



GARCH MODELS IN R

**There are old traders and
there are bold traders, but...**

Kris Boudt

Professor of finance and econometrics



About the instructor

- Kris Boudt
 - PhD in financial risk forecasting
 - Use GARCH models to win by not losing (much)
- R package rugarch of Alexios Ghalanos.





Calculating returns

- Relative financial gains and losses, expressed in terms of returns

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

- **Function** CalculateReturns **in** PerformanceAnalytics

```
# Example in R for daily S&P 500 prices (xts object)
library(PerformanceAnalytics)
SP500returns <- CalculateReturns(SP500prices)
```

Daily S&P 500 returns

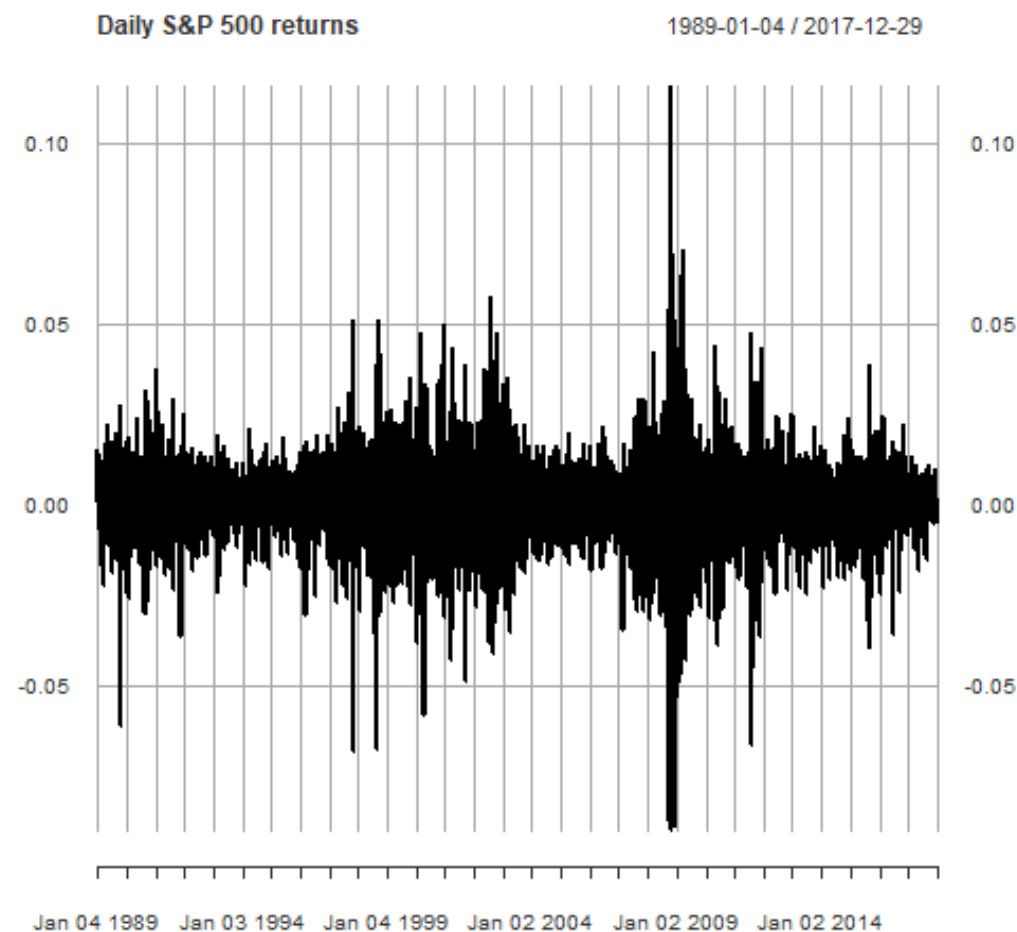
Properties of daily returns:

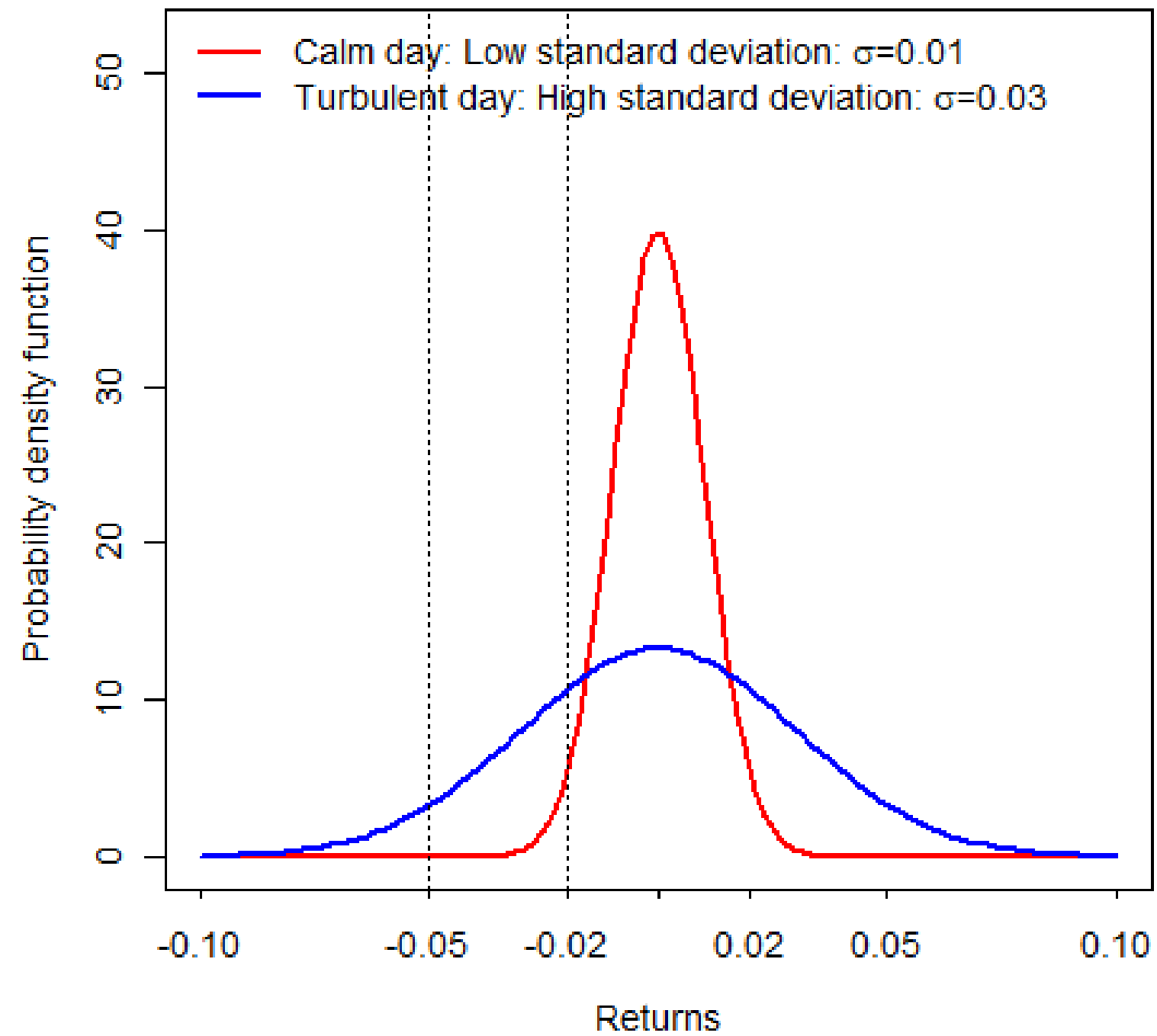
- The average return is zero
- Return variability changes through time

Standard deviation = measure of return variability.

Synonym: Return **volatility**.

Greek letter σ_t .







How to estimate return volatility

- Function `sd()` computes the standard deviation:

```
# Compute daily standard deviation  
> sd(sp500ret)  
[1] 0.01099357
```

- Corresponding formula for T daily returns:

$$\hat{\sigma} = \sqrt{\frac{1}{T-1} \sum_{t=1}^T (R_t - \hat{\mu})^2},$$

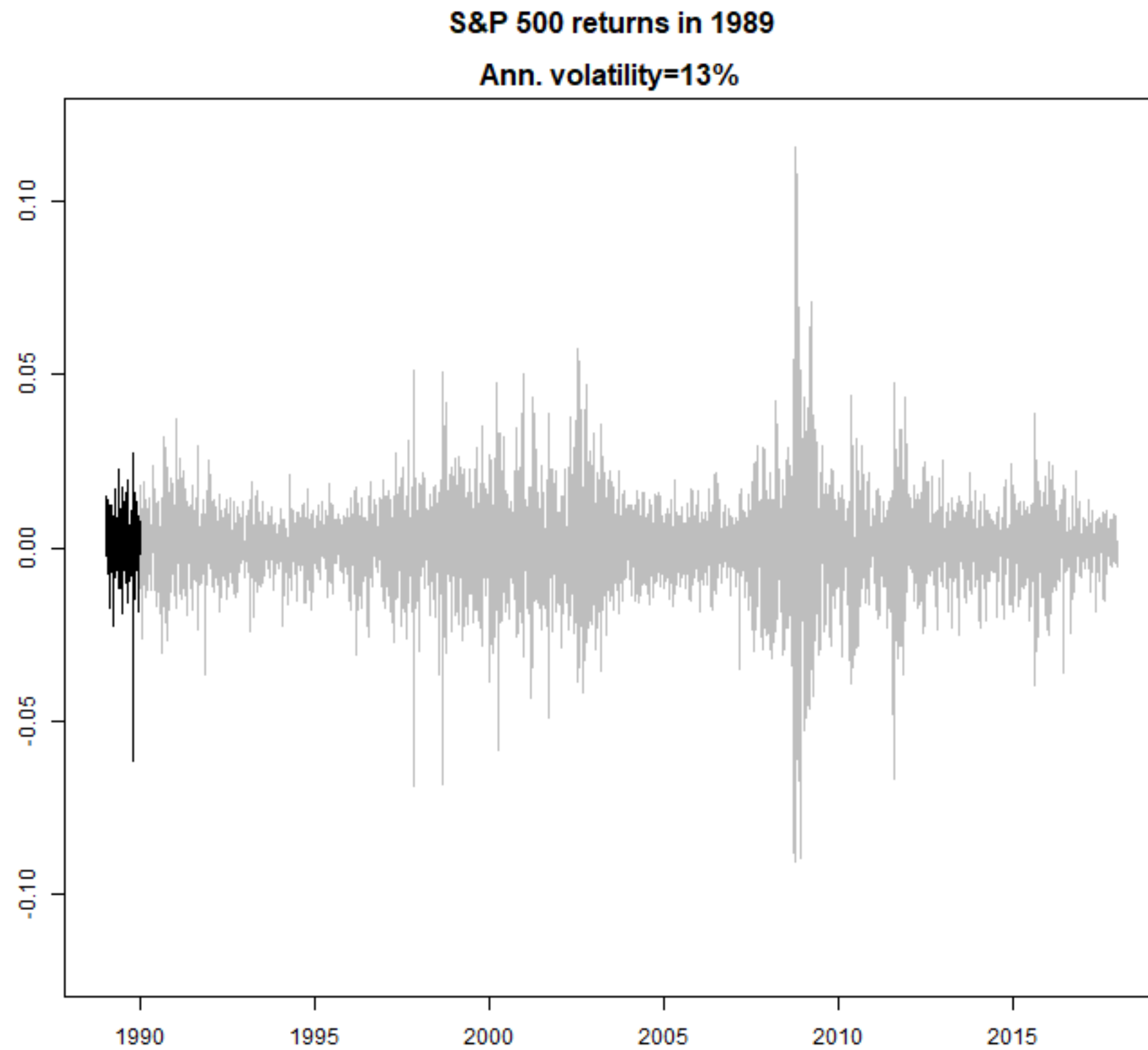
where $\hat{\mu}$ is the mean return.



