universität
innsbruck

Time Series Analysis

Models of Heteroskedasticity

Models of Heteroskedasticity

Contents

Contents

Models of Heteroskedasticity

Motivation and Infrastructure

Motivation

So far: Employ correlations for modeling/predicting conditional mean $\mathrm{E}(Y_t | Y_{t-1}, Y_{t-2}, \dots)$.

Now: Modeling/predicting conditional variance $\mathrm{Var}(Y_t | Y_{t-1}, Y_{t-2}, \dots)$.

But: ARIMA models cannot accommodate this. E.g., conditional one-step-ahead variance is always constant and equal to the innovation variance.

Application: In finance, the conditional variance of a return of a financial asset is a commonly used risk measure. Employed in pricing of assets and value-at-risk (VaR) computations.

Motivation

Approach:

- In an efficient market, the expected return (conditional mean) should be zero.
- Thus, the return series should be white noise, i.e., uncorrelated.
- Develop models for conditional heteroskedasticity in serially uncorrelated time series (e.g., ARCH, GARCH, . . .).
- Combine ARIMA and (G)ARCH models to model conditional mean and variance structure simultaneously.

Required infrastructure: Computational tools for financial time series data, i.e., in particular including daily and intra-day data.

Infrastructure: Times and dates

Classes: Implementations of time/date objects that capture all necessary information of underlying conceptual entities.

- "`numeric`"/"`integer`" (*base*): Annual/quarterly/monthly.
- "`yearqtr`"/"`yearmon`" (*zoo*): Quarterly/monthly.
- "`Date`" (*base*): Daily.
- "`chron`" (*chron*): Intra-day. No time zones, daylight savings time. Slightly non-standard interface.
- "`POSIXct`" (*base*): Intra-day. With time zones, daylight savings time. Computations in GMT are straightforward. Other time zones might require some more attention.
- "`timeDate`" (*timeDate*): Intra-day. With time zones, daylight savings time via concept of financial centers.

Infrastructure: Times and dates

Functions/methods: Computations and extraction of relevant information.

- Set up time from numeric or character input: Class constructors.
- Extract underlying numeric scale: `as.numeric()` method.
- Produce character label: `format()` and `as.character()` Method.
- Use for plotting: Input for `plot()`, e.g., `Axis()` method.
- Sequences of times with given spacing: `seq()` method.
- Time differences: Group generic functions or `difftime()`.
- Move forward/backward on time scale: Group generic functions.
- Comparison (less/greater): Group generic functions.

Infrastructure: Times and dates

Remarks:

- Use time/date class that is appropriate for your data (and not more complex). See Grothendieck and Petzoldt (2004), "R Help Desk: Date and Time Classes in R." *R News*, **4**(1), 29–32.
- Idea for many (but not all) time/date classes: Numeric vector (e.g., corresponding to years, days, seconds since origin/epoch) plus class attribute. Method dispatch handles matching/rounding correctly and provides coercion to other classes.
- Time/date objects are usually not interesting as standalone objects but are used to annotate other data.
- The most important application of this are *time series* where there is for each time point a vector of (typically numeric) observations.

Infrastructure: Times series

Types of time series:

- *Irregular* (unequally spaced).
- Strictly *regular* (equally spaced).
- With underlying regularity, i.e., created from a regular series by omitting some observations.

For strictly regular series: The whole time index can be reconstructed from start, end and time difference between two observations. The reciprocal value of the time difference is also called *frequency*.

For irregular series: All time indexes need to be stored in a vector of length n.

Infrastructure: Times series

Classes: Virtually all implementations are focused on numeric data and fix some particular class for the time index.

- "`ts`" (*base*): Regular "`numeric`" time index (e.g., annual, quarterly, monthly),
- "`its`" (*its*): Irregular time index of class "`POSIXct`",
- "`irts`" (*tseries*): Irregular time index of class "`POSIXct`",
- "`timeSeries`" (*timeSeries*): Irregular time index of class "`timeDate`",
- "`zoo`" (*zoo*): Regular or irregular time index of arbitrary class.
- "`xts`" (*xts*): Built on top of "`zoo`", with specialized infrastructure for time indexes of class "`Date`", "`POSIXct`", "`chron`", "`timeDate`", "`yearmon`", "`yearqtr`", ...

Infrastructure: Times series

Functions/methods:

- Visualization: `plot()`.
- Extraction of observations or associated times: `time()` (and `coredata()`).
- Lags and differences: `lag()` (note sign!) and `diff()`.
- Subsets in a certain time window: `window()`.
- Union and intersection of several time series: `merge()`.
- Aggregation along a coarser time grid: `aggregate()`.
- Rolling computations such as means or standard deviations: (`rollapply()`).

Infrastructure: Times series

Advantages of "`zoo`":

- Can be used with arbitrary time indexes (i.e., you could also provide your own specialized class).
- Standard interface: Employs R's standard methods and introduces only few new generics.
- Talks to all other classes: Coercion functions are available, e.g., `as.ts()`, `as.timeSeries()`, etc. that work if the time index is of the required class. The reverse `as.zoo()` always works.

Recommendations:

- Use "`zoo`" for storage and basic computations.
- If necessary, coerce to other classes for analysis etc.
- "`xts`" is helpful extension of "`zoo`" for date/time indexes.

Infrastructure: Illustration

Illustration: College Retirement Equities Fund (CREF). Fund of several thousand stocks, not openly traded in the stock market. Time series for trading days from August 26, 2004, to August 15, 2006.

In R: Provided in *TSA* as "`ts`" series without proper date information.

```
R> data("CREF", package = "TSA")
R> tsp(CREF)
```

```
[1]   1 501   1
```

```
R> CREF[1:5]
```

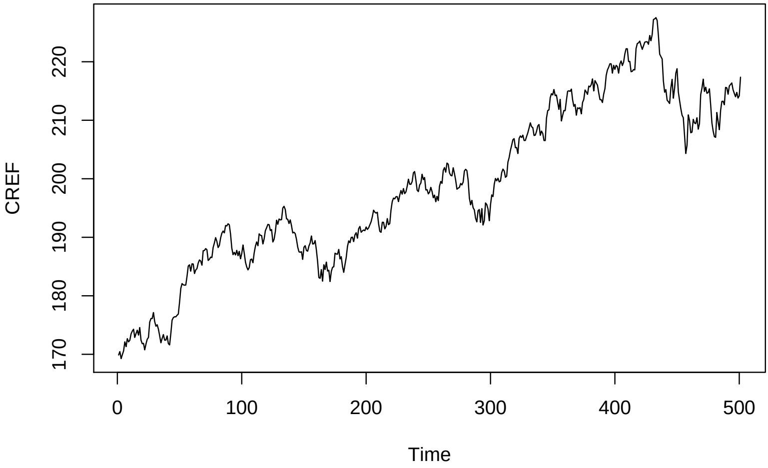
```
[1] 169.9 170.5 169.3 169.9 170.6
```

Alternatively: "`zoo`" series with "`Date`" index.

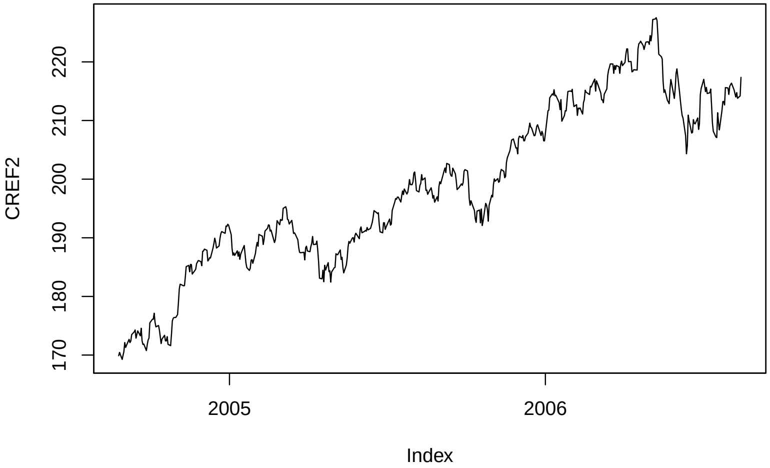
```
R> load("Data/CREF2.rda")
R> CREF2[1:5]
```

```
2004-08-26 2004-08-27 2004-08-30 2004-08-31 2004-09-01
     169.9      170.5      169.3      169.9      170.6
```

Infrastructure: Illustration



Infrastructure: Illustration



Infrastructure: Illustration

Note: The simpler "`ts`" version can easily be produced by stripping the index from the data and adding the trivial index $1, \dots, n$.

```
R> identical(CREF, ts(coredata(CREF2)))
[1] TRUE
```

Infrastructure: Illustration

Time coercions: "`Date`" objects are stored as days since 1970-01-01. Enables certain transformations. (Timezones are system-dependent).

```
R> start(CREF2)
[1] "2004-08-26"
R> as.numeric(start(CREF2))
[1] 12656
R> as.POSIXct(start(CREF2))
[1] "2004-08-26 02:00:00 CEST"
R> as.POSIXlt(start(CREF2))
[1] "2004-08-26 UTC"
R> as.POSIXlt(start(CREF2))$wday
[1] 4
```

Infrastructure: Illustration

Series coercions: "zoo" objects can be coerced to "ts" if an underlying regularity can be inferred.

