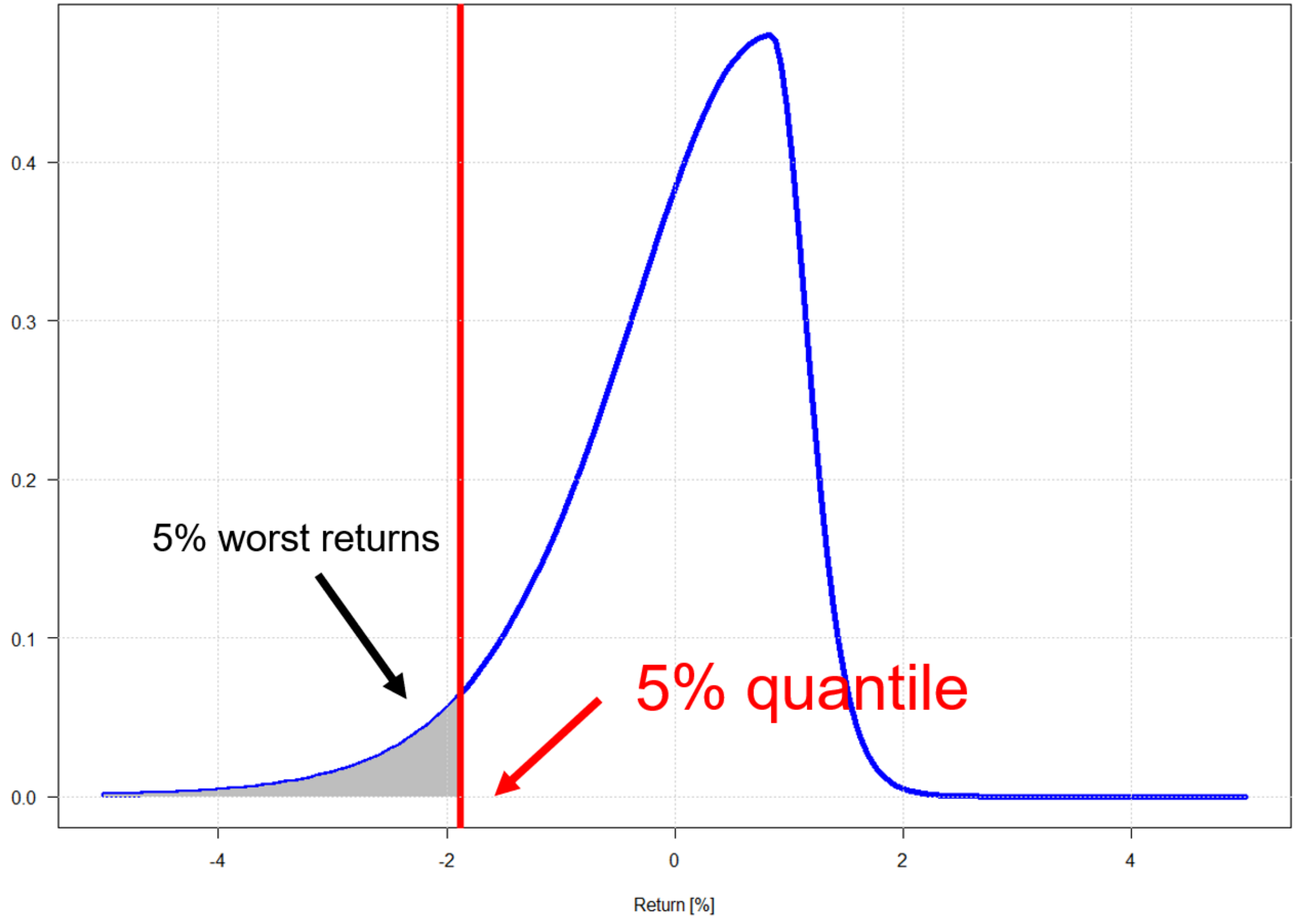
GARCH MODELS IN R

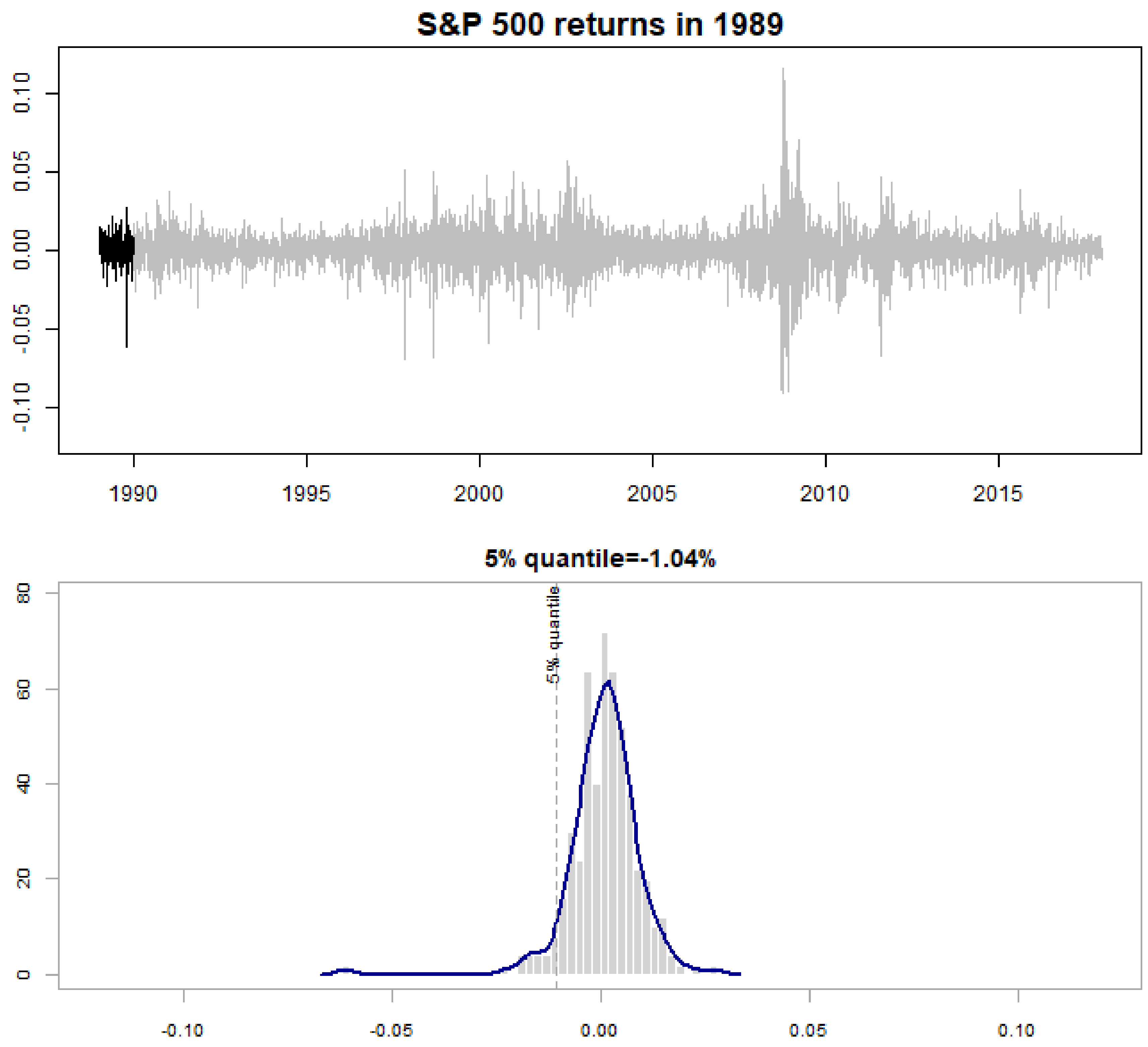
How much would you lose in the best of the 5% worst cases?

Kris Boudt
Professor of finance and econometrics

Value-at-risk

- A popular measure of downside risk: 5% value-at-risk. The 5% quantile of the return distribution represents the best return in the 5% worst scenarios.




S&P 500 returns in 1989
0.10
0.05
0.00
-0.05
-0.10
1990
1995
2000
2005
2010
2015
5% quantile=-1.04%
80
60
40
20
0
5% quantile
-0.10
-0.05
0.00
0.05
0.10

Forward looking approach is needed

- Quantiles of rolling windows of returns are backward looking:
  - ex post question: what has the 5% quantile been for the daily returns over the past year
  - ex ante question: what is the 5% quantile of the predicted distribution of the future return?
- Forward looking risk management uses the predicted quantiles from the GARCH estimation.
- How? Method `quantile()` applied to a `ugarchroll` object.