Annualized volatility

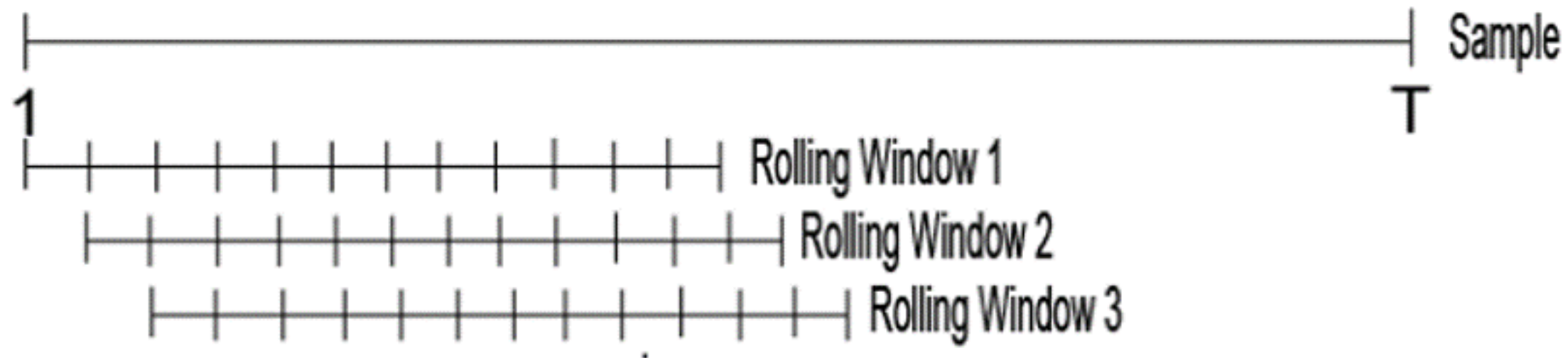
- `sd(sp500ret)` is daily volatility
- Annualized volatility = $\sqrt{252} \times$ daily volatility

```
# Compute annualized standard deviation  
> sqrt(252)*sd(sp500ret)  
[1] 0.1745175
```

