical implementatation of the Maximum Likelihood approach under four assumptions, Normal, Student-t, GED or skewed Student-t errors. The parameter estimates are checked by several diagnostic analysis tools including graphical features and hypothesis tests. One step ahead forecasts of both the conditional mean and variance are available.

In this document we consider as a first step univariate ARMA, models with GARCH, errors. The name GARCH is considered in a broader sense denoting a whole family of models. The number of GARCH models is immense, but the most influential models were the first. Beside the standard ARCH model introduced by Engle (1982) and the GARCH model introduced by Bollerslev (10986), we consider also the more general class of asymmetric power ARCH models (APARCH) introduced by Ding, Granger and Engle (1993). APARCH models itself include as special cases the TS-GARCH model of Taylor (1986) and Schwert (1989), the GJR-GARCH model of Glosten, Jaganathan, and Runkle (1993), the T-ARCH model of Zakoian (1993), the N-ARCH model of Higgins and Bera (1992), and the Log-ARCH model of Geweke (1986) and Pentula (1986).

Coupled with these models was a sophisticated analysis of the stochastic process of data generated by the underlying process as well as estimators for the unknown model parameters. Theorems for the autocorrelations, moments and stationarity and ergodicity of GARCH processes have been developed for many of the important cases. There exist a collection of review articles by Bollerslev, Chou and Kroner(1992), Bera and Higgins(1993), Bollerslev, Engle and Nelson(1994), Engle(2001), Engle and Patton(2001), Li, Ling and McAleer(2002) that give a good overview of the scope of the research.

The aim of this user guide is threefold. It gives a brief introduction to the concepts of GARCH time series modelling, it shows how the algorithms are implemented in the R environment using the S language, and gives some useful hints and suggestions for adding your own functions.

2 Mean and Variance Equation

We describe the mean equation of an univariate time series y_t by the process

$$y_t \;=\; E(y_t|\Omega_{t-1}) + \varepsilon_t \;, \tag{1}$$

where $E(\cdot|\cdot)$ denotes the conditional expectation operator, Ω_{t-1} the information set at tine $t-1$, and ε_t the innovations or residuals of the time series. ε_t describes uncorrelated disturbances with zero mean and plays the role of the unpredictable part of the time series. In the following we model the mean equation as an ARMA process, and the innovations are generated from a GARCH process.

2.1 ARMA Mean Equation

The ARMA(m,n) process of autoregressive order m and moving average order n can be described as

$$\begin{aligned} y_t \;&=\; \mu \;+\; \sum_{i=1}^{m} a_i y_{t-i} \;+\; \sum_{j=1}^{n} b_j \varepsilon_{t-j} \;, \\ &=\; \mu \;+\; a(\mathcal{B}) y_t \;+\; b(\mathcal{B}) \varepsilon_t \;, \end{aligned} \tag{2}$$

with mean μ, autoregressive coefficients a_i and moving average coefficients b_i. Note, that the model can be expressed in a quite comprehensive form using the backshift operator $\mathcal{B}$ defined by $\mathcal{B}x_t = x_{t-1}$. The functions $a(\mathcal{B})$ and $b(\mathcal{B})$ are polynomials of degree m and n respectively in the backward shift operator $\mathcal{B}$. If $m = 0$ we have a pure autoregressive process and if on the other hand $n = 0$ we have a pure moving average process.

The ARMA time series is *stationary* when the series $a(\mathcal{B})$, which is the generating function of the coefficients a, converges for $|\mathcal{B}| < 1$ that is, on or within the unit circle.

2.2 GARCH Variance Equation

The mean equation cannot take into account for heteroskedastic effects of the time series process typically observed in form of fat tails, as clustering of volatilities, and the leverage effect. In this context Engle (1982) introduced the *Autoregressive Conditional Heteroskedastic* model, named ARCH, later generalized by Bollerslev (1986), named GARCH.

The ε_t terms in the ARMA mean equation (2) are the innovations of the time series process. Engle (1982) defined them as an autoregressive conditional heteroscedastic process where all ε_t are of the form

$$\varepsilon_t \;=\; z_t \; \sigma_t \;, \tag{3}$$

where z_t is an *iid* process with zero mean and unit variance. Although ε_t is serially uncorrelated by definition its conditional variance equals σ_t^2 and, therefore, may change over time. All the GARCH models we consider in the following differ only in their functional form for the conditional variance.

The GARCH Model:

The variance equation of the GARCH(p,q) model can be expressed as:

$$\begin{aligned} \varepsilon_t &= z_t\sigma_t \ , \\ z_t &\sim \mathcal{D}(0,1) \ , \\ \sigma_t^2 &= \omega \ + \ \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 \ + \ \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \ , \\ &= \omega \ + \ \alpha(\mathcal{B})\varepsilon_{t-1}^2 \ + \ \beta(\mathcal{B})\sigma_{t-1}^2 \end{aligned} \tag{4}$$

where $\mathcal{D}(0,1)$ is the probability density function of the innovations or residuals with zero mean and unit variance. If all the coefficients β are zero the GARCH model is reduced to the ARCH model. Like for ARMA models a GARCH specification often leads to a more parsimonious representation of the temporal dependencies and thus provides a similar added flexibility over the linear ARCH model when parameterizing the conditional variance.

Bolerslev (1986) has shown that the GARCH(p,q) process is wide-sense stationary with $E(\varepsilon_t) = 0$, $var(\varepsilon_t) = \omega/(1-\alpha(1)-\beta(1))$ and $cov(\varepsilon_t, \varepsilon_s) = 0$ for $t \neq s$ if and only if $\alpha(1)+\beta(1) < 1$.

The APARCH Model:

The variance equation of the APARCH(p,q) model can be written as:

$$\begin{aligned} \varepsilon_t &= z_t\sigma_t \ , \\ z_t &\sim \mathcal{D}(0,1) \ , \\ \sigma_t^\delta &= \omega \ + \ \sum_{i=1}^{q} \alpha_i (|\varepsilon_{t-i}| - \gamma_i \varepsilon_{t-i})^\delta \ + \ \sum_{j=1}^{p} \beta_j \sigma_{t-j}^\delta \ , \end{aligned} \tag{5}$$

where $\delta > 0$ and $-1 < \gamma_i < 1$. This model has been introduced by Ding, Granger, and Engle (1993). It adds the flexibility of a varying exponent with an asymmetry coefficient to take the leverage effect into account. A stationary solution exists if $\omega > 0$, and $\Sigma_i \alpha_i \kappa_i + \Sigma_j \beta_j < 1$, where $\kappa_i = E(|z| + \gamma_i z)^\delta$. Note, that if $\gamma \neq 0$ and/or $\delta \neq 2$ the κ_i depend on the assumptions made on the innovation process.

The APARCH includes seven other ARCH extensions as special cases:

- *ARCH Model of Engle* when $\delta = 2$, $\gamma_i = 0$, and $\beta_j = 0$.
- *GARCH Model of Bollerslev* when $\delta = 2$, and $\gamma_i = 0$.
- *TS-GARCH Model of Taylor and Schwert* when $\delta = 1$, and $\gamma_i = 0$.
- *GJR-GARCH Model of Glosten, Jagannathan, and Runkle* when $delta = 2$.
- *T-ARCH Model of Zakoian* when $delta = 1$.
- *N-ARCH Model of Higgens and Bera* when $\gamma_i = 0$, and $\beta_j = 0$.
- *Log-ARCH Model of Geweke and Pentula* when $\delta \to 0$.

3 GARCH Modelling Process

The modelling of a GARCH process consists of several steps: The specification of the model, the generation of artificial time series for simulation and testing, the parameter estimation, the diagnostic analysis, and the computation of forecasts. For each step a R-function is available.

Functions for GARCH Modelling Process:

- `garchSpec` - specifies the GARCH model. The function creates a GARCH specification object of class `"garchSpec"`.
- `garchSim` - simulates an artificial GARCH time series.
- `garchFit` - fits the parameters to the model. This function estimates the time series coefficients and the distributional parameters of a specified GARCH model.
- `print`, `plot`, `summary`, - are methods for an object returned by the function `"garchFit"`. These functions print and plot results, create a summary report and perform a diagnostic analysis.
- `predict` - forecasts one step ahead from an estimated model. This function can be used to predict future volatility from a GARCH model.

3.1 Specification Structure

The R-function `garchSpec` creates a S4 object called *specification structure* which specifies a GARCH process. This function maintains information that defines a model used for time series simulation, parameter estimation and diagnostic analysis.

Class Representation for the Specification Structure:

```
setClass("garchSpec",
  representation(
    call = "call",
    formula = "formula",
    model = "list",
    distribution = "list",
    presample = "matrix",
    title = "character",
    description = "character"
  )
)
```

The slots of the object include the string which matches the call of the function, a formula expression that describes the model, a list with the model parameters, a list with the distributional specification of the innovations, and a presample matrix to initiate the GARCH process. Additionally, a title and a short description can be added to the specification structure.

The meaning of the arguments of the function `garchSpec` together with the default values is summarized in the following list:

Function: GARCH Specification Function - `garchSpec`

```
garchSpec =
    function (model = list(omega = 1e-6, alpha = 0.1, beta = 0.8),
    distribution = c("norm", "snorm", "ged", "sged", "std", "sstd"),
    presample = NA, title = NULL, description = NULL)
```

- `model` - a list with the model parameters as entries
  - `ar` - a vector of autoregressive coefficients of length `m` for the ARMA specification,
  - `ma` - a vector of moving average coefficients of length n for the ARMA specification,
  - `omega` - the variance value for GARCH/APARCH specification,
  - `alpha` - a vector of autoregressive coefficients of length `p` for the GARCH/APARCH specification,
  - `gamma` - a vector of leverage coefficients of length `p` for the APARCH specification,
  - `beta` - a vector of moving average coefficients of length `q` for the GARCH/APARCH specification,
  - `mu` - the mean value for ARMA/GARCH specification,
  - `delta` - the exponent value used in the variance equation,
  - `parm` - a vector of two shape parameters measuring the degree of skewness and kurtosis.

