GARCH MODELS IN R

Are the variables in your GARCH model relevant?

Kris Boudt
Professor of finance and econometrics

Example

- Case of AR(1) GJR GARCH model with skewed student t innovations

```
names(coef(flexgarchfit))

"mu"       "ar1"      "omega"   "alpha1" "beta1"   "gamma1" "skew"    "shape"
```

- Can you simplify the model? $\Longleftrightarrow$ Are there parameters zero?
  - If the `ar1` parameter is zero, you can use a constant mean model.
  - If the `gamma1` parameter is zero, there is no GARCH-in-mean and you can use a standard GARCH model instead of the GJR.

Challenge

- We don't know the true parameter value. It needs to be estimated.
- Caveat: even if the true parameter is 0, the estimated parameter will not be 0.
- Example: are the `ar1` and `gamma1` parameters 0?

```
round(coef(flexgarchfit), 6)

       mu       ar1     omega    alpha1     beta1    gamma1      skew     shape
-0.000021  0.000150  0.000000  0.034281  0.968688 -0.010093  1.013487  9.139252
```

Test of statistical significance

- Use of statistical tests to decide whether the magnitude of the estimated parameter is significantly large enough to conclude that the true parameter is not zero.
- How? By comparing the estimated parameter to its

standard error = the standard deviation of the parameter estimate

t-statistic

t-statistic = estimated parameter / standard error

Example for MSFT returns

- Specify and estimate the model

```
flexgarchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                            variance.model = list(model = "gjrGARCH"),
                            distribution.model = "sstd")
flexgarchfit <- ugarchfit(data = msftret, spec = flexgarchspec)
```

- Print table with parameter estimates, standard errors, and t-statistics using:

```
round(flexgarchfit@fit$matcoef, 6)
```

Parameter estimation table

```
round(flexgarchfit@fit$matcoef, 6)

        Estimate  Std. Error    t value Pr(>|t|)
mu      0.000610    0.000189   3.220843 0.001278
ar1    -0.037799    0.013718  -2.755532 0.005860
omega   0.000002    0.000001   2.617696 0.008853
alpha1  0.034574    0.003395  10.182558 0.000000
beta1   0.935927    0.007163 130.667531 0.000000
gamma1  0.055483    0.009772   5.677857 0.000000
skew    1.059959    0.020676  51.264435 0.000000
shape   4.392327    0.256700  17.110745 0.000000
```

- Note:
  - `3.220843 = 0.000610 / 0.000189`
  - `-2.755532 = -0.037799 / 0.013718`
  - `2.617696 = 0.000002 / 0.000001`
  - ...

Interpretation of t-statistic

t-statistic = estimated parameter / standard error

- Its absolute value is a distance measure: It tells you how many standard errors the estimated parameter is separated from 0. The larger the distance, the more evidence the parameter is not 0.

Rule of thumb for deciding:
|t-statistic| > 2: estimated parameter is statistically significant $\rightarrow$ conclude that true parameter $\neq$ 0.

Conclusion for MSFT returns using t-statistics

- All t-statistics are larger than 2: all parameter estimates are statistically significant.

```
         Estimate  Std. Error    t value Pr(>|t|)
mu       0.000610    0.000189   3.220843 0.001278
ar1     -0.037799    0.013718  -2.755532 0.005860
omega    0.000002    0.000001   2.617696 0.008853
alpha1   0.034574    0.003395  10.182558 0.000000
beta1    0.935927    0.007163 130.667531 0.000000
gamma1   0.055483    0.009772   5.677857 0.000000
skew     1.059959    0.020676  51.264435 0.000000
shape    4.392327    0.256700  17.110745 0.0000000
```

- Same conclusion can be reached using the p-values in the last column.

Interpretation of p-value

- The p-value measures how likely it is that the parameter is zero:
  - The lower its value, the more evidence the parameter is not zero

Rule of thumb for deciding:
p-value < 5%: estimated parameter is statistically significant $\rightarrow$ *conclude that true parameter* $\neq 0$

GARCH MODELS IN R

Let's practice your skill to analyze statistical significance.

GARCH MODELS IN R

Do the GARCH predictions fit will with the observed returns?

Kris Boudt
Professor of finance and econometrics

Evaluation criterion

- Depends on what you want to evaluate
  - the predicted mean
  - the predicted variance
  - the predicted distribution of the returns

1) Goodness of fit for the mean prediction

- The GARCH model leads to

Predicted mean: $\hat{\mu}_t = E[\widehat{R_t \ |} I_{t-1}]$

Prediction error: $e_t = R_t - \hat{\mu}_t$

Mean squared prediction error: $\frac{1}{T}\sum_{t=1}^{T} e_t^2$ (the lower the better)

- Implementation

```
e <- residuals(tgarchfit)
mean(e^2)
```

2) Goodness of fit for the variance prediction

- The GARCH model leads to

$$\text{Predicted variance: } \hat{\sigma}_t^2 = \widehat{var(R_t \mid} I_{t-1})$$

$$= E[(R_t - \widehat{\mu_t})^2 \mid I_{t-1}]$$

$$= E[\widehat{e_t^2 \mid I}_{t-1}]$$

$$\text{Prediction error: } d_t = e_t^2 - \hat{\sigma}_t^2$$

$$\text{Mean squared prediction error : } \frac{1}{T}\sum_{t=1}^{T} d_t^2 \text{ (the lower the better)}$$

- Implementation

```
e <- residuals(tgarchfit)
d <- e^2 - sigma(tgarchfit)^2
mean(d^2)
```

Example for EUR/USD returns

```
# Specify the model
tgarchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
+             variance.model = list(model = "sGARCH",
                                    variance.targeting = TRUE),
+             distribution.model = "std")

# Estimate the model
tgarchfit <- ugarchfit(data = EURUSDret, spec = tgarchspec)
```

```
# Compute mean squared prediction error for the mean
e <- residuals(tgarchfit)^2
mean(e^2)
```

```
3.836205e-05
```

```
# Compute mean squared prediction error for the variance
d <- e^2 - sigma(tgarchfit)^2
mean(d^2)
```

```
5.662366e-09
```

3) Goodness of fit for the distribution

- The GARCH model provides a predicted density for all the returns in the sample
  - The higher the density, the more likely the return is under the estimated GARCH model
  - The likelihood of the sample is based on the product of all these densities. It measures how likely it is to that the observed returns come from the estimated GARCH model
  - The higher the likelihood, the better the model fits with your data

Example for EUR/USD returns

- Calculation in R:

```
likelihood(tgarchfit)

18528.58
```

- Analyze by comparing with other models:

```
# Complex model with many parameters
flexgarchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                            variance.model = list(model = "gjrGARCH"),
                            distribution.model = "sstd")
flexgarchfit <- ugarchfit(data = EURUSDret, spec = flexgarchspec)
```

```
likelihood(flexgarchfit)
```

```
18530.49
```

Risk of overfitting

- Attention: We use an in-sample evaluation approach where the estimation sample and evaluation sample coincides.
- Danger of overfitting:

Overfitting consists of choosing overly complex models that provide a good fit for the returns in the estimation sample used but not for the future returns that are outside of the sample.

Solution: Balance goodness of fit with a penalty for the complexity

- A GARCH model is parsimonious when it has:
  - a high likelihood
  - and a relatively low number of parameters

Information criteria

- Parsimony is measured information criteria.

information criteria = - likelihood + penalty(number of parameters)

- The lower, the better.

Rule of thumb for deciding:
Choose the model with lowest information criterion.

Results information criteria

- Method `infocriteria()` prints the information criteria for various penalties

```
infocriteria(tgarchfit)

Akaike       -7.468081
Bayes        -7.462833
Shibata      -7.468083
Hannan-Quinn -7.466241
```

- Interpretation requires to compare with the information criteria of other models.