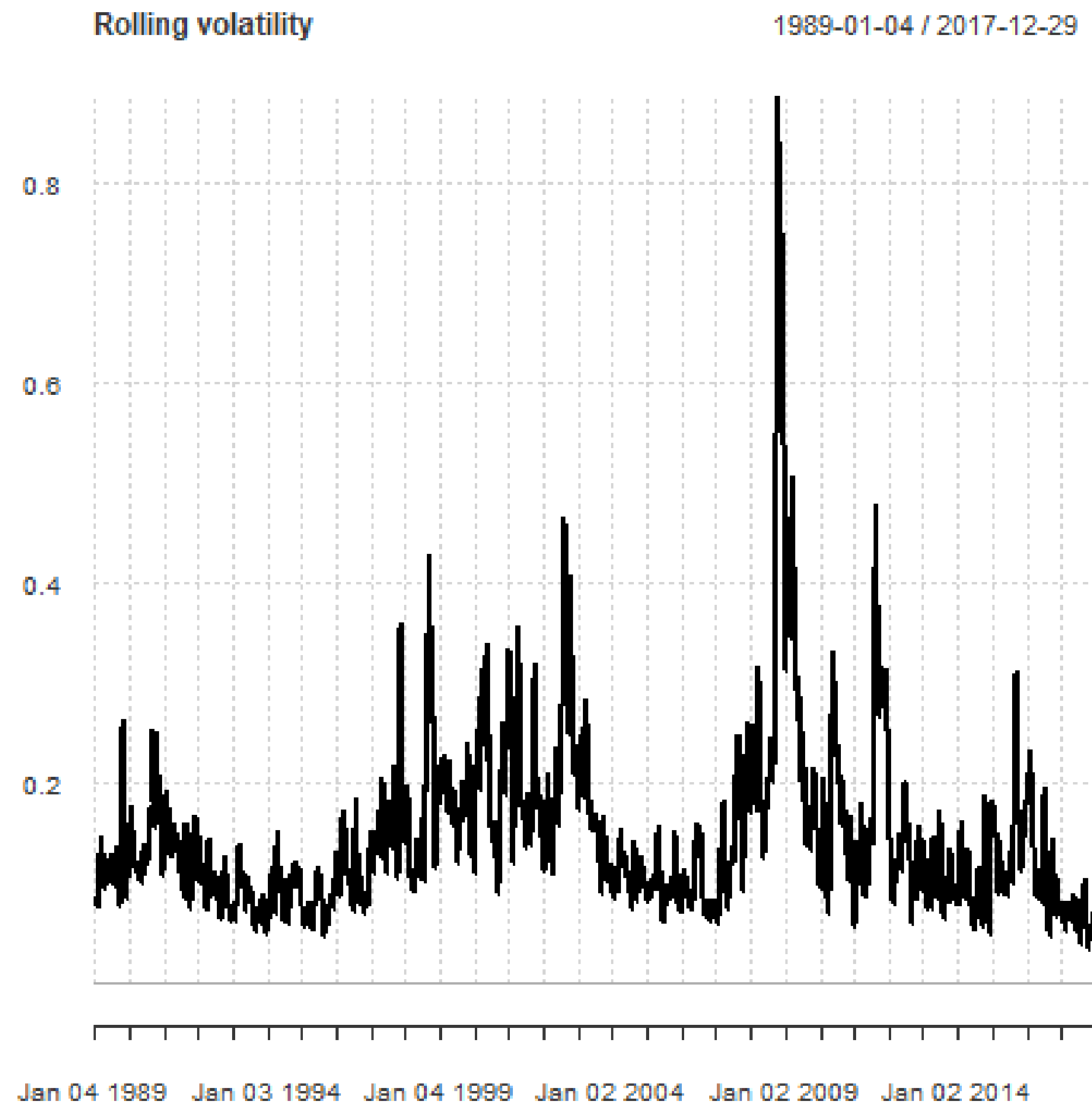


Rolling volatility estimation

- Rolling estimation windows :



- Window width? Multiple of 22 (trading days).





About GARCH models in R

- Estimation of σ_t requires time series models, like GARCH.



GARCH MODELS IN R

**Let's refresh the basics of
computing rolling standard
deviations in R**



GARCH MODELS IN R

GARCH models: The way forward

Kris Boudt

Professor of finance and econometrics

Inventors of GARCH models

Robert Engle

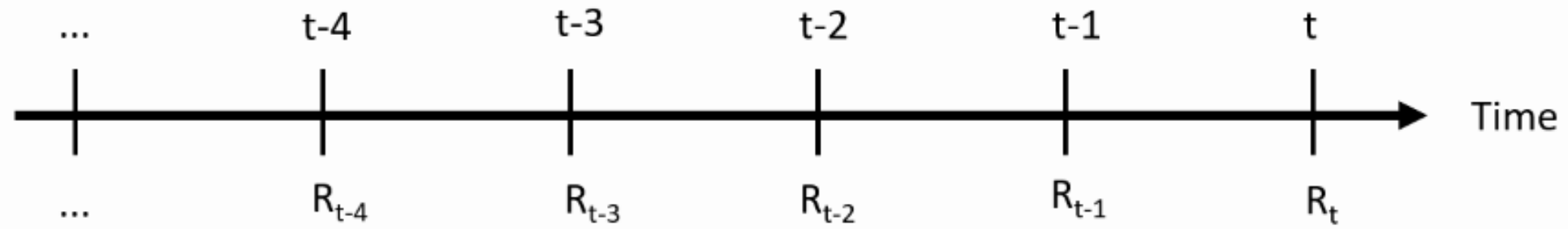


Tim Bollerslev



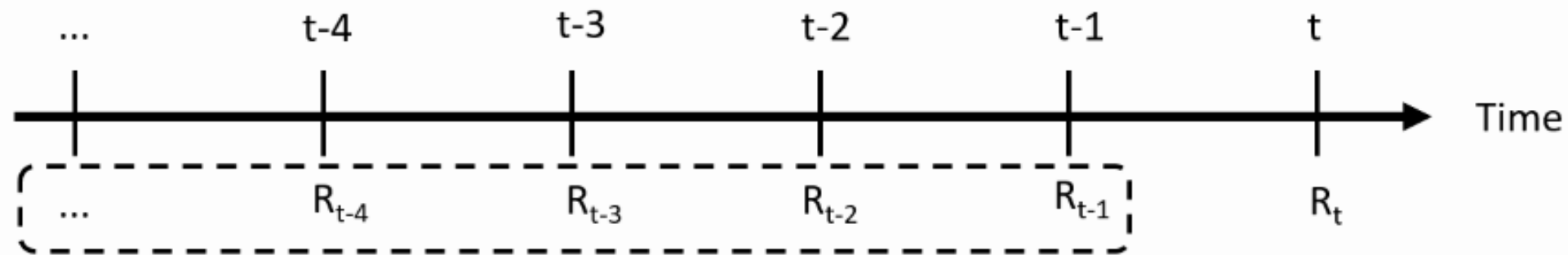
Notation (i)

- Input: Time series of returns



Notation (ii)

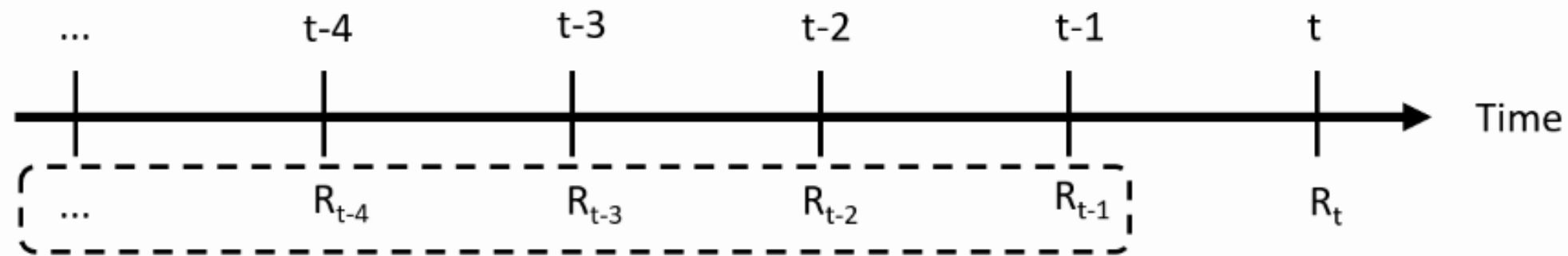
- At time $t-1$, you make the prediction about the the future return R_t , using the *information set* available at time $t - 1$:



I_{t-1} = Information set available at the time of prediction (t-1)

Notation (iii)

- Predicting the **mean return**: what is the best possible prediction of the actual return?