  By default a GARCH(1,1) process with $\omega = 10^{-6}$, $\alpha = 0.1$, and $\beta = 0.8$ with normal innovations will be created.
- `distribution` - a string selecting the desired distributional form of the innovations either `norm` for the Normal, `ged` for the Generalized Error, `std` for the Standardized Student-t, or `snorm`, `sged`, `sstd`, for one of the skewed distributions. By default the normal distribution will be selected.
- `presample` - a numeric `"matrix"` with `3` columns and at least `max(m,n,p,q)` rows. The first column holds the *pre-innovations*, the second the *pre-sigmas*, and the third the *pre-series*. Otherwise, if presample takes the value `NA`, which is the default, or is specified as a positive integer, then the presample will be automatically created of minimal or of the specified length.
- `title` - a character variable associated with a title. By default NULL, which means that the title will be automatically set.
- `description` - a character string associated with a description of the GARCH specification. By default NULL, which means that a brief description will be automatically set.

The function `garchSpec` has a print method which produces for the GARCH(1,1) default specification the following report:

Exercise: Print Method for GARCH Specification - `xmpGarchSpecification`

```
> print(garchSpec())

Call: garchSpec()

       Title:  ARMA - GARCH/APARCH Specification
     Formula:  ~garch(1, 1)
       Model:
       omega:  1e-06
       alpha:  0.1
        beta:  0.8
       delta:  2
Distribution:  norm
   Parameter:  NA    1
   Presample:
                 time              z            h              y
               1    0  -0.1422962187  1.000000e-05  0.000000e+00
 Description:
               Specification as of Wed Jan 07 20:34:51 2004
```

In the following we describe in more detail the specification of the formula and model object, the distribution object and the presample object.

The Formula and Model Specification:

The `formula` object describes the GARCH model and is automatically created from the `model` list, e.g. ARMA(m,n) + GARCH(p,q). ARMA can be missing or specified as AR(m) or MA(n) in the case of pure autoregressive or moving average models. GARCH(p,q) may be alternatively specified as ARCH(p) or APARCH(p,q).

The Distribution Specification:

The `distribution` object selects one from six possible probability functions. The default choice for the distribution of the innovations z_t to simulate or to estimate the parameters of a GARCH process is the *Standardized Normal Probability Function*[1]

$$f^\star(z) \;=\; \frac{1}{\sqrt{2\pi}}\, e^{-\frac{z^2}{2}} \;. \tag{6}$$

The probability function or density is named standardized, marked by a star *, because $f^\star(z)$ has zero mean and unit variance. This can easily be verified computing the *moments*

$$\mu_r^\star \;=\; \int_{-\infty}^{\infty} z^r f^\star(z) dz \;. \tag{7}$$

Note, that $\mu_0^\star \equiv 1$ is the normalization condition, that $\mu_1^\star$ defines the mean $\mu \equiv 0$, and $\mu_2^\star$ the variance $\sigma^2 \equiv 1$.

An arbitrary Normal distribution located around a mean value μ and scaled by the standard deviation σ can be obtained by introducing a *location* and a *scale* parameter through the transformation

$$f(z)dz \;\rightarrow\; \frac{1}{\sigma} f^\star\Big(\frac{z-\mu}{\sigma}\Big)dz \;=\; \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(z-\mu)^2}{2\sigma^2}} dz \;. \tag{8}$$

The *central moments* μ_r of $f(z)$ can simply be expressed in terms of the moments $\mu_r^\star$ of the standardized distribution $f^\star(z)$. Odd central moments are zero and those of even order can be computed from

$$\mu_{2r} \;=\; \int_{-\infty}^{\infty} (z-\mu)^{2r} f(z) dz \;=\; \sigma^{2r}\mu_{2r}^\star \;=\; \sigma^{2r}\, \frac{2^r}{\sqrt{\pi}}\, \Gamma\Big(r+\frac{1}{2}\Big) \;, \tag{9}$$

yielding $\mu_2 = 0$, and $\mu_4 = 3$. The degree of asymmetry γ_1 of a probability function, named *skewness*, and the degree of peakedness γ_2, named *excess kurtosis*, can be measured by normalized forms of the third and fourth central moments.

$$\gamma_1 = \frac{\mu_3}{\mu_2^{3/2}} = 0 \;, \qquad \gamma_2 = \frac{\mu_4}{\mu_2^2} - 3 = 0 \;. \tag{10}$$

However, if we like to model an asymmetric and/or leptokurtic shape of the innovations we have to draw or to model z_t from a standardized probability function which depends on additional

[1] The Normal probability function is selected by setting `distribution="norm"`.

shape parameters which modify the skewness and kurtosis. However, it is important that the probability has still zero mean and unit variance. Otherwise, it would be impossible, or at least difficult, to separate the fluctuations in the mean and variance from the fluctuations in the shape of the density. In a first step we consider still symmetric probability functions but with an additional shape parameter which models the kurtosis. As examples we consider the generalized error distribution and the Student-t distribution with unit variance, both relevant in modelling GARCH processes.

Generalized Error Distribution:

Nelson [1991] suggested to consider the family of *Generalized Error Distributions*[2], GED, already used by Box and Tiao [1973][3] and Harvey [1981]. $f^\star(z|\nu)$ can be expressed as

$$\begin{aligned} f^\star(z|\nu) &= \frac{\nu}{\lambda_\nu 2^{1+1/\nu}\Gamma(1/\nu)}\, e^{-\frac{1}{2}|\frac{z}{\lambda_\nu}|^\nu}\ , \\ \lambda_\nu &= \left(\frac{2^{(-2/\nu)}\Gamma\left(\frac{1}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)}\right)^{1/2} \end{aligned} \tag{11}$$

with $0 < \nu \leq \infty$. Note, that the density is standardized and thus has zero mean and unit variance. Arbitrary location and scale parameters μ and σ can be introduced via the transformation $z \rightarrow \frac{z-\mu}{\sigma}$. Since the density is symmetric, odd central moments of the GED are zero and those of even order can be computed from

$$\mu_{2r} \;=\; \sigma^{2r}\mu^\star_{2r} \;=\; \sigma^{2r}\,\frac{(2^{1/\nu}\lambda_\nu)^{2r}}{\Gamma\left(\frac{1}{\nu}\right)}\Gamma\Big(\frac{2r+1}{\nu}\Big)\ . \tag{12}$$

Skewness γ_1 and kurtosis γ_2 are given by

$$\gamma_1 = \frac{\mu_3}{\mu_2^{3/2}} = 0\ , \qquad \gamma_2 = \frac{\mu_4}{\mu_2^2} - 3 = \frac{\Gamma\left(\frac{1}{\nu}\right)\Gamma\left(\frac{5}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)^2} - 3\ . \tag{13}$$

For $\nu = 1$ the GED reduces to the Laplace distribution, for $\nu = 2$ to the Normal distribution, and for $\nu \rightarrow \infty$ to the uniform distribution as a special case[4].

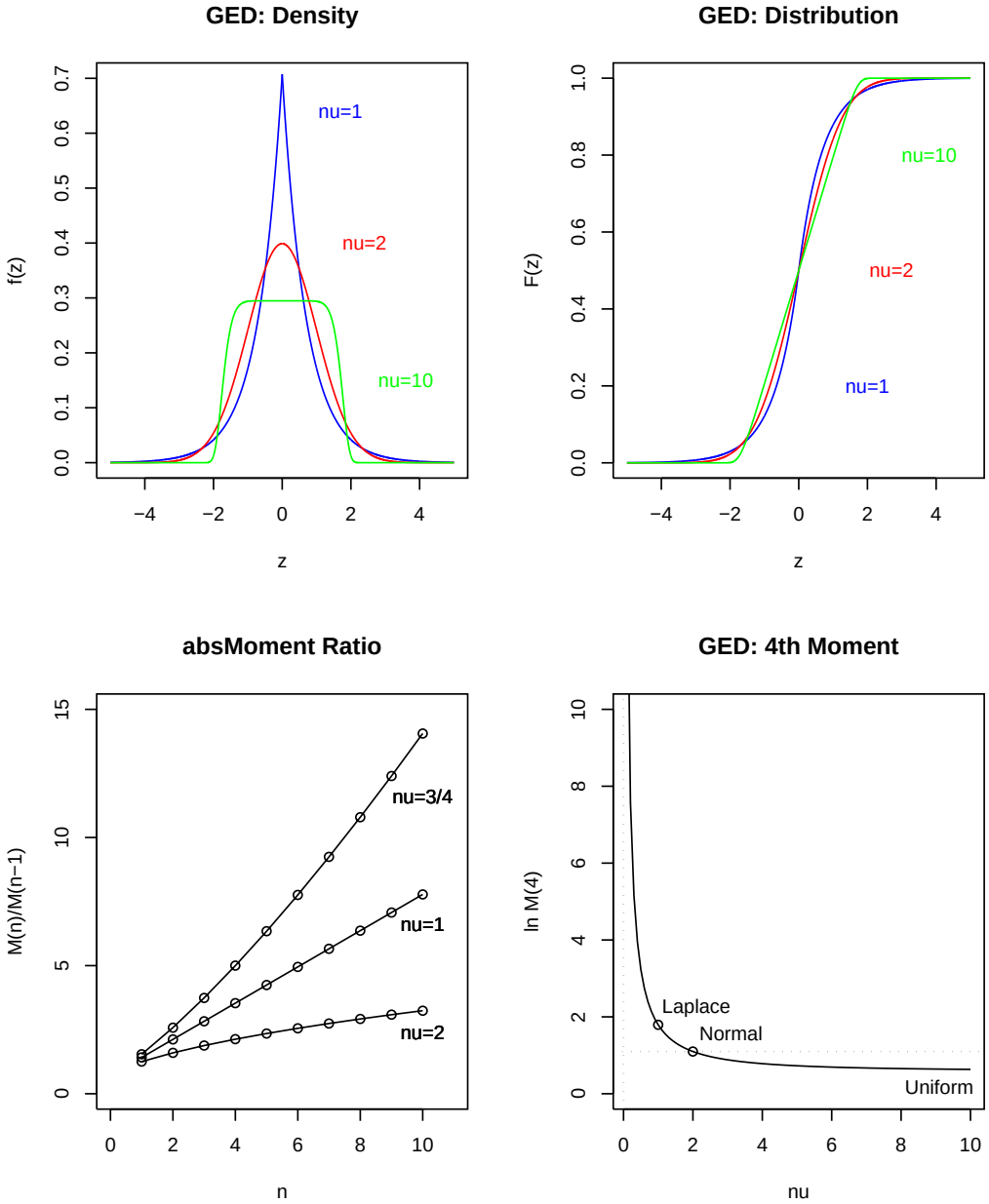
Function: GED Distribution - `[dpqr]ged`

> In this example we show how to write a function to compute density, distribution function, quantile function and how to generate random deviates for the generalized error distribution with mean equal to `mean`, standard deviation equal to `sd`, and shape parameter equal to `nu`.
> *Hint:* To Compute the distribution function $F(x) = \int_{-\infty}^{x} f(z)dz$ transform $\frac{1}{2}|\frac{z}{\lambda_\nu}|^\nu \rightarrow z$ and make use of the relationship to the Gamma distribution function which is part of R's base package.