Illustration on the EUR/USD returns

```
# Simple model with few parameters
tgarchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                         variance.model = list(model = "sGARCH",
                                          variance.targeting = TRUE),
                         distribution.model = "std")
tgarchfit <- ugarchfit(data = EURUSDret, spec = tgarchspec)

length(coef(tgarchfit))
likelihood(tgarchfit)

5
18528.58
```

```
# Complex model with many parameters
flexgarchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                            variance.model = list(model = "gjrGARCH"),
                            distribution.model = "sstd")
flexgarchfit <- ugarchfit(data = EURUSDret, spec = flexgarchspec)

length(coef(flexgarchfit))
likelihood(flexgarchfit)

8
18530.49
```

Which model is most parsimonious for EUR/USD returns?

```
# Simple model
infocriteria(tgarchfit)

Akaike       -7.468435
Bayes        -7.464499
Shibata      -7.468436
Hannan-Quinn -7.467055
```

```
# Complex model
infocriteria(flexgarchfit)

Akaike       -7.467239
Bayes        -7.456742
Shibata      -7.467244
Hannan-Quinn -7.463558
```

- The simple model has the lowest information criterion and should thus be preferred in case of the EUR/USD returns.

Result is case-specific. Case of MSFT returns

```
tgarchfit <- ugarchfit(data = msftret, spec = tgarchspec)
flexgarchfit <- ugarchfit(data = msftret, spec = flexgarchspec)
```

```
infocriteria(tgarchfit)

Akaike       -5.481895
Bayes        -5.477833
Shibata      -5.481896
Hannan-Quinn -5.480468

infocriteria(flexgarchfit)

Akaike       -5.489087
Bayes        -5.478255
Shibata      -5.489092
Hannan-Quinn -5.485282
```

GARCH MODELS IN R

KISS: Keep it Sophisticatedly Simple

GARCH MODELS IN R

Validate your assumptions about the mean and variance

Kris Boudt
Professor of finance and econometrics

Check 1: Mean and standard deviation of standardized returns

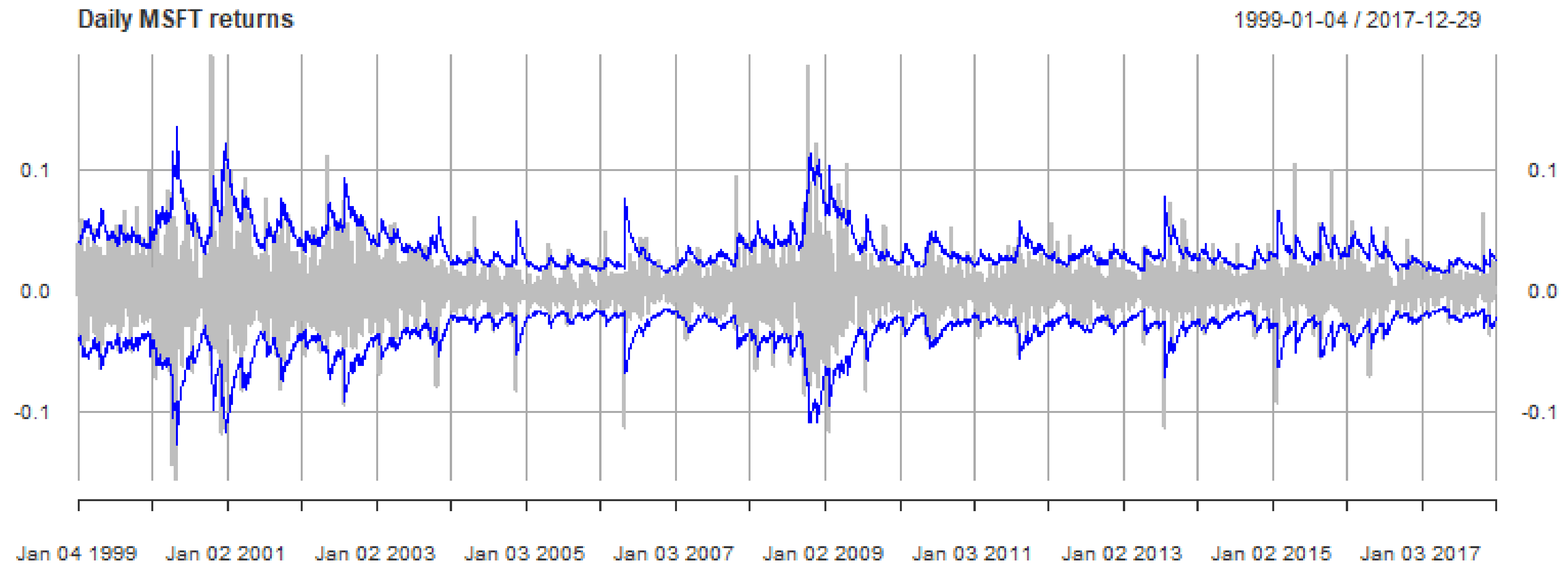
- Formula standardized returns

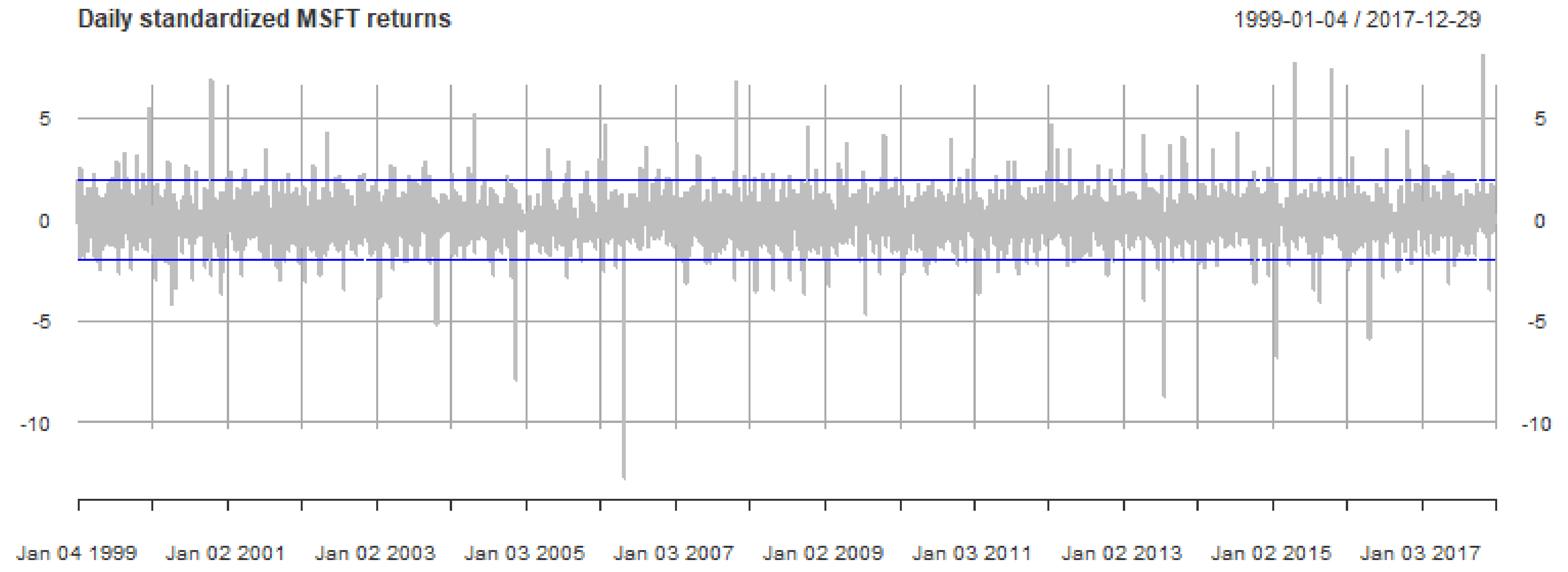
$$Z_t = \frac{R_t - \hat{\mu}_t}{\hat{\sigma}_t}$$

