COMPUTATIONAL FINANCE & RISK MANAGEMENT
UNIVERSITY of WASHINGTON
Department of Applied Mathematics

Univariate GARCH

Amath 546/Econ 589
Eric Zivot
Spring 2013
Updated: April 24, 2013

Introduction to ARCH and GARCH Models

- ARCH (AutoRegressive Conditional Heteroskedasticity) models were proposed by Engle in 1982.
- GARCH (Generalized ARCH) models proposed by Bollerslev in 1986.
- Engle received the Nobel price in 2003. The GARCH model framework is considered as one of the most important contributions in empirical finance over the last 20 years.
- Engle currently resides at NYU and heads the volatility institute

Robert Engle, NYU

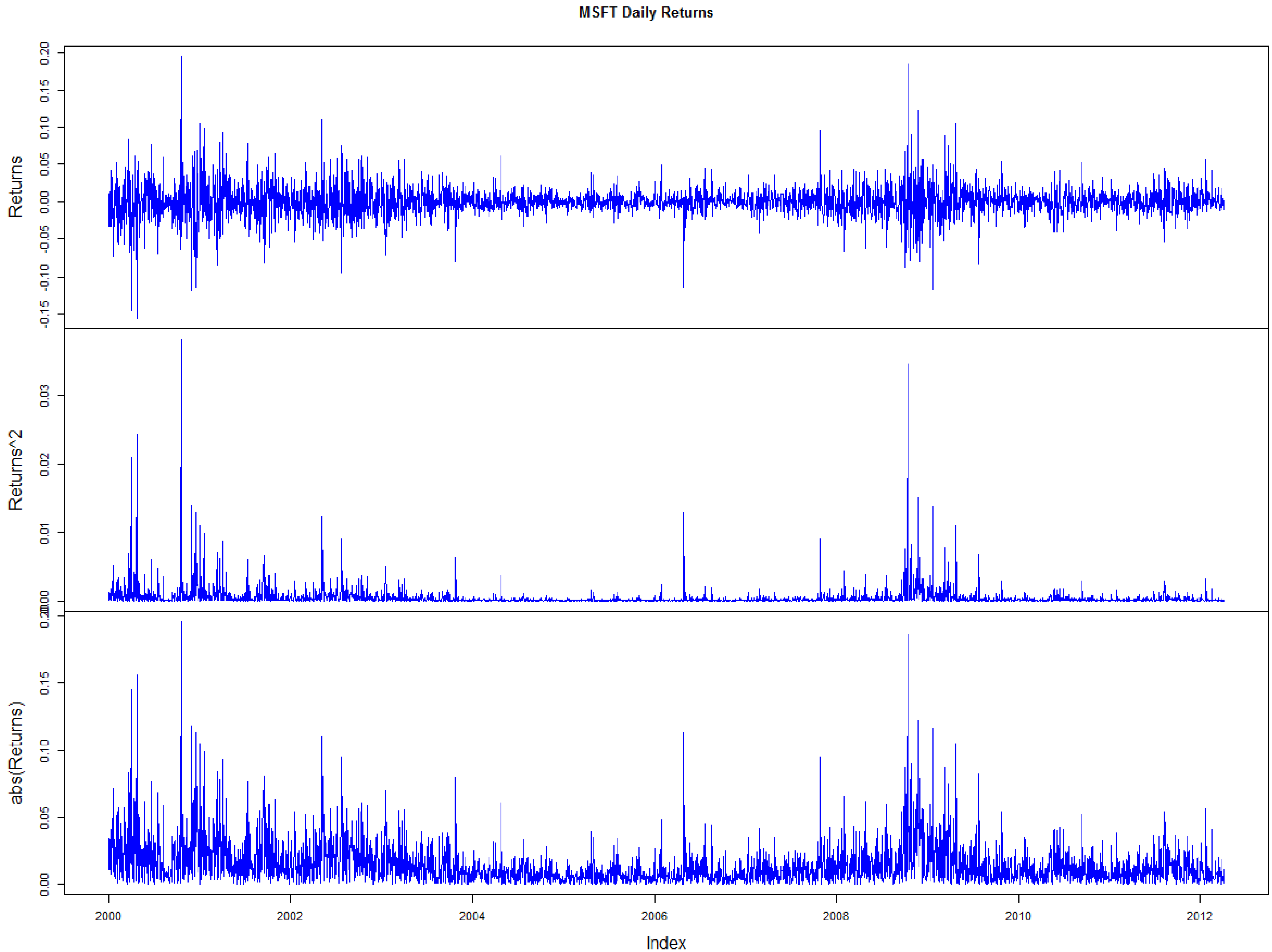
Nobel Prize citation:
"for methods of analyzing economic time series with time-varying volatility (ARCH)"

Champion Pairs Skater Too!

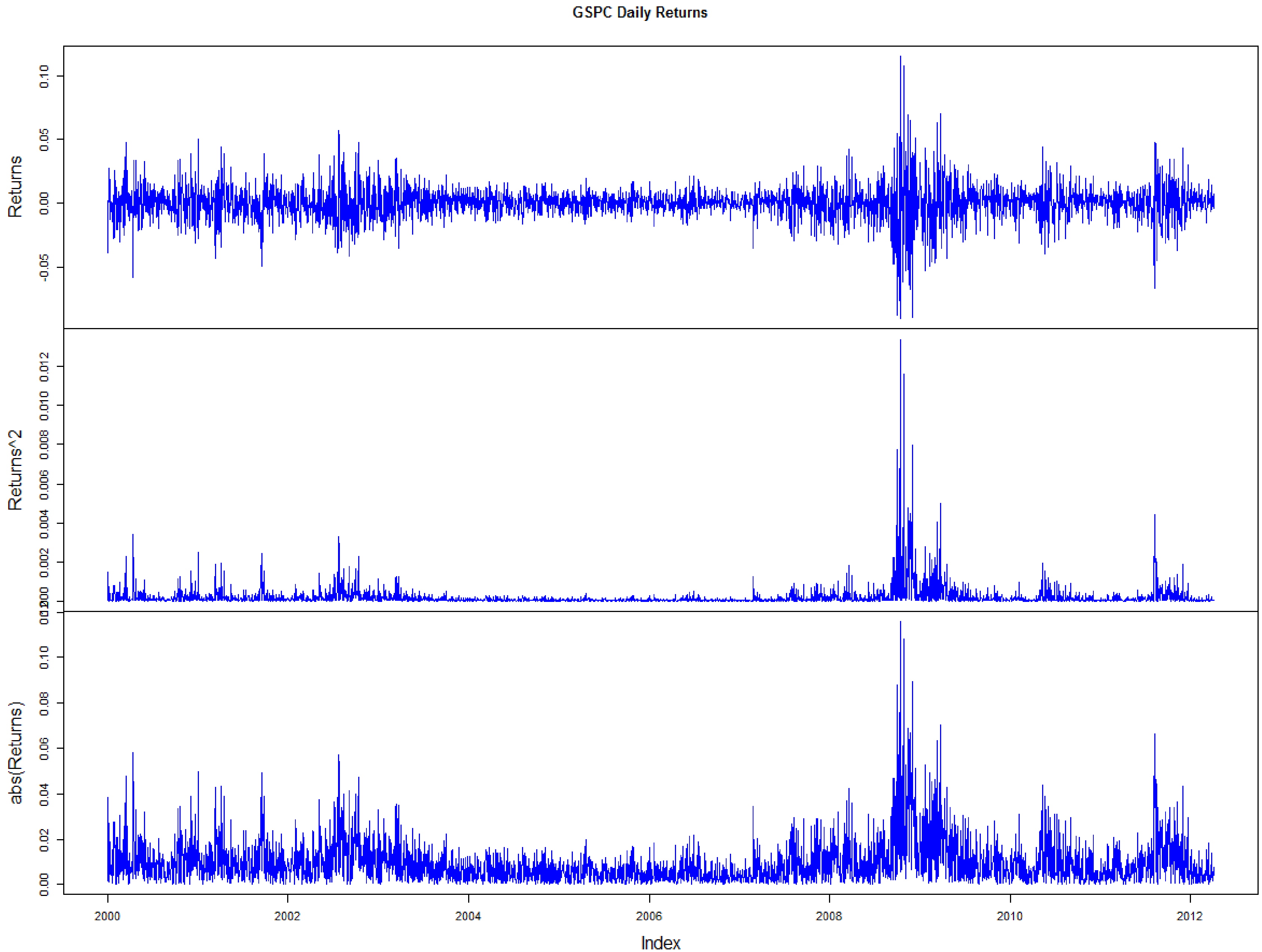
The ARCH Family

Bolerslev (2008) identified over 150 different ARCH models. Here are some of the most common:

- GJR-GARCH
- TARCH
- STARCH
- AARCH
- NARCH
- MARCH
- SWARCH
- SNPARCH
- APARCH
- TAYLOR-SCHWERT
- FIGARCH
- FIEGARCH
- Component
- Asymmetric Component
- SQGARCH
- CESGARCH
- Student t
- GED
- SPARCH