[2]The Generalized error distribution is selected by setting `distribution="ged"`.

[3]Box and Tiao call the GED exponential power distribution.

[4]The Laplace Distribution takes the form $f(z) = \frac{1}{\sqrt{2}}e^{-\sqrt{2}|z|}$, and the uniform distribution has range $\pm 2\sqrt{3}$



■ Figure 1: The figure shows on top the density and distribution for the GED. Three cases are considered: The leptokurtic *Laplace Distribution* with $\nu = 1$, the *Normal Distribution* with $\nu = 2$, and the almost *Uniform Distribution* for large $\nu = 10$. On the lower left the ratio of two succeeding absolute moments is displayed. For the Laplace distribution we get a straight line, for distributions with thinner tails than the Laplace distribution, the curve bends upwards, and for distributions with thicker tails the curve bends downwards. On the lower right the 4th moment as a function of the shape parameter ν is shown. Since the distribution is standardized, the fourth moment coincides with the value of the kurtosis. Note, to obtain the excess kurtosis we have to subtract 3. For $\nu \to 0$ the kurtosis diverges and for $\nu \to \infty$ the kurtosis tends to one. The circles mark the values for the Laplace and Normal distributions. The code to produces these figures can be found in the example file `xmpGarchDISTged`.

```
# Density Function:
  dged = function(x, mean = 0, sd = 1, nu = 2) {
    # x - Vector of quantiles
    x = (x - mean ) / sd
    lambda = sqrt ( 2^(-2/nu) * gamma(1/nu) / gamma(3/nu) )
    g = nu / ( lambda * (2^(1+1/nu)) * gamma(1/nu) )
    Density = g * exp (-0.5*(abs(x/lambda))^nu) / sd
    return(Density) }

# Distribution Function:
  pged = function(q, mean = 0, sd = 1, nu = 2) {
    # q - Vector of quantiles
    q = (q - mean ) / sd
    lambda = sqrt ( 2^(-2/nu) * gamma(1/nu) / gamma(3/nu) )
    g = nu / ( lambda * (2^(1+1/nu)) * gamma(1/nu) )
    h = 2^(1/nu) * lambda * g * gamma(1/nu) / nu
    s = 0.5 * ( abs(q) / lambda )^nu
    Distribution = 0.5 + sign(q) * h * pgamma(s, 1/nu)
    return(Distribution) }

# Quantile Function:
  pged = function(p, mean = 0, sd = 1, nu = 2) {
    # p - Vector of probabilities
    lambda = sqrt ( 2^(-2/nu) * gamma(1/nu) / gamma(3/nu) )
    g  = nu / ( lambda * (2^(1+1/nu)) * gamma(1/nu) )
    h = 2^(1/nu) * lambda * g * gamma(1/nu) / nu
    s = 0.5 * ( abs(p) / lambda )^nu
    Quantiles = qgamma(s, 1/nu)
    return(Quantiles) }

# Random Deviates:
  rged = function(n, mean = 0, sd = 1, nu = 2) {
    # n - Number of random deviates to be generated
    r = rgamma(n, shape = 1/nu, scale = nu)
    Deviates = r^(1/nu) * sign(runif(n)-0.5)
    return(Deviates) }
```

Student-t Distribution:

Bollerslev [1987], Hsieh [1989)], Baillie and Bollerslev [1989], Bollerslev, Chou, and Kroner [1992], Palm [1996], Pagan [1996)], and Palm and Vlaar [1997] among others showed that the Student-t distribution better captures the observed kurtosis in empirical log-return time series. The density $f^\star(z|\nu)$ of the *Standardized Student-t Distribution*[5] can be expressed as[6]

$$f^\star(z|\nu) \;=\; \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi(\nu-2)}\Gamma(\frac{\nu}{2})}\;\frac{1}{\left(1+\frac{z^2}{\nu-2}\right)^{\frac{\nu+1}{2}}} \;=\; \frac{1}{\sqrt{\nu-2}\,\mathrm{B}\left(\frac{1}{2},\frac{\nu}{2}\right)}\frac{1}{\left(1+\frac{z^2}{\nu-2}\right)^{\frac{\nu+1}{2}}}\;, \qquad (14)$$

where $\nu > 2$ is the shape parameter and $\mathrm{B}(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ the Beta function. Again, arbitrary location and scale parameters μ and σ can be introduced via the transformation

[5]The standardized Student-t probability function is selected by setting `distribution="std"`.

[6]Note, when setting $\mu = 0$ and $\sigma^2 = \nu/(\nu-2)$ formula (**??**) results in the usual one-parameter expression for the Student-t distribution as implemented in the R function *dt*.

$z \rightarrow \frac{z-\mu}{\sigma}$. Odd central moments of the standardized Student-t distribution are zero and those of even order can be computed from

$$\mu_{2r} = \sigma^{2r}\mu^{\star}_{2r} \;=\; \sigma^{2r}\;(\nu-2)^{\frac{r}{2}}\;\frac{\mathrm{B}(\frac{r+1}{2},\frac{\nu-r}{2})}{\mathrm{B}(\frac{1}{2},\frac{\nu}{2})}\;. \tag{15}$$

Skewness γ_1 and kurtosis γ_2 are given by

$$\gamma_1 = \frac{\mu_3}{\mu_2^{3/2}} = 0\;, \qquad \gamma_2 = \frac{\mu_4}{\mu_2^2} - 3 =?\;. \tag{16}$$

Function: Student-t Distribution - `[dpqr]std`

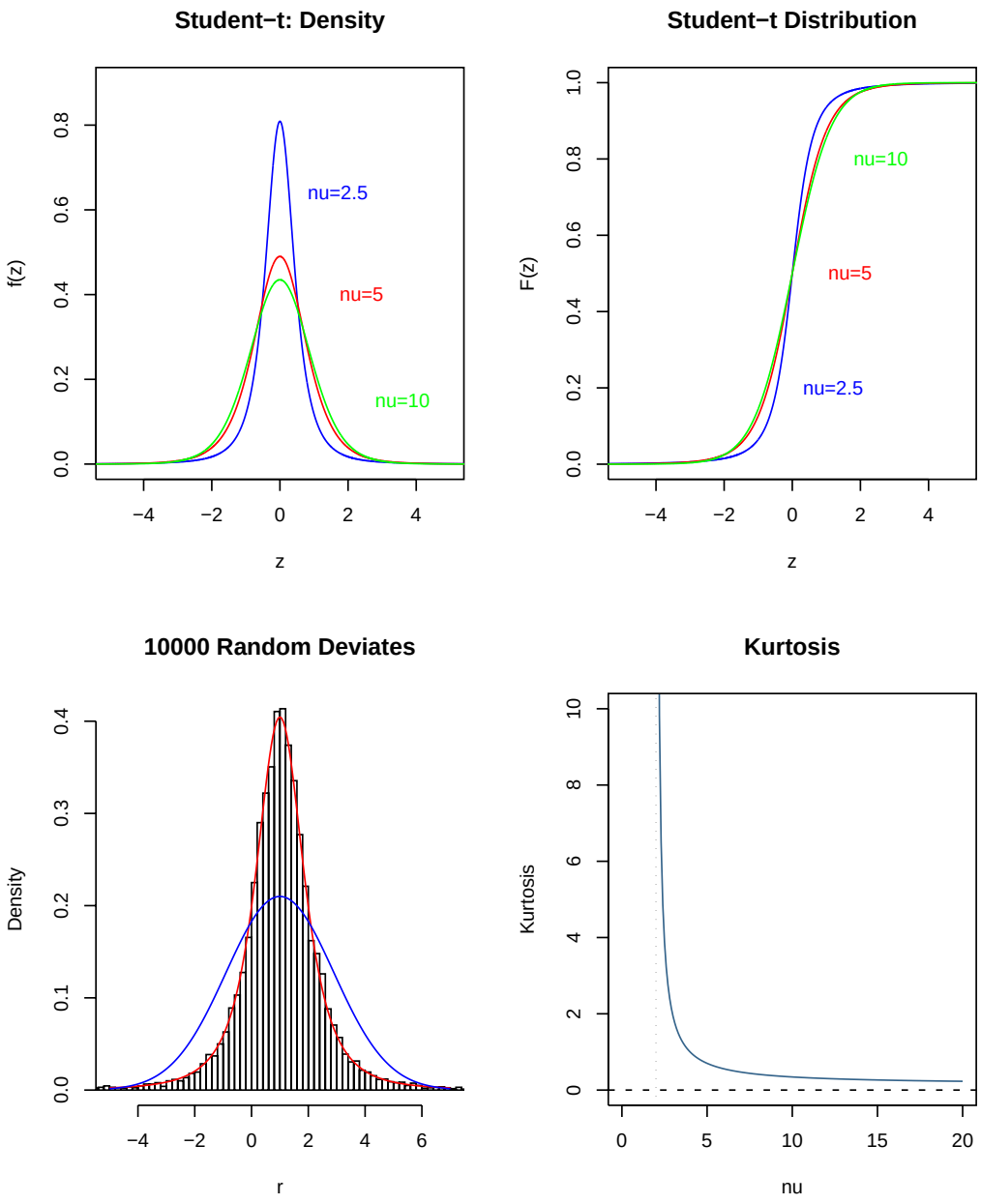
In this example we show how to write a function to compute density, distribution function, quantile function and how to generate random deviates for the Student-t distribution with mean equal to `mean`, standard deviation equal to `sd`, and shape parameter equal to `nu`.