Workflow to obtain predicted 5% quantiles from ugarchroll

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

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

- `ugarchroll()`: Estimate the GARCH model on rolling estimation samples

```
garchroll <- ugarchroll(garchspec, data = sp500ret, n.start = 2500,
                        refit.window = "moving", refit.every = 100)
```

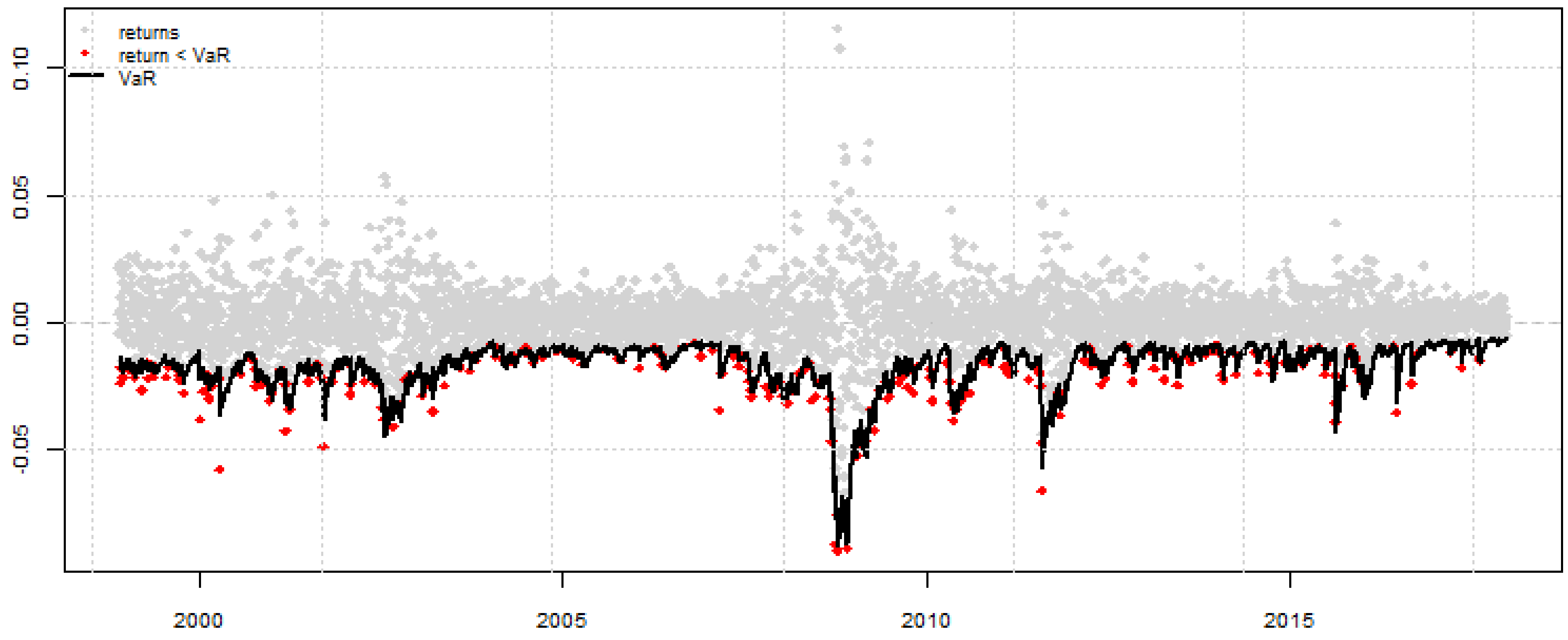
- `quantile()`: Compute the predicted quantile

```
garchVaR <- quantile(garchroll, probs = 0.05)
```

- (or any other loss probability that you wish to use: 1% and 2.5% are also popular)

Value-at-risk plot for loss probability 5%

```
actual <- xts(as.data.frame(garchroll)$Realized, time(garchVaR))
VaRplot(alpha = 0.05, actual = actual, VaR = garchVaR)
```



Exceedance and VaR coverage

A VaR exceedance occurs when the actual return is less than the predicted value-at-risk: $R_t < VaR_t$.

The frequency of VaR exceedances is called the VaR coverage.

```
# Calculation of coverage for S&P 500 returns and 5% probability level
mean(actual < garchVaR)
```

```
0.05159143
```

VaR coverage and model validation

- Interpretation of coverage for VaR at loss probability α (e.g. 5%):
  - Valid prediction model has a coverage that is close to the probability level α used.
  - If coverage $\gg \alpha$: too many exceedances: the predicted quantile should be more negative. Risk of losing money has been underestimated.
  - If coverage $\ll \alpha$: too few exceedances, the predicted quantile was too negative. Risk of losing money has been overestimated.

Factors that deteriorate the performance

- `distribution.model = "std"` instead of `distribution.model = "sstd"`:

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

- Rolling estimation and 5% VaR prediction:

```
garchroll <- ugarchroll(garchspec, data = sp500ret, n.start = 2500,
                        refit.window = "moving", refit.every = 100 )

garchVaR <- quantile(garchroll, probs = 0.05)

mean(actual < garchVaR)
```

```
0.05783233
```

Further deterioration

- `variance.model = list(model = "sGARCH")`

  instead of

  `variance.model = list(model = "gjrGARCH")`:

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

- Rolling estimation and 5% VaR prediction:

```
garchroll <- ugarchroll(garchspec, data = sp500ret, n.start = 2500,
                        refit.window = "moving", refit.every = 100)

garchVaR <- quantile(garchroll, probs=0.05)

mean(actual < garchVaR)
```

```
0.06074475
```

Even further deterioration

- `refit.every = 1000`

  instead of

  `refit.every = 100`:

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

garchroll <- ugarchroll(garchspec, data = sp500ret, n.start = 2500,
                        refit.window = "moving", refit.every = 1000)

garchVaR <- quantile(garchroll, probs = 0.05)

mean(actual<garchVaR)

0.06199293
```

GARCH MODELS IN R

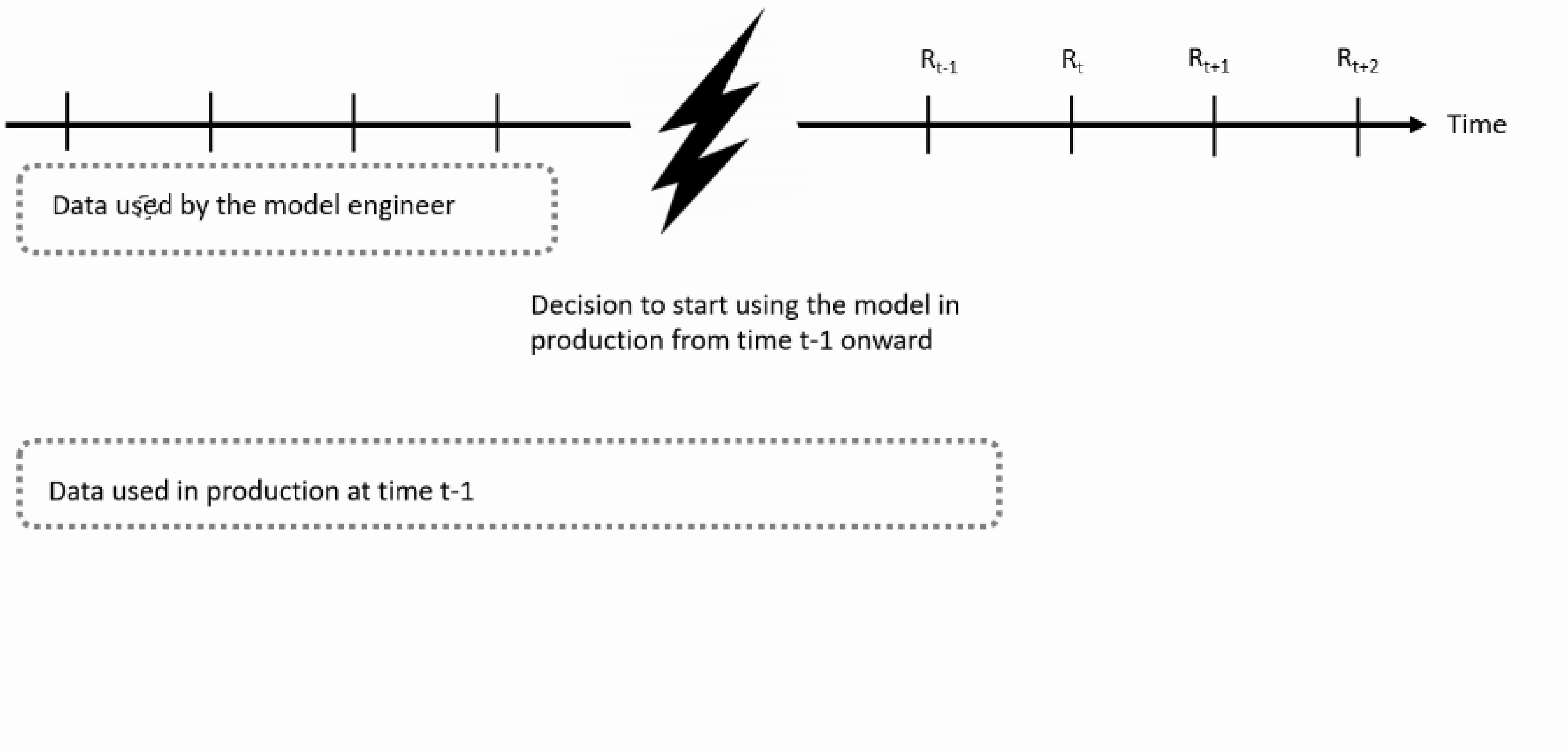
Downside risk means thinking about predicted quantiles.

GARCH MODELS IN R

Use the validated GARCH model in production

Kris Boudt
Professor of finance and econometrics

Use in production



New functionality

- Use `ugarchfilter()` for analyzing the recent dynamics in the mean and volatility
- Use `ugarchforecast()` applied to a `ugarchspec` object (instead of `ugarchfit()`) object for making the predictions about the future mean and volatility

Example on MSFT returns

- `msftret`: 1999-2017 daily returns.
- Suppose the model fitting was done using the returns available at year-end 2010.
- You use this model at year-end 2017 to analyze past volatility dynamics and predict future volatility.

Step 1: Defines the final model specification

- Fit the best model using the `msftret` available at year-end 2010:

```
# specify AR(1)-GJR GARCH model with skewed student t distribution
garchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                        variance.model = list(model = "gjrGARCH"),
                        distribution.model = "sstd")
# estimate the model
garchfit <- ugarchfit(data = msftret["/2010-12"], spec = garchspec)
```

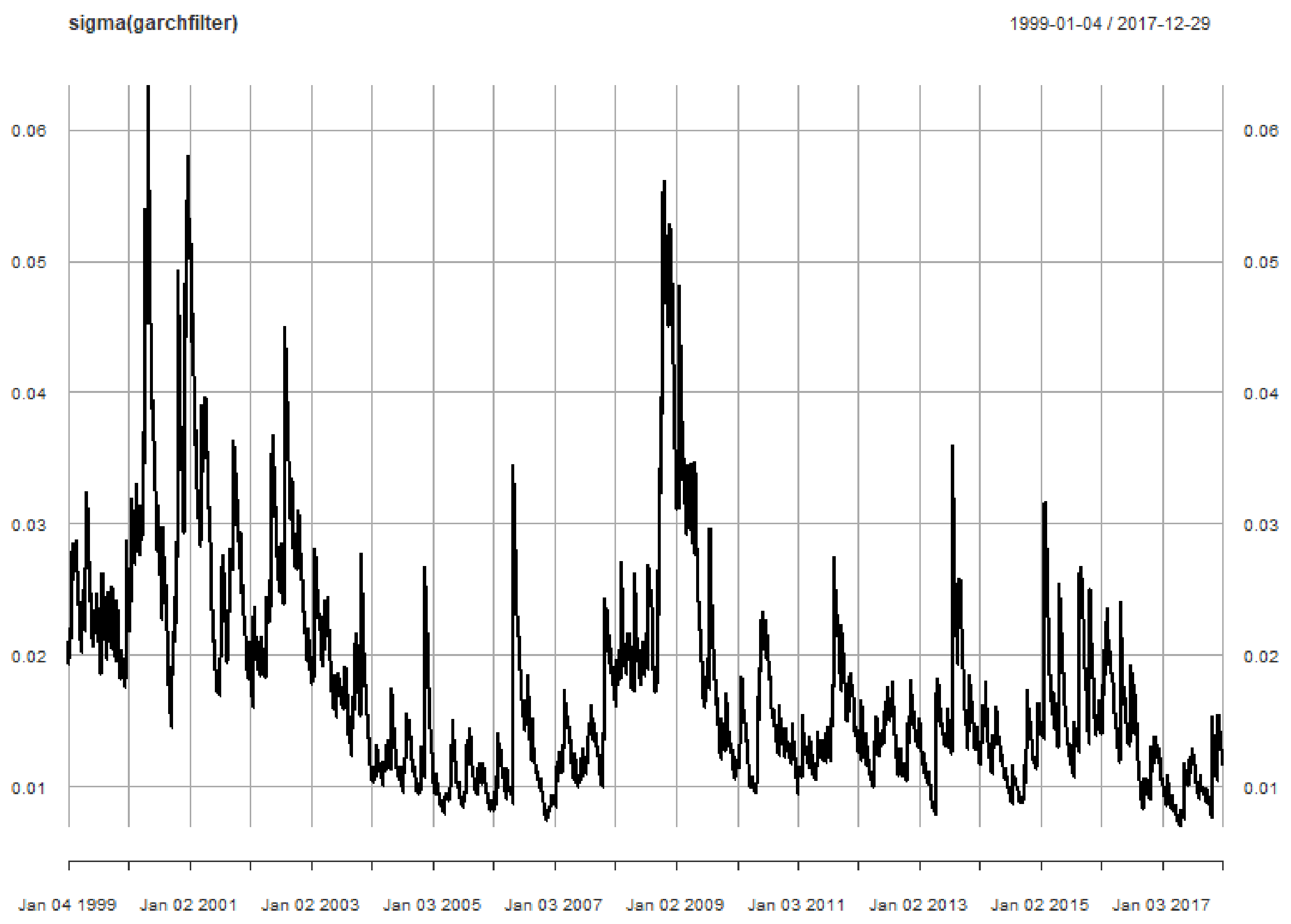
- Define `progarchspec` as the specification to be used in production and use the instruction `setfixed(progarchspec) <- as.list(coef(garchfit))`:

```
progarchspec <- garchspec
setfixed(progarchspec) <- as.list(coef(garchfit))
```

Step 2: Analysis of past mean and volatility dynamics

- Use the `ugarchfilter()` function:

```
garchfilter <- ugarchfilter(data = msftret, spec = progarchspec)
plot(sigma(garchfilter))
```



Step 3: Make predictions about future returns

```
# Make the predictions for the mean and vol for the next ten days
garchforecast <- ugarchforecast(data = msftret,
                                fitORspec = progarchspec,
                                n.ahead = 10)
```

```
cbind(fitted(garchforecast), sigma(garchforecast))
```

```
       2017-12-29 2017-12-29
T+1  0.0004781733 0.01124870
T+2  0.0003610470 0.01132550
T+3  0.0003663683 0.01140171
T+4  0.0003661265 0.01147733
T+5  0.0003661375 0.01155238
T+6  0.0003661370 0.01162688
T+7  0.0003661371 0.01170083
T+8  0.0003661371 0.01177424
T+9  0.0003661371 0.01184712
T+10 0.0003661371 0.01191948
```

Use in simulation

- Instead of applying the complete model to analyze observed returns, you can use it to simulate artificial log-returns:

$$r_t = \log(P_t) - \log(P_{t-1})$$

- Useful to assess the randomness in future returns and the impact on prices, since the future price equals:

$$P_{t+h} = P_t \exp(r_{t+1} + r_{t+2} + \ldots + r_{t+h})$$

Step 1: Calibrate the simulation model

- Use the log-returns in the estimation

```
# Compute log returns
msftlogret <- diff(log(MSFTprice))[(-1)]
```

- Estimate the model and assign model parameters to the simulation model

```
garchspec <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                        variance.model = list(model = "gjrGARCH"),
                        distribution.model = "sstd")
# Estimate the model
garchfit <- ugarchfit(data = msftlogret, spec = garchspec)

# Set that estimated model as the model to be used in the simulation
simgarchspec <- garchspec
setfixed(simgarchspec) <- as.list(coef(garchfit))
```

Step 2: Run the simulation with ugarchpath()

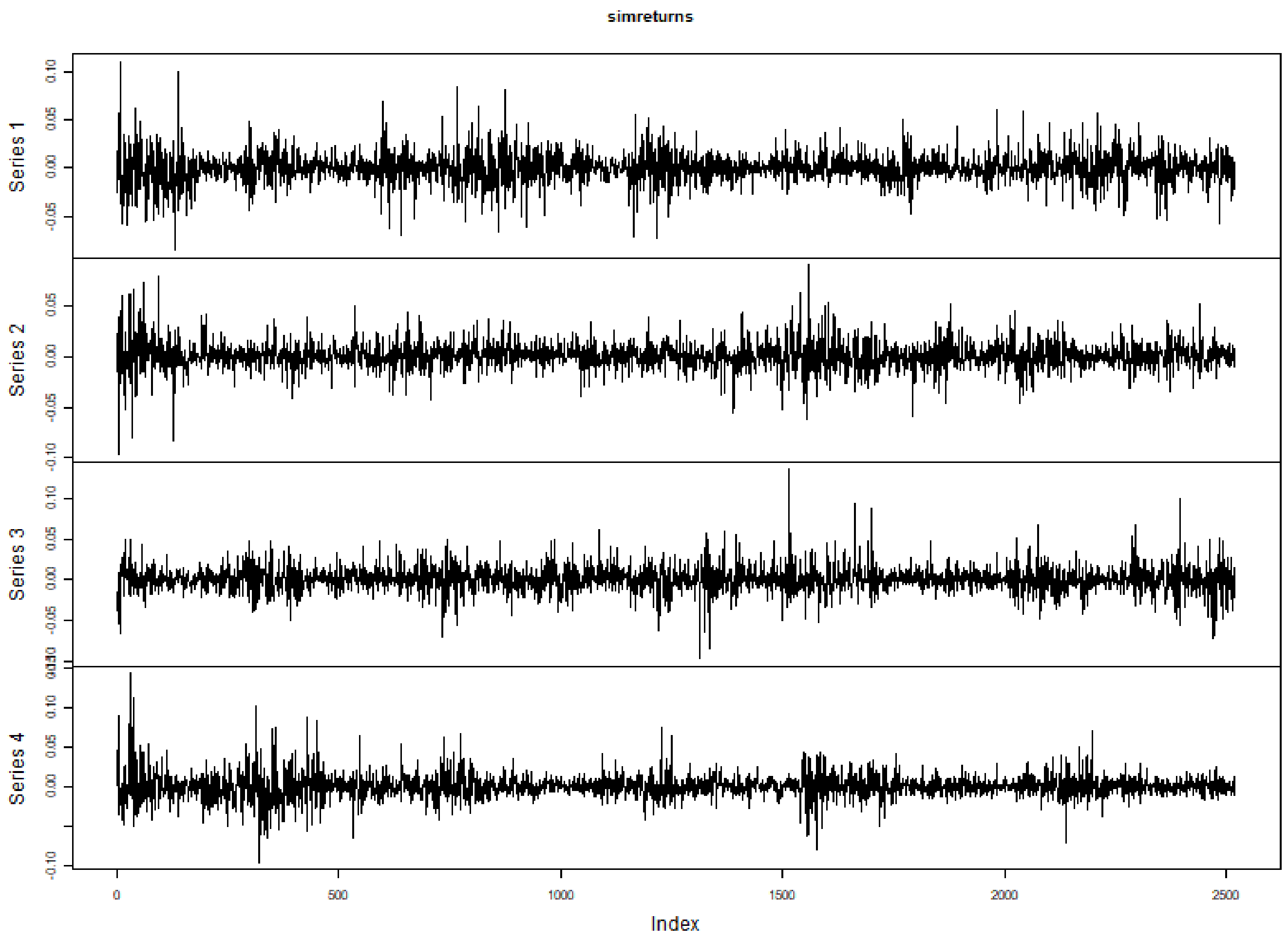
- Simulation using the `ugarchpath()` function requires to choose:
  - `spec` : completely specified GARCH model
  - `m.sim` : number of time series of simulated returns you want
  - `n.sim`: number of observations in the simulated time series (e.g. 252)
  - `rseed` : any number to fix the seed used to generate the simulated series (needed for reproducibility)