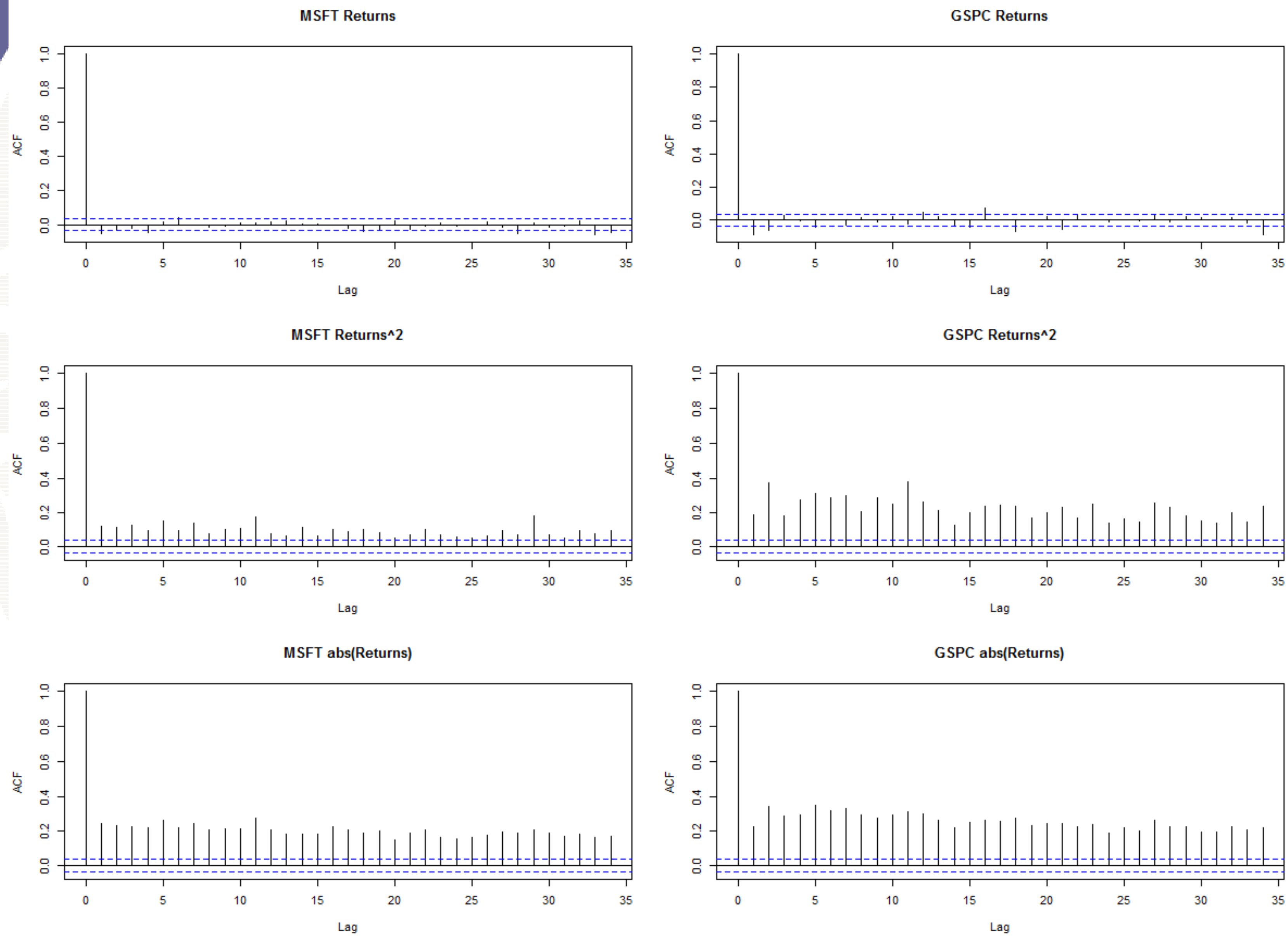


MSFT Daily Returns
Returns
Returns^2
abs(Returns)
Index
2000
2002
2004
2006
2008
2010
2012



GSPC Daily Returns
Returns
Returns^2
abs(Returns)
Index
2000
2002
2004
2006
2008
2010
2012

Return Autocorrelations



Summary Statistics

```
> table.Stats(MSFT.GSPC.ret)
                         MSFT      GSPC
Observations        3082.0000 3082.0000
NAs                    0.0000    0.0000
Minimum               -0.1560   -0.0903
Quartile 1            -0.0093   -0.0061
Median                 0.0000    0.0006
Arithmetic Mean        0.0001    0.0001
Geometric Mean        -0.0001    0.0000
Quartile 3             0.0095    0.0063
Maximum                0.1955    0.1158
SE Mean                0.0004    0.0002
LCL Mean (0.95)       -0.0006   -0.0004
UCL Mean (0.95)        0.0009    0.0006
Variance               0.0005    0.0002
Stdev                  0.0214    0.0137
Skewness               0.2500    0.0298
Kurtosis               9.0241    7.3286
```

Specify ARCH(1) Process in rugarch

$$r_t = \sigma_t z_t$$
$$z_t \sim iid\ N(0,1)$$
$$\sigma_t^2 = 0.1 + 0.8\varepsilon_{t-1}^2$$

```
# Use functions from rugarch package
> arch1.spec=ugarchspec(variance.model=list(garchOrder=c(1,0)),
+                        mean.model = list(armaOrder=c(0,0)),
+                        fixed.pars=list(mu = 0, omega=0.1,
+                                        alpha1=0.8))

> class(arch1.spec)
[1] "uGARCHspec"
attr(,"package")
[1] "rugarch"
```

Specify ARCH(1) Process

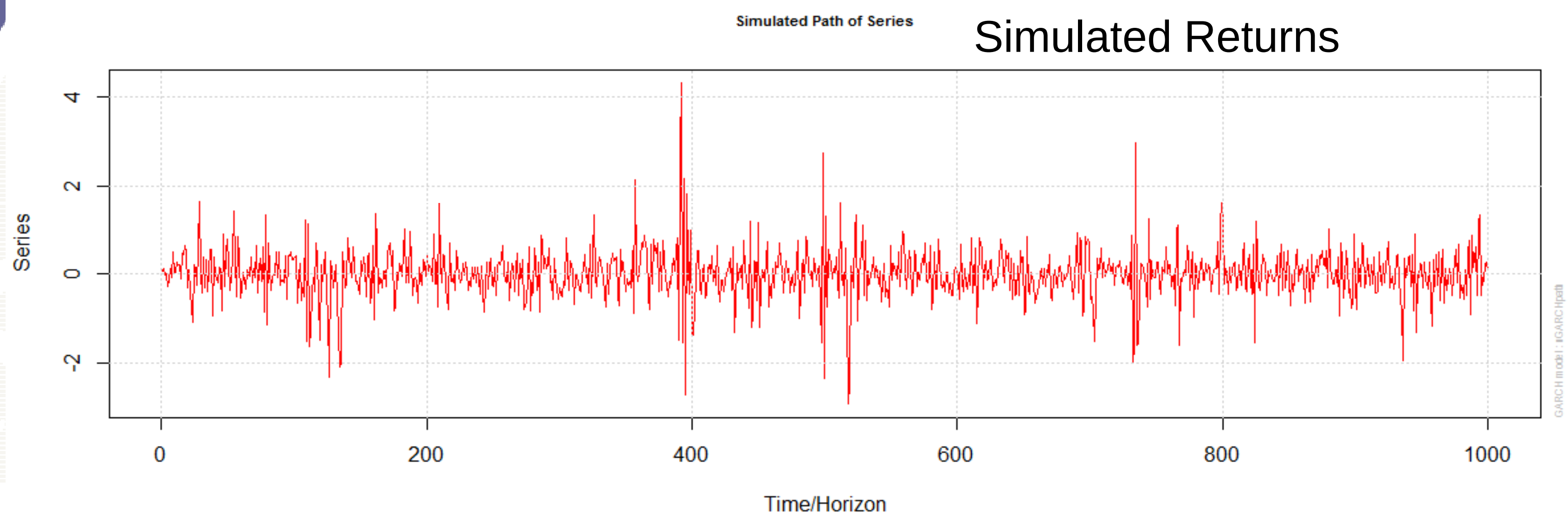
```
> show(arch1.spec)

*---------------------------------*
*       GARCH Model Spec          *
*---------------------------------*

Conditional Variance Dynamics
------------------------------------
GARCH Model             : sGARCH(1,0)
Variance Targeting      : FALSE

Conditional Mean Dynamics
------------------------------------
Mean Model              : ARFIMA(0,0,0)
Include Mean            : TRUE
GARCH-in-Mean           : FALSE

Conditional Distribution
------------------------------------
Distribution    :  norm
Includes Skew   :  FALSE
Includes Shape :  FALSE
Includes Lambda        :  FALSE
```