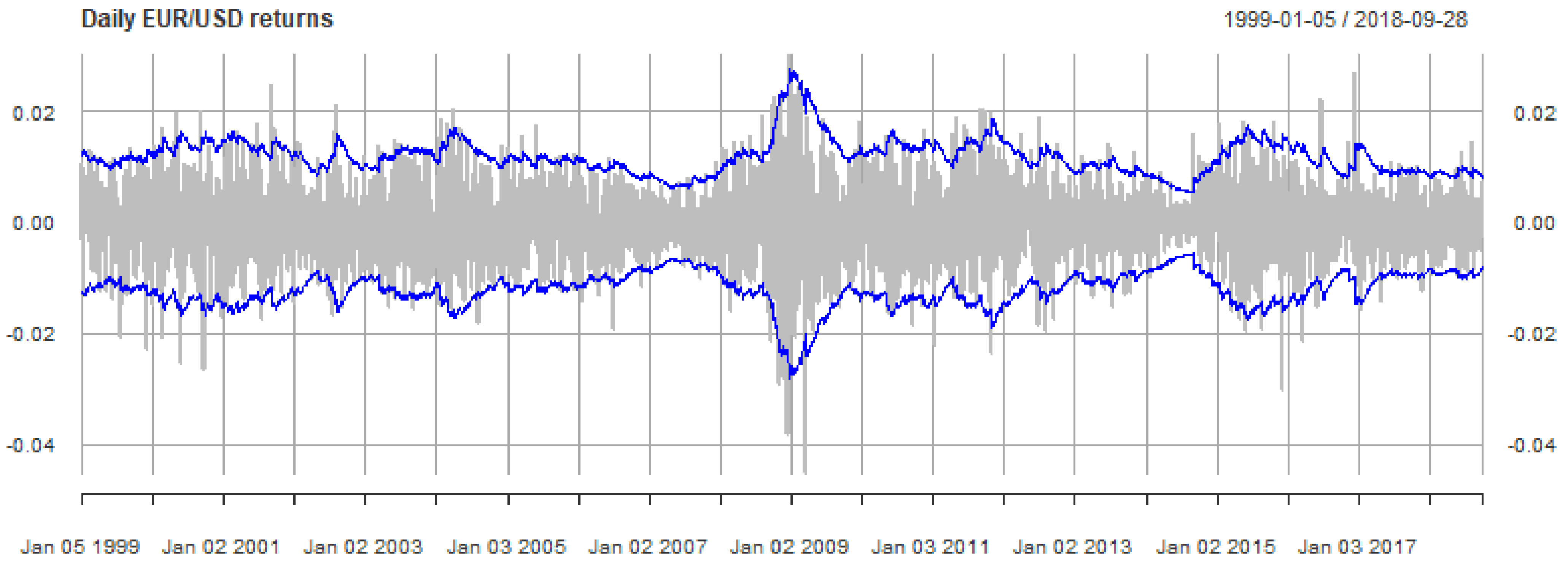
- First check of model validity:
  - Sample mean of standardized returns ≈ 0
  - Sample standard deviation of standardized returns ≈ 1

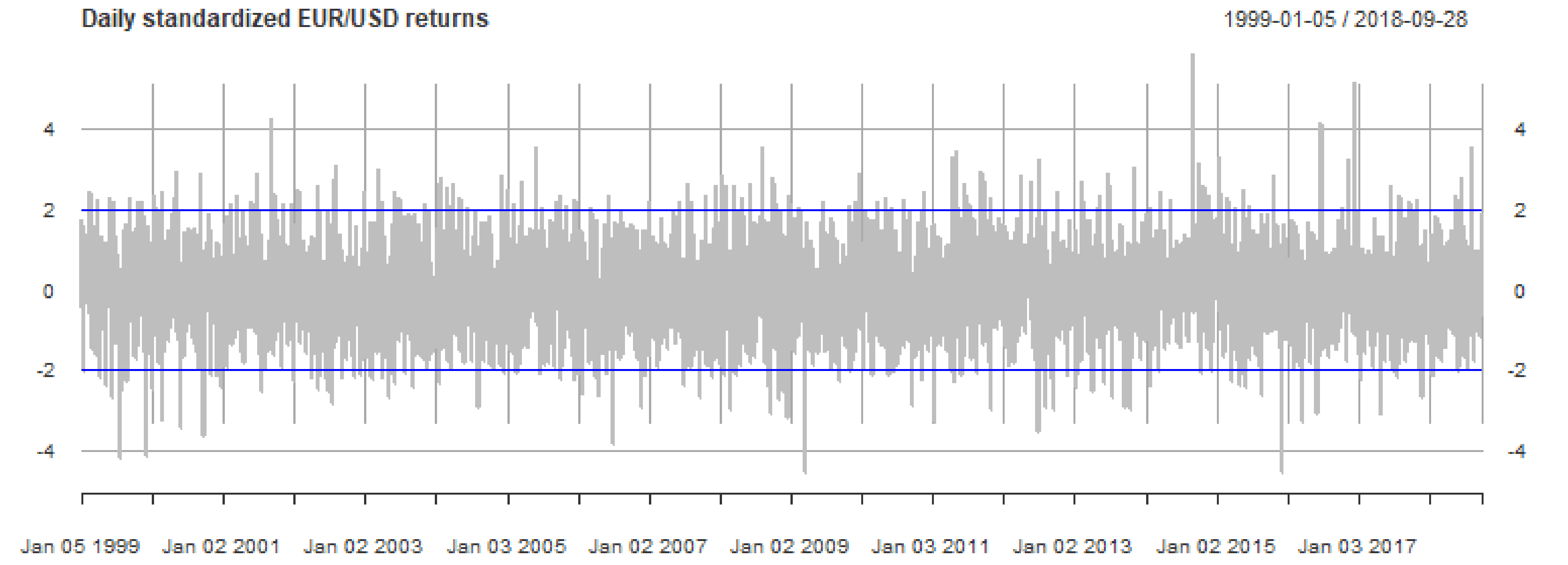
Check 2: Time series plot of standardized returns

- Second check of model validity:
  - time series plot of standardized returns
  - standardized returns should have constant variability


Daily MSFT returns
1999-01-04 / 2017-12-29
0.1
0.0
-0.1
Jan 04 1999
Jan 02 2001
Jan 02 2003
Jan 03 2005
Jan 03 2007
Jan 02 2009
Jan 03 2011
Jan 02 2013
Jan 02 2015
Jan 03 2017


Daily standardized MSFT returns
1999-01-04 / 2017-12-29
5
0
-5
-10
Jan 04 1999
Jan 02 2001
Jan 02 2003
Jan 03 2005
Jan 03 2007
Jan 02 2009
Jan 03 2011
Jan 02 2013
Jan 02 2015
Jan 03 2017


Daily EUR/USD returns
1999-01-05 / 2018-09-28
0.02
0.00
-0.02
-0.04
Jan 05 1999
Jan 02 2001
Jan 02 2003
Jan 03 2005
Jan 02 2007
Jan 02 2009
Jan 03 2011
Jan 02 2013
Jan 02 2015
Jan 03 2017


Daily standardized EUR/USD returns
1999-01-05 / 2018-09-28
4
2
0
-2
-4
Jan 05 1999
Jan 02 2001
Jan 02 2003
Jan 03 2005
Jan 02 2007
Jan 02 2009
Jan 03 2011
Jan 02 2013
Jan 02 2015
Jan 03 2017

Check 3: No predictability in the absolute standardized returns

- Third check of model validity:
  - verify that there is no correlation between the past absolute standardized return and the current absolute standardized return.
  - this means: $Corr(|Z_{t-k}|, |Z_t|) \approx 0$, for $k > 0$
- Why?
  - The magnitude of the absolute standardized return should be constant $\rightarrow$ no correlations in the absolute standardized returns.

Autocorrelations

- Such within time series correlations are called autocorrelations of order k
  - $k = 1$: $Corr(|Z_{t-1}|, |Z_t|)$: Correlation of the current absolute standardized return and its previous value.
  - $k = 2$: $Corr(|Z_{t-2}|, |Z_t|)$: Correlation of the current absolute standardized return and its value two periods ago.
  - ...
- All of these should be 0. Except:
  - $k = 0$: $Corr(|Z_t|, |Z_t|)$: Correlation of the absolute standardized return with itself: equals 1.