Hint: To Compute the distribution function make use of the relationship to the (non-standardized) Student-t distribution function which is part of R's base package.

```
# Density Function:
  dstd = function(x, mean = 0, sd = 1, nu = 5) {
    # x - Vector of quantiles
    s = sqrt(nu/(nu-2))
    z = (x-mean) / sd
    Density = dt(x = z*s, df = nu) * sd / s)
    return(Density) }

# Distribution Function:
  pstd = function (q, mean = 0, sd = 1, nu = 5) {
    # q - Vector of quantiles
    s = sqrt(nu/(nu-2))
    z = (q-mean) / sd
    Distribution = pt(q = z*s, df = nu)
    return(Distribution) }

# Quantile Function:
  qstd = function (p, mean = 0, sd = 1, nu = 5) {
    # p - vector of probabilities
    s = sqrt(nu/(nu-2))
    Quantiles = qt(p = p, df = nu) * sd / s + mean
    return(Quantiles) }

# Random Deviates:
  rstd = function(n, mean = 0, sd = 1, nu = 5) {
    # n - Number of random deviates to be generated
    s = sqrt(nu/(nu-2))
    Deviates = rt(n = n, df = nu) * sd / s + mean
    return(Deviates) }
```

Standardized Skewed Distributions:

Fernandez and Steel [1998] proposed a quite general approach that allows the introduction of skewness in any continuous unimodal and symmetric distribution by changing the scale at each



■ Figure 2: The figure shows on top the density and distribution for the Student-t distribution with unit variance for three different degrees of freedom, $\nu = 2.5, 5, 10$. The graph for the largest value of ν is almost undistinguishable from a normal distribution. The lower left shows a histogram of 1000 random deviates distributed for $\nu = 2.5$. The thick line fits the theoretical density and the thin line corresponds to a normal distribution with the same variance. The last graph shows the kurtosis as function of the number of degrees of freedom. The kurtosis diverges for $\nu \to 2$ and its asymptotic values for $\nu \to \infty$ approaches zero, the value for the normal distribution.

side of the mode

$$f(z|\xi) \;=\; \frac{2}{\xi+\frac{1}{\xi}}\left[f(\xi z)H(-z) \;+\; f(\frac{z}{\xi})H(z)\right] \;, \tag{17}$$

where $0 < \xi < \infty$ is a shape parameter which describes the degree of asymmetry. $\xi = 1$ yields the symmetric distribution with $f(z|\xi = 1) = f(z)$. $H(z) = (1+\text{sign}(z))/2$ is the Heaviside unit step function. Mean μ_ξ and variance σ_ξ of $f(z|\xi)$ depend on ξ and are given by

$$\begin{aligned}
\mu_\xi &= M_1\big(\xi - \frac{1}{\xi}\big) \\
\sigma_\xi^2 &= (M_2 - M_1^2)\big(\xi^2 + \frac{1}{\xi^2}\big) + 2M_1^2 - M_2 \\
M_r &= 2\int_0^\infty x^r\; f(x)\; dx \;,
\end{aligned} \tag{18}$$

where M_r is the r-th absolute moment of $f(x)$ on the positive real line. Note, that $M_2 \equiv 1$ if $f(x)$ is a standardized distributions.

Function: Compute Absolute Moments - `xmpGarchDISTmoments`

In this example we show how to write a function which computes the absolute moments M_r for the standardized normal, GED and Student-t distributions or any other standardized distribution.

```
absMoments =
function(n, density, ...) {

  # "density" is a character string giving the name of a
  #    standardized distribution, e.g. dnorm, dged, dstd, ...
  # "..." are additional arguments passed to the
  #    standardized distribution, e.g. nu for dged and dstd.

  # norm - Normal Distribution:
  if (density == "dnorm") {
     return (sqrt(2)^n * gamma((n+1)/2) / sqrt(pi)) }

  # ged - Generalized Error Distribution:
  if (density == "ged") {
     parm = function(n, nu) {
       lambda = sqrt ( 2^(-2/nu) * gamma(1/nu) / gamma(3/nu) )
       return ((2^(1/nu)*lambda)^n * gamma((n+1)/nu) / gamma(1/nu)) }
     return(parm(n, ...)) }

  # std - Student-t Distribution:
  # Note: nu > 2*n
  if (density == "std") {
     parm = function(n, nu) {
       return (beta(1/2+2*n, nu/2-2*n)/beta(1/2, nu/2) * sqrt(nu-2)) }
     return(parm(n, ...)) }

  # Any other standardized Distribution will be numerically integrated ...
  fun = match.fun(density)
  moments = function(x, n, ...) { x^n * fun(x, ...) }
  M = absMoments.error <<-NULL
```

```
  for (i in n) {
      I = integrate(moments, 0, Inf, n=i, ...)
      M = c(M, 2*I$value)
      absMoments.error <<- c(absMoments.error, 2*I$abs.error) }
  return(M)
}

# Example - Try GED:
> absMoments(1:5, density = "dged", nu = 1)
[1]  0.7071068  1.0000000  2.1213203  6.0000000 21.2132034

# Example - Try Standardized Laplace Distribution:
# We expect the same results ...
> dlaplace = function(x) { exp(-sqrt(2)*abs(x)) / sqrt(2) }
> absMoments(1:5, density = "dlaplace")
[1]  0.7071068  1.0000000  2.1213203  6.0000000 21.2132034
> absMoments.error
[1] 1.7577e-07  2.728e-06  1.550e-05  3.867e-05  2.365e-06
```

Now we introduce a re-parametrization in such a way that the skewed distribution becomes standardized with zero mean and unit variance[7]. We call a skewed distribution function with zero mean and unit variance *Standardized Skewed Distribution* function.

The probability function $f^\star(z|\xi)$ of a standardized skewed distribution can be expressed in a compact form as

$$\begin{aligned} f(z|\xi\theta) &= \frac{2\sigma}{\xi + \frac{1}{\xi}}\, f^\star(z_{\mu_\xi\sigma_\mu\xi}|\theta) \\ z_{\mu_\xi\sigma_\mu\xi} &= \xi^{\text{sign}(\sigma_\xi\, z+\mu_\xi)}(\sigma_\xi z + \mu_\xi\ )\ , \end{aligned} \tag{19}$$

where $f^\star(x|\theta)$ may be any standardized symmetric unimodal distribution function, like the Standardized Normal distribution (6), the Standardized Generalized Error distribution (11) or the Standardized Student-t distribution (**??**). μ_ξ and σ_ξ can be calculated via equation (18).

Transforming $z \to \frac{z-\mu}{\sigma}$ yields skewed distributions, where the parameters have the following interpretation:

- μ is the mean or location parameter,
- σ is the standard deviation or the dispersion parameters,
- $0 < \xi < \infty$ a shape parameter that models the skewness[8].
- θ an optional set of shape parameters that model higher moments of even order like ν in the GED and Student-t distributions.

Exercise: Standardized Skewed Distributions

In this example we show how to write a function to compute density, distribution function, quantile function and how to generate random deviates for the standardized Student-t distribution with zero mean and unit variance, and shape parameter equal to `nu`.

[7]Lambert and Laurent (2001) have also reparametrized the density of Fernandez and Steel (1998) as a function of the conditional mean and of the conditional variance in such a way that again the innovation process has zero mean and unit variance.

[8]Setting $\xi = 1$ yields the symmetric distribution

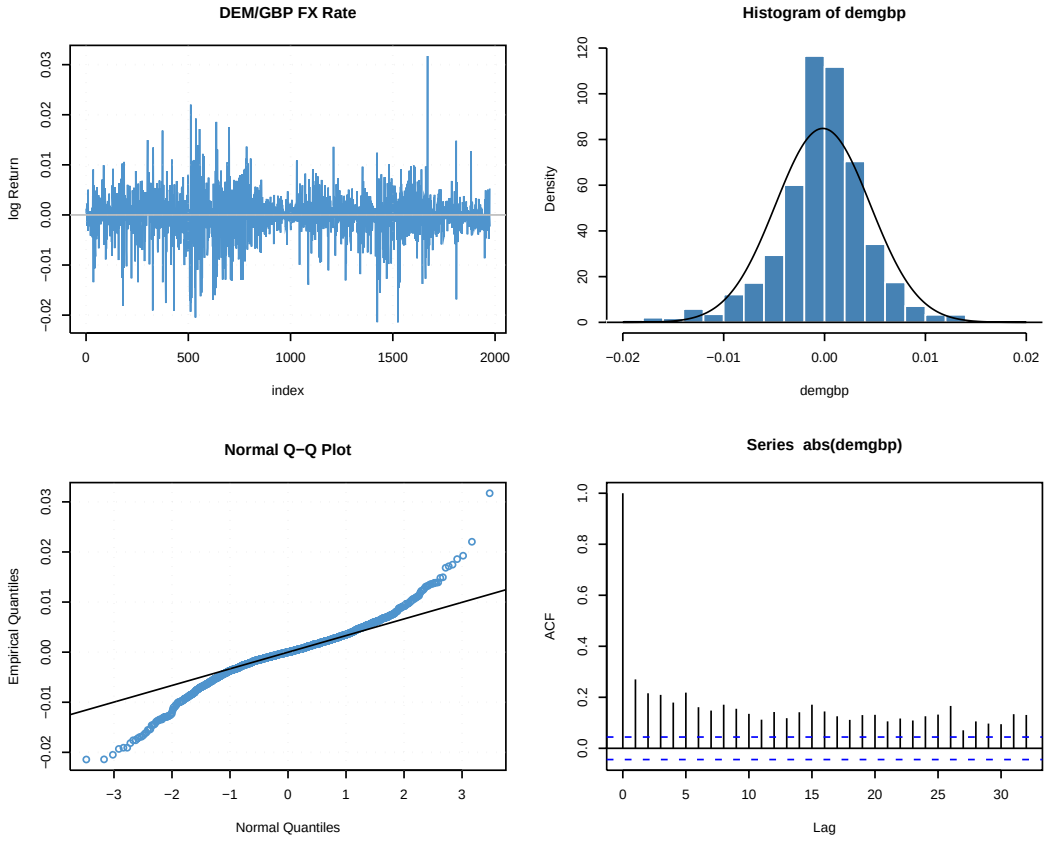
```
# Density Function:
  .dsstd = function(x, nu, xi) {
    # Standardize:
    m1 = 2 * sqrt(nu-2) / (nu-1) / beta(1/2, nu/2)
    mu = m1*(xi-1/xi)
    sigma =  sqrt((1-m1^2)*(xi^2+1/xi^2) + 2*m1^2 - 1)
    z = x*sigma + mu
    # Compute:
    Xi = xi^sign(z)
    g = 2 / (xi + 1/xi)
    Density = g * dstd(x = z/Xi, nu = nu) * sigma
    return(Density) }

# Distribution Function:
  .psstd = function(q, nu, xi) {
    # Standardize:
    m1 = 2 * sqrt(nu-2) / (nu-1) / beta(1/2, nu/2)
    mu = m1*(xi-1/xi)
    sigma =  sqrt((1-m1^2)*(xi^2+1/xi^2) + 2*m1^2 - 1)
    z = q*sigma + mu
    # Compute:
    Xi = xi^sign(z)
    g = 2 / (xi + 1/xi)
    Distribution = H(z) - sign(z) * g * Xi * pstd(q = -abs(z)/Xi, nu = nu)
    return(Distribution) }

# Quantile Function:
  .qsstd = function(p, nu, xi) {
    # Standardize:
    m1 = 2 * sqrt(nu-2) / (nu-1) / beta(1/2, nu/2)
    mu = m1*(xi-1/xi)
    sigma =  sqrt((1-m1^2)*(xi^2+1/xi^2) + 2*m1^2 - 1)
    # Compute:
    g = 2  / (xi + 1/xi)
    sig = sign(p-1/2)
    Xi = xi^sig
    p = (H(p-1/2)-sig*p) / (g*Xi)
    Quantiles = (-sig*qstd(p=p, sd=Xi, nu=nu) - mu ) / sigma
    return(Quantiles) }

# Random Deviates:
  .rsstd = function(n, nu, xi) {
    # Generate Random Deviates:
    weight = xi / (xi + 1/xi)
    z = runif(n, -weight, 1-weight)
    Xi = xi^sign(z)
    Random = -abs(rstd(n, nu=nu))/Xi * sign(z)
    # Scale:
    m1 = 2 * sqrt(nu-2) / (nu-1) / beta(1/2, nu/2)
    mu = m1*(xi-1/xi)
    sigma =  sqrt((1-m1^2)*(xi^2+1/xi^2) + 2*m1^2 - 1)
    Deviates = (Random - mu ) / sigma
    return(Deviates) }
```