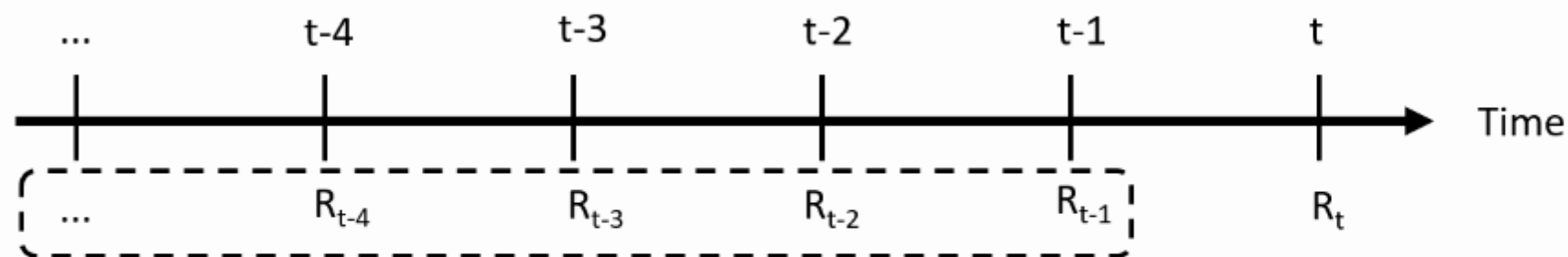
I_{t-1} = Information set available at the time of prediction (t-1)

$$\mu_t = E[R_t \mid I_{t-1}]$$

Prediction error: $e_t = R_t - \mu_t$

Notation (iv)

- We then predict the **variance**: how far off the return can be from its mean?



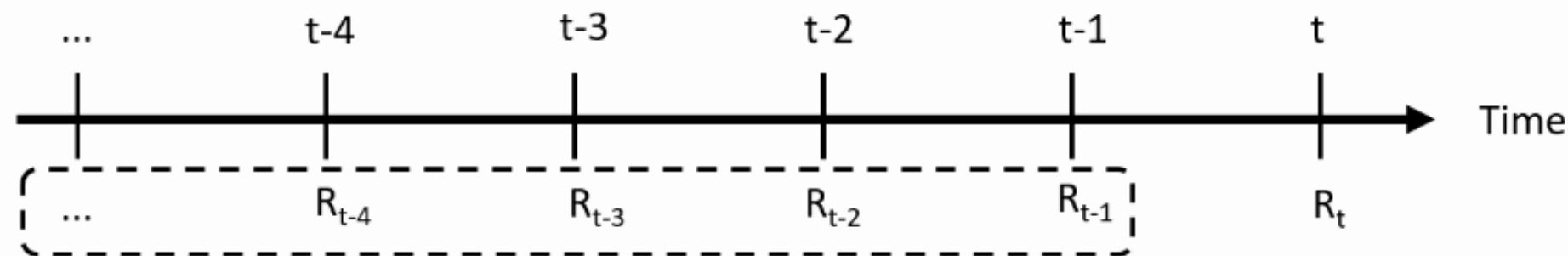
I_{t-1} = Information set available at the time of prediction (t-1)

$$\begin{aligned}\sigma_t^2 &= \text{var}(R_t \mid I_{t-1}) \\ &= E[(R_t - \mu_t)^2 \mid I_{t-1}] \\ &= E[e_t^2 \mid I_{t-1}]\end{aligned}$$

$$\sigma_t = \sqrt{\sigma_t^2}$$

From theory to practice: Models for the mean

- We need an equation that maps the past returns into a prediction of the mean

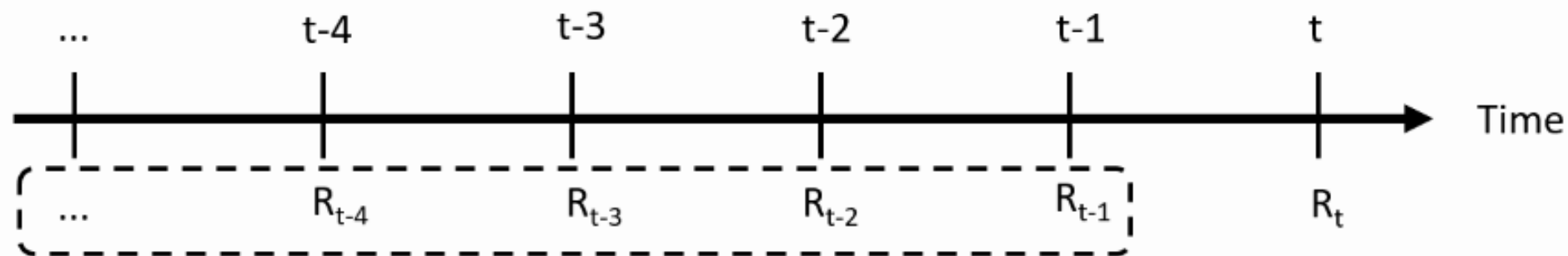


Rolling mean model: $\mu_t = \frac{1}{M} \sum_{i=1}^M R_{t-i}$

For AR(MA) models for the mean, see Datacamp course on [time series analysis](#).