Simulate ARCH(1) Process

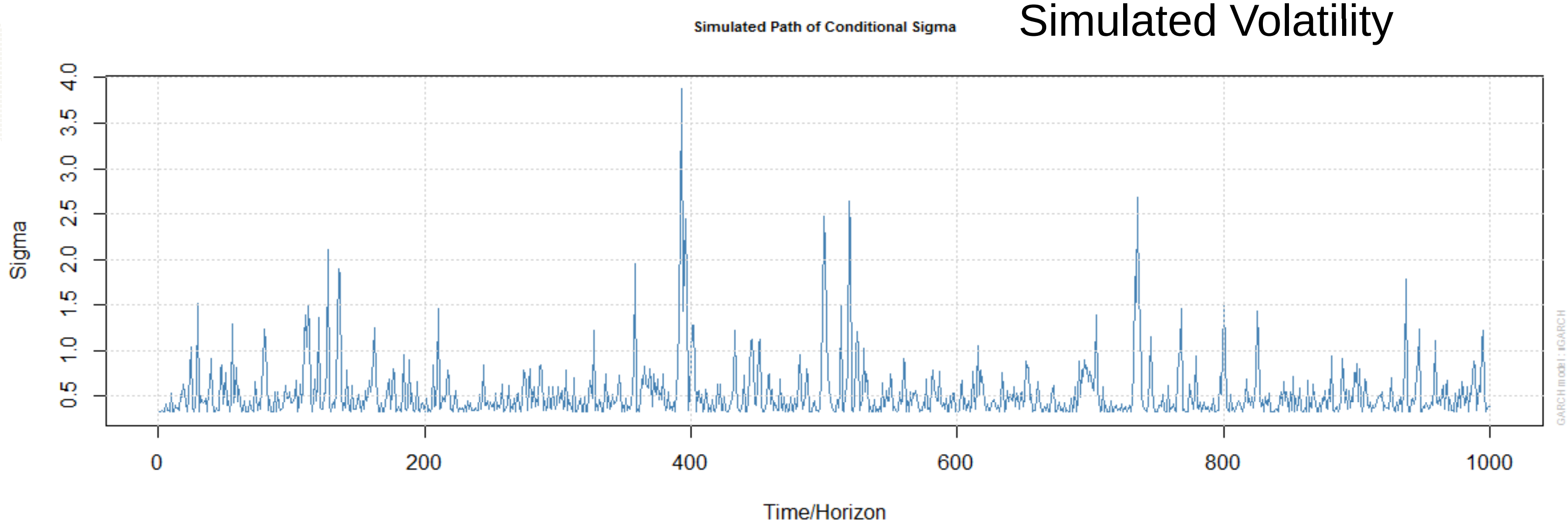
```
# Use functions from rugarch package
> set.seed(123)
> arch1.sim = ugarchpath(arch1.spec, n.sim=1000)

> class(arch1.sim)
[1] "uGARCHpath"
attr(,"package")
[1] "rugarch“

> slotNames(arch1.sim)
[1] "path"  "model" "seed“

> names(arch1.sim@path)
[1] "sigmaSim"  "seriesSim" "residSim“

> plot(arch1.sim)
```