For the skewed normal and skewed generalized error distribution we have only to change the function names and to exchange the expression for the first absolute moment `m1` with the appropriate value. To generate skewed distributions with arbitrary mean and standard deviation, we have only to add a shift and scale transformation, e.g.

```
# Density Function:
dsstd = function (x, mean = 0, sd = 1, nu = 5, xi = 1.5) {
  .dsstd(x = (x - mean)/sd, nu = nu, xi = xi)/sd }
```

For details of the implementation inspect the functions:

```
[dpqr]snorm(x, mean, sd, xi)
[dpqr]sged(x, mean, sd, nu, xi)
[dpqr]sstd(x, mean, sd, nu, xi)
```



■ Figure 3: The figure shows on top the density and distribution for the GED. Three cases are considered: The leptokurtic Laplace distribution with $\nu = 1$, the Normal distribution with $\nu = 2$, and the almost uniform distribution for large $\nu = 10$. On the lower left the ratio of two succeeding absolute moments is displayed. For the Laplace distribution we get a straight line, for distributions with thinner tails than the Laplace distribution, the curve bends upwards, and for distributions with thicker tails the curve bends downwards. On the lower right the 4th moment as a function of the shape parameter ν is shown. Since the distribution is standardized, the fourth moment coincides with the value of the kurtosis. Note, to obtain the excess kurtosis we have to subtract 3. For $\nu \to 0$ the kurtosis diverges and for $\nu \to \infty$ the kurtosis tends to one. The circles mark the values for the Laplace and Normal distributions.

Distributional Parameter Fits:

If the distribution of z_i's is $F(z|\theta)$ and θ is an unknown vector of distributional parameters, then we say that the distribution is parametric. In such a setting the *method of maximum likelihood* is the appropriate technique for the estimation and inference on θ. For a continuous distribution $F(z|\theta)$ the density will be written as $f(z|\theta)$ and the joint density of a random sample $\{z_i\}$ is

$\Pi_{i=1}^{N} f(z_i|\theta)$. The likelihood of the sample is this joint density evaluated at the observed sample values, viewed as a function of θ. The log-likelihood function $\mathcal{L}_N(\theta)$ is its natural logarithm

$$\mathcal{L}_N(\theta) \;=\; \sum_{i=1}^{N} \ln f(z_i|\theta) \;. \tag{20}$$

The *maximum likelihood estimator* or *MLE*, named $\widehat{\theta}$, are the values of the parameter set which maximizes the likelihood or equivalently, which maximizes the log-likelihood

$$\widehat{\theta} \;=\; \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_N(\theta) \;. \tag{21}$$

Function: MLE Density Parameter Estimation - `dFit`

In this example we show how to write a function which returns a S4 object of "nlmFit" with the estimated parameters of an arbitrary probability function via the maximum log-likelihood approach.

```
setClass("nlmFit", representation(
  call = "call", estimate = "numeric", fit = "list",
  title = "character", description = "character") )

dFit =
function(x, density, parm)
{
  # "x" is a numeric vector of random deviates
  # "density" is a character string giving the name of the
  #    probability function whose parameters should be estimated
  # "parm" is a vector of valued parameters passed to the
  #    probability function with an initial guess for the optimizer.

  # Internal MLE Function:
  d = match.fun(density)
  if (length(parm) == 1)
      d.mle = function(x, y = x) { -sum(log(d(x=y, x[1]))) }
  if (length(parm) == 2)
      d.mle = function(x, y = x) { -sum(log(d(x=y, x[1], x[2]))) }
  if (length(parm) == 3)
      d.mle = function(x, y = x) { -sum(log(d(x=y, x[1], x[2], x[3]))) }
  if (length(parm) == 4)
      d.mle = function(x, y = x) { -sum(log(d(x=y, x[1], x[2], x[3], x[4]))) }

  # Estimate:
  fit = nlm(f = dsnorm.mle, p = parm, hessian = TRUE, print.level = 0, y = x)

  # Add names Attribute:
  names(fit$estimate) = names(fit$gradient) = c("mean", "  sd", "  xi")

  # Return Value:
  return(new("nlmFit",
      call = as.call(match.call()), estimate = fit$estimate, fit = fit,
      title = as.character(paste(density, "Fit")),
      description = as.character("MLE Parameter Estimation")))
}
```

Note, that the function `dFit` implemented in the `finGARCH` package also optionally allows to trace the iteration path of the optimization process and to plot the results. For the print method we refer to the source code.

In the following example we show how to fit a simulated GED sample:

```
z = rsged(n = 1000, mean = 1, sd = 0.5, nu = 2.5, xi = 1.25)
dFit(z, density = "dsged", parm = c(mean(z), sqrt(var(z), 2, 1))
```

The Presample Specification:

Because the mean equation and the variance equation is recursive in nature, they require *pre-sample data* to initiate the simulation.

For automatically generated presample data the presample is created as a three column matrix of length `max(m,n,p,q)`. The three columns have the time ordering $t_{1-L}, \ldots, t_{1-\ell}, \ldots, t_0$, where $1 \le \ell \le L$ counts the rows, and have the following meaning:

The Default Presample Matrix

- *Column 1:* holds the pre-innovations z_t. The pre-innovations are random deviates generated from the distribution as specified in the specification structure.
- *Column 2:* holds the pre-sigmas σ_t. All pre-sigmas are set to the expectation value of the conditional variance.
- *Column 3:* holds the pre-series x_t. All pre-series values are set to the expectation value of the mean.

To minimize transient effects, we recommend to specify the `presample` long enough. For a given value of $L > max(m, n, p, q)$ the iteration of the ARMA/GARCH process starts at time $t_{-L+\max(m,n,p,q)}$ and the pre-sigmas and pre-series values ar overwritten by the iterated time series values. Alternatively, the `presample` can be fully specified by the user.

Example: GARCH Presample Generation

The following lines of code show how the presample is generated

...

The next example shows how to specify a presample for a typical model from the ARMA-GARCH family.

Example: GARCH Specification - `xmpGarchSpecification`:

Use the function `garchSpec()` to specify models from the ARMA-GARCH/APARCH family:

```
# 1. Specify an AR(1)-GARCH(1,1) with default presample
  seed(4711)
  model = list(ar=1, omega=1e-6, alpha=0.1, beta=0.3)
  spec = garchSpec(model=model)
  print(spec)

# 2. AR(1)-GARCH(1,1) with default presample of length 10:
  seed(4711)
  model = list(ar=1, omega=1e-6, alpha=0.1, beta=0.3)
  spec = garchSpec(model=model, presample=10)
  print(spec)

# 3. AR(1)-GARCH(1,1) with user defined presample:
```

```
seed(4711)
model = list(ar=1, omega=1e-6, alpha=0.1, beta=0.3)
presamle.innovations = dnorm(10)
sigma = sqrt(model$omega/(1-model$alpha-model$beta))
presample.sigmas = rep(sigma, times=10)
presample.series = rep(0, times=10)
spec = garchSpec(model=model, presample=presample)
print(spec)
```

The third example generates the following presample,

Presample for an AR(1)-GARCH(1,1) Model

...

it is the same as the default presample from the second example.

3.2 Simulation of Time Series

The R-function `garchSim(n, spec)` creates with the information provided by the *specification structure* an artificial ARMA-GARCH time series. The function requires only two arguments, the length of the time series to be simulated and the specification structure. The function `garchSim` returns the sample path for the simulated return series, for the innovations, and for the conditional standard deviations.

Function: GARCH Time Series Simulation - `garchSim`

This function simulates an artificial ARMA time series with GARCH/APARCH errors.

```
garchSim = function(n, spec)
```

- `n` - the length of the time series to be simulated,
- `spec` - the model specification, a S4 object of class `"garchSpec"` as returned from the the function `garchSpec`. Only the `garchSpec@model` slot is required to simulate the artificial GARCH process, the remaining arguments are optional and have default values.

The following example shows how to use the function `garchSim` and how to simulate some typical GARCH Models.

Example: GARCH Simulation - `xmpGarchSimulation`

Use the function `garchSpec` to specify models from the ARMA-GARCH family:

```
# Specify ARCH(2):
  model <- list(omega=1e-6, alpha=c(0.1, 0.3), h0=1e-7)
  spec = garchSpec(model)
  x <- garchSim(model)
  plot(x, type="l", main="ARCH(2) Model")

# Specify GARCH(1,1):
  model <- list(omega=1e-6, alpha=0.1, beta=0.8, h0=1e-7)
  x <- garchSim(model)
  plot(x, type="l", main="GARCH(1,1) Model")
```

Default estimation of a GARCH(1,1) model on the Bollerslev and Ghysels data

Package	Parameters and *t*-ratios							
	μ	$t(\mu)$	α_0	$t(\alpha_0)$	α_1	$t(\alpha_1)$	β_1	$t(\beta_1)$
E-VIEWS	−0.00540	−0.64	0.0096	8.01	0.143	11.09	0.821	53.83
GAUSS-FANPAC	−0.00600	−0.75	0.0110	3.67	0.153	5.67	0.806	23.71
LIMDEP	−0.00619	−0.71	0.0108	3.44	0.153	5.61	0.806	26.73
MATLAB	−0.00619	−0.73	0.0108	8.13	0.153	10.96	0.806	48.67
MICROFIT	−0.00621	−0.73	0.0108	3.78	0.153	5.78	0.806	24.02
SAS	−0.00619	−0.74	0.0108	8.15	0.153	10.97	0.806	48.60
SHAZAM	−0.00613	−0.73	0.0107	5.58	0.154	7.91	0.806	36.96
RATS	−0.00625	−0.71	0.0108	3.76	0.153	5.79	0.806	23.93
TSP	−0.00619	−0.67	0.0108	1.66	0.153	2.86	0.806	11.11

■ The figure shows ... .