From theory to practice: Models for the variance

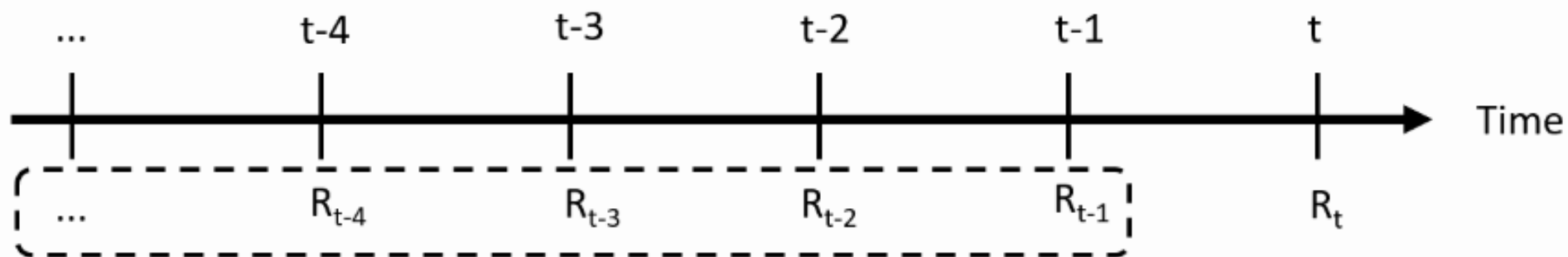
- We need an equation that maps the past returns into predictions of the variance



Rolling variance model: $\sigma_t^2 = \frac{1}{M} \sum_{i=1}^M e_{t-i}^2$

ARCH(p) model: Autoregressive Conditional Heteroscedasticity

- We need an equation that maps the past returns into predictions of the variance

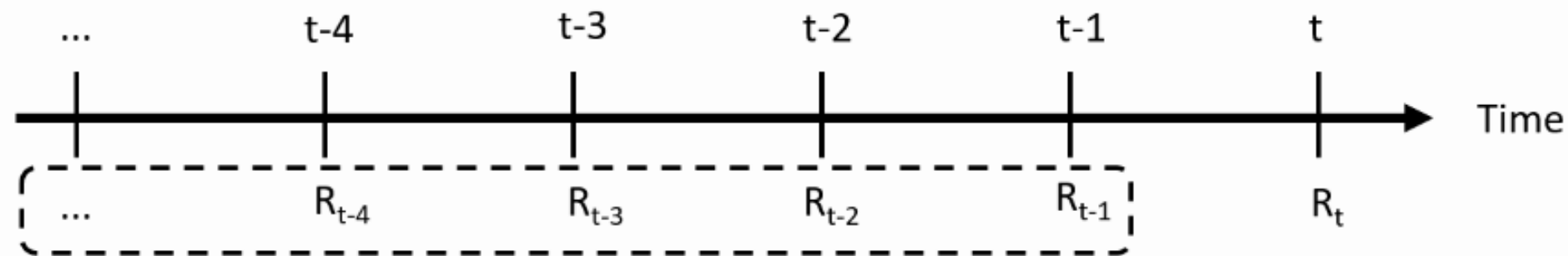


Rolling variance model: $\sigma_t^2 = \frac{1}{M} \sum_{i=1}^M e_{t-i}^2$

ARCH(p) model: $\sigma_t^2 = \omega + \sum_{i=1}^p \alpha_i e_{t-i}^2$

GARCH(1,1) model: Generalized ARCH

- We need an equation that maps the past returns into predictions of the variance



I_{t-1} = Information set available at the time of prediction (t-1)

$$\text{ARCH}(p) \text{ model: } \sigma_t^2 = \omega + \sum_{i=1}^p \alpha_i e_{t-i}^2$$

$$\text{GARCH}(1,1) \text{ model: } \sigma_t^2 = \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2$$



Parameter restrictions

To make the GARCH process realistic, we need that:

1. ω , α and β are > 0 : this ensures that $\sigma_t^2 > 0$ at all times.
2. $\alpha + \beta < 1$: this ensures that the predicted variance σ_t^2 always returns to the long run variance:
 - The variance is therefore "mean-reverting"
 - The long run variance equals $\frac{\omega}{1-\alpha-\beta}$

R implementation - Specify the inputs

- Let's familiarize ourselves with the GARCH equations using R code:

$$\sigma_t^2 = \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2$$

```
# Set parameter values
alpha <- 0.1
beta <- 0.8
omega <- var(sp500ret) * (1-alpha-beta)
# Then: var(sp500ret) = omega / (1-alpha-beta)
```

```
# Set series of prediction error
e <- sp500ret - mean(sp500ret) # Constant mean
e2 <- e^2
```



R implementation - compute predicted variances

```
# We predict for each observation its variance.
nobs <- length(sp500ret)
predvar <- rep(NA, nobs)
```

```
# Initialize the process at the sample variance
predvar[1] <- var(sp500ret)
```

```
# Loop starting at 2 because of the lagged predictor
for (t in 2:nobs){
  # GARCH(1,1) equation
  predvar[t] <- omega + alpha * e2[t - 1] + beta * predvar[t-1]
}
```