Here: The underlying days since the origin are used and hence non-trading days are includes as NAs.

```
R> CREF2a <- as.ts(CREF2)
R> tsp(CREF2a)

[1] 12656 13375     1

R> CREF2a[1:20]

 [1] 169.9 170.5    NA    NA 169.3 169.9 170.6 172.1 171.3    NA
[11]    NA    NA 172.7 172.1 172.4 173.5    NA    NA 174.0 174.3

R> as.Date(12656)

[1] "2004-08-26"

R> as.Date(13375)

[1] "2006-08-15"
```

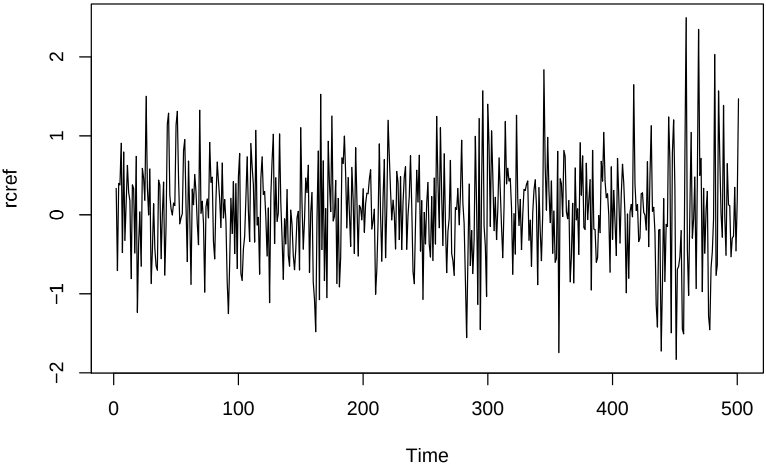
Infrastructure: Illustration

However: Time series from trading days are typically treated as if they were regularly spaced.

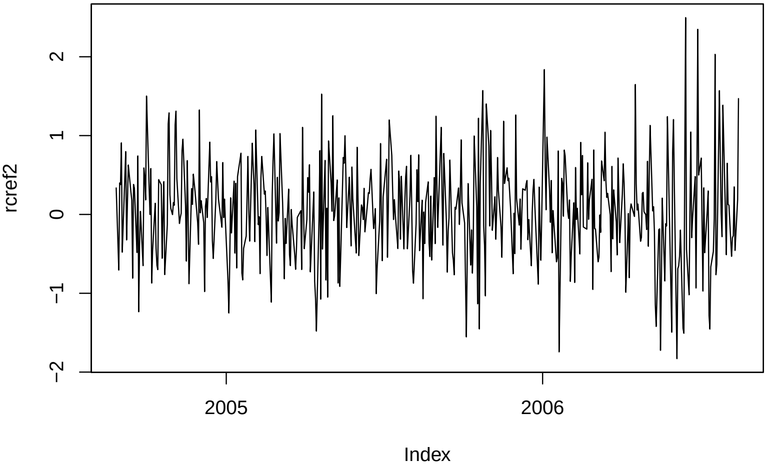
Hence: In computations, e.g., for returns, either use the regularly spaced "ts" series with auxiliary index or the irregularly spaced "zoo" series with proper index.

```
R> rcref  <- 100 * diff(log(CREF))
R> tsp(rcref)

[1]   2 501   1

R> rcref[1:5]

[1]  0.3361 -0.7056  0.3987  0.3785  0.9080

R> rcref2 <- 100 * diff(log(CREF2))
R> rcref2[1:5]

2004-08-27 2004-08-30 2004-08-31 2004-09-01 2004-09-02
    0.3361    -0.7056     0.3987     0.3785     0.9080
```

Infrastructure: Illustration



Infrastructure: Illustration



Models of Heteroskedasticity

Some Common Features of Financial Time Series

Some common features

Goal: Model (continuously compounded) return series $\{r_t\}$ computed from the prices $\{p_t\}$ of a financial asset.

$$r_t = \log(p_t) - \log(p_{t-1})$$

Sometimes transformed to percentage points by multiplication of 100. Often easier to interpret and numerically more stable.

Stylized facts: For return series from financial assets.

- *Volatility clustering:* Alternating extended periods of low and high volatility, respectively.
- *Heavy tails:* Extreme returns occur more often than would be expected under normality.

Some common features

Jargon: Volatility typically refers to the conditional standard deviation of a financial instrument, which varies with time.

Empirical evidence:

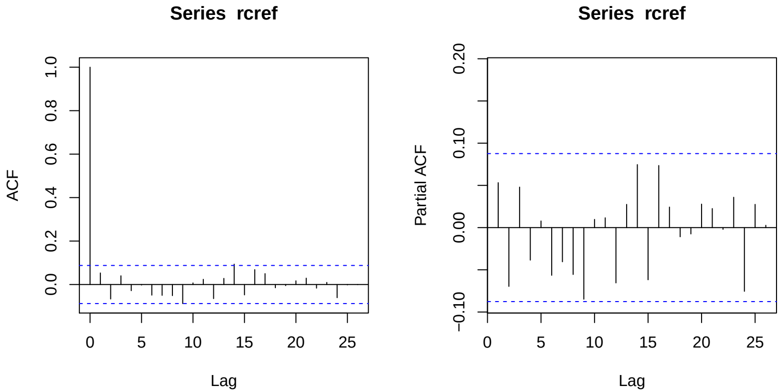
- Return series does not have significant (partial) autocorrelations.
- Absolute or squared returns are positively autocorrelated.

Thus: Distinguish independence and lack of correlation.

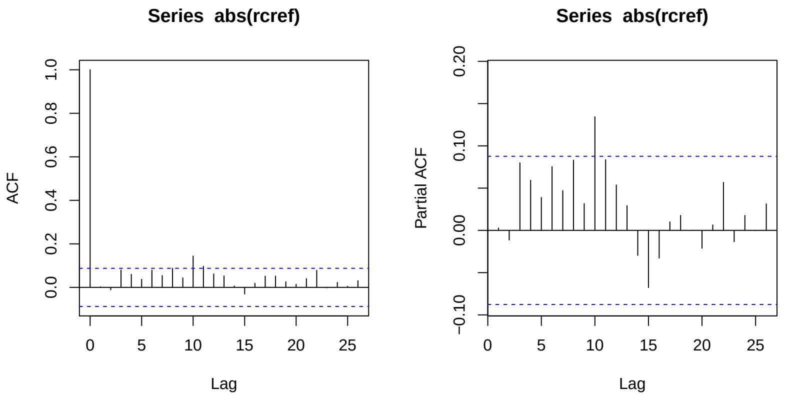
- If Y_t and Y_{t-k} are independent, $g(Y_t)$ and $g(Y_{t-k})$ are, too. Both for linear and nonlinear transformations $g(\cdot)$.
- Correlation is only a measure of linear dependence. Not necessarily pertained in nonlinear transformations.

Illustration: ACF and PACF for CREF returns and transformations. The overall mean is 0.0493 with standard error 0.0288.