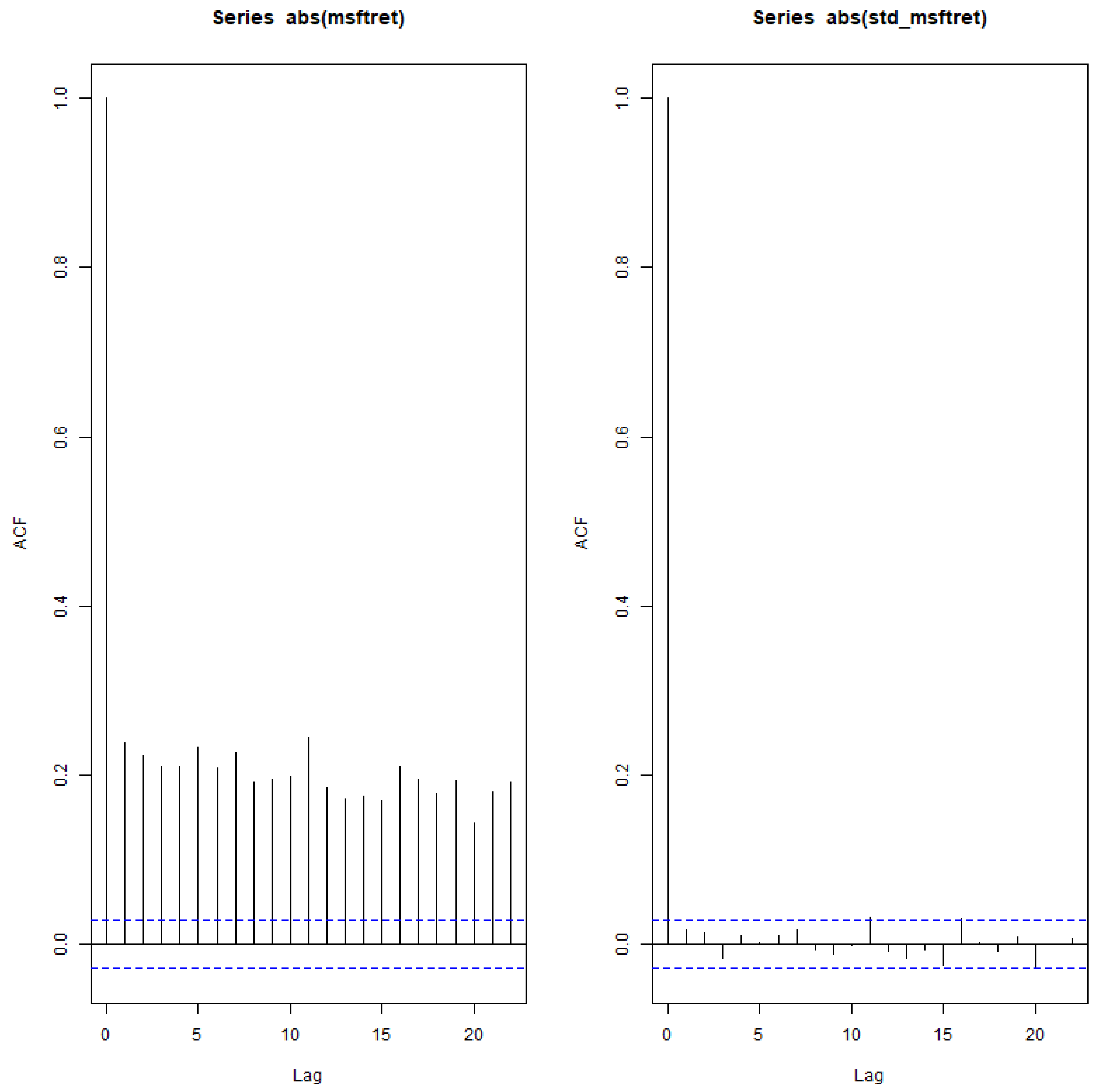
acf()

- In R, we can compute those autocorrelations using the autocorrelation function denoted by `acf()`
  - Input: Time series, maximum order
  - Output: The **correlogram** : plot showing the values of the autocorrelation for different orders $k = 0, 1, ...$

Application to MSFT

```
garchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                        variance.model = list(model = "gjrGARCH"),
                        distribution.model = "sstd")

garchfit <- ugarchfit(data = msftret, spec = garchspec)
stdmsftret <- residuals(garchfit, standardize = TRUE)
```

```
acf(abs(msftret), 22)
acf(abs(stdmsftret), 22)
```


Series abs(msftret)
Series abs(std_msftret)
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20

Check 4: Ljung-Box test

- Fourth check of model validity:
  - Ljung-Box test that the first k autocorrelations in the absolute standardized returns $|Z_t|$ are zero:

$$H_0 : Corr(|Z_t|, |Z_{t-1}|) = Corr(|Z_t|, |Z_{t-2}|) = ... = Corr(|Z_t|, |Z_{t-k}|) = 0$$

  - Similar to a t-test for statistical significance of the estimated parameters, but here we want to have 0 in order to have a good model.

Rule of thumb: p-value less than 5% indicates that the model used is not valid.

Ljung-Box test in R

- In R: function `Box.test()` with 3 arguments:
  - series
  - maximum order for which autocorrelations are zero
  - `type="Ljung-Box"`

Example:

```
Box.test(abs(stdmsftret), 22, type = "Ljung-Box")
```

- Output: p-value
  - Rule of thumb: p-value less than 5% indicates that the model used is not valid.

Box.test using absolute standardized MSFT returns

- Test on absolute standardized returns:

```
Box.test(abs(stdmsftret), 22, type = "Ljung-Box")

    Box-Ljung test
data:  abs(stdmsftret)
X-squared = 25.246, df = 22, p-value = 0.2855
```

Note: p-value is 28.55% > 5%. We cannot reject that:

$$H_0 : Corr(|Z_t|, |Z_{t-1}|) = Corr(|Z_t|, |Z_{t-2}|) = ... = Corr(|Z_t|, |Z_{t-22}|) = 0$$

GARCH MODELS IN R

Let's diagnose the absolute standardized returns.

GARCH MODELS IN R

Use only the data that were available at the time of prediction

Kris Boudt
Professor of finance and econometrics

Volatility prediction by applying sigma() to ugarchforecast object

```
garchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                        variance.model = list(model = "sGARCH"),
                        distribution.model = "norm")

garchfit <- ugarchfit(data = sp500ret,
                      spec = garchspec)

garchforecast <- ugarchforecast(fitORspec = garchfit,
                                n.ahead = 5)

sigma(garchforecast)

      2017-12-29
T+1  0.005034754
T+2  0.005127582
T+3  0.005217770
T+4  0.005305465
T+5  0.005390797
```

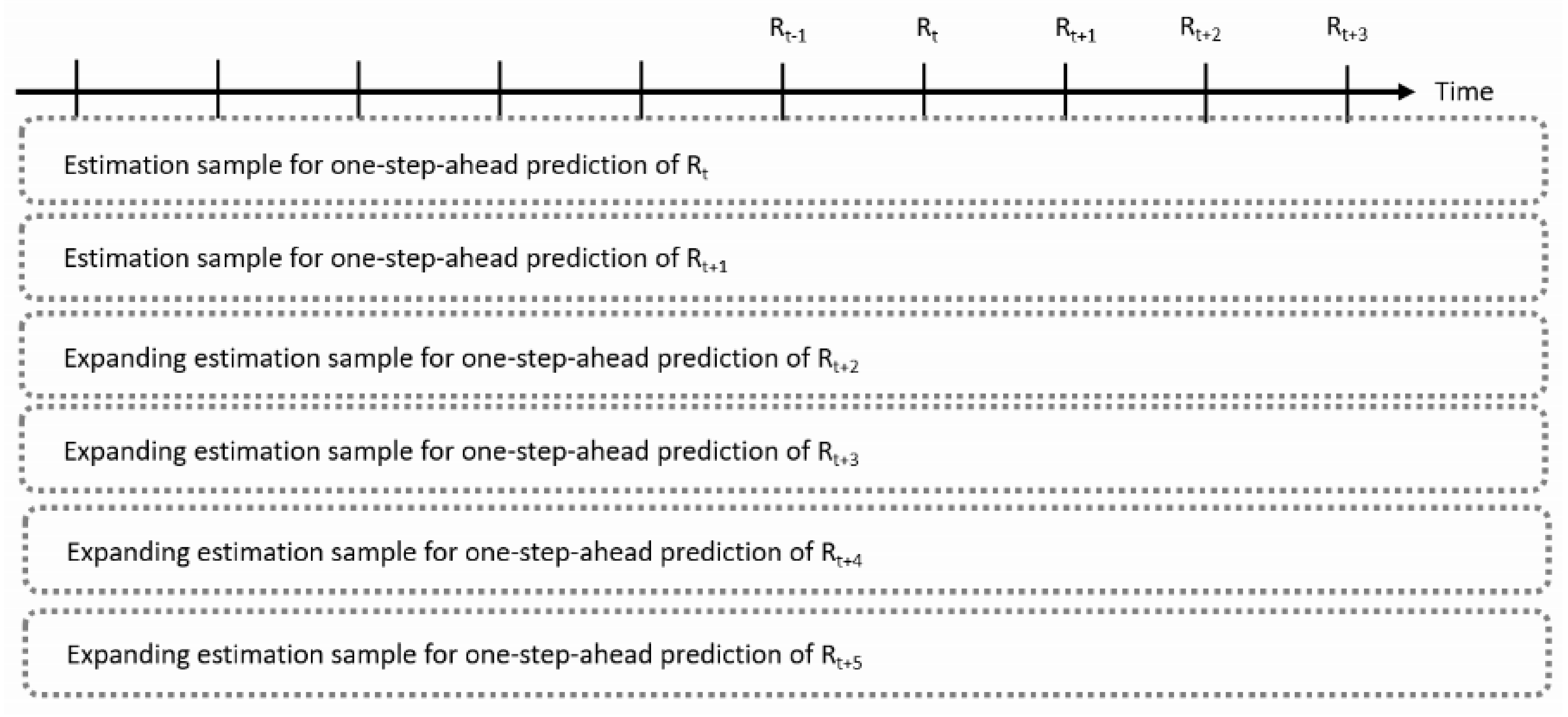
Volatility estimation by applying sigma() to ugarchfit object (i)

```
head(sigma(garchfit), 5)

                 [,1]
1989-01-04 0.01099465
1989-01-05 0.01129167
1989-01-06 0.01084294
1989-01-09 0.01042072
1989-01-10 0.01000925
```

```
tail(sigma(garchfit), 5)

                  [,1]
2017-12-22 0.005252819
2017-12-26 0.005142349
2017-12-27 0.005051926
2017-12-28 0.004947569
2017-12-29 0.004862908
```