```
simgarch <- ugarchpath(spec = simgarchspec, m.sim = 4,
                       n.sim = 10 * 252, rseed = 12345)
```

Step 3: Analysis of simulated returns

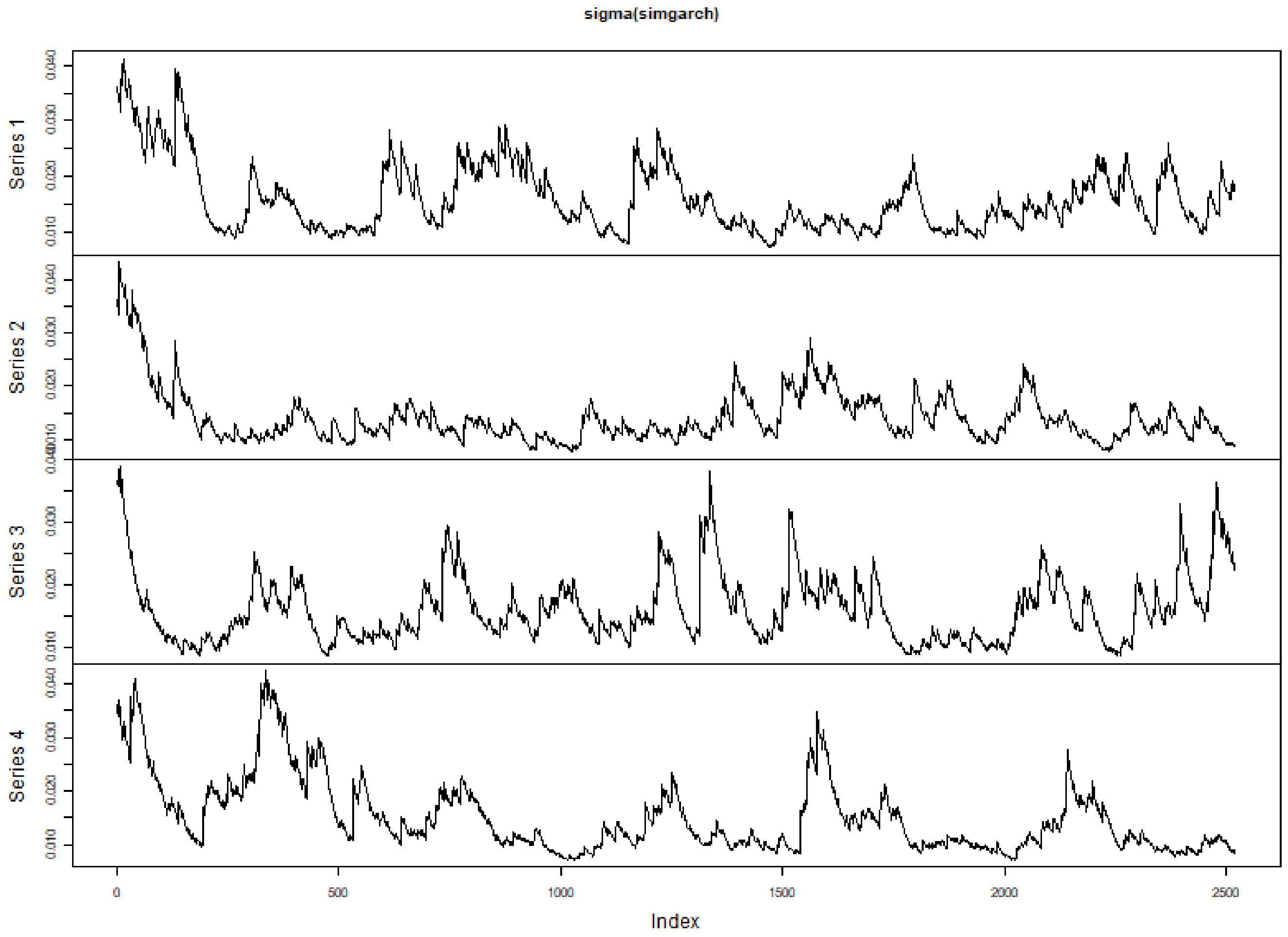
- Method `fitted()` provides the simulated returns:

```
simret <- fitted(simgarch)
plot.zoo(simret)
```



Analysis of simulated volatility

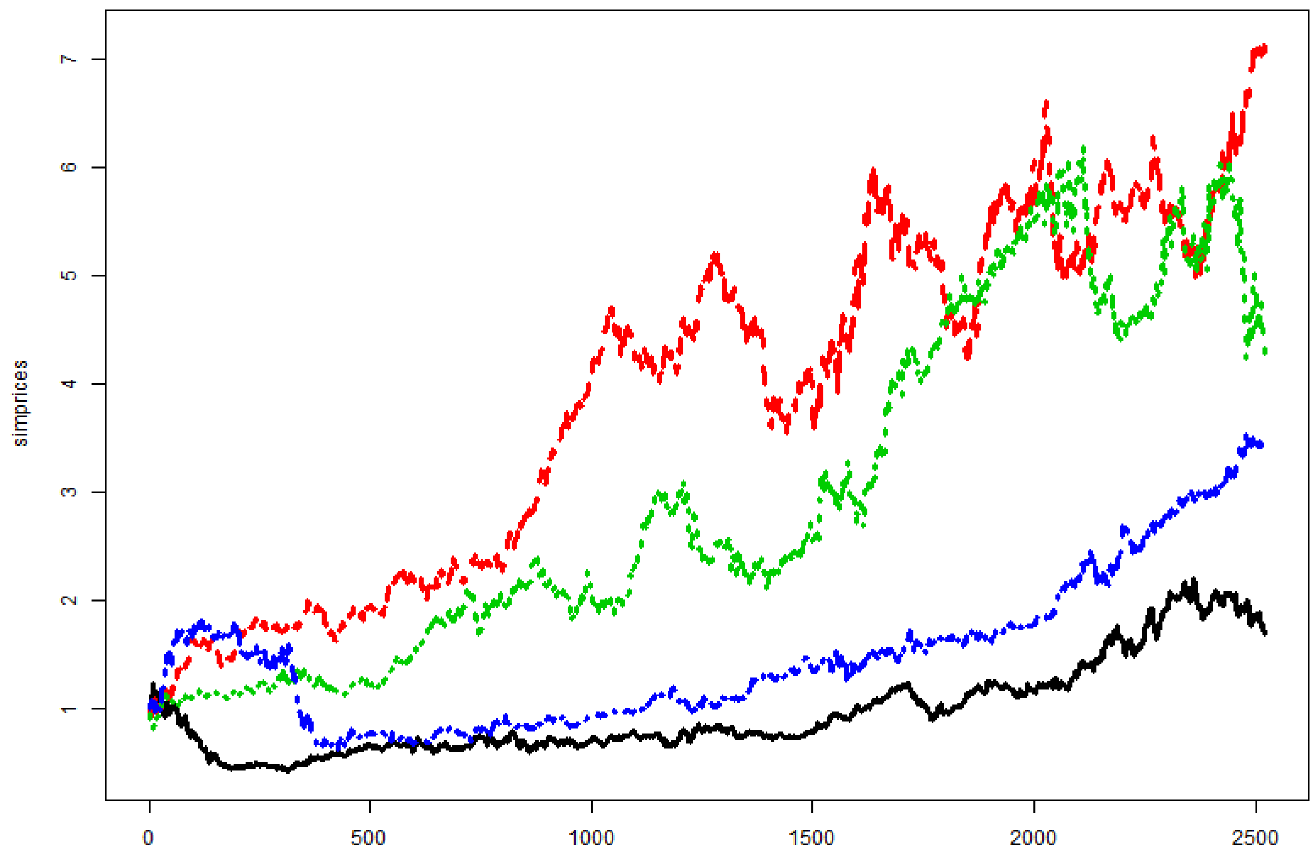
```
plot.zoo(sigma(simgarch))
```



Analysis of simulated prices

- Plotting 4 simulations of 10 years of stock prices, with initial price set at 1:

```
simprices <- exp(apply(simret, 2, "cumsum"))
matplot(simprices, type = "l", lwd = 3)
```



GARCH MODELS IN R

Time to practice with setfixed(), ugarchfilter(), ugarchforecast() and ugarchpath()

GARCH MODELS IN R

Model risk is the risk of using the wrong model

Kris Boudt
Professor of finance and econometrics

Sources of model risk and solutions

Sources:

- modeling choices
- starting values in the optimization
- outliers in the return series

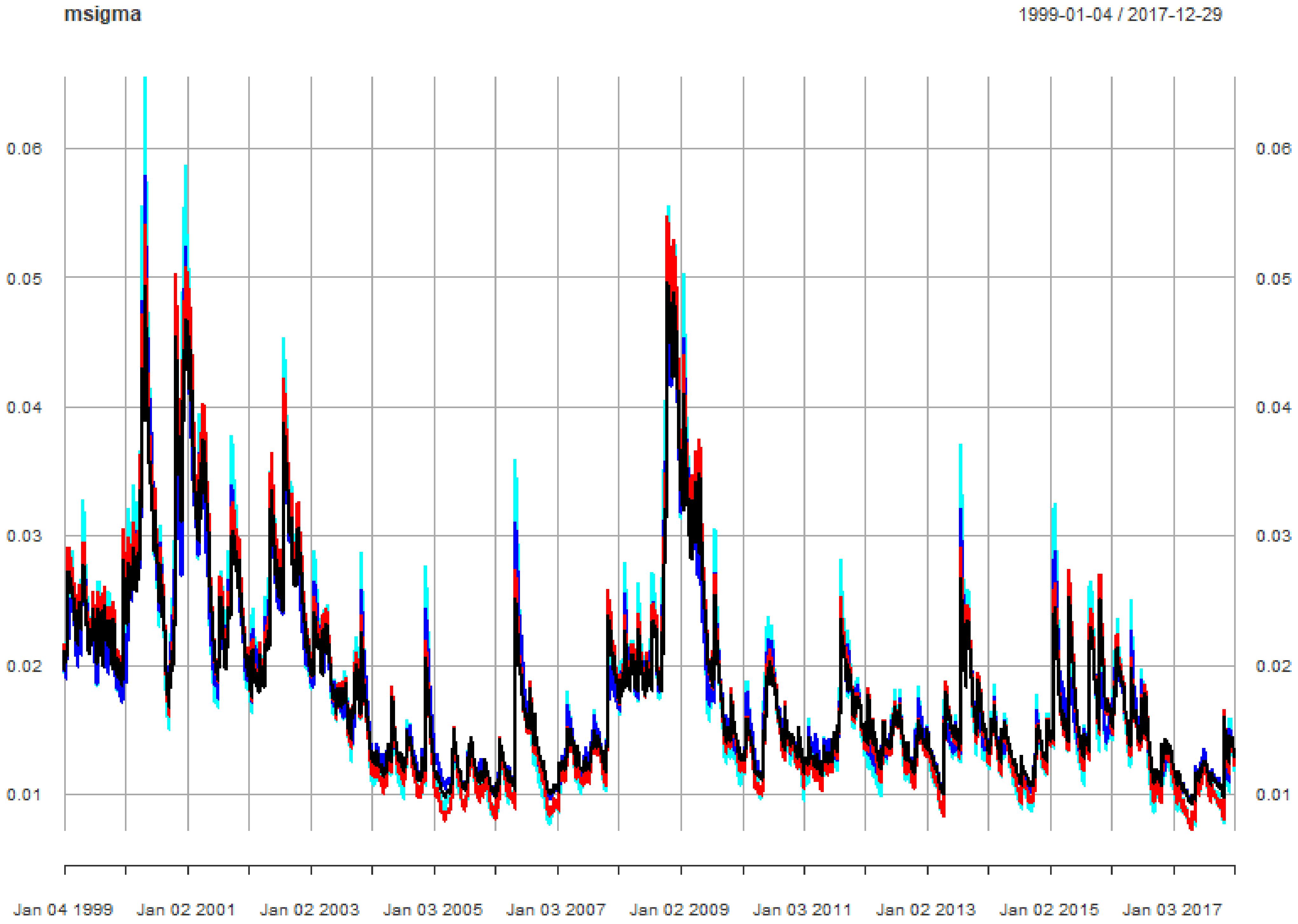
Solution: Protect yourself through a robust approach

- model-averaging: averaging the predictions of multiple models
- trying several starting values and choosing the one that leads to the highest likelihood
- cleaning the data

Model averaging

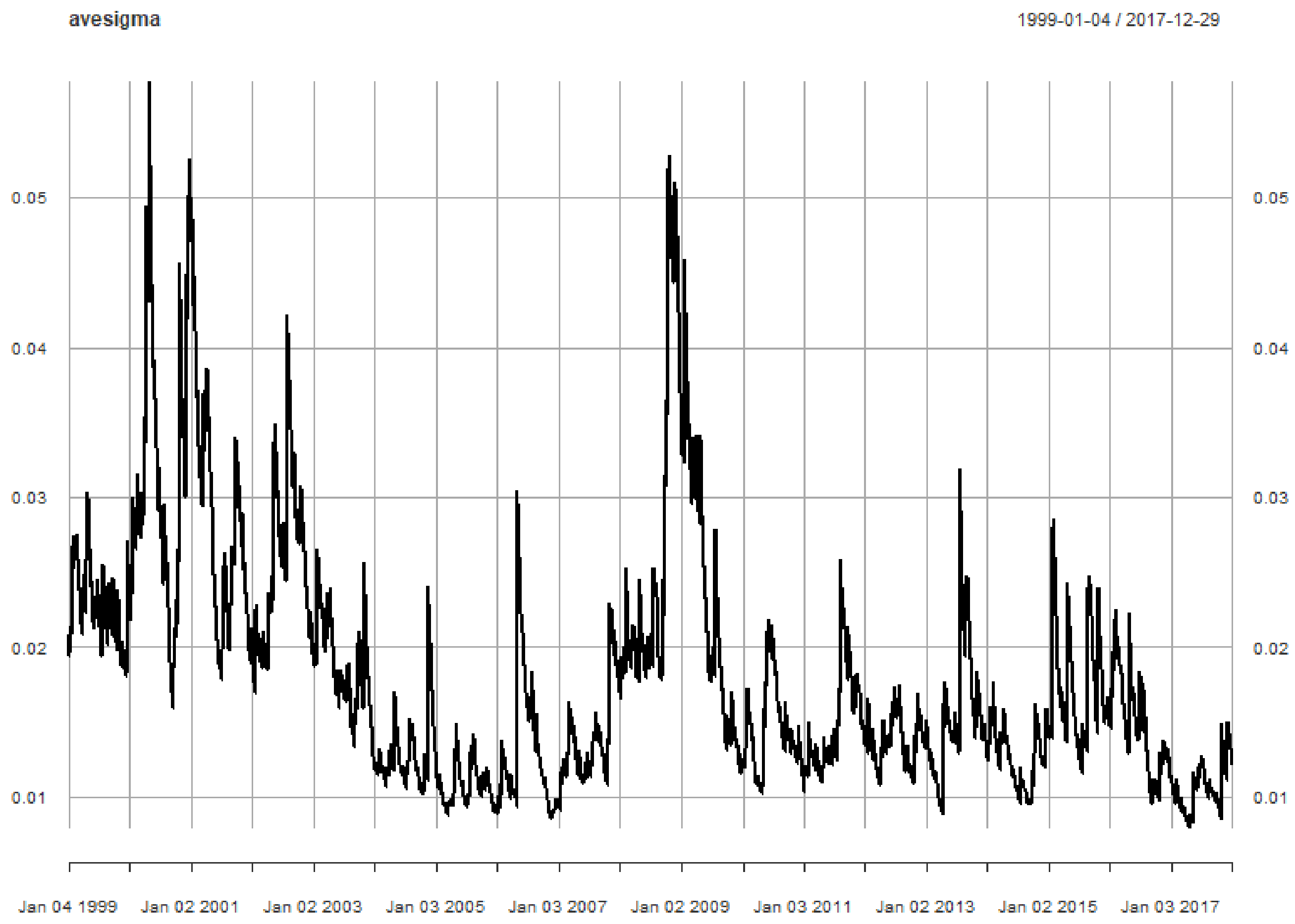
- If you cannot choose which model to use, you could estimate them all