Simulated ARCH(1) Process

Simulated Returns

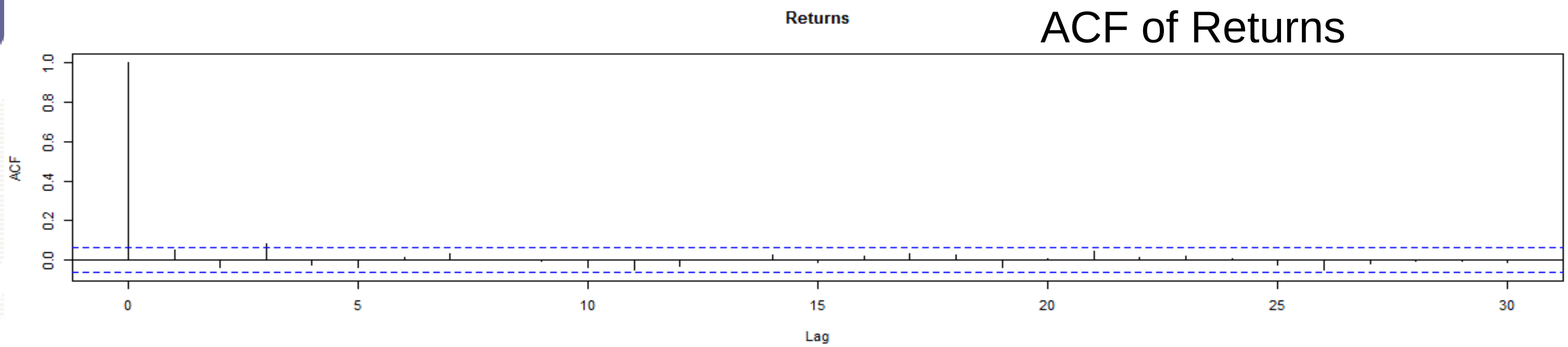


Simulated Volatility

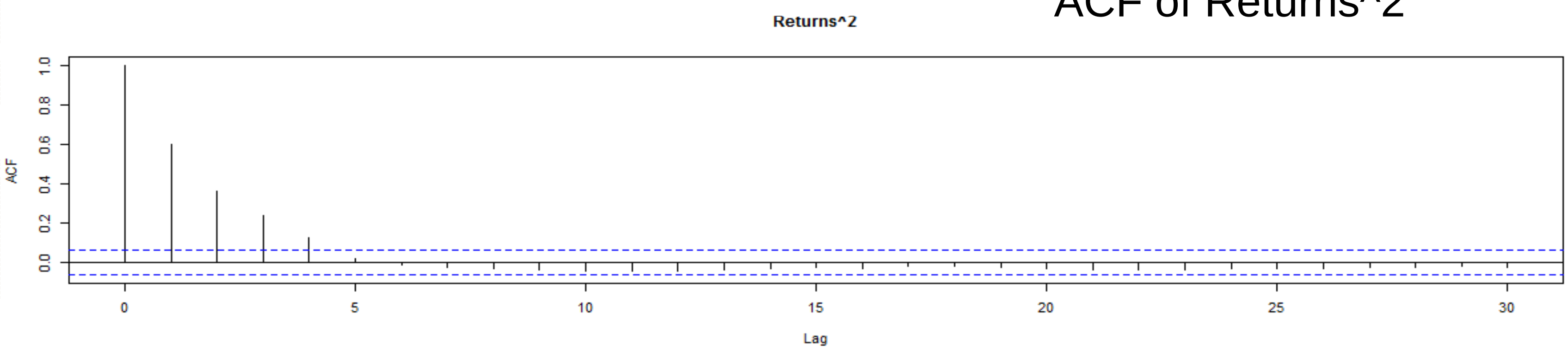


ARCH(1) Autocorrelations

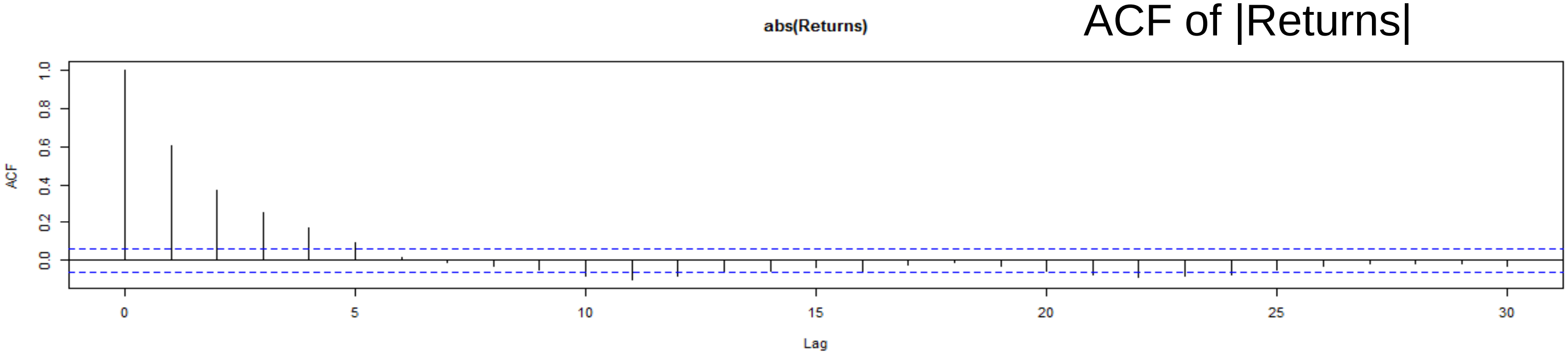
ACF of Returns



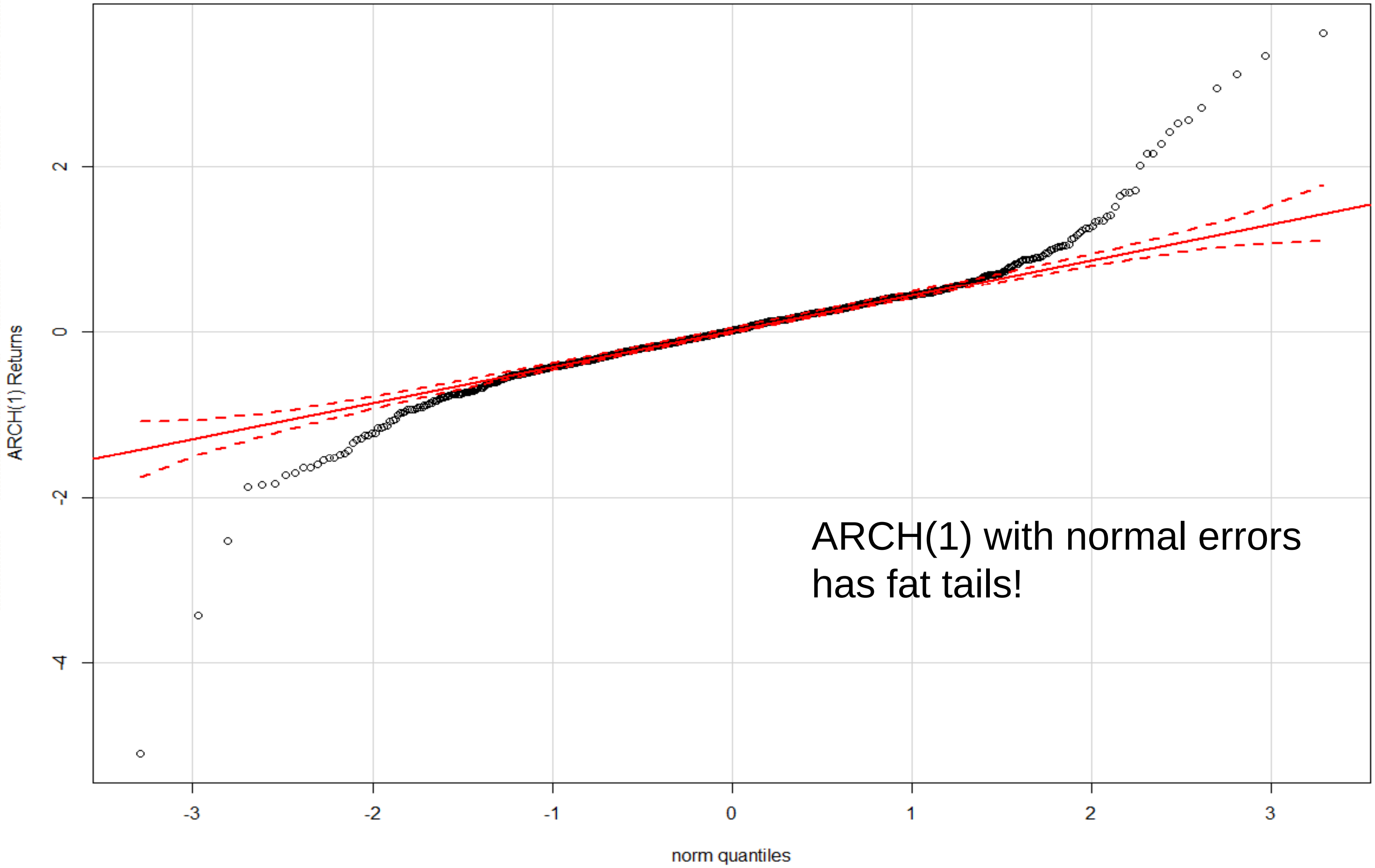
ACF of Returns^2



ACF of |Returns|



ARCH(1) Normal QQ-Plot



ARCH(1) with normal errors has fat tails!

Simulate GARCH(1,1) Process

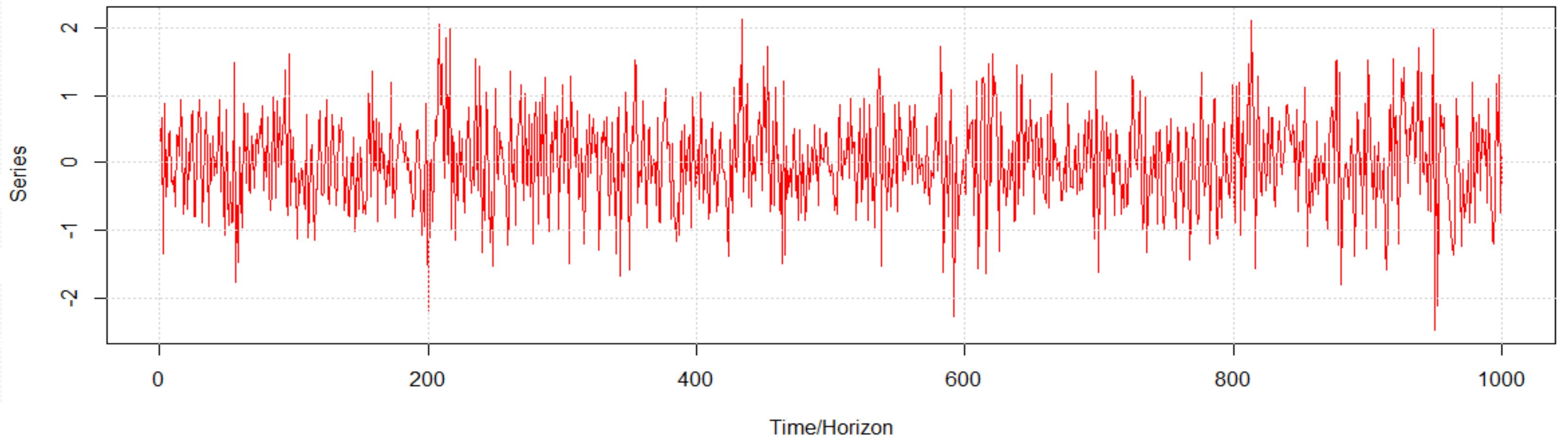
```
> garch11.spec=ugarchspec(variance.model=list(garchOrder=c(1,1)),
+                         mean.model = list(armaOrder=c(0,0)),
+                         fixed.pars=list(mu = 0, omega=0.1,
+                                         alpha1=0.1,
+                                         beta1 = 0.7))

> set.seed(123)
> garch11.sim = ugarchpath(garch11.spec, n.sim=1000)
```

Note: alpha1 + beta1 = 0.8, same as for the ARCH(1)