3.3 Parameter Estimation

Given a model for the conditional mean and variance and an observed univariate return series, we use the maximum likelihood, estimation approach to fit the parameters for the specified model of the return series. The procedure infers the process innovations or residuals by inverse filtering. Note, that this filtering transforms the observed process ε_t into an uncorrelated white noise process z_t. The log-likelihood function then uses the inferred innovations z_t to infer the corresponding conditional variances ${\sigma_t}^t$ via recursive substitution into the model-dependent conditional variance equations. Finally, the procedure uses the inferred innovations and conditional variances to evaluate the appropriate log-likelihood objective function. The procedure is implemented in the function `garchFit()`.

Maximum Likelihood Estimator:

As already mentioned, the GARCH models are estimated using a maximum likelihood estimation, MLE, approach. The MLE concept interprets the density as a function of the parameter set, conditional on a set of sample outcomes. The Normal distribution is the standard distribution when estimating and forecasting GARCH models. Using $\varepsilon = z_t\sigma_t$ the log-likelihood function of the normal distribution is given by

$$\mathcal{L}_N(\theta) = \sum_{t=1}^{T}[log(d(\varepsilon\sigma_t^{-1} - log\sigma_t] \tag{22}$$

$$= -\frac{1}{2}\sum_{t=1}^{T}[log(2\pi) + log(\sigma_t^2) + z_t^2] \tag{23}$$

where T is the number of observations.

Parameter Constraints:

If we use a solver for unconstrained numerical optimization to maximize the log-likelihood function with respect to the vector of parameters θ, the inspected range of the parameter space is unlimited. However, the problem is that $\omega > 0$ and δ have to be constrained in a semifinite interval, that α and *beta* have to be constrained in a finite interval $[0, 1)$, and γ in $(-1, 1)$. To impose constraints on a finite interval one can easy perform the transformation

$$\theta^\star \leftarrow u + \frac{v - u}{1 + e^{-\theta}} \tag{24}$$

which relates monotonously the finite interval $[u, v]$ to the infinite one. So, applying unconstrained optimization of the log-likelihood function with respect to θ is equivalent to applying constrained optimization with respect to $\theta^\star$.

Optimization Algorithm:

The function `garchFit` uses the R solver `nlm`. This optimizer uses a Newton-type algorithm, Dennis and Schnabel [1983], Schnabel, Koontz and Weiss [1985]. If the function to be optimized has an attribute called `gradient` or both `gradient` and `hessian` attributes[9], these will be used in the calculation of updated parameter values, otherwise, numerical derivatives are used. The optimizer returns the following components:

Function: Return Values of the Optimizer - `nlm`

- `minimum` - the value of the estimated minimum.
- `estimate` - the point at which the minimum value is obtained.
- `gradient` - the gradient at the estimated minimum.
- `hessian` - the hessian at the estimated minimum.
- `code` - the termination status of the optimization process
  - `1` - the relative gradient is close to zero, current iterate is probably the solution.
  - `2` - successive iterates within tolerance, current iterate is probably solution.
  - `3` - last global step failed to locate a point lower than the estimate "estimate". Either "estimate" is an approximate local minimum of the function or the step tolerance "steptol' is too small.
  - `4` - the iteration limit was exceeded.
  - `5` - the maximum step size "stepmax" exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or "stepmax" is too small.
- `iterations` - the number of iterations performed.

The result of the optimization is added to the specification structure and globally accessible.

Initial Parameter Estimates:

The optimizer requires a vector of initial parameter estimates for the ARMA coefficients a_i and b_j, for the GARCH/APARCH coefficients ω, α_i, γ_i, and β_j, and additionally for the mean μ and exponent δ. If the parameters for the distribution function should also be estimated, these parameters have also to be initialized. By default all parameters are automatically initialized by the function `garchFit`. On the other hand all parameters can be initialized by the user, thus the control of the initialization remains with the user.

[9] Analytical expressions for the gradient and hessian are not yet implemented.

Example: GARCH Parameter Estimation - `xmpGarchSpecification`

Use the function `garchSpec()` to specify models from the ARMA-GARCH family:

```
# Specify ARCH(2):
  model <- list(omega=1e-6, alpha=c(0.1, 0.3), h0=1e-7)
  x <- garchSim(model)
  plot(x, type="l", main="ARCH(2) Model")

# Specify GARCH(1,1):
  model <- list(omega=1e-6, alpha=0.1, beta=0.8, h0=1e-7)
  x <- garchSim(model)
  plot(x, type="l", main="GARCH(1,1) Model")
```

Putting All Together:

3.4 Diagnostic Analysis

The R-functions `print`, `plot`, `summary` and `print.summary` are methods to print and plot the results of the parameter estimation and to create a report summarizing the results of a diagnostic analysis.

Print and Plot the Results from the Fit:

The `print` function takes as input a S4 object of class `"garchFit"` which is the result of a GARCH maximum log-likelihood estimation. The report lists following aspects of the fit:

...

In the same way the `plot` function takes also as input a S4 object of class `"garchFit"` and displays the following graphs:

...

Estimation Statistics:

Q(20), $Q^2(20)$, P(50) AIC, Log-Lik

Forecast Performance:

- Mean Squared Errors (MSE)
- Median Squared Error (MedSE)
- Mean Absolute Error (MAE)
- Adjusted Mean Absolute Percentage Error (AMAPE)
- Theil Inequality Coefficient (TIC)
- Mincer-Zarnowitz R2 (R2)

The first three criteria are well-known and self-explanatory. The AMAPE is

the TIC is

and the R2 statistic from this regression therefore provides the proportion of variances explained by the forecast (i.e., the higher the R2 , the better the forecasts).

Diagnostic Analysis:

The postprocessing investigates the residuals z_t, and the conditional standard deviations σ_t.

If you plot the ACF of the squared standardized innovations, they also show no correlation: acf(innovations/sigmas)2). Now compare the ACF of the squared standardized innovations in this figure to the ACF of the squared returns prior to fitting the default model. The comparison shows that the default model sufficiently explains the heteroscedasticity in the raw returns.

3. Quantify and Compare Correlation of the Standardized Innovations. Compare the results below of the Q-test and the ARCH test with the results of these same tests in the preestimation analysis. In the preestimation analysis, both the Q-test and the ARCH test indicate rejection (H = 1 with pValue = 0) of their respective null hypotheses, showing significant evidence in support of GARCH effects. In the postestimate analysis, using standardized innovations based on the estimated model, these same tests indicate acceptance (H = 0 with highly significant pValues) of their respective null hypotheses and confirm the explanatory power of the default model.

Example: GARCH Specification - `xmpGarchSpecification`:

Use the function `garchSpec()` to specify models from the ARMA-GARCH family:

```
# Specify ARCH(2):
  model = list(omega=1e-6, alpha=c(0.1, 0.3), h0=1e-7)
  spec = garchSpec()
  x <- garchSim(model)
  plot(x, type="l", main="ARCH(2) Model")

# Specify GARCH(1,1):
  model = list(omega=1e-6, alpha=0.1, beta=0.8, h0=1e-7)
  spec = garchSpec()
  x <- garchSim(n = 1000, spec)
  plot(x, type="l", main="GARCH(1,1) Model")

# Specify AR(1)-APARCH(1,1):
  model = list(omega=1e-6, alpha=0.1, beta=0.8, h0=1e-7)
  spec = garchSpec()
  x <- garchSim(n = 1000, spec)
  plot(x, type="l", main="AR(1)-APARCH(1,1) Model")
```

3.5 Forecasting

One of the major aspects in the investigation of heteroskedastic time series is to produce forecasts. Function for forecasts of both the conditional mean and the conditional variance are available as well as for several forecast error measures.

Forecasting the Conditional Mean:

The optimal h-step-ahead predictor of x_{t+h} given the information up to time t is

$$\widehat{\sigma}^2_{t+h|t} \; = \tag{25}$$

Forecasting the Conditional Variance:

The conditional variance can be forecasted independently from the conditional mean.

For a GARCH(p,q) process, the optimal h-step-ahead forecast of the conditional variance $\widehat{\omega}^2_{t+h|t}$ is computed recursively from

$$\widehat{\sigma}^2_{t+h|t} \;=\; \widehat{\omega} \;+\; \sum_{i=1}^{q} \widehat{\alpha}_i \varepsilon^2_{t+h-i|t} \;+\; \sum_{j=1}^{p} \widehat{\beta}_j \sigma^2_{t+h-j|t} \;, \tag{26}$$

where $\varepsilon^2_{t+i|t} = \sigma^2_{t+i|t}$ for $i > 0$ while $\varepsilon^2_{t+i|t} = \varepsilon^2_{t+i}$ and $\sigma^2_{t+i|t} = \sigma^2_{t+i}$ for $i \leq 0$.

For a APARCH(p,q) process the distributional of the innovations may have an effect on the forecast, the optimal h-step-ahead forecast of the conditional variance $\widehat{\omega}^2_{t+h|t}$ is computed recursively from

$$\begin{aligned} \widehat{\sigma}^{\delta}_{t+h|t} &= E(\sigma^{\delta}_{t+h}|\Omega_t) && (27) \\ &= \widehat{\omega} + \sum_{i=1}^{q} \widehat{\alpha}_i E[(|\varepsilon_{t+h-i}| - \widehat{\gamma}_i \varepsilon_{t+h-i})^{\widehat{\delta}}|\Omega_t] + \sum_{j=1}^{p} \widehat{\beta}_j \sigma^{\widehat{\delta}}_{t+h-j|t} && (28) \end{aligned}$$

where ...