Volatility estimation by applying sigma() to ugarchfit object (ii)

- Look ahead bias: Future returns are used to make the volatility estimate.

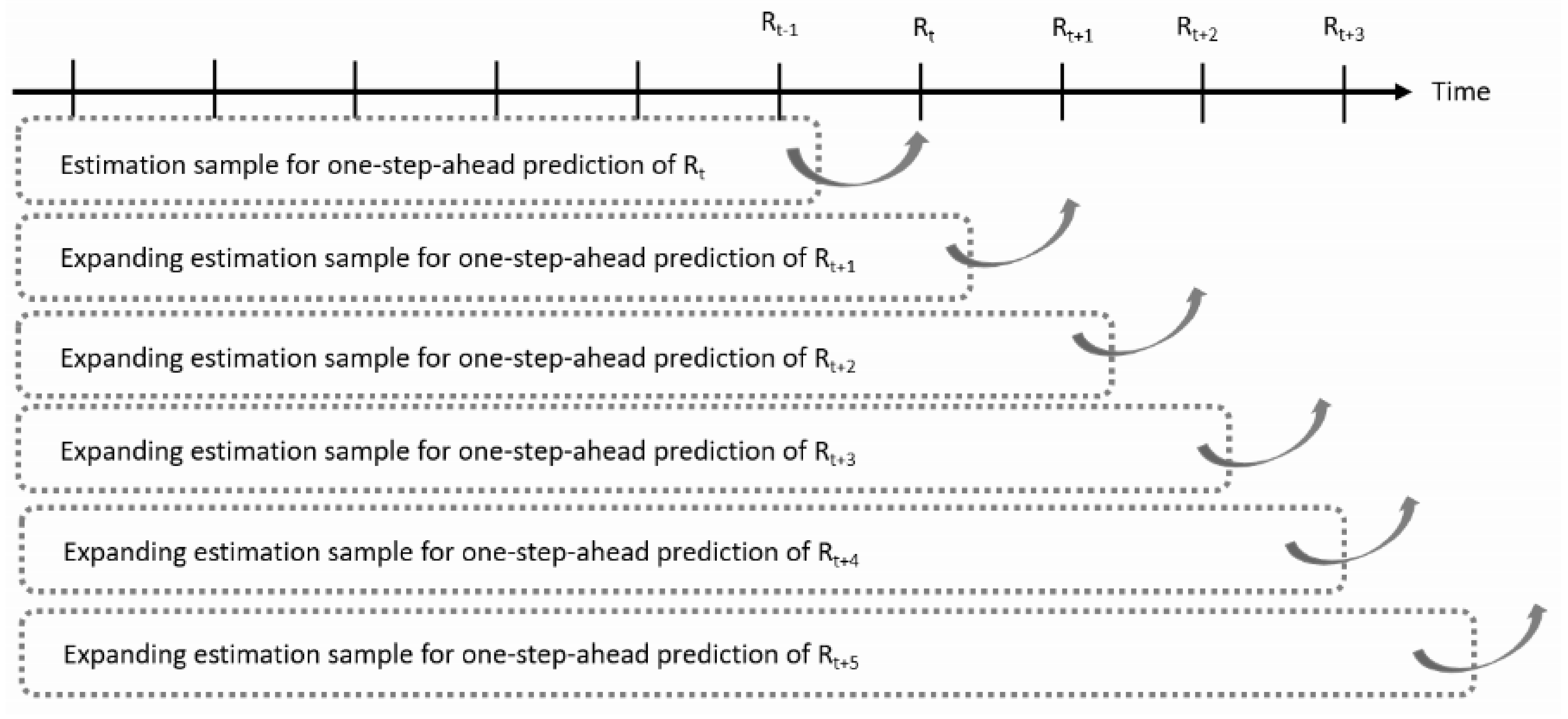

R_{t-1}
R_t
R_{t+1}
R_{t+2}
R_{t+3}
Time
Estimation sample for one-step-ahead prediction of R_t
Estimation sample for one-step-ahead prediction of R_{t+1}
Expanding estimation sample for one-step-ahead prediction of R_{t+2}
Expanding estimation sample for one-step-ahead prediction of R_{t+3}
Expanding estimation sample for one-step-ahead prediction of R_{t+4}
Expanding estimation sample for one-step-ahead prediction of R_{t+5}

Solution to avoid look-ahead bias: Rolling estimation

- Program a `for` loop: Iterate over the prediction times and use `ugarchfit()` to estimate the model and `ugarchforecast()` to make the volatility forecast
- Built-in function `ugarchroll()` in the `rugarch` package
- Options:
  - Length of the estimation sample
  - Frequency of model estimation