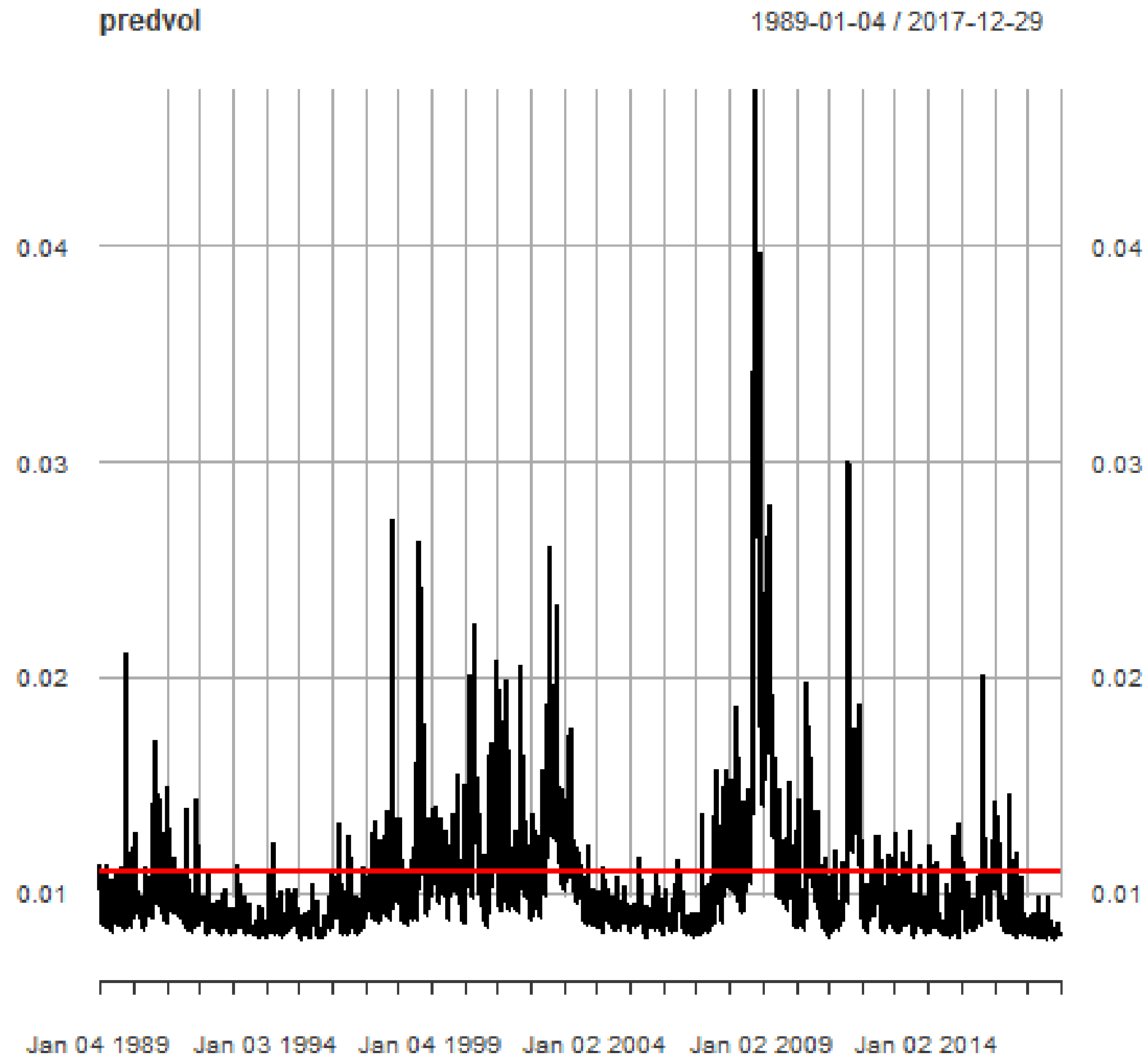


R implementation - Plot of GARCH volatilities

```
# Volatility is sqrt of predicted variance
predvol <- sqrt(predvar)
predvol <- xts(predvol, order.by = time(sp500ret))
```

```
# We compare with the unconditional volatility
uncvol <- sqrt(omega / (1 - alpha-beta))
uncvol <- xts(rep(uncvol, nobs), order.by = time(sp500ret))
```

```
# Plot
plot(predvol)
lines(uncvol, col = "red", lwd = 2)
```





GARCH MODELS IN R

Let's practice!



GARCH MODELS IN R

Alpha - Beta - Sigma: The rugarch package

Kris Boudt

Professor of finance and econometrics



The normal GARCH(1,1) model with constant mean

- The normal GARCH model

$$\begin{aligned}R_t &= \mu + e_t \\ e_t &\sim N(0, \sigma_t^2) \\ \sigma_t^2 &= \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2\end{aligned}$$

- Four parameters: $\mu, \omega, \alpha, \beta$.
- Estimation by **maximum likelihood**: find the parameter values for which the GARCH model is most likely to have generated the observed return series.



Alexios Ghalanos

```
library(rugarch)
citation("rugarch")
```

When using rugarch in publications, please cite:

To cite the rugarch package, please use:

Alexios Ghalanos (2018). rugarch: Univariate GARCH models. R package version 1.4

Workflow

- Three steps:
 - `ugarchspec()`: Specify which GARCH model you want to use (mean μ_t , variance σ_t^2 , distribution of e_t)
 - `ugarchfit()`: Estimate the GARCH model on your time series with returns R_1, \dots, R_T .
 - `ugarchforecast()`: Use the estimated GARCH model to make volatility predictions for R_{T+1}, \dots



Workflow in R

- `ugarchspec()`: Specify which GARCH model you want to use.

```
# Constant mean, standard garch(1,1) model
garchspec <- ugarchspec(
  mean.model = list(armaOrder = c(0,0)),
  variance.model = list(model = "sGARCH"),
  distribution.model = "norm")
```

- `ugarchfit()`: Estimate the GARCH model

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

- `ugarchforecast()`: Forecast the volatility of the future returns

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

ugarchfit object

- The `ugarchfit` yields an object that contains all the results related to the estimation of the garch model.
- **Methods** `coef`, `uncvar`, `fitted` and `sigma`:

```
# Coefficients
garchcoef <- coef(garchfit)

# Unconditional variance
garchuncvar <- uncvariance(garchfit)

# Predicted mean
garchmean <- fitted(garchfit)

# Predicted volatilities
garchvol <- sigma(garchfit)
```