Example: GARCH Forecast - `xmpGarchSpecification`:

Use the function `garchSpec()` to specify models from the ARMA-GARCH family:

```
# Specify ARCH(2):
  model <- list(omega=1e-6, alpha=c(0.1, 0.3), h0=1e-7)
  x <- garchSim(model)
  plot(x, type="l", main="ARCH(2) Model")

# Specify GARCH(1,1):
  model <- list(omega=1e-6, alpha=0.1, beta=0.8, h0=1e-7)
  x <- garchSim(model)
  plot(x, type="l", main="GARCH(1,1) Model")
```

Forecast Error Measures:

4 Numerical Investigations

When analyzing and forecasting GARCH models we have to implement the algorithms. Through the complexity of the GARCH models it is evident that different software implementations have different functionalities, drawbacks and features and may lead to differences in the numerical results. McCullough and Renfro [1999], Brooks, Burke and Persand [2001], and Laurent and Peters [2003] compared the results of several software packages. These investigations demonstrate that there could be major differences between estimated GARCH parameters from different software packages. Therefore we use the benchmark suggested by Fiorentini, Calzolari, and Panattoni [1996] to test our *R*-package.

4.1 GARCH(1,1) Benchmark

We use as *the benchmark* the GARCH(1,1) model estimated with the software[10] as implemented by Fiorentini, Calzolari, and Panattoni [1996] using analytical derivatives. For the time series data we take the DEM/GBP daily exchange rates[11] as published by Bollerslev and Ghysels [1996]. The series contains a total of 1975 daily observations sampled during the period from January 2, 1984, to December 31, 1991. This benchmark was also used by McCullough and Renfro [1999], and by Brooks, Burke, and Persand [2001] in their analysis of several GARCH software packages.

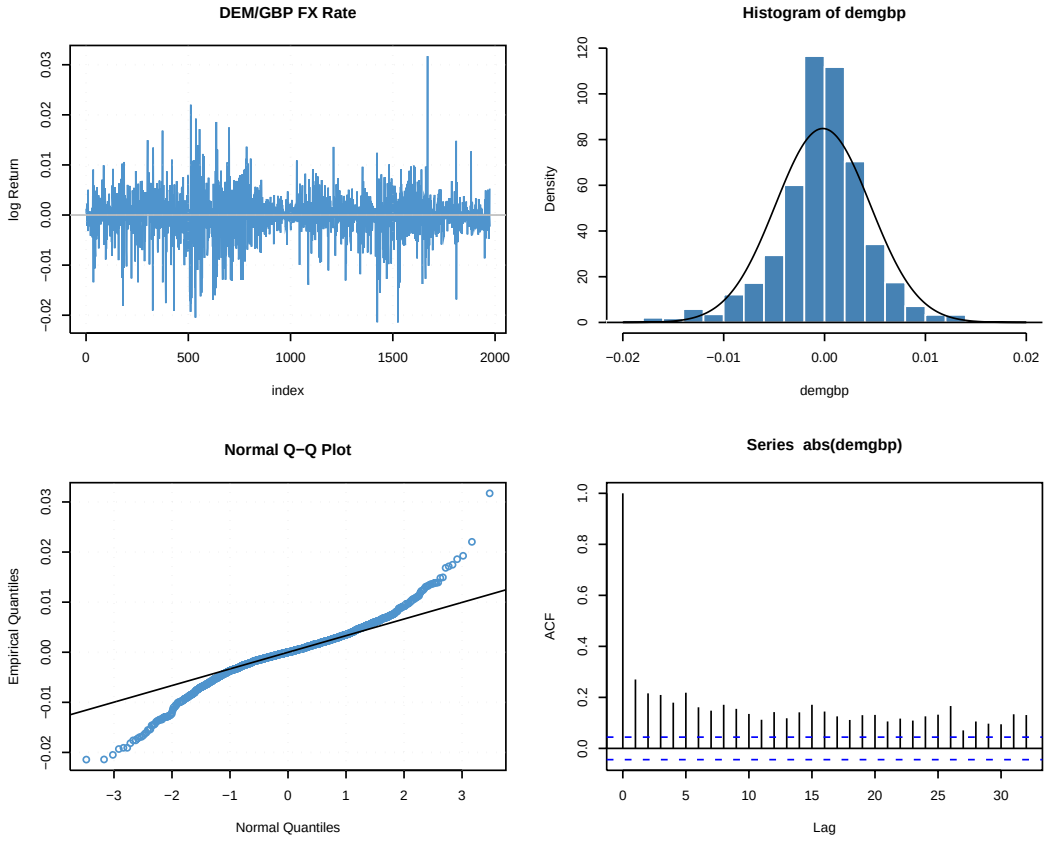
4.2 Modelling DEM/GBP Rates

Figure 4.1 shows some stylized facts of the log-returns of the daily DEM/GBP FX rates. The distribution is leptokurtic and skewed to negative values. The log-returns have a mean value of $\mu = -0.00016$ of almost zero, a skewness of $\varsigma = -0.25$, and an excess kurtosis of $\kappa = 3.62$. The histogram of the density and the quantile-quantile plot graphically display this behavior. Furthermore, the autocorrelation function of the absolute values of the returns, which measure volatility, decay slowly.

Example: GARCH(1,1) - Stylized Facts `xmpGarchDEMGBP`:

```
# Load Data Set:
  data(dem2gbp)
  demgbp = dem2gbp[, 1]/100
# Plot Log-Return Series:
  ts.plot(demgbp, xlab = "index", ylab = "log Return", col = "steelblue3")
  title(main = "DEM/GBP FX Rate")
# Histogram Plot:
  hist(demgbp, n= 20, col = "steelblue", probability = TRUE,
    border = "white", xlim = c(-0.02, 0.02) )
  x = seq(-0.02, 0.02, length = 201)
  lines(x, dnorm(x = x, mean = mean(demgbp), sd = sqrt(var(demgbp))))
```

[10] $http://qed.econ.queensu.ca/jae/1996-v11.4/fiorentini-calzolari-panattoni/$ SOFTWARE
[11] $ftp://www.amstat.orgJBES_View/96-2-APR/bollerslev_ghysels/bollerslev.sec41.dat.$ DATA



■ The figure shows the returns, their distribution, the quantile-quantile plot, and the autocorrelation function of the volatility.

```
# Quantile - Quantile Plot:
  qqnorm(demgbp, xlab = "Normal Quantiles", ylab = "Empirical Quantiles",
    col = "steelblue3")
  qqline(demgbp)
# Partial Autocorrelation:
  acf(abs(demgbp))
```

We use the default GARCH(1,1) model to fit the parameters of the time series and compare later the results with the benchmark.

Example: GARCH(1,1) Benchmark - `xxx`:

```
# Get the Data:
  data(demgpb)
# Specify the Model:
  spec = garchSpec()
# Estimate the Parameters:
  fit = garchFit(x, spec)
  print(fit)
# Diagnostic Analysis:
  summary(fit)
# Predict one Step Ahead:
  predict(fit)
```

The result are the following:

4.3 Benchmark Comparison

Brooks, Burke, and Persand [2001] used the GARCH benchmark to compare the accuracy of GARCH(1,1) model estimations among several econometric software packages. The results of the default estimation of the GARCH(1,1) model in each package is shown in figure XX.

Default estimation of a GARCH(1,1) model on the Bollerslev and Ghysels data

Package	Parameters and t-ratios							
	μ	$t(\mu)$	α_0	$t(\alpha_0)$	α_1	$t(\alpha_1)$	β_1	$t(\beta_1)$
E-VIEWS	−0.00540	−0.64	0.0096	8.01	0.143	11.09	0.821	53.83
GAUSS-FANPAC	−0.00600	−0.75	0.0110	3.67	0.153	5.67	0.806	23.71
LIMDEP	−0.00619	−0.71	0.0108	3.44	0.153	5.61	0.806	26.73
MATLAB	−0.00619	−0.73	0.0108	8.13	0.153	10.96	0.806	48.67
MICROFIT	−0.00621	−0.73	0.0108	3.78	0.153	5.78	0.806	24.02
SAS	−0.00619	−0.74	0.0108	8.15	0.153	10.97	0.806	48.60
SHAZAM	−0.00613	−0.73	0.0107	5.58	0.154	7.91	0.806	36.96
RATS	−0.00625	−0.71	0.0108	3.76	0.153	5.79	0.806	23.93
TSP	−0.00619	−0.67	0.0108	1.66	0.153	2.86	0.806	11.11

■ The figure shows ... .

We have added to the table the results as obtained from our `finGARCH` package, from the SPlus software package and from the GARCH Ox software package. In addition we have reinvestigated the FX rates with the GARCH toolbox from MATLAB.

Also listed is the benchmark result obtained by Fiorentini, Calzolari, and Panattoni [1996].

Accuracy of estimates[a]

Package	Parameter	Coefficient	Standard error				
			Hessian	OPG	QMLE	IM	BW
LIMDEP	μ	6.0	–	–	–	–	6.0
	α_0	6.0	–	–	–	–	6.0
	α_1	6.0	–	–	–	–	6.0
	β_1	6.0	–	–	–	–	6.0
MATLAB	μ	4.6	–	4.7	–	–	–
	α_0	6.0	–	5.1	–	–	–
	α_1	4.9	–	5.1	–	–	–
	β_1	5.6	–	5.2	–	–	–
MICROFIT	μ	2.5	2.9	–	–	–	–
	α_0	4.2	3.5	–	–	–	–
	α_1	2.7	2.7	–	–	–	–
	β_1	3.8	4.0	–	–	–	–
RATS	μ	1.9	1.4	3.4	1.1	–	–
	α_0	4.1	2.5	2.3	2.8	–	–
	α_1	4.4	2.8	2.3	2.5	–	–
	β_1	3.8	2.4	2.6	2.4	–	–
SAS	μ	2.6	3.1	5.0	2.8	–	–
	α_0	4.4	4.5	4.7	4.8	–	–
	α_1	4.6	4.9	4.6	5.0	–	–
	β_1	5.2	4.8	4.9	5.1	–	–
SHAZAM	μ	3.2	–	3.1	–	4.3	5.0
	α_0	3.4	–	3.0	–	3.4	3.6
	α_1	4.1	–	3.5	–	3.9	4.2
	β_1	4.5	–	3.3	–	3.7	4.0
TSP	μ	6.0	6.0	6.0	6.0	–	–
	α_0	6.0	6.0	6.0	6.0	–	–
	α_1	6.0	6.0	6.0	6.0	–	–
	β_1	6.0	6.0	6.0	6.0	–	–

[a] Cell entries are errors measured relative to the FCP benchmark values from FCP using the same method for estimating the first and second derivatives. 6.0 is the maximum possible score for any given package in this exercise since only six digits of accuracy are given in the FCP paper.