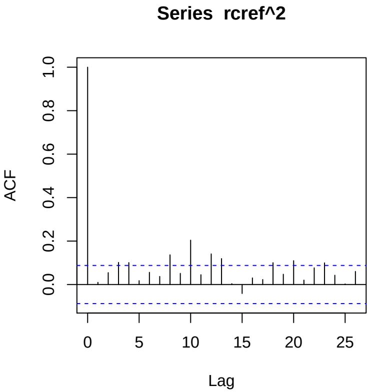
Some common features

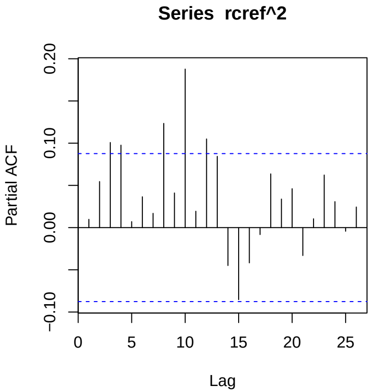


Some common features



Some common features





Some common features

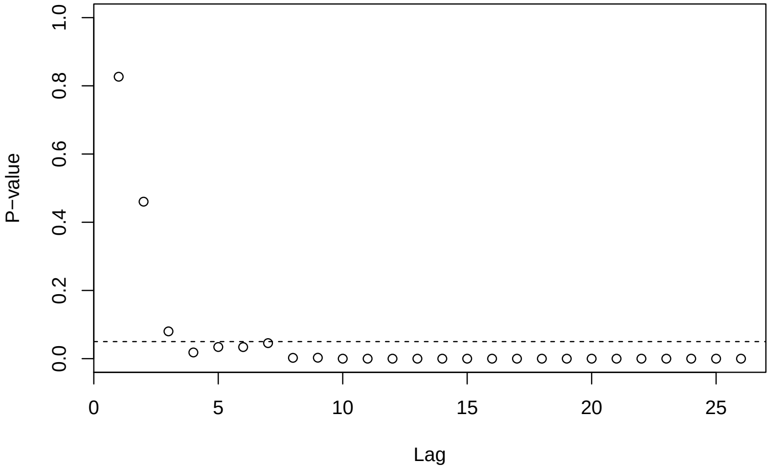
Test: Formally test for autocorrelation in volatility.

- Employ Box-Ljung test for squared returns (if the levels are not serially correlated).
- More generally, employ squared residuals from an ARIMA model.
- Degrees of freedom do not need to be adjusted (i.e., `fitdf = 0`) because no correlations in the squared values are modeled.
- Test is also referred to as McLeod-Li test.

In R: Test can be easily performed graphically for varying lags.

```
R> plot(1:26, sapply(1:26, function(i)
+    Box.test(rcref^2, lag = i, type = "Ljung-Box")$p.value),
+    xlab = "Lag", ylab = "P-value", ylim = c(0, 1))
R> abline(h = 0.05, lty = 2)
```

Some common features



Some common features

Furthermore: To assess nonnormality, consider sample skewness and (excess) kurtosis.

Skewness: For a random variable Y with mean μ and standard deviation σ, the skewness is $\mathrm{E}\{(Y-\mu)^3\}/\sigma^3$. It can be estimated by

$$
\begin{aligned}
g_1 &= \frac{1}{n\,\widehat{\sigma}^3}\sum_{i=1}^{n}(Y_i-\bar{Y})^3 \\
\widehat{\sigma}^2 &= \frac{1}{n}\sum_{i=1}^{n}(Y_i-\bar{Y})^2
\end{aligned}
$$

Sample is left/right-skewed for values less/greater than zero.

In R: `skewness()` from *e1071*. Various definitions of sample skewness exist. `skewness()` by default (`type = 3`) computes version with finite sample correction $g_1\sqrt{n(n-1)}/(n-2)$.

Some common features

Kurtosis: Defined as $\mathrm{E}\{(Y-\mu)^4\}/\sigma^4 - 3$, so that normal distribution has value zero. It can be estimated by

$$g_2 \;=\; \frac{1}{n\,\widehat{\sigma}^4}\sum_{i=1}^{n}(Y_i-\bar{Y})^4 \;-\; 3$$

Sample is light/heavy-tailed for values less/greater than zero.

In R: `kurtosis()` from *e1071*. Again various definitions exist and `kurtosis()` by default computes version with finite sample correction.

```
R> library("e1071")
R> skewness(rcref, type = 1)
[1] 0.116
R> kurtosis(rcref, type = 1)
[1] 0.6274
```

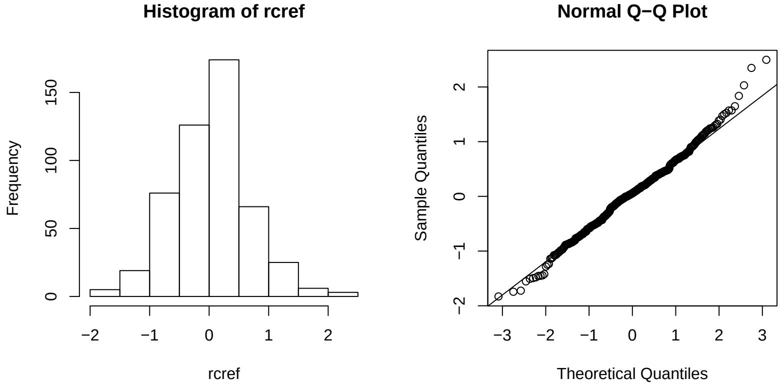
Some common features

Formal test: Assess the null hypothesis of zero skewness and zero excess kurtosis. Jarque-Bera test statistic is sum of two terms where each is asymptotically χ^2_1-distributed under the null hypothesis of normality. The joint distribution is χ^2_2.

$$JB = \frac{ng_1^2}{6} + \frac{ng_2^2}{24}$$

```
R> length(rcref) * c(skewness(rcref, type = 1)^2/6,
+     kurtosis(rcref, type = 1)^2/24)
[1] 1.121 8.201
R> jarque.bera.test(rcref)

        Jarque Bera Test

data:  rcref
X-squared = 9.3, df = 2, p-value = 0.009
```

Some common features



Some common features

Summary: Results for CREF returns.

- Serially uncorrelated.
- Higher-order dependence structure: Volatility clustering and heavy tails.
- Typical for financial time series.
- Often much more extreme. Potentially also with skewness, etc.

Some common features

Further illustration: Daily closing prices (regular "`ts`" series in business days) the German DAX stock index.

```
R> data("EuStockMarkets", package = "datasets")
R> rdax <- 100 * diff(log(EuStockMarkets[, "DAX"]))
R> coeftest(lm(rdax ~ 1))
```

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.0652     0.0239    2.73   0.0064
```

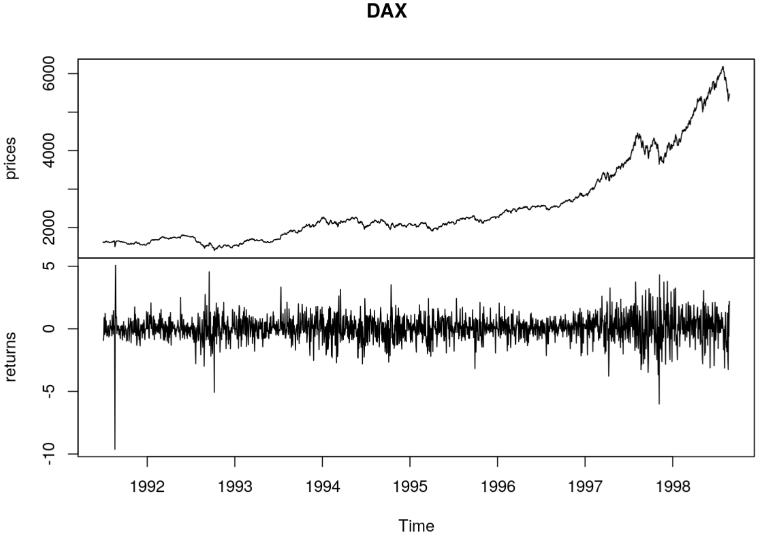
```
R> Box.test(rdax^2)
```

```
        Box-Pierce test

data:  rdax^2
X-squared = 12, df = 1, p-value = 7e-04
```

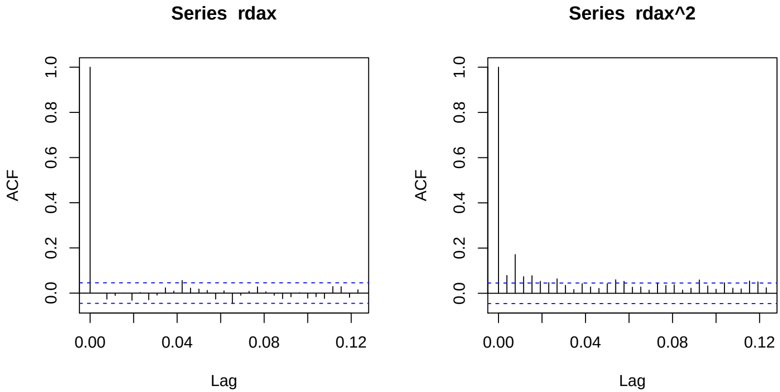
```
R> length(rdax) * c(skewness(rdax, type = 1)^2/6,
+    kurtosis(rdax, type = 1)^2/24)
```