Estimated GARCH coefficients for daily S&P 500 returns

```
print(garchcoef)
```

mu	omega	alpha1	beta1
5.728020e-04	1.220515e-06	7.792031e-02	9.111455e-01

- Estimated model:

$$R_t = 5.7 * 10^{-4} + e_t$$

$$e_t \sim N(0, \hat{\sigma}_t^2)$$

$$\hat{\sigma}_t^2 = 1.2 * 10^{-6} + 0.08e_{t-1}^2 + 0.91 \hat{\sigma}_{t-1}^2$$

```
sqrt(garchuncvar)
```

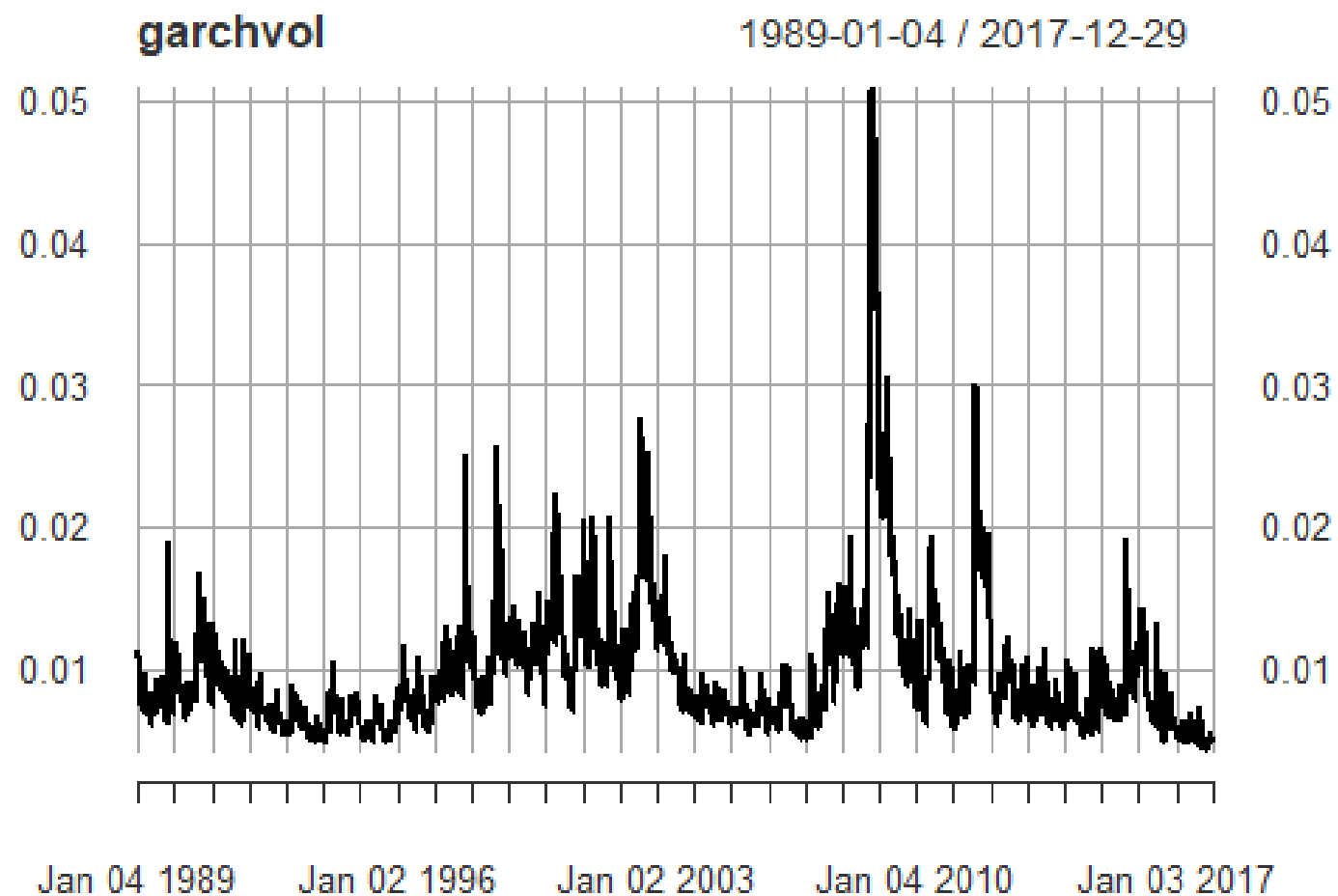
```
0.01056519
```



Estimated volatilities

```
garchvol <- sigma(garchfit)
```

```
plot(garchvol)
```





What about future volatility?

```
tail(garchvol, 1)
```

```
2017-12-29 0.004862908
```

- What about the volatility for the days following the end of the time series?



Forecasting h-day ahead volatilities

- Applying the `sigma()` method to the `ugarchforecast` object gives the volatility

forecasts:

```
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
```



Forecasting h-day ahead volatilities

Applying the `fitted()` method to the `ugarchforecast` object gives the mean forecasts:

```
fitted(garchforecast)
```

```
      2017-12-29  
T+1  0.000572802  
T+2  0.000572802  
T+3  0.000572802  
T+4  0.000572802  
T+5  0.000572802
```


Application to tactical asset allocation

- A portfolio that invests a percentage w in a risky asset (with volatility σ_t) and keeps $1 - w$ on a risk-free bank deposit account has volatility equal to

$$\sigma_p = w\sigma_t.$$

- How to set w ? One approach is **volatility targeting**: w is such that the predicted annualized portfolio volatility equals a target level, say 5%. Then:

$$w^* = 0.05/\sigma_t$$

- Since GARCH volatilities change, the optimal weight changes as well.



GARCH MODELS IN R

Let's play with rugarch!