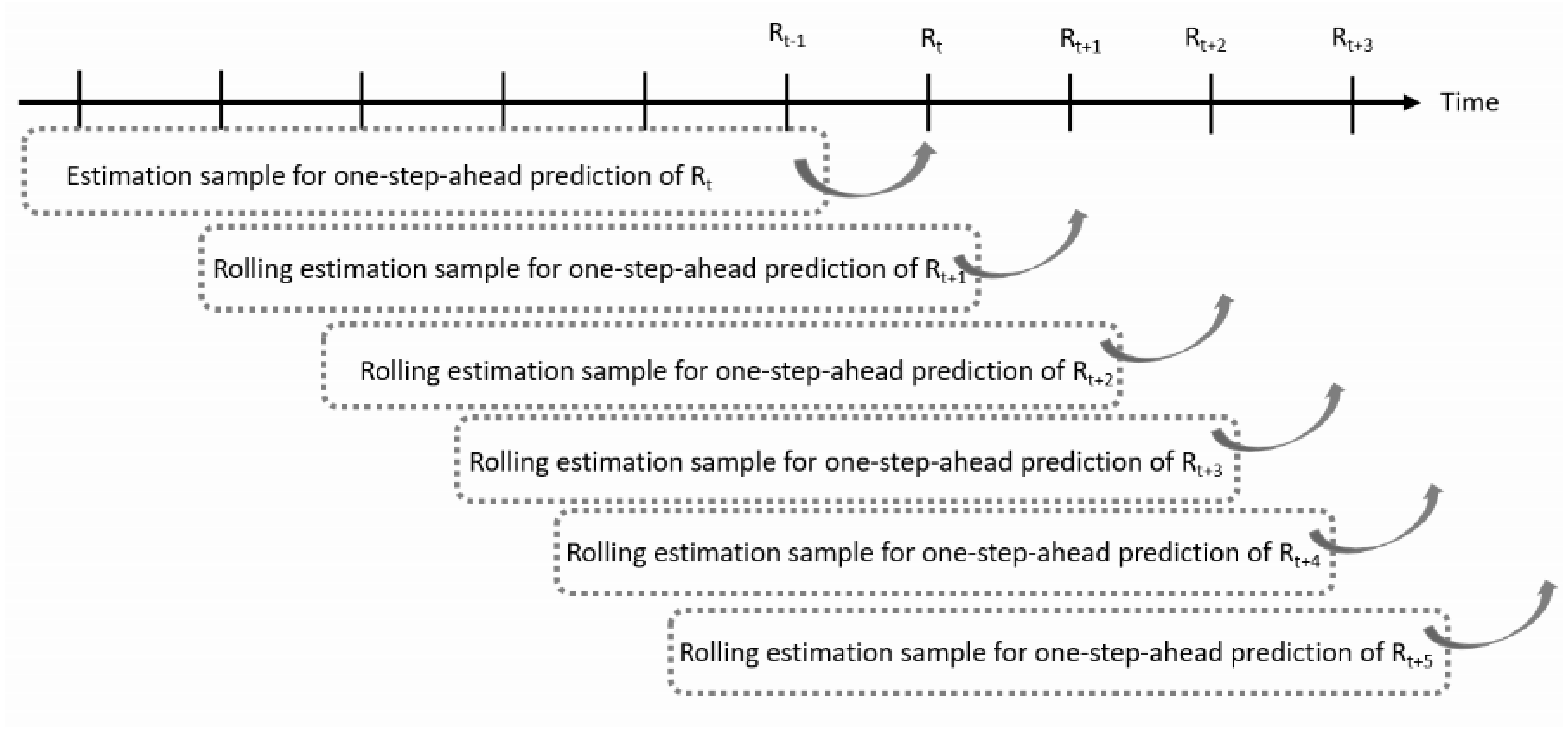
Expanding window estimation

- Use all available returns at the time of prediction


R_{t-1}
R_t
R_{t+1}
R_{t+2}
R_{t+3}
Time
Estimation sample for one-step-ahead prediction of R_t
Expanding estimation sample for one-step-ahead prediction of R_{t+1}
Expanding estimation sample for one-step-ahead prediction of R_{t+2}
Expanding estimation sample for one-step-ahead prediction of R_{t+3}
Expanding estimation sample for one-step-ahead prediction of R_{t+4}
Expanding estimation sample for one-step-ahead prediction of R_{t+5}

Moving window estimation

- Only use a fixed number of the most return observations available at the time of prediction

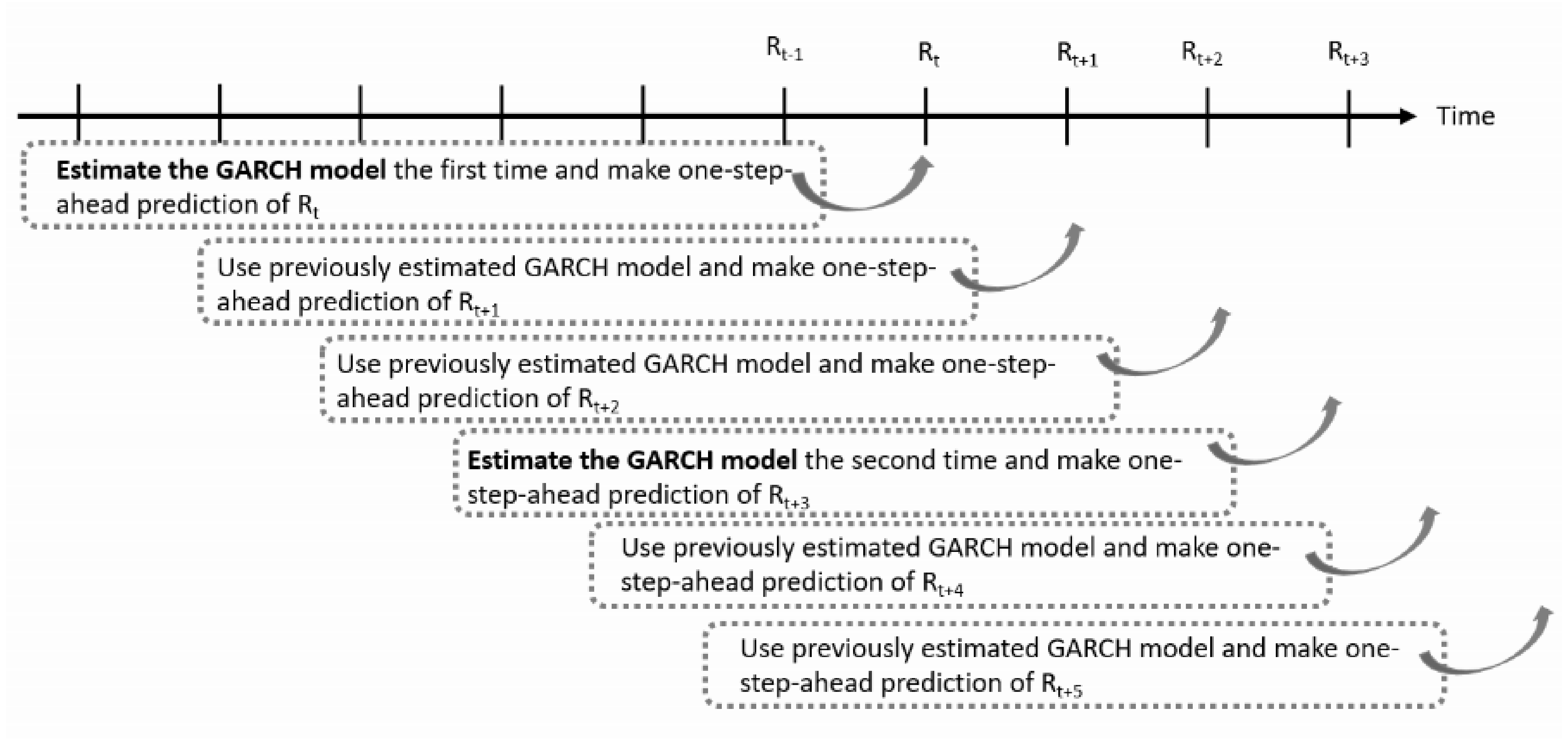

R_{t-1}
R_t
R_{t+1}
R_{t+2}
R_{t+3}
Time
Estimation sample for one-step-ahead prediction of R_t
Rolling estimation sample for one-step-ahead prediction of R_{t+1}
Rolling estimation sample for one-step-ahead prediction of R_{t+2}
Rolling estimation sample for one-step-ahead prediction of R_{t+3}
Rolling estimation sample for one-step-ahead prediction of R_{t+4}
Rolling estimation sample for one-step-ahead prediction of R_{t+5}

Properties of rolling window estimation

- Rolling window lets you adapt to changes in the model parameters
- Computational cost of `ugarchroll()` is the loop across prediction times:
  - Can be reduced by re-estimate the model at a lower frequency than the frequency of prediction

Rolling and re-estimation

- Reduce the computational cost by estimating the model every K observations


R_{t-1}
R_t
R_{t+1}
R_{t+2}
R_{t+3}
Time
Estimate the GARCH model the first time and make one-step-ahead prediction of R_t
Use previously estimated GARCH model and make one-step-ahead prediction of R_{t+1}
Use previously estimated GARCH model and make one-step-ahead prediction of R_{t+2}
Estimate the GARCH model the second time and make one-step-ahead prediction of R_{t+3}
Use previously estimated GARCH model and make one-step-ahead prediction of R_{t+4}
Use previously estimated GARCH model and make one-step-ahead prediction of R_{t+5}