```
[1]   95.11 3054.53
```

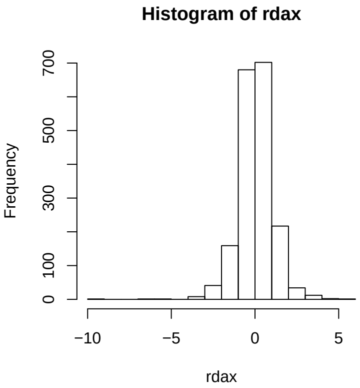
Some common features

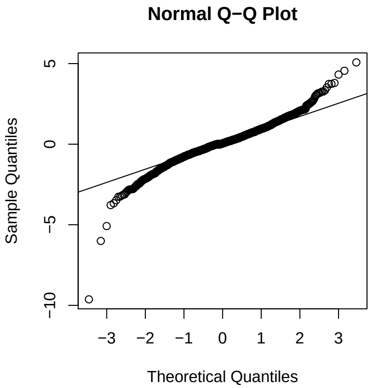


Some common features



Some common features





Models of Heteroskedasticity

The ARCH(1) Model

The ARCH(1) model

Approach: Model changing variance of a time series.

- Assume $\{r_t\}$ is serially uncorrelated with zero mean.
- Capture conditional variance or *conditional volatility*

$$\sigma^2_{t|t-1} = \text{Var}(r_t \mid r_{t-1}, r_{t-2}, \dots)$$

- Note that r_t^2 (if it is available) is an unbiased estimator of $\sigma^2_{t|t-1}$.
- Employ autoregressive model for $\sigma^2_{t|t-1}$ with past squared returns r^2_{t-k} as regressors.

Name: *Autoregressive conditional heteroskedasticity* (ARCH) model.

Remark: First proposed by Engle (1982)

The ARCH(1) model

Example: ARCH(1).

$$\begin{aligned} r_t &= \sigma_{t|t-1}\,\varepsilon_t \\ \sigma^2_{t|t-1} &= \omega + \alpha r^2_{t-1} \end{aligned}$$

with

- ω and α unknown parameters.
- $\{\varepsilon_t\}$ is an independently and identically distributed (i.i.d.) series of innovations with zero mean and unit variance.
- Innovations ε_t are independent of previous observations r_{t-k} with $k \geq 1$.
- Thus, $\sigma_{t|t-1}$ is the conditional volatility of r_t.

The ARCH(1) model

Formally:

$$
\begin{aligned}
\mathrm{Var}(r_t \mid r_{t-1}, r_{t-2}, \ldots) &= \mathrm{E}(r_t^2 \mid r_{t-1}, r_{t-2}, \ldots) \\
&= \mathrm{E}(\sigma_{t|t-1}^2 \, \varepsilon_t^2 \mid r_{t-1}, r_{t-2}, \ldots) \\
&= \sigma_{t|t-1}^2 \mathrm{E}(\varepsilon_t^2 \mid r_{t-1}, r_{t-2}, \ldots) \\
&= \sigma_{t|t-1}^2 \mathrm{E}(\varepsilon_t^2) \\
&= \sigma_{t|t-1}^2
\end{aligned}
$$

The ARCH(1) model

Illustration: Artificial ARCH(1) series with $\omega = 0.01$ and $\alpha = 0.9$.

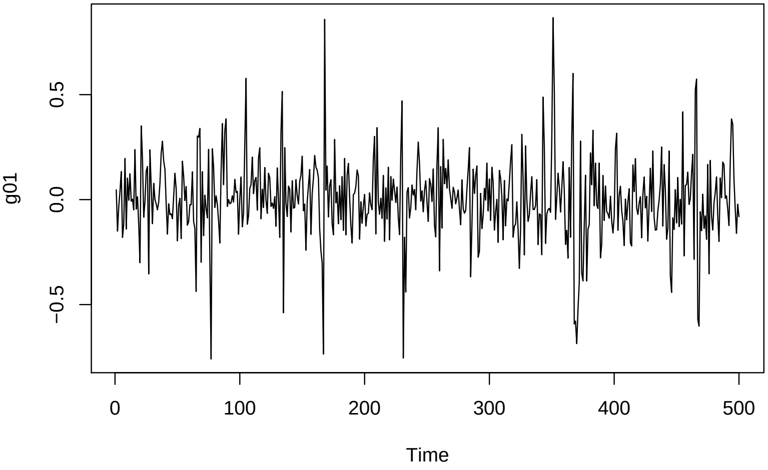
In R: `garch.sim()` in *TSA* (returning a "`numeric`" vector).

As before: Loading *TSA* is problematic, however, as several functions are overwritten.

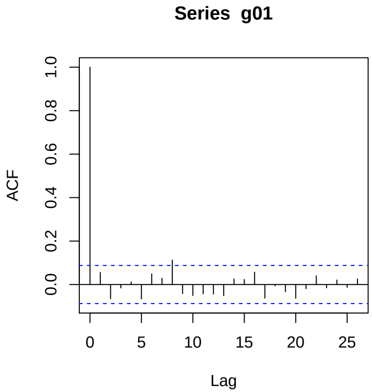
For replication: Simulated series `g01` and `g11` from Cryer & Chan (2008) are provided in `garch-sim.rda`.

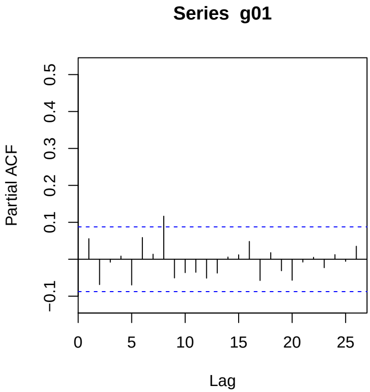
```
R> load("Data/garch-sim.rda")
R> plot(g01)
```

The ARCH(1) model

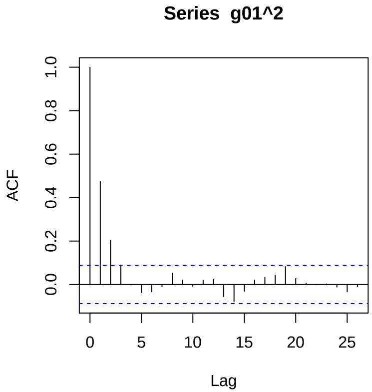


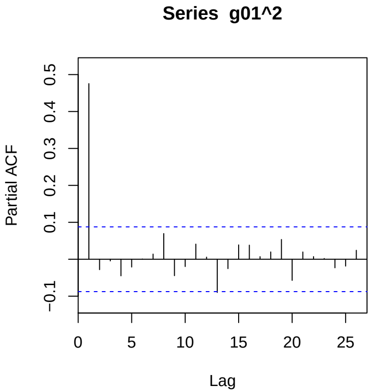
The ARCH(1) model





The ARCH(1) model



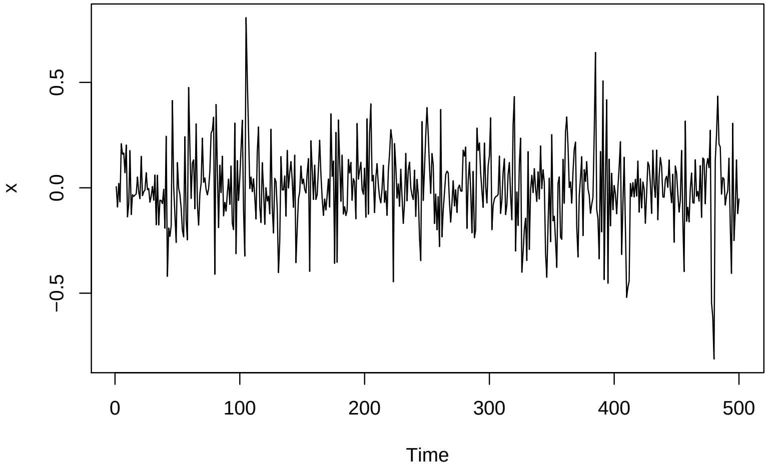


The ARCH(1) model

Alternatively: More flexible function `garchSim()` in *fGarch* (returning a "`timeSeries`" object).