```
variance.models <- c("sGARCH", "gjrGARCH")
distribution.models <- c("norm", "std", "std")
c <- 1
for (variance.model in variance.models) {
    for (distribution.model in distribution.models) {
        garchspec <- ugarchspec(mean.model = list(armaOrder = c(0, 0)),
                                variance.model = list(model = variance.model),
                                distribution.model = distribution.model)
        garchfit <- ugarchfit(data = msftret, spec = garchspec)
        if (c==1) {
          msigma <- sigma(garchfit)
        } else {
          msigma <- merge(msigma, sigma(garchfit))
        }
        c <- c + 1
     }
}
```


msigma
1999-01-04 / 2017-12-29
0.06
0.05
0.04
0.03
0.02
0.01
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

The average vol prediction

```
avesigma <- xts(rowMeans(msigma), order.by = time(msigma))
```



Robustness to starting values

- GARCH models have many parameters, like

```
coef(garchfit)

          mu        omega       alpha1        beta1         skew        shape
5.669200e-04 6.281258e-07 7.462984e-02 9.223701e-01 9.436331e-01 6.318621e+00
```

- Those estimates are the result of a complex optimization of the likelihood function
- Optimization is numeric and iterative: step by step improvement, which can be sensitive to starting values
- `rugarch` has a default approach in getting sensible starting values
- You can specify your own starting values by applying the `setstart()` method to your `ugarchspec()` GARCH model specification

Example with setstart() - default starting values

- Estimation with default starting values

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

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

coef(garchfit)

          mu        omega       alpha1        beta1         skew        shape
5.669200e-04 6.281258e-07 7.462984e-02 9.223701e-01 9.436331e-01 6.318621e+00

likelihood(garchfit)

24280.33
```

Example with setstart() - modified starting values

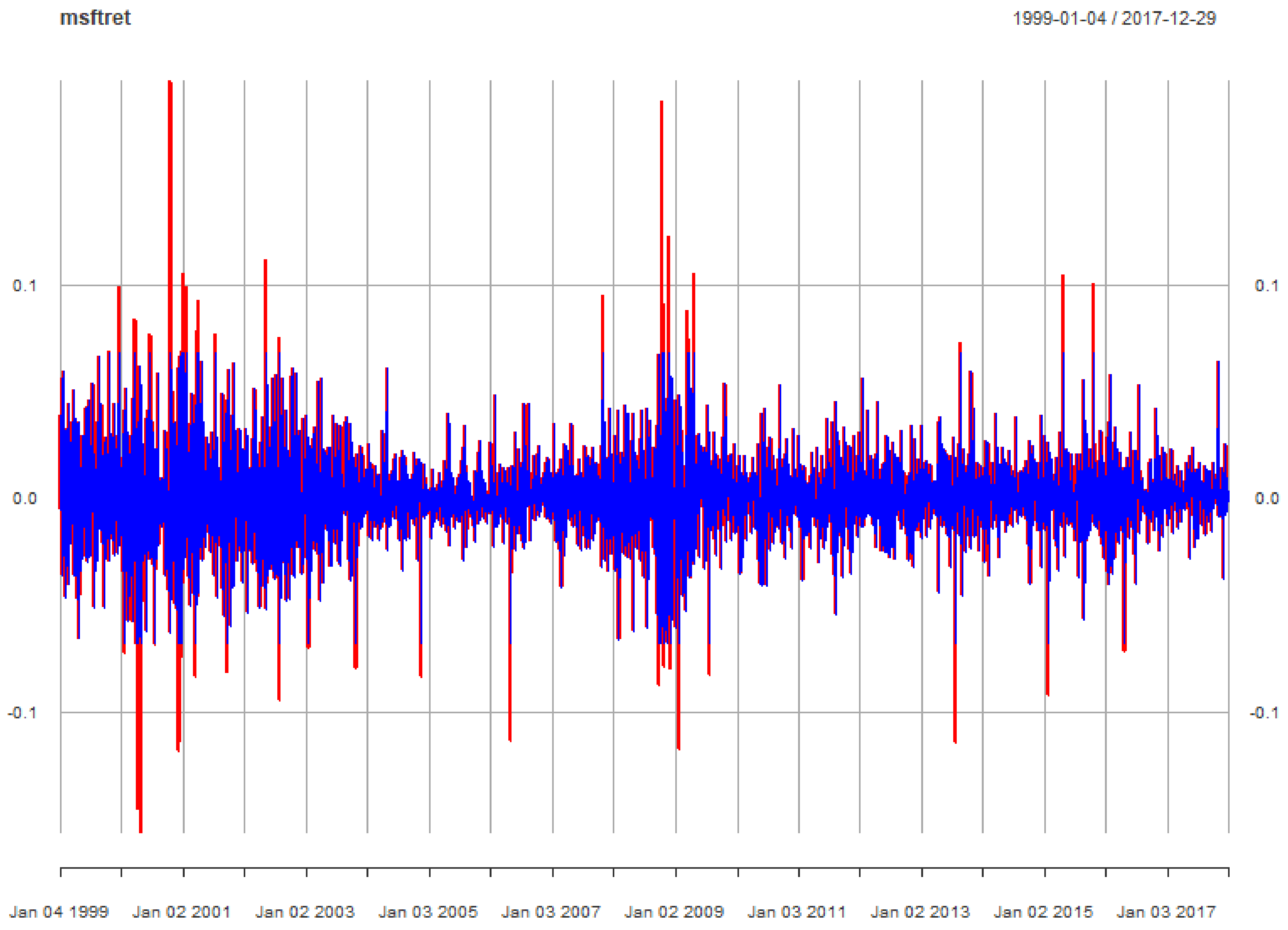
- Estimation with modified starting values

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

setstart(garchspec) <- list(alpha1 = 0.05, beta1 = 0.9, shape = 8)

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

coef(garchfit)

          mu        omega       alpha1        beta1         skew        shape
5.638002e-04 6.303949e-07 7.466503e-02 9.224117e-01 9.438978e-01 6.309185e+00

likelihood(garchfit)

24280.33
```

Cleaning the data

- Avoid that outliers distort the volatility predictions
- How? Through winsorization: reduce the magnitude of the return to an acceptable level using the function `Return.clean()` in the package `PerformanceAnalytics` with `method="boudt"`:

```
# Clean the return series
library(PerformanceAnalytics)
clmsftret <- Return.clean(msftret, method = "boudt")

# Plot them on top of each other
plotret <- plot(msftret, col = "red")
plotret <- addSeries(clmsftret, col = "blue", on = 1)
```


msftret
1999-01-04 / 2017-12-29
0.1
0.0
-0.1
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

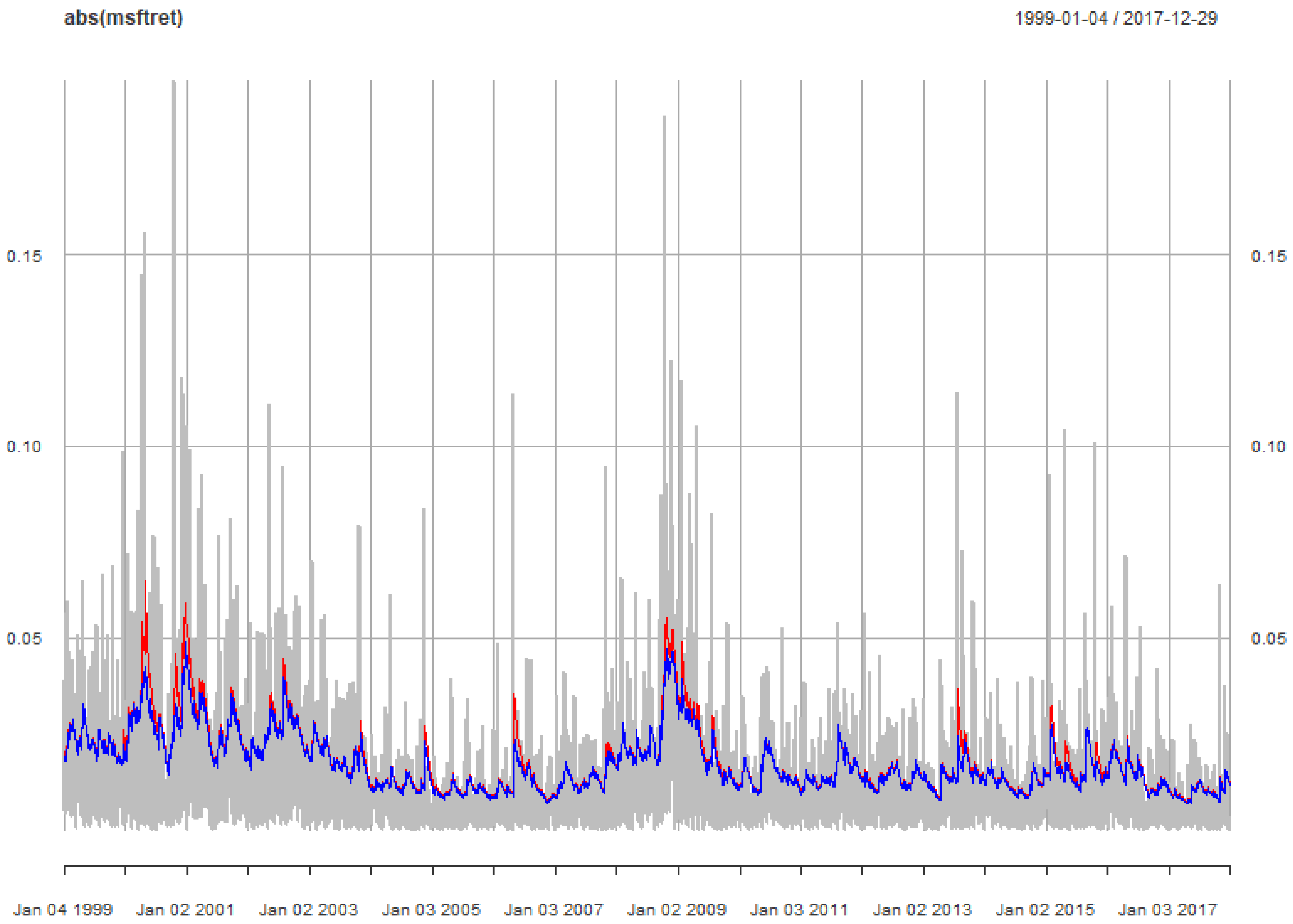
Impact of cleaning on volatility prediction

- Make the volatility predictions using raw and cleaned Microsoft returns

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

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

clgarchfit <- ugarchfit(data = clmsftret, spec = garchspec)
```