Function ugarchroll

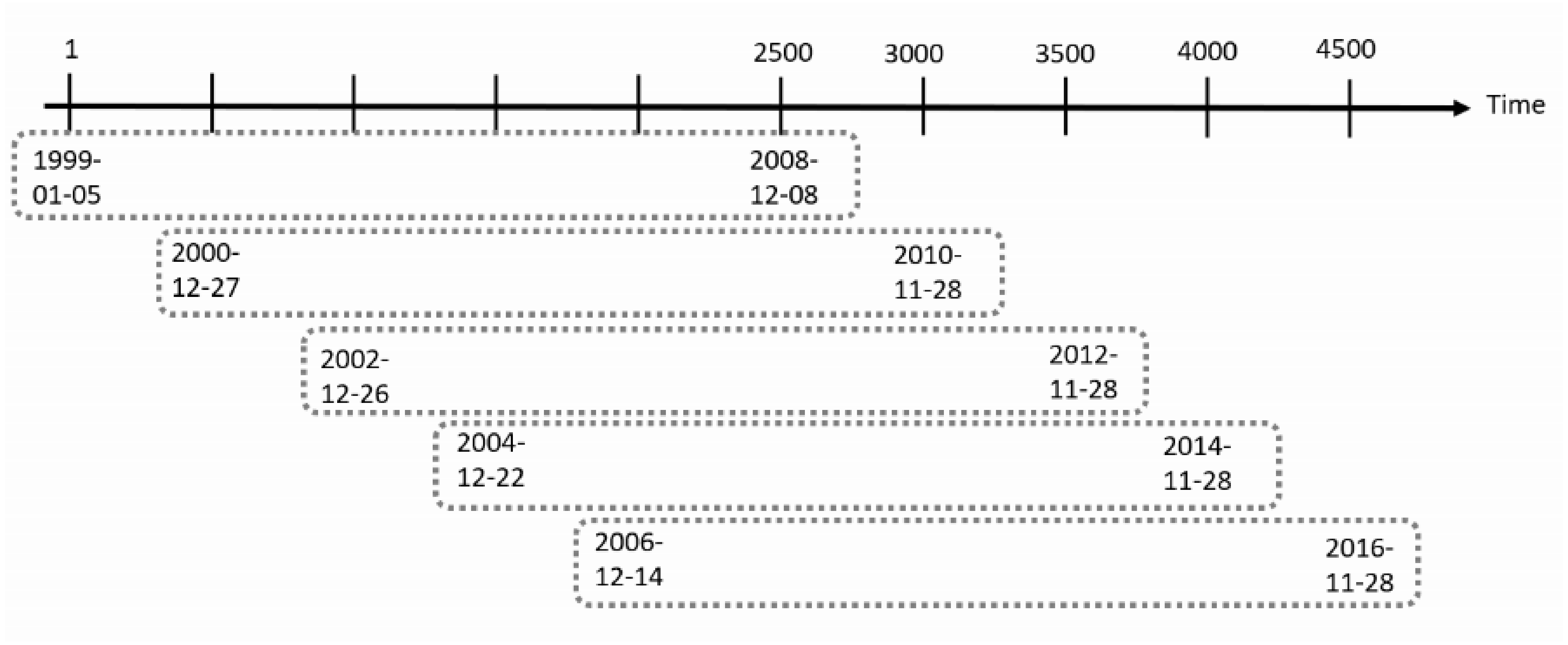
```
tgarchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                        variance.model = list(model = "sGARCH"),
                        distribution.model = "std")

garchroll <- ugarchroll(tgarchspec, data = EURUSDret, n.start = 2500,
                       refit.window = "moving", refit.every = 500)
```

- Arguments to specify:
    1. GARCH specification used
    2. `data` : return data to use
    3. `n.start`: the size of the initial estimation sample
    4. `refit.window`: how to change that sample through time: "moving" or "expanding
    5. `refit.every`: how often to re-estimate the model

Example on the Jan 1999-Dec 2018 daily EUR/USD returns

- For the 4961 EUR/USD returns with 4961 observations, starting on 1999-01-05

  and using a moving estimation window of 2500 observations:



Output of changing parameters

- Method `coef()`

```
coef(tgarchroll)
```

produces a list with the estimated coefficients for each estimation

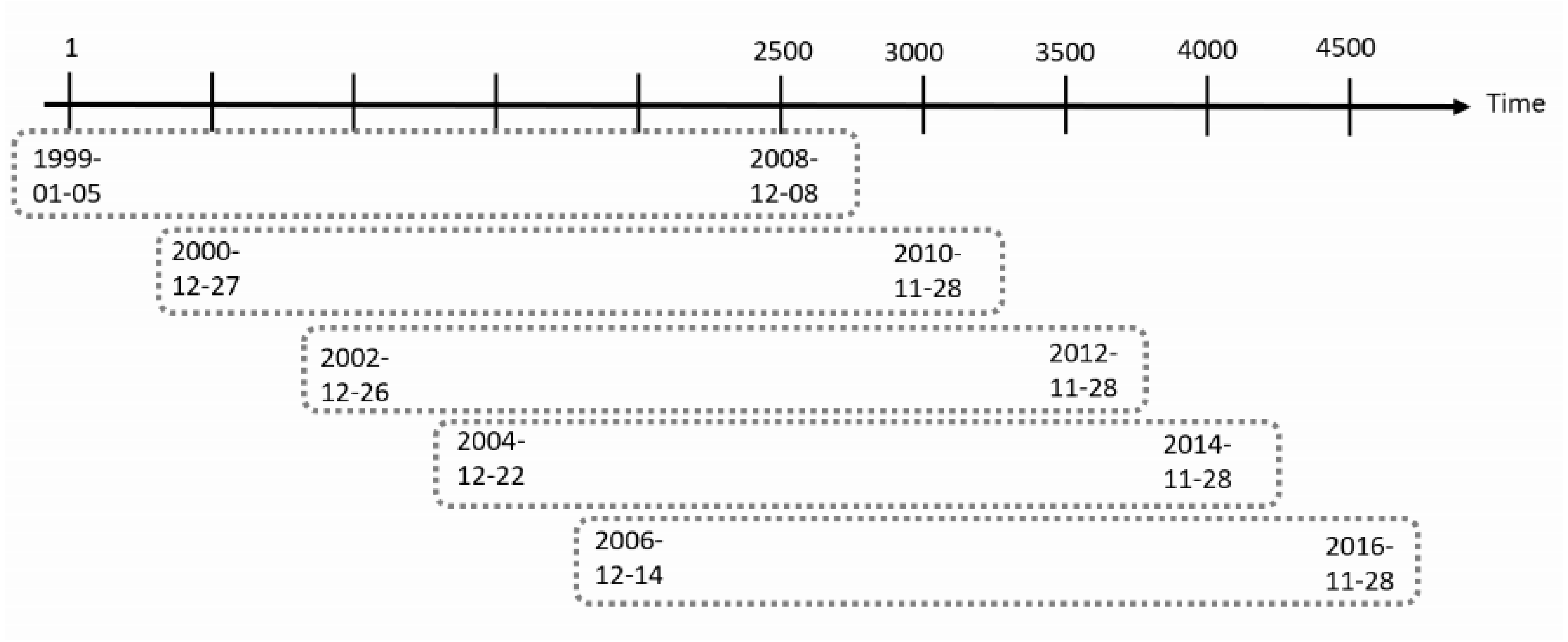
```
coef(garchroll)[[1]]

$`index`
"2008-12-08"

$coef

             Estimate   Std. Error     t value  Pr(>|t|)
mu      -1.480000e-04 1.330915e-04 -1.11201737 0.2661307
ar1     -2.953484e-03 1.985344e-02 -0.14876432 0.8817396
omega    7.498928e-08 5.587618e-06  0.01342062 0.9892922
alpha1   2.805079e-02 6.401139e-02  0.43821564 0.6612300
beta1    9.709400e-01 6.080197e-02 15.96889122 0.0000000
shape    1.098068e+01 2.609293e+01  0.42082981 0.6738794
```

Changes between first and fifth estimation window

```
coef(garchroll)[[1]] # 2008-12-08

$coef

            Estimate   Std. Error     t value   Pr(>|t|)
mu     -1.480000e-04 1.330915e-04 -1.11201737 0.2661307
ar1    -2.953484e-03 1.985344e-02 -0.14876432 0.8817396
omega   7.498928e-08 5.587618e-06  0.01342062 0.9892922
alpha1  2.805079e-02 6.401139e-02  0.43821564 0.6612300
beta1   9.709400e-01 6.080197e-02 15.96889122 0.0000000
shape   1.098068e+01 2.609293e+01  0.42082981 0.6738794

coef(garchroll)[[5]] # 2016-11-28

$coef

            Estimate   Std. Error      t value     Pr(>|t|)
mu      2.339788e-05 2.637007e-04  0.088728907 9.292974e-01
ar1    -9.244175e-04 3.980756e-02 -0.023222161 9.814731e-01
omega   8.982527e-08 2.639680e-05  0.003402885 9.972849e-01
alpha1  4.149787e-02 2.550381e-01  0.162712424 8.707449e-01
beta1   9.573885e-01 2.195782e-01  4.360125676 1.299878e-05
shape   7.980116e+00 5.361855e+01  0.148831267 8.816868e-01
```

What is the rolling predicted mean and volatility?