```
R> set.seed(1)
R> x <- garchSpec(model = list(omega = 0.01, alpha = 0.9, beta = 0))
R> x
Formula:
 ~ arch(1)
Model:
 omega: 0.01
 alpha: 0.9
Distribution:
 norm
Presample:
  time       z   h y
1    0 -0.6265 0.1 0
R> x <- garchSim(x, n = 500)
R> x <- ts(as.vector(coredata(as.zoo(x))))
```

The ARCH(1) model



The ARCH(1) model

Question: Can the ARCH(1) model explain the stylized facts: volatility clustering and heavy tails?

Answer 1: By construction, volatility clustering is captured.

Moreover: The series

$$\eta_t = r_t^2 - \sigma_{t|t-1}^2$$

can be shown to be serially uncorrelated with zero mean. Hence, the model equation can be written as an AR(1) for the squared returns.

$$r_t^2 = \omega + \alpha r_{t-1}^2 + \eta_t$$

Consequently: Standard model selection techniques can be applied to the squared returns r_t^2.

The ARCH(1) model

Hence: Assuming stationarity, taking expectations yields

$$\sigma^2 = \omega + \alpha\sigma^2 = \frac{\omega}{1-\alpha}$$

It can also be shown that a necessary and sufficient condition for stationarity of the ARCH(1) process is $0 \leq \alpha < 1$ and $\omega > 0$.

Remarks:

- Similarly to AR processes (which are stationary but have nonconstant conditional mean) ARCH processes are stationary but have nonconstant conditional variance.
- ARCH processes are (weak) white noise.

The ARCH(1) model

Answer 2: Even for normal innovations $\{\varepsilon_t\}$, the resulting ARCH(1) r_t have heavy tails.

Explanation: Assume i.i.d. standard normal ε_t.

$$
\begin{aligned}
\mathsf{E}(r_t^4) &= \mathsf{E}\{\mathsf{E}(\sigma_{t|t-1}^4 \varepsilon_t^4 \mid r_{t-1}, r_{t-2}, \dots)\} \\
&= \mathsf{E}\{\sigma_{t|t-1}^4 \mathsf{E}(\varepsilon_t^4 \mid r_{t-1}, r_{t-2}, \dots)\} \\
&= \mathsf{E}\{\sigma_{t|t-1}^4 \mathsf{E}(\varepsilon_t^4)\} \\
&= 3\mathsf{E}(\sigma_{t|t-1}^4)
\end{aligned}
$$

Assuming $\mathsf{E}(\sigma_{t|t-1}^4)$ exists, say a finite value τ, (and assuming stationarity), taking expectations in the squared model equation yields:

$$\tau = \omega^2 + 2\omega\alpha\sigma^2 + \alpha^2 3\tau = \frac{\omega^2 + 2\omega\alpha\sigma^2}{1 - 3\alpha^2}$$

The ARCH(1) model

Remarks:

- Thus, for finite fourth moments, $0 \leq \alpha < 1/\sqrt{3}$ is needed.
- Hence, stationary ARCH models may not have finite fourth moments.
- It can be shown that $\tau > \sigma^4$ and hence $\mathrm{E}(r_t^4) > 3\sigma^4$ implying a positive excess kurtosis (also known as heavy tails).

The ARCH(1) model

Predictions: Recursion for ℓ-step-ahead conditional variance.

$$\begin{aligned}
\sigma^2_{t+\ell|t} &= \mathsf{E}(r^2_{t+\ell} \mid r_t, r_{t-1}, \dots) \\
&= \mathsf{E}\{\mathsf{E}(\sigma^2_{t+\ell|t+\ell-1}\varepsilon^2_{t+\ell} \mid r_t, r_{t-1}, \dots) \mid r_t, r_{t-1}, \dots\} \\
&= \mathsf{E}\{\sigma^2_{t+\ell|t+\ell-1}\mathsf{E}(\varepsilon^2_{t+\ell}) \mid r_t, r_{t-1}, \dots\} \\
&= \mathsf{E}(\sigma^2_{t+\ell|t+\ell-1} \mid r_t, r_{t-1}, \dots) \\
&= \omega \,+\, \alpha\, \mathsf{E}(r^2_{t+\ell-1} \mid r_t, r_{t-1}, \dots) \\
&= \omega \,+\, \alpha\, \sigma^2_{t+\ell-1|t}
\end{aligned}$$

with convention $\sigma^2_{t+\ell|t} = r^2_{t+\ell}$ for $\ell < 0$.

Specifically, for $\ell = 1$, weighted average of last squared return and long-run variance.

$$\sigma^2_{t+1|t} \,=\, \omega \,+\, \alpha r^2_t \,=\, (1-\alpha)\sigma^2 \,+\, \alpha r^2_t$$

Models of Heteroskedasticity

GARCH Models

GARCH models

Problem: Predictions only employ last observed return.

Conceivable solutions:

- Include up to q lagged squared returns: ARCH(q), Engle (1982).
- Include p lagged conditional volatilities: generalized ARCH (GARCH) effects, Bollerslev (1986) and Taylor (1986).
- Combinations of both: GARCH(p, q) model.

Model equation: GARCH(p, q).

$$\sigma^2_{t|t-1} = \omega + \beta_1 \sigma^2_{t-1|t-2} + \cdots + \beta_p \sigma^2_{t-p|t-p-1} + \alpha_1 r^2_{t-1} + \cdots + \alpha_q r^2_{t-q}$$

Or more compactly

$$\beta(B)\sigma^2_{t|t-1} = \omega + \alpha(B) r^2_t$$

Note that textbooks/software may interchange p and q.

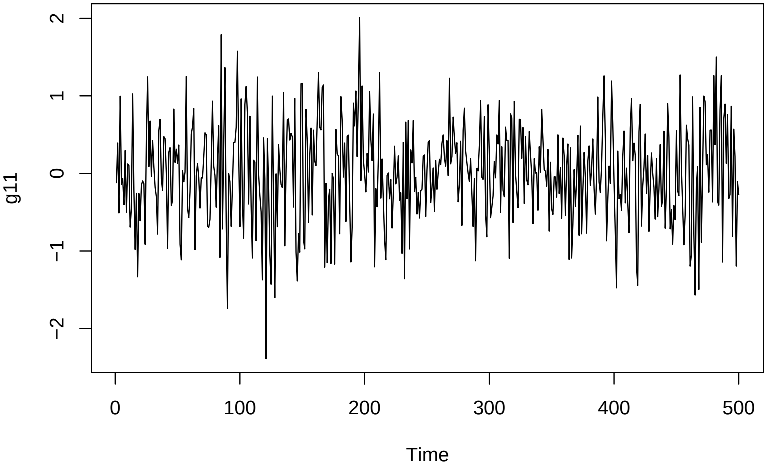
GARCH models

Remarks:

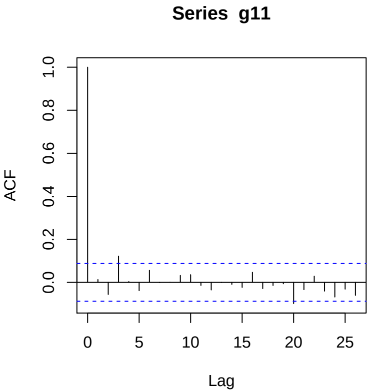
- Coefficients in GARCH models are often constrained to be nonnegative because the conditional variances must be nonnegative.
- However, nonnegative parameters are not necessary to obtain nonnegative conditional variances with probability 1.

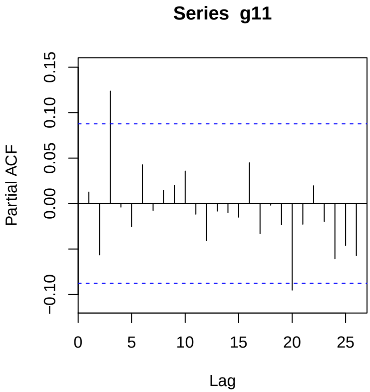
Illustration: Artificial GARCH(1, 1) series with $\omega = 0.02$, $\alpha = 0.05$, and $\beta = 0.9$. Inclusion of lagged conditional volatility increases the persistence in the volatility.

GARCH models

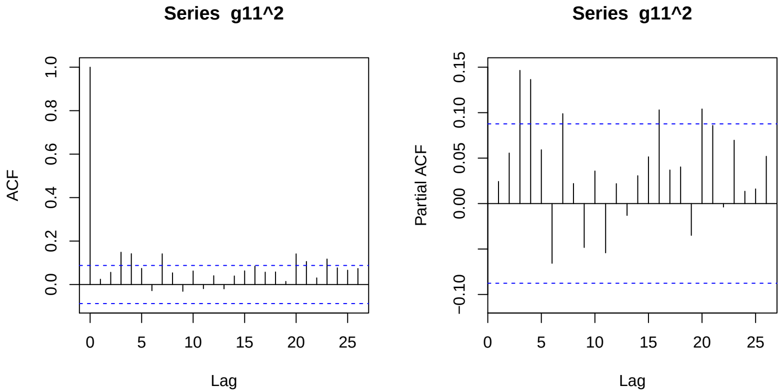


GARCH models





GARCH models



GARCH models

Model identification: Similarly to the ARCH(1) case, $\eta_t = r_t^2 - \sigma_{t|t-1}^2$ can be shown to be serially uncorrelated and uncorrelated with past squared returns. Thus,

$$\begin{aligned} r_t^2 &= \omega + (\beta_1 + \alpha_1) r_{t-1}^2 + \dots + (\beta_{\max(p,q)} + \alpha_{\max(p,q)}) r_{t-\max(p,q)}^2 \\ &\quad + \eta_t - \beta_1 \eta_{t-1} - \dots - \beta_p \eta_{t-p} \end{aligned}$$

with $\beta_k = 0$ for $k > p$ and $\alpha_k = 0$ for $k > q$.

Thus: GARCH(p, q) model for r_t implies ARMA($\max(p, q)$, p) model for squared returns r_t^2. Below employ $m = \max(p, q)$ for ease of notation.

Naive estimation: ARMA(1, 1) for squared artificial GARCH(1, 1) series lead to surprisingly accurate estimates.