■ The figure shows ... .

.

5 Summary and Next Steps

This is the first approach to an *educational* software implementation of the GARCH modelling process. Much value was put to show how to use the S language and how to implement functions to the R environment.

The aim was to implement the whole software in R without using any partial implementations in Fortran, C, or C++. Although this would speed up the execution time for the parameter estimation essentially, we did not make use of this opportunity. This keeps the software more transparent for everybody who is interested in the code and applied algorithms and would learn programming techniques under R.

The next steps we will follow are,

- to implement analytical derivatives for the gradient vector and hessian matrix,
- to add long memory behavior in the mean equation, and
- to add further volatility models for the variance equation.

Analytical derivatives for GARCH model were derived by, for the APARCH mode by , and for the FIGARCH model by . The long memory behavior in the mean equation will be modelled by ... The volatility models we like to include are ...

Everybody who likes to contribute to this package will be welcome.

6 GARCH Function Summary

The GARCH functions are part of Rmetrics `fSeries` package All functions are entirely written in S. So it becomes very easy to inspect the code which is especially useful for teaching financial engineers and for teaching computational finance.

The `fSeries` package can be downloaded from the Rmetrics website *www.rmetrics.org*.

6.1 List of R Functions

The following summary gives an overview over the GARCH functions available in the `fSeries` package. The functions can be grouped according to their purpose in *Distribution Functions*, *GARCH Modelling Functions*, and *Utility Functions*.

Distribution Functions

The collection of distribution functions adds the Generalized Error Distribution and a re-parameterized Student-t Distribution to the library. Both distributions have parameterizations where the location parameter has the meaning of the mean, and the squared scale parameter has the meaning of the variance. Furthermore, a function is added which allows to create skewed distributions from any standardized probability function. In addition the collection includes a program which computes absolute moments for standardized distributions, and a function which fits the parameters of a density vie the max log likelihood estimation approach.

```
[dpqr]ged               Generalized Error Distribution
[dpqr]std               Modified Student-t Duistribution
[dpqr]skew              Skewed Distributions from a Standardized PDF
absMoments              Compute absolute Moments
nlmFit                  MLE Fit of distributional parameters
```

ARMA-GARCH Modelling Functions

The collection of ARMA-GARCH modelling functions includes functions to simulate artificial GARCH/APARCH processes, to estimate the parameters for a specified model from empirical time series data, to perform a diagnostic analysis of the fit and to forecast one step ahead in time.

```
garchSpec               Specify an ARMA-GARCH/APARCH model
garchFit                Fit the GARCH parameters from empirical data
garchDiag               Perform a diagnostic analysis
garchPredict            Predict one step ahead
```

Utility Functions

The collection of utility functions includes a function to compute the Heaviside unit step function and some related functions.

```
H                       Heaviside unit step function
Sign                    Just another signum function
boxcar                  The boxcar function
ramp                    The ramp function
```

6.2 List of Data Sets

The `fSeries` package includes for GARCH modelling three data sets used in examples and for benchmarks.

DEM/GBP Exchange Rate Data: A price and return series which' contains daily observations of the German Mark versus British Pound foreign exchange rates. The sample period is from January 2, 1984, to December 31,1991, for a total of 1975 daily observations of. The data can be downloaded from the JBES FTP site[12].

NASDAQ Composite Index Data: The series contains daily closing values of the NASDAQ Composite Index. The sample period is from January 2, 1990, to December 31, 2001, for a total of 3028 daily equity index observations. The data can be downloaded from the Market Data section of the NASDAQ Web site[13].

NYSE Composite Index Data: The series contains daily closing values of the New York Stock Exchange Composite Index. The sample period is from January 2, 1990, to December 31, 2001, for a total of 3028 daily equity index observations of the NYSE Composite Index. The data can be downloaded from the NYSE web site[14].

CAC40 French Index: The series contains daily values for the CAC40 French Index ranging from January, 1995 up to December, 1999, in total 1249 observations. The data were used in examples of the GARCH Ox package, Laurent and Peters [2002], and are part of their software distribution[15].

```
demgbp.csv              DEM-GBP Foreign exchange rates
cac40.csv               CAC-40 log-returns and volume data
nasdaq                  NASDAQ Index
```

6.3 List of Examples

The `fSeries` package includes several example and demonstration files for GARCH modelling, which show how to use the functions:

```
xmpDist                 Distribution Examples
xmpGarch                Garch modelling examples
```

6.4 Software Licence

[12] $ftp://www.amstat.orgJBES_View/96-2-APR/bollerslev_ghysels/bollerslev.sec41.dat.$
[13] $http://www.marketdata.nasdaq.com/mr4b.html$
[14] $http://www.nyse.com/marketinfo/marketinfo.html$
[15] $http://www.core.ucl.ac.be/\ laurent/G@RCH/site/download.htm$

References

[1] ... not yet complete

[2] Bera, A.K., Higgins H.L., (1993); *A Survey of ARCH Models: Properties, Estimation and Testing*, Journal of Economic Surveys 7, 305–366.

[3] Bollerslev, T., (1986); *Generalised Autoregressive Conditional Heteroskedasticity*, Journal of Econometrics 31, 307–327.

[4] Bollerslev T., Chou R.Y., Kroner K.F., (1992); *ARCH Modelling in Finance: A Review of the Theory and Empirical Evidence*; Journal of Econometrics 52(5), 5–59.

[5] Bollerslev T., Engle R.F., NelsonD.B., (1994); *ARCH Model*, Chapter 49 of Handbook of Econometrics Volume IV, edited by Engle and McFadden), Elsevier Science, pp2961–3031.

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

[7] Box G.E.P., Tiao G.C., (1973); *Bayesian Inference in Statistical Analysis*, Addison-Wesley, Publishing Reading, MA.

[8] Brooks R.D., Faff R.W., McKenzie M.D., Mitchell H., (2000); *A Multi-Country Study of Power ARCH Models and National Stock Market Return*, Journal of International Money and Finance 19, 377–397. PDF

[9] Brooks C., Burke S.P., Persand G., (2001); *Benchmarks and the Accuracy of GARCH Model Estimation*, International Journal of Forecasting 17, 45-56. LNK

[10] Dennis J.E., Schnabel R.B., (1983); *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.

[11] Ding, Z., Granger C.W.J., Engle R.F. (1993); *A Long Memory Property of Stock Market Returns and a New Model*, Journal of Empirical Finance 1, 83–106.

[12] Engle, R.f. (1982); *Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation*, Econometrica 50, 987–1007.

[13] Engle, R. F., Bollerslev T. (1986); *Modelling the Persistence of Conditional Variances* Econometric Reviews 5, 1–50.

[14] Engle, R.F. (2001); *GARCH 101: An Introduction to the Use of ARCH/GARCH models in Applied Econometrics*, forthcoming Journal of Economic Perspectives. PDF

[15] Engle, R. F., (2002); *New Frontiers for ARCH Models* NYU Preprint, 29 pp. PDF

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

[17] Geweke, J. (1986); *Modeling the Persistence of Conditional Variances: A Comment*, Econometric Review 5, 57–61.

[18] Glosten, L., Jagannathan R., Runkle D. (1993); *On the Relation Between Expected Value and the Volatility of the Nominal Excess Return on Stocks*, Journal of Finance 48, 1779–1801.

[19] Hansen, B. (1994); *Autoregressive Conditional Density Estimation*, International Economic Review 35, 705–730.

[20] Harvey, A.C. (1981); *The Econometric Analysis of Time Series*, Oxford.

[21] Higgins, M.L., Bera A.K. (1992); *A Class of Nonlinear Arch Models International*, Economic Review 33, 137–158.

[22] Lambert, P., Laurent S. (2000); *Modelling Skewness Dynamics in Series of Financial Data*, Discussion Paper, Institut de Statistique, Louvain-la-Neuve. PDF

[23] Lambert, P., Laurent S. (2001); *Modelling Financial Time Series Using GARCH-Type Models and a Skewed Student Density*, Mimeo, Universite de Liege.

[24] Laurent S., Lambert, P. (2002); *A Tutorial for GARCH 2.3, a Complete Ox Package for Estimating and Forecasting ARCH Models*, GARCH 2.3 Tutorial, 71 pp. PDF

[25] Laurent S. (2003); *Analytical derivates of the APARCH model*, Priprint, University of Namur, 8 pp. PDF

[26] Ling, S., McAleer M. (2002); *Stationarity and the Existence of Moments of a Family of GARCH processes*, Journal of Econometrics 106, 109–117.

[27] Ling, S., McAleer M. (2002); *Necessary and Sufficient Moment Conditions for the GARCH(r,s) and Asymmetric Power GARCH(r,s) Models*, Econometric Theory 18, 722–729.

[28] Lombardi, M., Gallo G. (2001); *Analytic Hessian Matrices and the Computation of FI-GARCH Estimates*, Manuscript, Universita degli studi di Firenze. PDF

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

[30] Nelson, D.B., (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

[31] Pentula, S. (1986); *Modeling the Persistence of Conditional Variances: A Comment*, Econometric Review 5, 71–74.

[32] Peters J.P. (2001); *Estimating and Forecasting Volatility of Stock Indices Using Asymmetric GARCH Models and (Skewed) Student-t Densities*, Preprint, University of Liege, Belgium, 20 pp. PDF

[33] Poon, S.H. Granger C.W.J. (2001); *Forecasting Financial Market Volatility: A Review*, Manuscript, Department of Economics, UCSD. PDF

[34] Rabemananjara R., Zakoian J. M. (1993); *Threshold Arch Models and Asymmetries in Volatility*, Journal of Applied Econometrics 8, 31–49.

[35] Schnabel R.B., Koontz J.E., Weiss B.E., (1985); *A Modular System of Algorithms for Unconstrained Minimization*, ACM Trans. Mathematical Software 11, 419–440.

[36] Schwert, W. (1990); *Stock Volatility and the Crash of 87*, Review of Financial Studies 3, 77–102.

[37] Taylor, S. (1986); *Modelling Financial Time Series*, Wiley, New York.

[38] Zakoian, J.-M. (1994); *Threshold Heteroskedasticity Models*, Journal of Economic Dynamics and Control 15, 931–955.