Simulated GARCH(1,1) Process

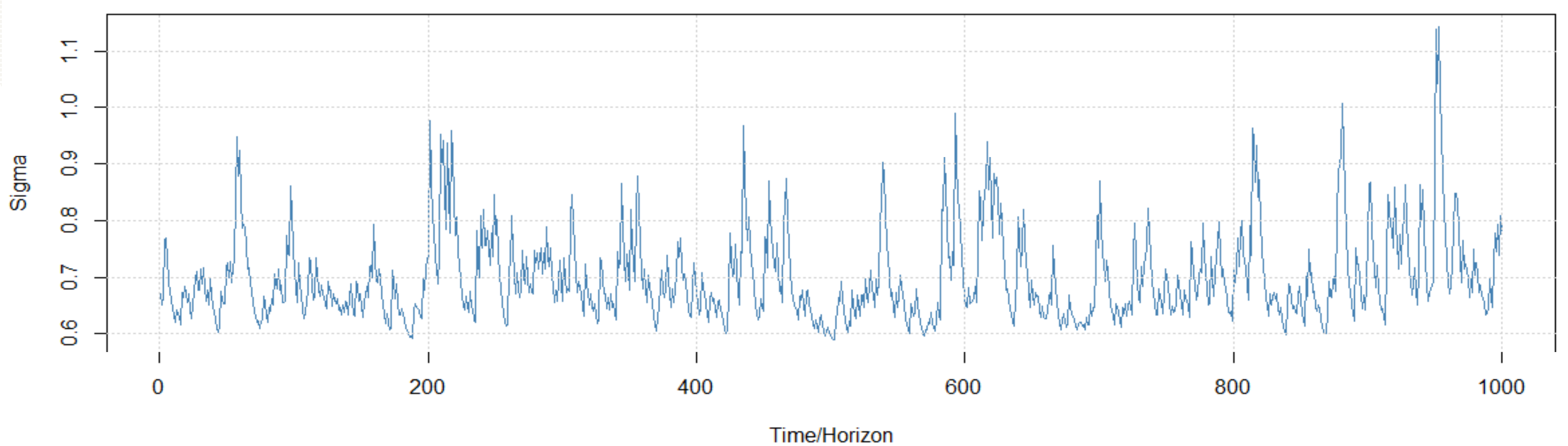
Simulated Path of Series

Simulated Returns

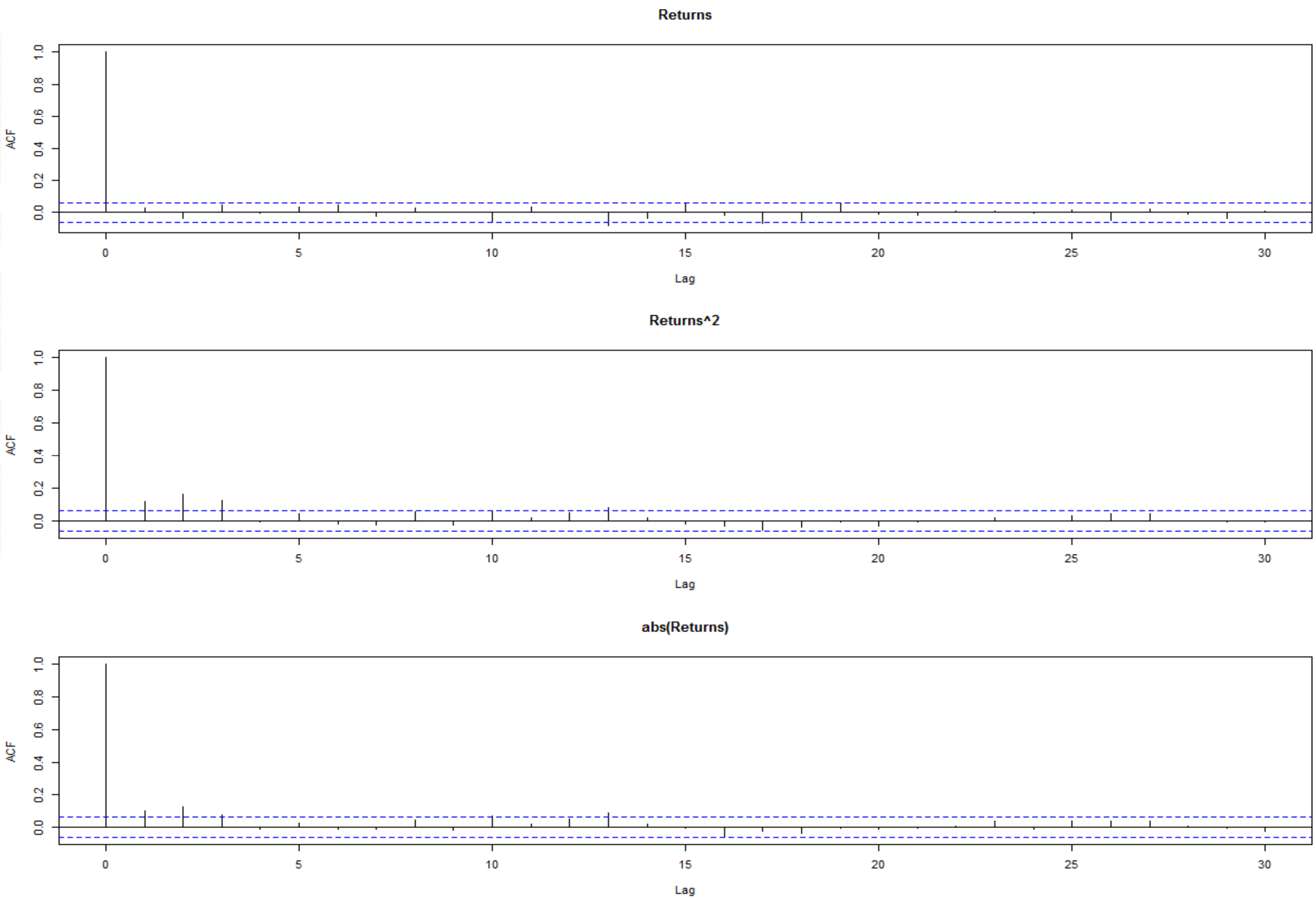


Simulated Volatility

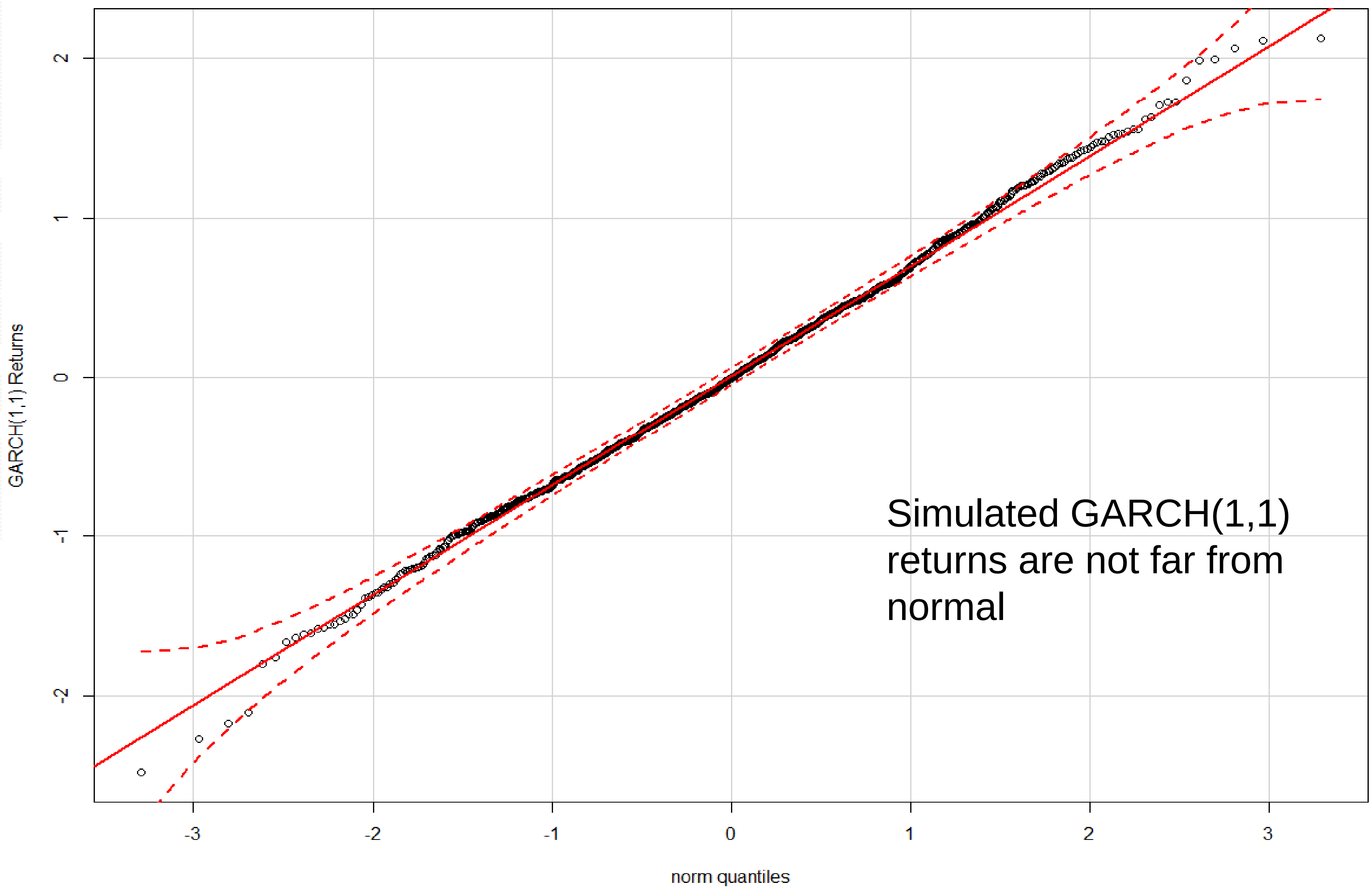
Simulated Path of Conditional Sigma



GARCH(1,1) Autocorrelations



GARCH(1,1) Normal QQ-Plot



Testing for ARCH/GARCH Effects

```
# use Box.test from stats package
> Box.test(coredata(MSFT.ret^2), type="Ljung-Box", lag = 12)

        Box-Ljung test

data:  coredata(MSFT.ret^2)
X-squared = 503.4529, df = 12, p-value < 2.2e-16

> Box.test(coredata(GSPC.ret^2), type="Ljung-Box", lag = 12)

        Box-Ljung test

data:  coredata(GSPC.ret^2)
X-squared = 2973.828, df = 12, p-value < 2.2e-16
```

Q-stat on squared returns

Testing for ARCH/GARCH Effects

```
# Engle's LM ArchTest() function from FinTS package
> ArchTest(MSFT.ret)

      ARCH LM-test; Null hypothesis: no ARCH effects

data:  MSFT.ret
Chi-squared = 246.8778, df = 12, p-value < 2.2e-16

> ArchTest(GSPC.ret)

      ARCH LM-test; Null hypothesis: no ARCH effects

data:  GSPC.ret
Chi-squared = 879.794, df = 12, p-value < 2.2e-16
```

Fit GARCH(1,1) to MSFT Returns

$$r_t = \mu + \varepsilon_t,\ \ \varepsilon_t = \sigma_t z_t$$

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

$$z_t \sim iid\ N(0,1)$$

```
# specify GARCH(1,1) with constant in mean equation
> garch11.spec = ugarchspec(variance.model=list(garchOrder=c(1,1)),
+                           mean.model = list(armaOrder=c(0,0)))

# estimate GARCH(1,1) by MLE
> MSFT.garch11.fit = ugarchfit(spec=garch11.spec, data=MSFT.ret,
+                              solver.control=list(trace = 1))

Iter: 1 fn: -8042.0206       Pars:  0.000489367 0.000004681
0.071961600 0.918437848
Iter: 2 fn: -8042.0206       Pars:  0.000489347 0.000004681
0.071962506 0.918437308
solnp--> Completed in 2 iterations
```

uGARCHfit Object

```
> class(MSFT.garch11.fit)
[1] "uGARCHfit"
attr(,"package")
[1] "rugarch“

> slotNames(MSFT.garch11.fit)
[1] "fit"   "model“

> names(MSFT.garch11.fit@fit)
 [1] "hessian"          "cvar"             "var"
 [4] "sigma"            "z"                "LLH"
 [7] "log.likelihoods" "residuals"        "coef"
[10] "robust.cvar"      "scores"           "se.coef"
[13] "tval"             "matcoef"          "robust.se.coef"
[16] "robust.tval"      "robust.matcoef"   "fitted.values"
[19] "convergence"      "kappa"            "persistence"
[22] "timer"            "ipars"            "solver"

> names(MSFT.garch11.fit@model)
 [1] "modelinc"   "modeldesc"  "modeldata"  "pars"      "start.pars"
 [6] "fixed.pars" "maxOrder"   "pos.matrix" "fmodel"      "pidx"
[11] "n.start"
```

Method Functions

Function	Description
`coef()`	Extract estimated coefficients
`infocriteria()`	Calculate information criteria for fit
`likelihood()`	Extract likelihood
`nyblom()`	Calculate Hansen-Nyblom coefficient stability test
`signbias()`	Calculate Engle-Ng sign bias test
`newsimpact()`	Calculate news impact curve
`as.data.frame()`	Extract data, fitted data, residuals and conditional vol
`sigma()`	Extract conditional volatility estimates
`residuals()`	Extract residuals
`fitted()`	Extract fitted values
`getspec()`	Extract model specification
`gof()`	Compute goodness-of-fit statistics
`uncmean()`	Extract unconditional mean
`uncvariance()`	Extract unconditional variance
`plot()`	Produce various plots
`persistence()`	Calculate persistence of fitted model
`halflife()`	Calculate half-life of fitted model

Summary of GARCH(1,1) Fit

```
> MSFT.garch11.fit

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(0,0,0)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.000489    0.000273   1.7894 0.073557
omega   0.000005    0.000001   4.6888 0.000003
alpha1  0.071963    0.010254   7.0177 0.000000
beta1   0.918437    0.011102  82.7242 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.000489    0.000298   1.6407 0.100856
omega   0.000005    0.000003   1.7766 0.075641
alpha1  0.071963    0.025959   2.7722 0.005568
beta1   0.918437    0.025206  36.4367 0.000000
```

MLE standard errors

QMLE standard errors

Summary of GARCH(1,1) Fit

```
LogLikelihood : 8042

Information Criteria
------------------------------------

Akaike       -5.2161
Bayes        -5.2083
Shibata      -5.2161
Hannan-Quinn -5.2133

Q-Statistics on Standardized Residuals
------------------------------------
      statistic p-value
Lag10     11.19  0.3430
Lag15     17.78  0.2742
Lag20     26.32  0.1554

H0 : No serial correlation

Q-Statistics on Standardized Squared Residuals
------------------------------------
      statistic p-value
Lag10     1.081  0.9998
Lag15     2.300  0.9999
Lag20     2.930  1.0000
```

Tests for ARCH/GARCH behavior in standardized residuals.