```
R> arma11 <- arima(g11^2, order = c(1, 0, 1))
```

GARCH models

```
R> arma11

Call:
arima(x = g11^2, order = c(1, 0, 1))

Coefficients:
        ar1      ma1  intercept
      0.982  -0.947      0.412
s.e.  0.013   0.022      0.074

sigma^2 estimated as 0.342:  log likelihood = -441.7,  aic = 891.4
R> b <- - coef(arma11)[2]
R> a <- coef(arma11)[1] - b
R> o <- coef(arma11)[3] * (1 - a - b)
R> par <- c(o, a, b)
R> names(par) <- c("omega", "alpha", "beta")
R> par
   omega    alpha     beta
0.007241 0.035436 0.946982
```

GARCH models

Unconditional variance: Under the assumption of stationarity, taking expectations yields

$$\sigma^2 \;=\; \omega \;+\; \sigma^2 \sum_{i=1}^{m} (\alpha_i + \beta_i) \;=\; \frac{\omega}{1 - \sum_{i=1}^{m} (\alpha_i + \beta_i)}$$

which is finite if $\sum_{i=1}^{m} (\alpha_i + \beta_i) < 1$. This can be shown to be a necessary and sufficient condition for weak stationarity.

Conditional variance: ℓ-step-ahead prediction for $\ell > p$.

$$\sigma^2_{t+\ell|t} \;=\; \omega \;+\; \sum_{i=1}^{m} (\alpha_i + \beta_i) \sigma^2_{t+\ell-i|t}$$

GARCH models

For $\ell \geq 1$: Formula is more involved.

$$\sigma^2_{t+\ell|t} \; = \; \omega \; + \; \sum_{i=1}^{q} \alpha_i \sigma^2_{t+\ell-i|t} \; + \; \sum_{i=1}^{p} \beta_i \widehat{\sigma}^2_{t+\ell-i|t+\ell-i-1}$$

with $\sigma^2_{t+\ell|t} = r^2_{t+\ell}$ for $\ell < 0$ and

$$\widehat{\sigma}^2_{t+\ell-i|t+\ell-i-1} \; = \; \begin{cases} \sigma^2_{t+\ell-i|t} & \text{for } \ell - i - 1 > 0 \\ \sigma^2_{t+\ell-i|t+\ell-i-1} & \text{otherwise} \end{cases}$$

GARCH models

Example: GARCH(1, 1).

$$\begin{aligned}\sigma^2_{t|t-1} &= \omega + \alpha r^2_{t-1} + \beta\sigma^2_{t-1|t-2} \\ &= (1-\alpha-\beta)\sigma^2 + \alpha r^2_{t-1} + \beta\sigma^2_{t-1|t-2} \\ &= \sigma^2 + \alpha(r^2_{t-1} + \beta r^2_{t-2} + \beta^2 r^2_{t-3} + \dots)\end{aligned}$$

Remarks:

- Conditional volatility is weighted average of long-run variance, previous squared return, and previous conditional volatility.
- Conditional volatility has MA(∞) representation in squared returns.
- Naive finite moving averages of squared returns have larger bias.
- If $\alpha + \beta = 1$ the model is nonstationary. Also called IGARCH (integrated GARCH).

Models of Heteroskedasticity

Maximum Likelihood Estimation

Maximum likelihood estimation

Goal: More efficient estimation of model parameters using maximum likelihood (ML).

Approach: Derive likelihood based on distributional assumption for innovations ε_t.

Example: Stationary GARCH(1, 1) with standard normal innovations.

Recursion for conditional variances

$$\sigma^2_{t|t-1} \;=\; \omega \;+\; \alpha r^2_{t-1} \;+\; \beta \sigma^2_{t-1|t-2}$$

for $t \geq 2$ with initial value $\sigma^2_{1|0} = \sigma^2 = \omega(1 - \alpha - \beta)$.

Conditional probability density function (PDF)

$$f(r_t \mid r_{t-1}, \dots r_1) \;=\; \frac{1}{\sqrt{2\pi\sigma^2_{t|t-1}}} \exp\{-1/2 \cdot r_t^2 / \sigma^2_{t|t-1}\}$$

Maximum likelihood estimation

Joint PDF can be defined recursively

$$f(r_n, \ldots, r_1) \;=\; f(r_{n-1}, \ldots, r_1) f(r_n \mid r_{n-1}, \ldots, r_1)$$

Taking logs yields

$$\log L(\omega, \alpha, \beta) \;=\; -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{n}\left\{\log(\sigma^2_{t-1|t-2}) \;+\; r_t^2/\sigma^2_{t|t-1}\right\}$$

Remarks: "As usual" for nonlinear ML problems.

- ML estimator has no closed-form solution but numerical techniques are required.
- Covariance matrix can be estimated using various types of estimators (Hessian, outer product of gradients, etc.), may be sensitive to distributional assumptions.
- Inference is typically based on central limit theorem.

Maximum likelihood estimation

In R: `garch()` in *tseries* and `garchFit()` in *fGarch*.

tseries: `garch()`.

- GARCH(p, q) with Gaussian innovations.
- Default is GARCH(1, 1).
- Employs custom quasi-Newton optimizer.
- Standard errors are based on outer product of gradients (relying on normality).
- Rather standard interface with `coef()`, `vcov()`, `logLik()`, … extractor functions.
- Coefficient naming: `a0` corresponds to ω. Remaining `a*` coefficients correspond to α, `b*` to β.

Maximum likelihood estimation

fGarch: `garchFit()`. Contained in *Rmetrics* suite of packages.

- GARCH(q, p) (note order!) with various types of innovations: Gaussian, t, GED (generalized error distribution), and skewed generalizations.
- Default is GARCH(1, 1) with Gaussian innovations.
- Several maximization algorithms, default is `nlminb()`, with two methods for initializing recursions.
- Standard errors are based on Hessian.
- A constant is included by default, not possible in `garch()`.
- Furthermore, ARMA models can be added for modeling conditional mean.
- Less standard interface, e.g., no `vcov()` or `logLik()` methods. But quantities can be extracted manually.

Maximum likelihood estimation

Illustration: GARCH(1, 1) with $\omega = 0.02$, $\alpha = 0.05$, $\beta = 0.9$.