- Compare them in a time series plot

```
plotvol <- plot(abs(msftret), col = "gray")
plotvol <- addSeries(sigma(garchfit), col = "red", on = 1)
plotvol <- addSeries(sigma(clgarchfit), col = "blue", on = 1)
plotvol
```


abs(msftret)
1999-01-04 / 2017-12-29
0.15
0.10
0.05
0.15
0.10
0.05
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

GARCH MODELS IN R

Be a robustnik: it is better to be roughly right than exactly wrong

GARCH MODELS IN R

GARCH volatility leads to time-varying variability of the returns

Kris Boudt
Professor of finance and econometrics

GARCH covariance

- If two asset returns $R_{1,t}$ and $R_{2,t}$ have correlation ρ and time varying volatility $\sigma_{1,t}$ and $\sigma_{2,t}$, then their covariance is:

$$\sigma_{12,t} = \rho\, \sigma_{1,t}\, \sigma_{2,t}$$

GARCH covariance estimation in four steps

- Step 1: Use `ugarchfit()` to estimate the GARCH model for each return series.

```
msftgarchfit <- ugarchfit(data = msftret, spec = garchspec)
wmtgarchfit <- ugarchfit(data = wmtret, spec = garchspec)
```

- Step 2: Use `residuals()` to compute the standardized returns.

```
stdmsftret <- residuals(msftgarchfit, standardize = TRUE)
stdwmtret <- residuals(wmtgarchfit, standardize = TRUE)
```

- Step 3: Use `cor()` to estimate ρ as the sample correlation of the standardized returns.

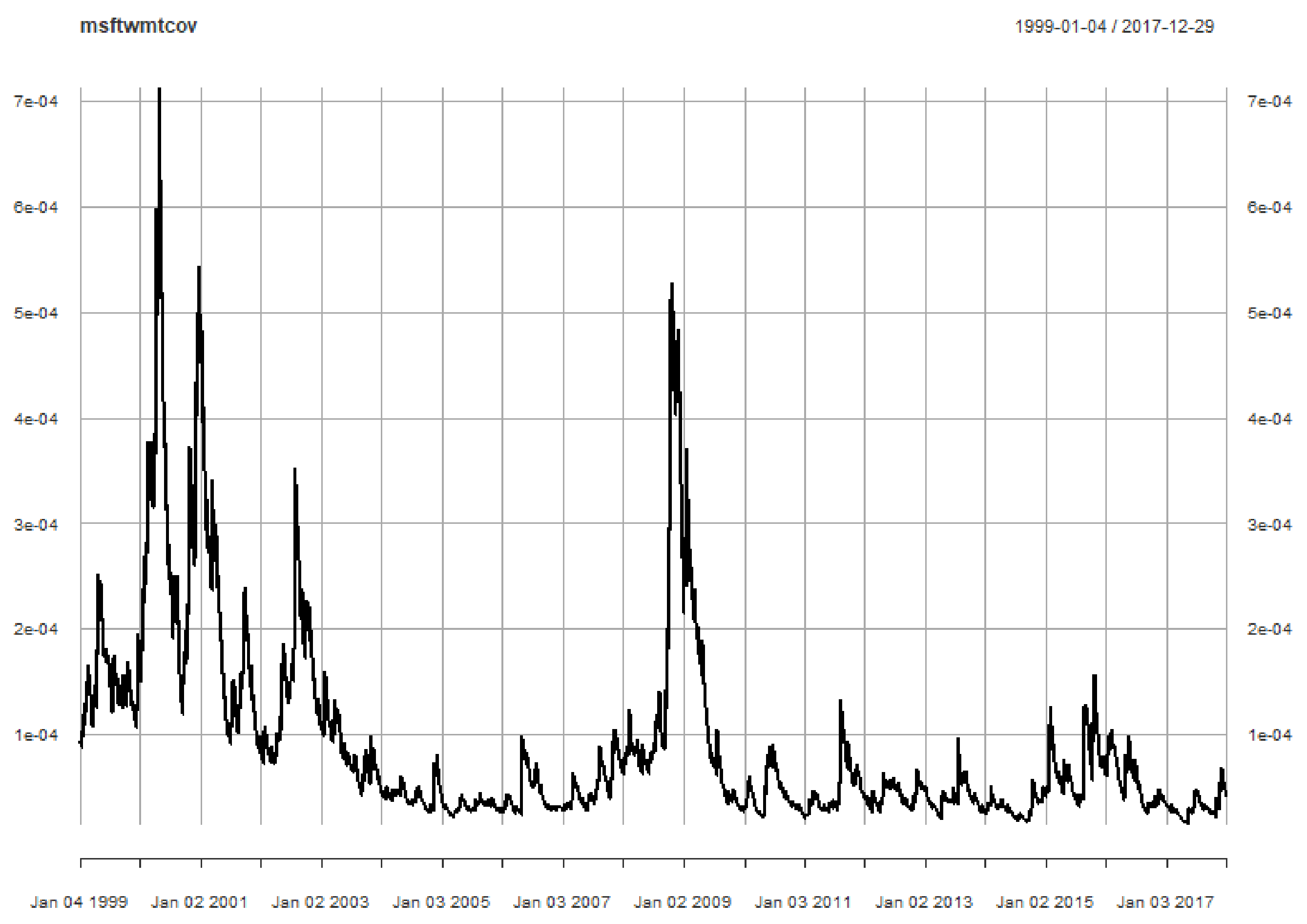
```
msftwmtcor <- as.numeric(cor(stdmsftret, stdwmtret))
msftwmtcor
```

```
0.298795
```

GARCH covariance estimation in four steps

- Step 4: Compute the GARCH covariance by multiplying the estimated correlation and volatilities

```
msftwmtcov <- msftwmtcor * sigma(msftgarchfit) * sigma(wmtgarchfit)
```



Applications of covariance in finance

- Numerous!
- Important case: Optimizing the variance of the portfolio.
- It depends on the:
  - portfolio weights
  - the variance of all the assets
  - the covariance between the asset returns

Application to portfolio optimization

- Variance of portfolio of two assets with weight $w_{1,t}$ invested in asset 1 and $(1 - w_{1,t})$ in asset 2:

$$\sigma^2_{p,t} = w^2_{1,t}\ \sigma^2_{1,t} + (1 - w_{1,t})^2\ \sigma^2_{2,t} + 2\ w_{1,t}\ (1 - w_{1,t})\sigma_{12,t}$$

- Many ways to define optimal $w_{1,t}$. One appraoch is to set $w_{1,t}$ such that the portfolio variance σ^2_t is minimized.
- First order condition to find minimum variance portfolio

$$\frac{d\sigma^2_{p,t}}{dw_{1,t}} = 2w_{1,t}(\sigma^2_{1,t} + \sigma^2_{2,t} - 2\sigma_{12,t}) - 2\ (\sigma^2_{2,t} - \sigma_{12,t}) = 0$$