No evidence of serial correlation in squared residuals

Summary of GARCH(1,1) Fit

```
ARCH LM Tests
------------------------------------
             Statistic DoF P-Value
ARCH Lag[2]     0.2991   2  0.8611
ARCH Lag[5]     0.7069   5  0.9826
ARCH Lag[10]    1.1019  10  0.9997

Nyblom stability test
------------------------------------
Joint Statistic:  0.9803
Individual Statistics:
mu     0.09264
omega  0.06068
alpha1 0.33424
beta1  0.12796

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:           1.07 1.24 1.6
Individual Statistic:      0.35 0.47 0.75
```

Tests for coefficient stability (structural change)

No evidence for unstable parameters

Summary of GARCH(1,1) Fit

```
Sign Bias Test
------------------------------------
                     t-value    prob sig
Sign Bias             2.1124 0.03473  **
Negative Sign Bias    0.8984 0.36904
Positive Sign Bias    0.2570 0.79721
Joint Effect          5.2995 0.15114
```

Tests for leverage effects (discuss later)

Some evidence of asymmetric effects

```
Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     125.1    1.233e-17
2    30     133.7    2.195e-15
3    40     156.9    4.420e-16
4    50     165.8    1.351e-14
```

Tests for Distribution goodness-of-fit

Normal distribution assumption is strongly rejected!

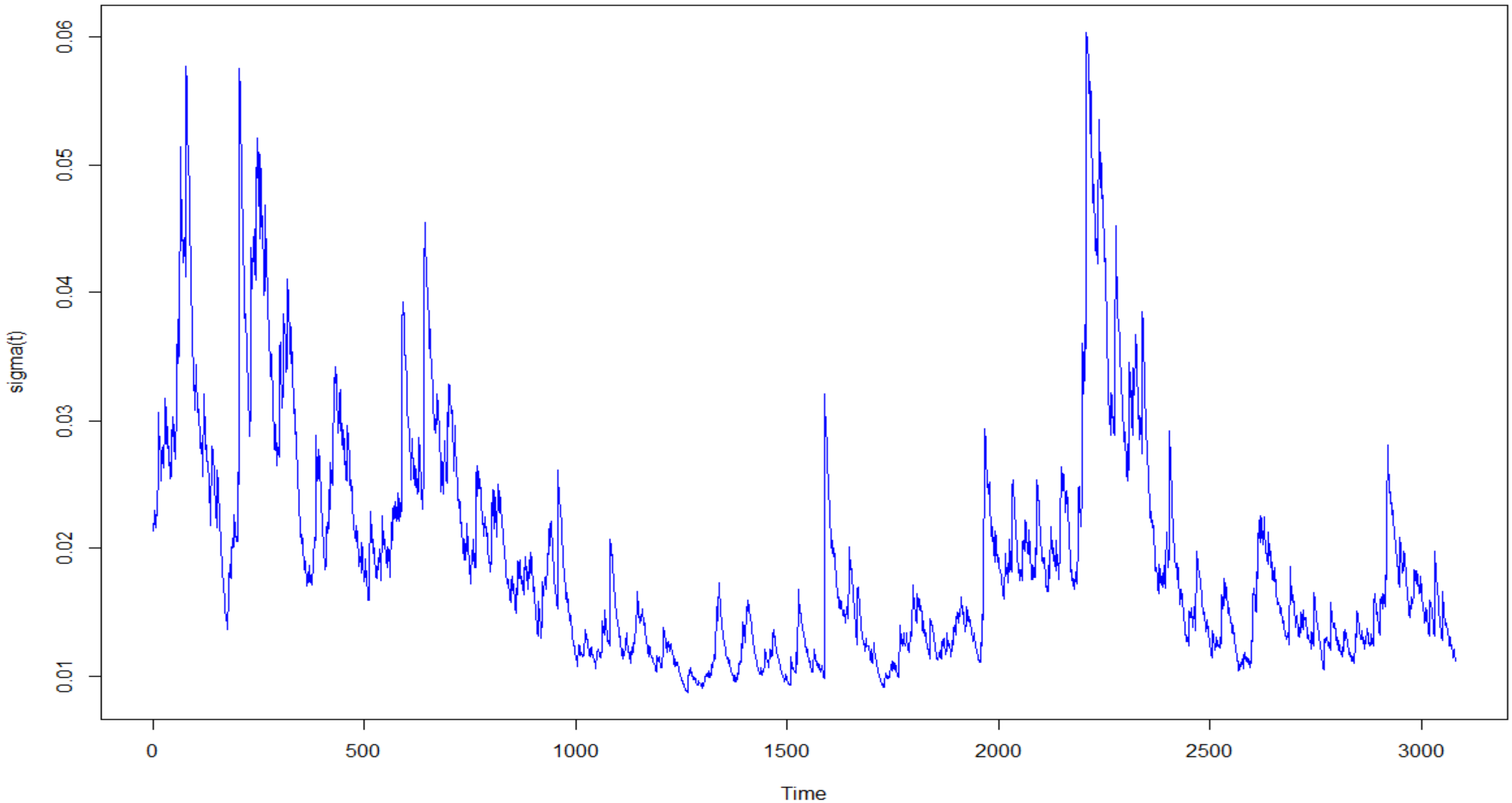
```
Elapsed time : 0.4252
```

Extractor Functions

```
# estimated coefficients
> coef(MSFT.garch11.fit)
       mu      omega     alpha1      beta1
4.893e-04 4.681e-06 7.196e-02 9.184e-01
# unconditional mean in mean equation
> uncmean(MSFT.garch11.fit)
       mu
0.0004893
# unconditional variance: omega/(alpha1 + beta1)
> uncvariance(MSFT.garch11.fit)
unconditional
    0.0004876
# persistence: alpha1 + beta1
> persistence(MSFT.garch11.fit)
persistence
     0.9904
# half-life: ln(0.5)/(ln(alpha1 + beta1))
> halflife(MSFT.garch11.fit)
Half-Life
    71.85
```

Conditional Volatility: σ_t

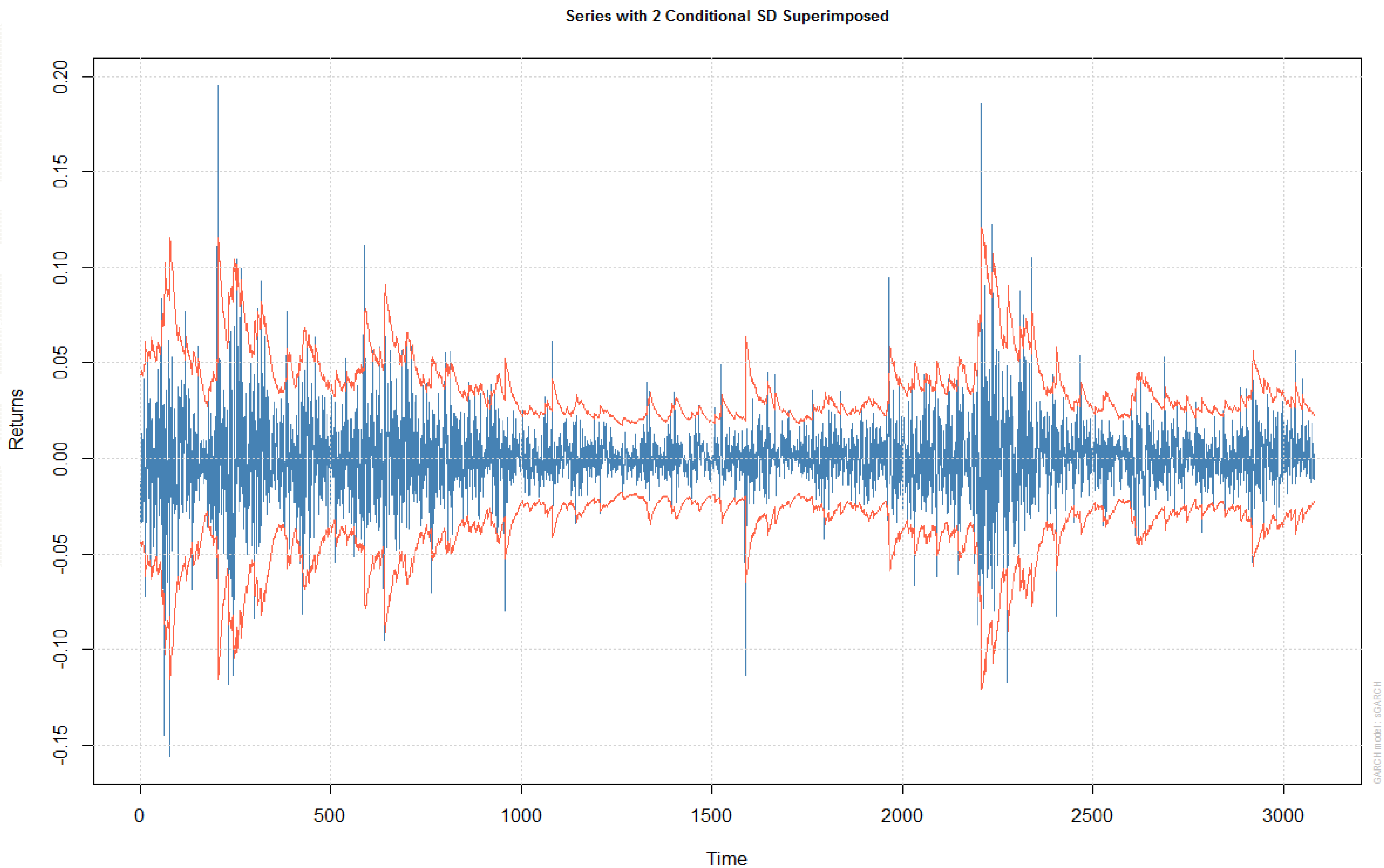
```
> plot.ts(sigma(MSFT.garch11.fit), ylab="sigma(t)", col="blue")
```