```
R> g11_g1 <- garch(g11, order = c(1, 1), trace = FALSE)
R> g11_g2 <- garch(g11, order = c(2, 2), trace = FALSE)
R> g11_gf1 <- garchFit(~ garch(1,1), data = g11, trace = FALSE,
+    include.mean = FALSE)
R> g11_gf2 <- garchFit(~ garch(2,2), data = g11, trace = FALSE,
+    include.mean = FALSE)
R> sapply(list(g11_gf1, g11_g1), coef)
```

```
           [,1]     [,2]
omega  0.007656 0.007575
alpha1 0.047266 0.047184
beta1  0.935066 0.935377
```

```
R> sapply(list(g11_gf2, g11_g2), coef)
```

```
            [,1]      [,2]
omega  9.266e-03 1.835e-02
alpha1 1.000e-08 9.976e-16
alpha2 5.436e-02 1.136e-01
beta1  8.790e-01 3.369e-01
beta2  4.523e-02 5.100e-01
```

Maximum likelihood estimation

```
R> summary(g11_g1)
Call:
garch(x = g11, order = c(1, 1), trace = FALSE)

Model:
GARCH(1,1)

Residuals:
     Min       1Q   Median       3Q      Max
-3.30703 -0.63798  0.00916  0.74198  3.01944

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0   0.00757     0.00759     1.00    0.318
a1   0.04718     0.02231     2.12    0.034
b1   0.93538     0.03584    26.10   <2e-16

Diagnostic Tests:
```

Maximum likelihood estimation

```
        Jarque Bera Test

data:  Residuals
X-squared = 0.83, df = 2, p-value = 0.7


        Box-Ljung test

data:  Squared.Residuals
X-squared = 0.54, df = 1, p-value = 0.5
R> summary(g11_g2)
Call:
garch(x = g11, order = c(2, 2), trace = FALSE)

Model:
GARCH(2,2)

Residuals:
     Min       1Q   Median       3Q      Max
-3.34683 -0.63188  0.00847  0.73611  3.20234
```

Maximum likelihood estimation

```
Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0  1.84e-02    1.51e-02     1.21    0.226
a1  9.98e-16    4.72e-02     0.00    1.000
a2  1.14e-01    5.86e-02     1.94    0.052
b1  3.37e-01    3.70e-01     0.91    0.362
b2  5.10e-01    3.58e-01     1.43    0.154

Diagnostic Tests:
        Jarque Bera Test

data:  Residuals
X-squared = 0.42, df = 2, p-value = 0.8


        Box-Ljung test

data:  Squared.Residuals
X-squared = 0.0053, df = 1, p-value = 0.9
```

Maximum likelihood estimation

```
R> logLik(g11_g1)
'log Lik.' -476 (df=3)
R> logLik(g11_g2)
'log Lik.' -475.5 (df=5)
R> AIC(g11_g1, g11_g2)
       df AIC
g11_g1  3 958
g11_g2  5 961
R> summary(g11_gf1)
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = g11, include.mean = FALSE,
    trace = FALSE)
```

Maximum likelihood estimation

```
Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x563e88ef0c10>
 [data = g11]

Conditional Distribution:
 norm

Coefficient(s):
    omega     alpha1      beta1
0.0076562  0.0472661  0.9350665

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
omega   0.007656    0.006251    1.225   0.2206
alpha1  0.047266    0.020207    2.339   0.0193
beta1   0.935066    0.029441   31.760   <2e-16
```

Maximum likelihood estimation

```
Log Likelihood:
 -476.5    normalized:  -0.953

Description:
 Mon Mar  4 15:50:03 2019 by user: zeileis

Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  0.7971    0.6713
 Shapiro-Wilk Test  R    W      0.998     0.8163
 Ljung-Box Test     R    Q(10)  9.972     0.4429
 Ljung-Box Test     R    Q(15)  10.87     0.762
 Ljung-Box Test     R    Q(20)  18.09     0.5815
 Ljung-Box Test     R^2  Q(10)  7.953     0.6334
 Ljung-Box Test     R^2  Q(15)  15.57     0.4111
 Ljung-Box Test     R^2  Q(20)  18.26     0.5704
 LM Arch Test       R    TR^2   10.32     0.5876

Information Criterion Statistics:
  AIC   BIC   SIC  HQIC
1.918 1.943 1.918 1.928
```

Models of Heteroskedasticity

Model Diagnostics

Model diagnostics

Basic tool: Residuals. For GARCH model with mean μ

$$r_t = \mu + \sigma_{t|t-1}\varepsilon_t$$

the standardized residuals are

$$\hat{\varepsilon}_t = \frac{r_t - \hat{\mu}}{\widehat{\sigma}_{t|t-1}}$$

tseries:

- `fitted()` computes bivariate series with $\widehat{\sigma}_{t|t-1}$, $-\widehat{\sigma}_{t|t-1}$.
- `residuals()` computes $\hat{\varepsilon}_t$ with $\mu = 0$ (as estimation of μ is not supported).

fGarch:

- `volatility()` computes $\widehat{\sigma}_{t|t-1}$ by default. Conditional variances are also available.
- `residuals()` computes $r_t - \hat{\mu}$ by default. $\hat{\varepsilon}_t$ is available with option `standardize = TRUE`.

Model diagnostics

Idea: If the model is correctly specified, the $\hat{\varepsilon}_t$ are approximately i.i.d. Employ standard diagnostics.

- Autocorrelation: Ljung-Box test, ...
- Skewness/kurtosis: Jarque-Bera test, ...
- Normality: QQ plot, Shapiro-Wilk test, ...

Caveat:

- Jarque-Bera test for GARCH residuals no longer has χ^2 distribution.
- Test becomes liberal, i.e., more rejections under the null hypothesis than specified significance level.
- Modified version has been suggested in the literature, however, rarely applied in practice.
- `summary()` methods for both "`garch`" and "`fGARCH`" objects employ the (incorrect) unmodified version.

Model diagnostics

```
R> g11_res <- na.omit(residuals(g11_g1))
R> c(skewness(g11_res), kurtosis(g11_res))
[1] -0.08824 -0.10409
R> jarque.bera.test(g11_res)
        Jarque Bera Test

data:  g11_res
X-squared = 0.83, df = 2, p-value = 0.7
R> Box.test(g11_res^2, type = "Ljung-Box")
        Box-Ljung test

data:  g11_res^2
X-squared = 0.54, df = 1, p-value = 0.5
```

Note: *timeDate* from *Rmetrics* suite has different `skewness()` and `kurtosis()` methods. Order of loading determines which is found first.

GARCH Modeling of CREF Returns

GARCH modeling of CREF returns

Goal: Find a satisfactory model for returns from College Retirement Equities Fund (2004-08-27 to 2006-08-15).

Strategy:

- Consider GARCH(p, q) models with $p = 0, 1, 2$ and $q = 1, 2$, with and without mean μ.
- Estimation of $\mu = 0$ models is carried out using `garch()` from *tseries*.
- Estimation of models with non-zero μ is carried out using `garchFit()` from *fGarch*. Some convenience glue code is needed.
- Choose best model via information criteria (AIC, BIC).
- Assess adequacy using model diagnostics.

GARCH modeling of CREF returns

Set up grid of parameters.

```
R> par <- expand.grid(p = 0:2, q = 1:2)
```

Compute list of fitted "garch" models.

```
R> fm <- lapply(1:nrow(par), function(i)
+    garch(rcref, order = c(par$p[i], par$q[i]), trace = FALSE))
```

Combine parameter grid with information about fitted log-likelihood, AIC, and BIC.

```
R> par <- cbind(par,
+    logLik = sapply(fm, logLik),
+    AIC = sapply(fm, AIC),
+    BIC = sapply(fm, AIC, k = log(length(rcref)))
+  )
```

GARCH modeling of CREF returns

Convenience function that generates a formula like `~ garch(1, 1)` from given integers `p` and `q` (note ordering!).

```
R> garch_formula <- function(p, q)
+    as.formula(sprintf("~ garch(%f, %f)", q, p))
```

To facilitate extraction of log-likelihood (and hence AIC/BIC), provide convenience S3 `logLik()` method for "`fGARCH`" models.

```
R> logLik.fGARCH <- function(object, ...)
+    structure(-object@fit$llh, df = length(coef(object)),
+      class = "logLik")
```

Compute list of "`fGARCH`" models, including mean.

```
R> fm2 <- lapply(1:nrow(par), function(i)
+    garchFit(garch_formula(par$p[i], par$q[i]),
+      data = rcref, trace = FALSE))
```

GARCH modeling of CREF returns

Collect fitted fitted log-likelihoods, AICs, and BICs.

```
R> par <- cbind(par,
+    logLik2 = sapply(fm2, logLik),
+    AIC2 = sapply(fm2, AIC),
+    BIC2 = sapply(fm2, AIC, k = log(length(rcref)))
+  )
R> round(par, digits = 1)
  p q logLik   AIC   BIC logLik2  AIC2   BIC2
1 0 1 -490.6 985.1 993.5  -489.7 985.3  998.0
2 1 1 -481.8 969.6 982.2  -479.8 967.6  984.5
3 2 1 -480.7 969.4 986.3  -479.8 969.6  990.7
4 0 2 -488.7 983.4 996.0  -488.3 984.6 1001.4
5 1 2 -485.1 978.2 995.0  -479.4 968.7  989.8
6 2 2 -480.2 970.3 991.4  -479.4 970.7  996.0
```

Conclusion: GARCH(1, 1) seems to be favored (both by AIC, and BIC). AIC also prefers inclusion of mean μ.

GARCH modeling of CREF returns

Pick selected model from list.