- For the EUR/USD returns with `n.start = 2500`, the first prediction is for observation 2501, namely 2008-12-09:



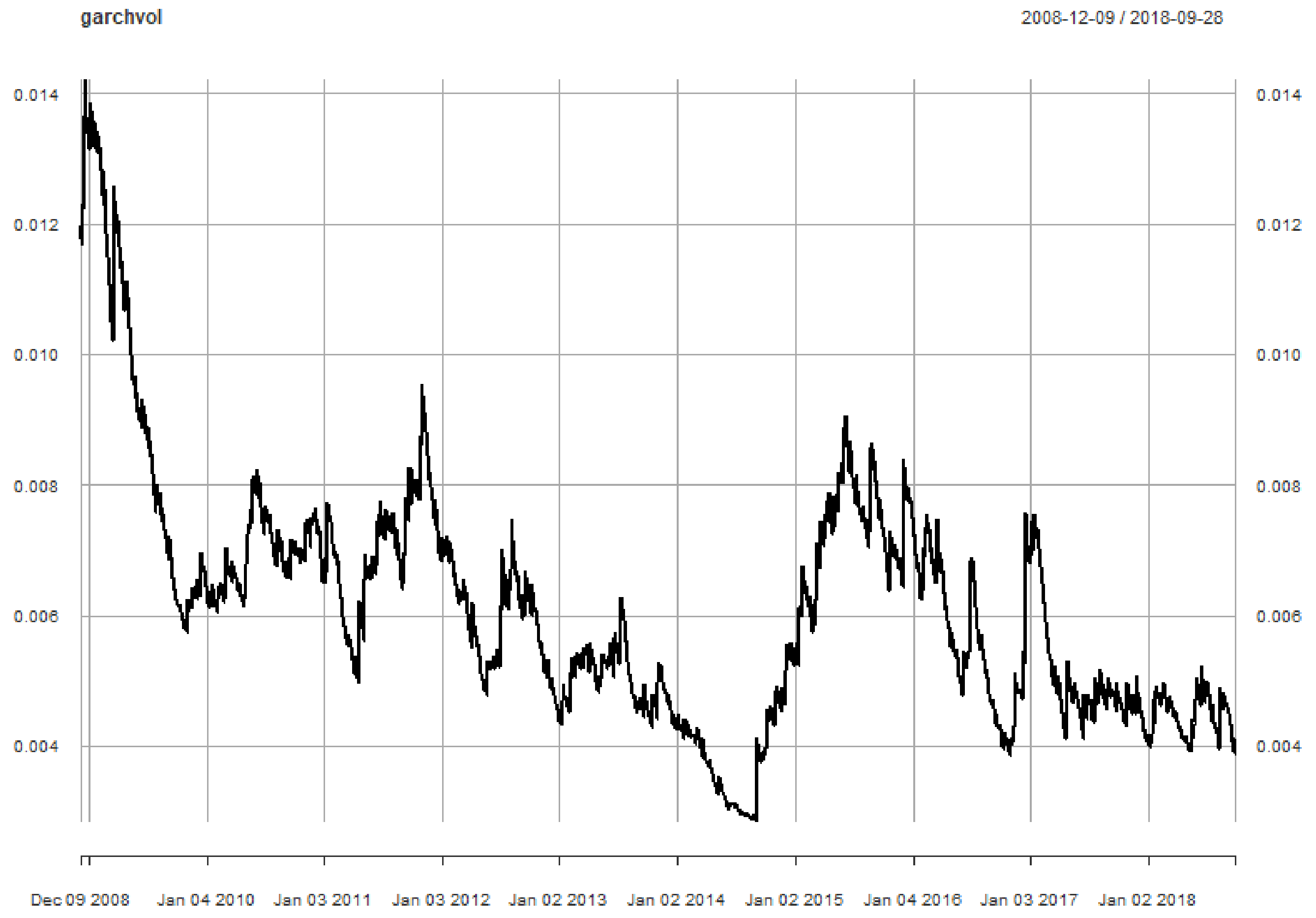
Method as.data.frame()

```
preds <- as.data.frame(garchroll)

head(preds)

                      Mu      Sigma Skew    Shape Shape(GIG)      Realized
2008-12-09 -8.271288e-05 0.01196917    0 10.98068          0  0.0003864884
2008-12-10 -1.495786e-04 0.01179742    0 10.98068          0 -0.0066799754
2008-12-11 -1.287079e-04 0.01167929    0 10.98068          0 -0.0203099142
2008-12-12 -8.845214e-05 0.01199756    0 10.98068          0 -0.0041201588
2008-12-15 -1.362683e-04 0.01184438    0 10.98068          0 -0.0230532787
2008-12-16 -8.034966e-05 0.01228900    0 10.98068          0 -0.0105720492
```

- Note:
  - `preds$Mu`: series of predicted mean values
  - `preds$Sigma`: series of predicted volatility values

Predicted volatilities

```
garchvol <- xts(preds$Sigma, order.by = as.Date(rownames(preds)))
plot(garchvol)
```



Accuracy of rolling predictions

```
preds <- as.data.frame(garchroll)

head(preds)

                      Mu      Sigma Skew    Shape Shape(GIG)      Realized
2008-12-09 -8.271288e-05 0.01196917    0 10.98068          0  0.0003864884
2008-12-10 -1.495786e-04 0.01179742    0 10.98068          0 -0.0066799754
2008-12-11 -1.287079e-04 0.01167929    0 10.98068          0 -0.0203099142
2008-12-12 -8.845214e-05 0.01199756    0 10.98068          0 -0.0041201588
2008-12-15 -1.362683e-04 0.01184438    0 10.98068          0 -0.0230532787
2008-12-16 -8.034966e-05 0.01228900    0 10.98068          0 -0.0105720492
```

- Evaluate accuracy of `preds$Mu` and `preds$Sigma` by comparing with

  `preds$Realized`

Mean squared prediction error for the mean

```
# Prediction error for the mean
e  <- preds$Realized - preds$Mu

mean(e^2)

3.867998e-05
```

Mean squared prediction error for the variance

```
# Prediction error for the mean
e  <- preds$Realized - preds$Mu

# Prediction error for the variance
d  <- e^2 - preds$Sigma^2

mean(d^2)

6.974161e-09
```

Compare two models

- Standard GARCH with student t distribution

```
tgarchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                         variance.model = list(model = "sGARCH"),
                         distribution.model = "std")

garchroll <- ugarchroll(tgarchspec, data = EURUSDret, n.start = 2500,
                        refit.window = "moving", refit.every = 500)
```

- versus GJR GARCH with skewed student t distribution

```
# Specification
gjrgarchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                           variance.model = list(model = "gjrGARCH"),
                           distribution.model = "sstd")
# Estimation
gjrgarchroll <- ugarchroll(gjrgarchspec, data = EURUSDret, n.start = 2500,
                           refit.window = "moving", refit.every = 500)
```

Comparison of prediction accuracny

- Standard GARCH with student t distribution

```
preds <- as.data.frame(garchroll)
e  <- preds$Realized - preds$Mu
d  <- e^2 - preds$Sigma^2

mean(d^2)

 6.974161e-09
`
```

- versus GJR GARCH with skewed student t distribution

```
gjrpreds <- as.data.frame(gjrgarchroll)
e  <- gjrpreds$Realized - gjrpreds$Mu
d  <- e^2 - gjrpreds$Sigma^2

mean(d^2)

 6.965095e-09
```

GARCH MODELS IN R

The proof of the pudding is in the eating