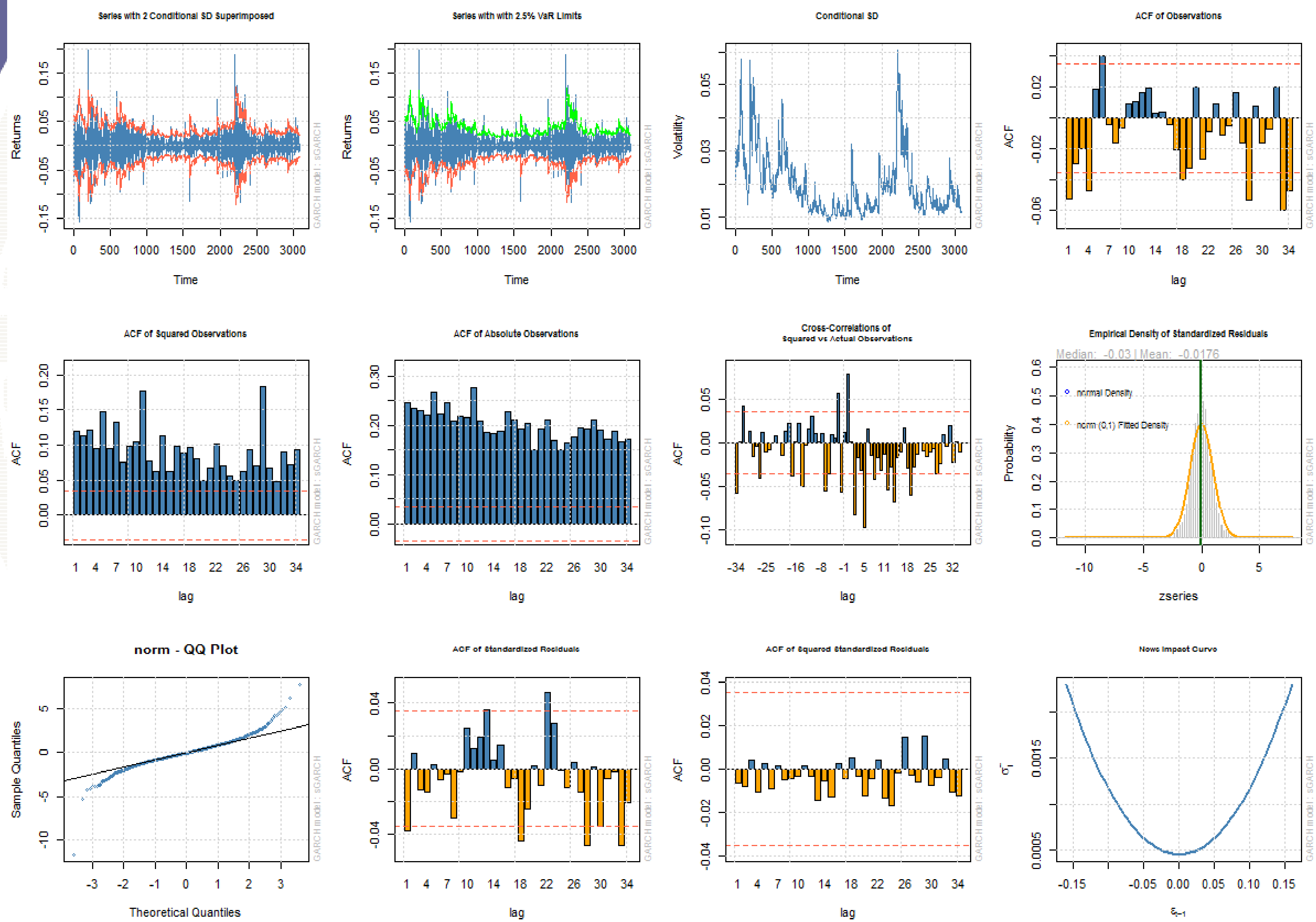
uGARCHfit plot method

```
> plot(MSFT.garch11.fit)

Make a plot selection (or 0 to exit):

 1:   Series with 2 Conditional SD Superimposed
 2:   Series with 2.5% VaR Limits (with unconditional mean)
 3:   Conditional SD
 4:   ACF of Observations
 5:   ACF of Squared Observations
 6:   ACF of Absolute Observations
 7:   Cross Correlation
 8:   Empirical Density of Standardized Residuals
 9:   QQ-Plot of Standardized Residuals
10:   ACF of Standardized Residuals
11:   ACF of Squared Standardized Residuals
12:   News-Impact Curve

Selection:
```