```
R> cref_g11 <- fm2[[2]]
R> summary(cref_g11)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = garch_formula(par$p[i], par$q[i]), data = rcref,
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x563e88830170>
 [data = rcref]

Conditional Distribution:
 norm
```

GARCH modeling of CREF returns

```
Coefficient(s):
      mu     omega    alpha1     beta1
0.062828  0.017698  0.049061  0.908419

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu       0.06283     0.02743    2.291   0.0220
omega    0.01770     0.01041    1.700   0.0890
alpha1   0.04906     0.01938    2.532   0.0114
beta1    0.90842     0.03687   24.640   <2e-16

Log Likelihood:
 -479.8    normalized:  -0.9596

Description:
 Mon Mar  4 15:50:03 2019 by user: zeileis
```

GARCH modeling of CREF returns

```
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  0.8842    0.6427
 Shapiro-Wilk Test  R    W      0.9966    0.3785
 Ljung-Box Test     R    Q(10)  11.02     0.3557
 Ljung-Box Test     R    Q(15)  19.38     0.197
 Ljung-Box Test     R    Q(20)  22.34     0.3222
 Ljung-Box Test     R^2  Q(10)  8.634     0.5672
 Ljung-Box Test     R^2  Q(15)  18.16     0.2546
 Ljung-Box Test     R^2  Q(20)  20.22     0.4443
 LM Arch Test       R    TR^2   15.61     0.2098

Information Criterion Statistics:
  AIC   BIC   SIC  HQIC
1.935 1.969 1.935 1.948
```

GARCH modeling of CREF returns

GARCH(1, 1) long-run variance and sample variance are very close.

```
R> coef(cref_g11)["omega"] /
+    (1 - coef(cref_g11)["alpha1"] - coef(cref_g11)["beta1"])
```

```
 omega
0.4162
```

```
R> var(rcref)
```

```
[1] 0.4161
```

GARCH modeling of CREF returns

Extract standardized residuals

```
R> cref_res <- residuals(cref_g11, standardize = TRUE)
```

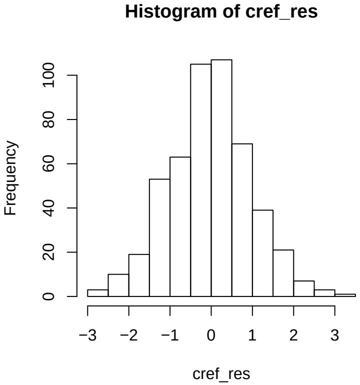
which can then be visualized using `hist()`, `qqnorm()`, `acf()`, etc.

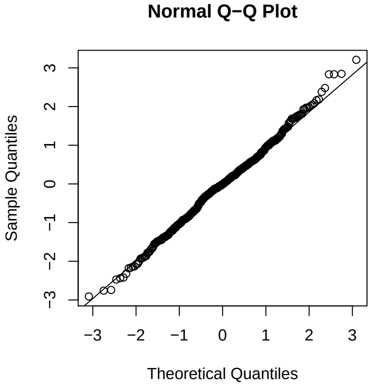
Furthermore, `plot(cref_g11)` produces various diagnostic plots (chosen from an interactive menu). Especially, a plot of $\mu \pm 1.96\sigma_{t|t-1}$ is often used for model visualization.

Remarks:

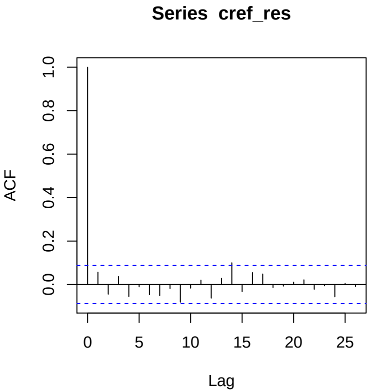
- Model fit depends on scaling of observations (such as multiplication by 100).
- `garchFit()` reports scaled information criteria, divided by *n*.

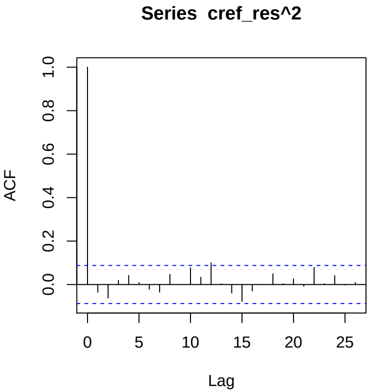
GARCH modeling of CREF returns



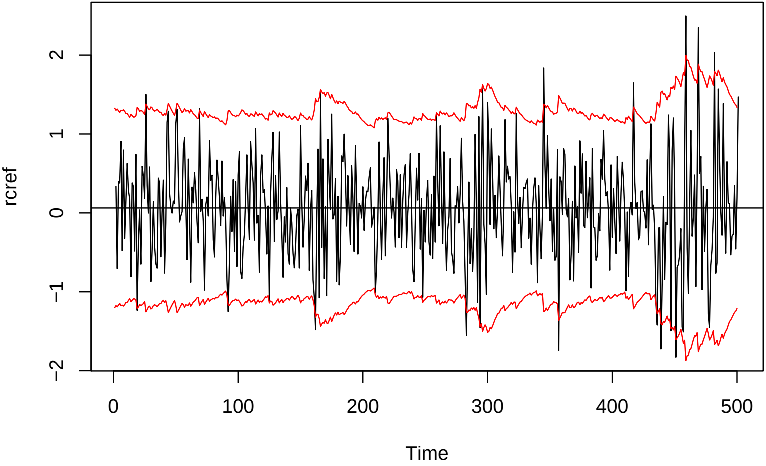


GARCH modeling of CREF returns





GARCH modeling of CREF returns



Models of Heteroskedasticity

Some Extensions of the GARCH Model

Some extensions of the GARCH model

Extensions: Numerous variations and generalizations have been suggested in the literature, e.g., incorporating nonlinearities, asymmetries, links, thresholds, etc.

- GJR-GARCH: Asymmetry via threshold.
- TGARCH: Threshold GARCH.
- EGARCH: Exponential GARCH.
- ...

Generalization: APARCH (asymmetric power ARCH) comprises many GARCH-type models, including ARCH, GARCH, Taylor/Schwert GARCH, GJR-GARCH, TARCH, NARCH, log-ARCH, ...

Furthermore: Combination of ARMA model for mean and GARCH-type model for variance.

In R: `garchFit()` from *fGarch*.

Some extensions of the GARCH model

ARMA-GARCH: Mean equation is ARMA(P, Q).

$$Y_t \;=\; \theta_0 \;+\; \sum_{i=1}^{P} \phi_i Y_{t-i} \;+\; \sum_{j=1}^{Q} \theta_j e_{t-j} \;+\; e_t$$

Variance equation is GARCH(p, q).

$$\begin{aligned} e_t &= \sigma_{t|t-1}\varepsilon_t \\ \varepsilon_t &\sim \mathcal{D}_\vartheta(0,1) \quad \text{i.i.d.} \\ \sigma^2_{t|t-1} &= \omega \;+\; \sum_{i=1}^{q} \alpha_i e^2_{t-i} \;+\; \sum_{j=1}^{p} \beta_j \sigma^2_{t-j|t-j-1} \end{aligned}$$

Some extensions of the GARCH model

ARMA-APARCH: Mean equation is ARMA(P, Q).

$$Y_t \;=\; \theta_0 \;+\; \sum_{i=1}^{P} \phi_i Y_{t-i} \;+\; \sum_{j=1}^{Q} \theta_j e_{t-j} \;+\; e_t$$

Variance equation is APARCH(p, q).

$$\begin{aligned}
e_t &= \sigma_{t|t-1}\varepsilon_t \\
\varepsilon_t &\sim \mathcal{D}_\vartheta(0,1) \quad \text{i.i.d.} \\
\sigma^\delta_{t|t-1} &= \omega \;+\; \sum_{i=1}^{q} \alpha_i(|e_{t-i}| - \gamma_i e_{t-i})^\delta \;+\; \sum_{j=1}^{p} \beta_j \sigma^\delta_{t-j|t-j-1}
\end{aligned}$$

with $\delta > 0$ and leverage parameters $-1 < \gamma_i < 1$.

Some extensions of the GARCH model

Remarks: For ARMA-GARCH models.

- Maximum likelihood estimators of ARMA and GARCH parameters are approximately independent if innovations ε_t are symmetric (e.g., normal or t).
- Then, also the standard errors are approximately equal to those from separate models.
- However, for skewed innovations ARMA and GARCH estimators are correlated.