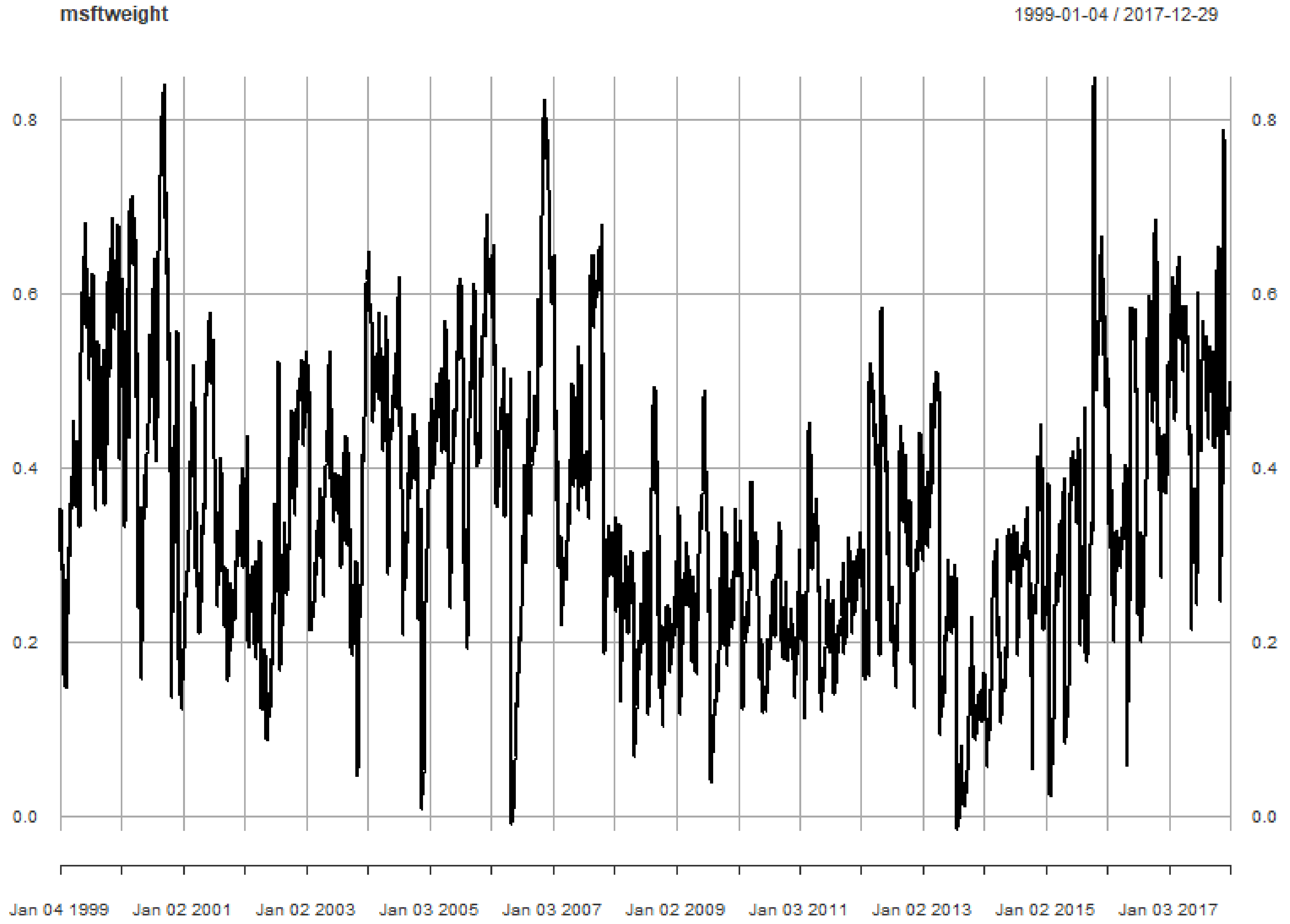
Minimum variance portfolio weights

- Solution:

$$w_{1,t}^{*} = \frac{\sigma_{2,t}^{2} - \sigma_{12,t}}{\sigma_{1,t}^{2} + \sigma_{2,t}^{2} - 2\sigma_{12,t}}$$

- Calculation in R

```
msftvar <- sigma(msftgarchfit)^2
wmtvar <- sigma(wmtgarchfit)^2
msftwmtcov <- msftwmtcor * sigma(msftgarchfit) * sigma(wmtgarchfit)
msftweight <- (wmtvar - msftwmtcov) / (msftvar + wmtvar - 2 * msftwmtcov)
```


msftweight
1999-01-04 / 2017-12-29
0.8
0.6
0.4
0.2
0.0
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

Dynamic beta

- The estimation of a stock's beta: systematic risk of a stock
- Defined as the covariance of the stock return and the market return, divided by the variance of the market returns

$$\beta_t = \frac{\textit{covariance between the stock return and the market return}}{\textit{variance of the market return}}$$

- Needed to compute the risk premium. The higher it is, the more risky the stock and thus the higher the required rate of return.
- For US stocks, the market return is the return on the S&P 500.

The daily beta of MSFT

- Compute the covariance between MSFT and S&P 500 returns

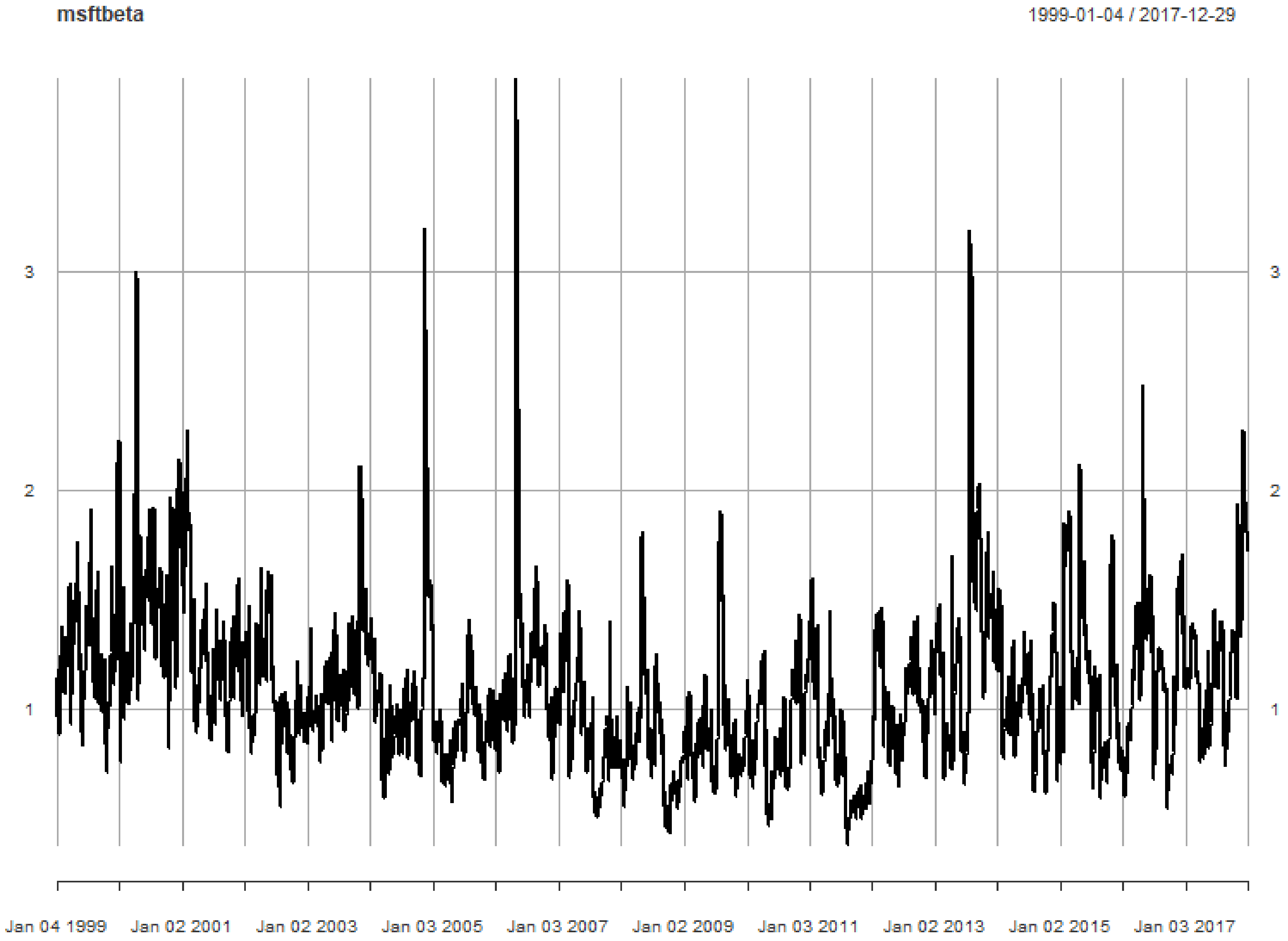
```
msftsp500cor <- as.numeric(cor(stdmsftret, stdsp500ret))
msftsp500cov <- msftsp500cor * sigma(msftgarchfit) * sigma(sp500garchfit)
```

- Compute the variance of the S&P 500 returns

```
sp500var <- sigma(sp500garchfit)^2
```

- Compute the beta

```
msftbeta <- msftsp500cov / sp500var
```


msftbeta
1999-01-04 / 2017-12-29
3
2
1
3
2
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

GARCH MODELS IN R

Let's practice!

GARCH MODELS IN R

Congratulations! You have learned three languages

Kris Boudt
Professor of finance and econometrics

Language of GARCH models

- Volatility σ_t at its clusters
- Information set and predictions
- Mean, variance and distribution assumptions
- Leverage effect and the GJR GARCH model
- Skewness, fat tails and the skewed student t distribution
- Model validation using the mean squared error, significance testing, standardized returns and Ljung-Box test
- Applications to value-at-risk, dynamic beta calculation and optimization of financial portfolios

Language of rugarch

- `ugarchspec()`
- `ugarchfit()`
- `ugarchroll()`
- `ugarchforecast()`
- `ugarchfilter()`
- `ugarchpath()`
- Many useful methods `sigma()`, `fitted()`, `coef()`, `infocriteria()`, `likelihood()`, `setfixed()`, `setbounds()`, `quantile()`...

GARCH MODELS IN R

@OptimizeRisk