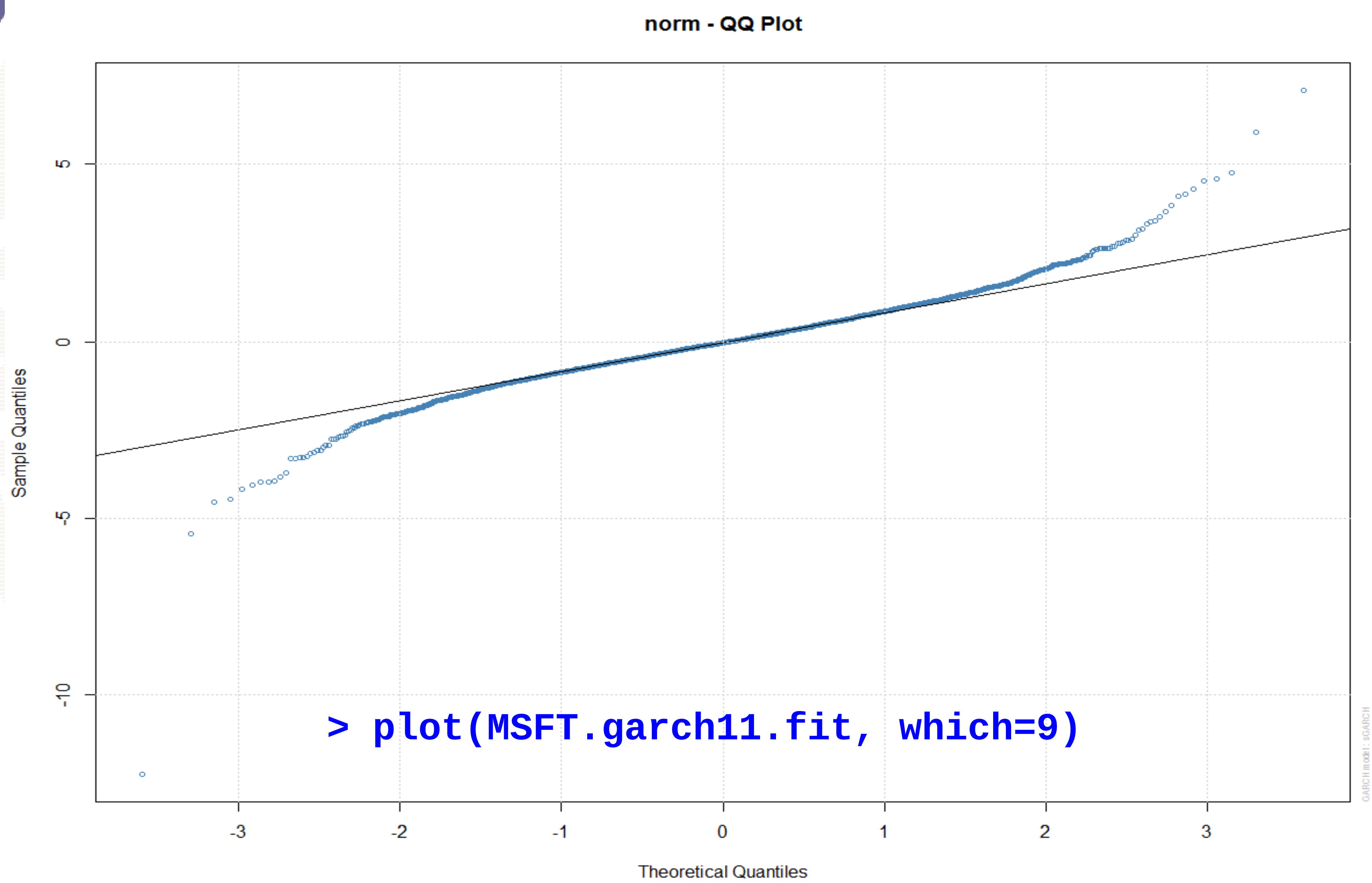
Plot Method: `plot(x, which=1)`



Plot Method : `plot(x, which="all")`



Normality Assumption is Bad




```
> plot(MSFT.garch11.fit, which=9)
```

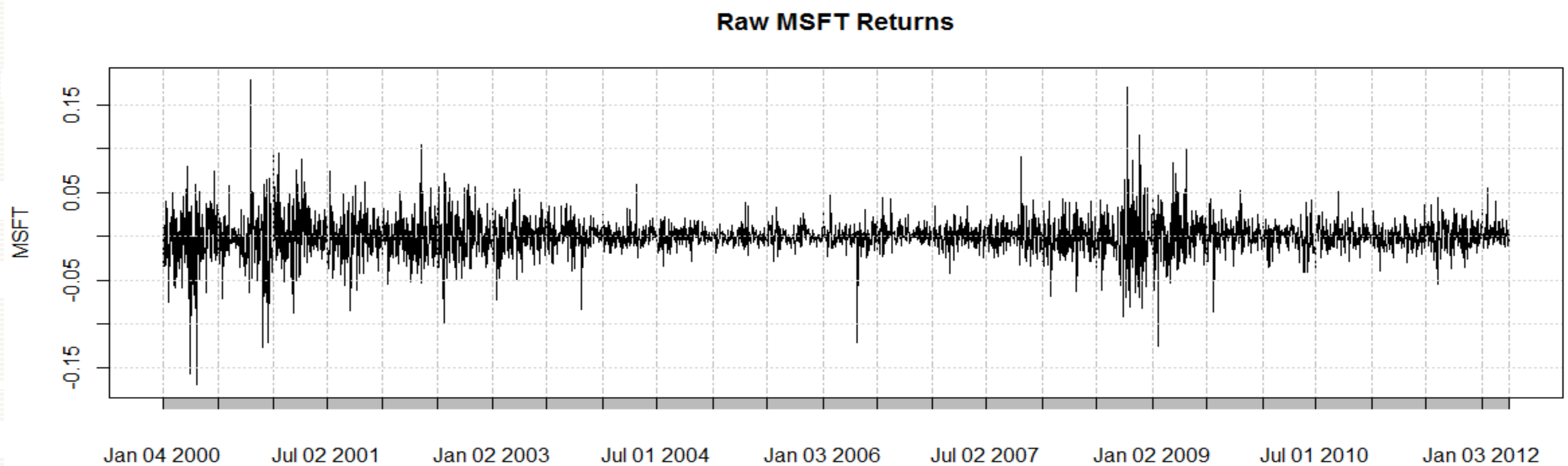
Convergence Problems

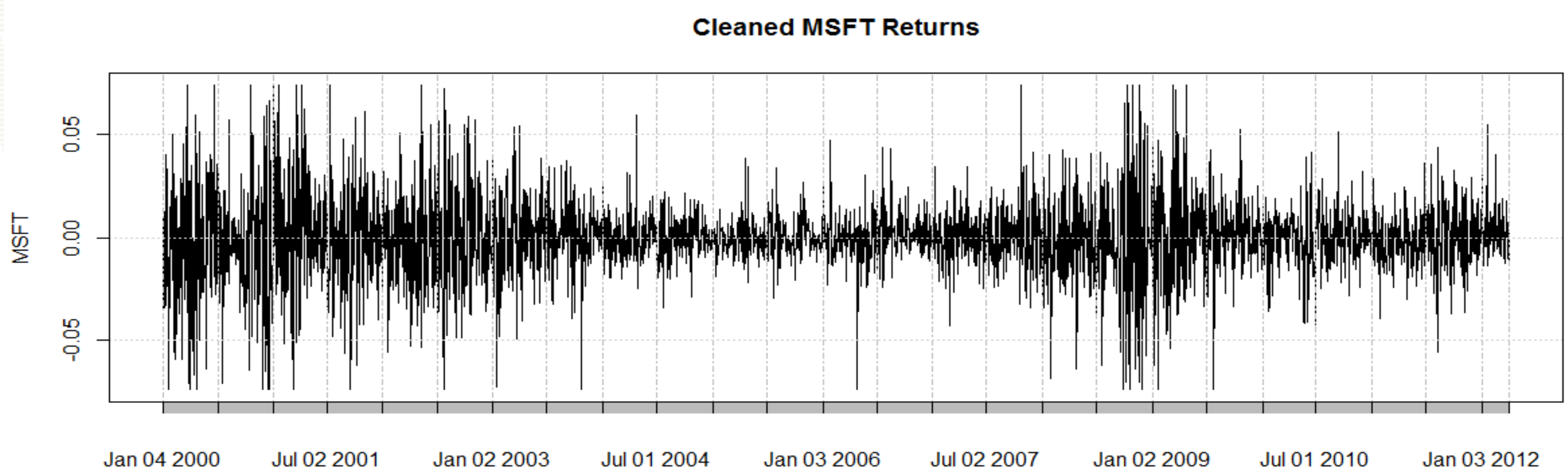
```
# Fit ARCH(1) to MSFT
> arch1.spec = ugarchspec(variance.model = list(garchOrder=c(1,0)),
+                          mean.model = list(armaOrder=c(0,0)))
> MSFT.arch1.fit = ugarchfit(spec=arch1.spec, data=MSFT.ret,
+                             solver.control=list(trace = 1))

Iter: 1 fn: -7659.3402  Pars:  -0.0004496  0.0002867  0.4711622
solnp--> Solution not reliable....Problem Inverting Hessian.
```

Convergence problems could be due to some extreme observations in the data. Sometime “cleaning” the data of “outliers” can help with convergence

Cleaned Data

```
> MSFT.ret.clean = Return.clean(MSFT.ret, method="boudt")
```





ARCH(1) on Cleaned Data

```
> MSFT.clean.arch1.fit = ugarchfit(spec=arch1.spec, data=MSFT.ret.clean,
+                                   solver.control=list(trace = 1))

Iter: 1 fn: -7845.5508     Pars:  -0.0001778  0.0002673  0.3586366
Iter: 2 fn: -7845.5508     Pars:  -0.0001777  0.0002673  0.3586415
solnp--> Completed in 2 iterations
> MSFT.clean.arch1.fit

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,0)
Mean Model       : ARFIMA(0,0,0)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu     -0.000178    0.000309 -0.57434  0.56574
omega   0.000267    0.000009 28.64710  0.00000
alpha1  0.358641    0.034496 10.39666  0.00000
```

Model Selection

```
> arch.order = 1:5
> arch.names = paste("arch", arch.order, sep="")

# fit all arch models with p <= 5
> arch.list = list()
> for (p in arch.order) {
+   arch.spec = ugarchspec(variance.model = list(garchOrder=c(p,0)),
+                          mean.model = list(armaOrder=c(0,0)))
+   arch.fit = ugarchfit(spec=arch.spec, data=MSFT.ret.clean,
+                        solver.control=list(trace = 0))
+   arch.list[[p]] = arch.fit
+ }

> names(arch.list) = arch.names

# Add GARCH(1,1) refit to cleaned data to list
> arch.list$garch11 = garch11.fit
```

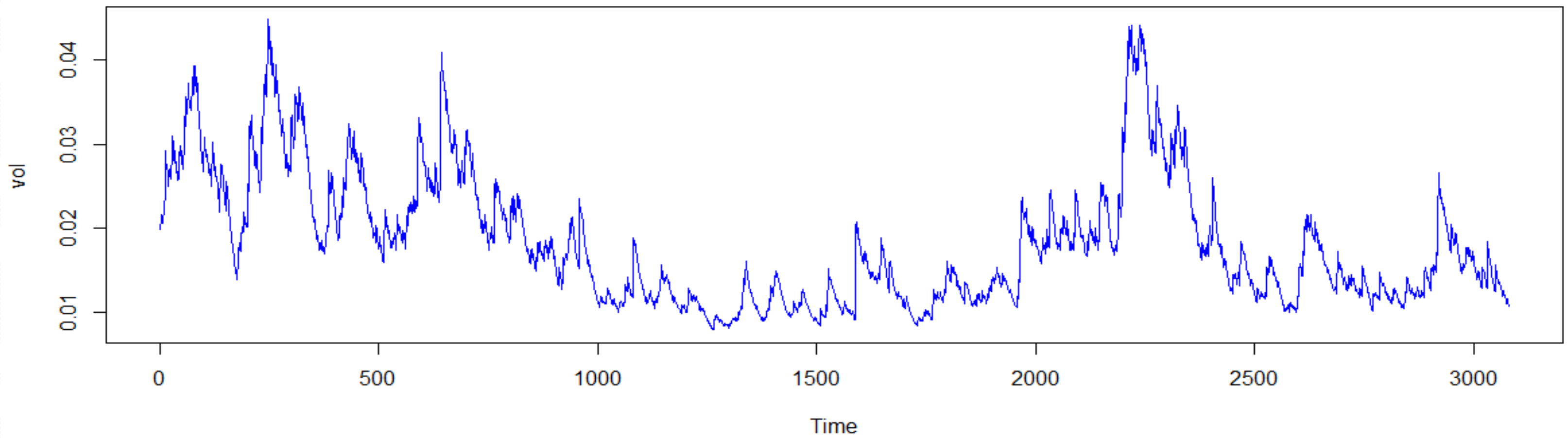
Model Selection

```
# Compute information criteria using infocriteria() function
> info.mat = sapply(arch.list, infocriteria)
> rownames(info.mat) = rownames(infocriteria(arch.list[[1]]))
> info.mat
                arch1  arch2  arch3  arch4  arch5 garch11
Akaike         -5.089 -5.140 -5.180 -5.218 -5.243  -5.319
Bayes          -5.083 -5.132 -5.170 -5.206 -5.230  -5.311
Shibata        -5.089 -5.140 -5.180 -5.218 -5.243  -5.319
Hannan-Quinn -5.087 -5.137 -5.177 -5.213 -5.238  -5.316
```

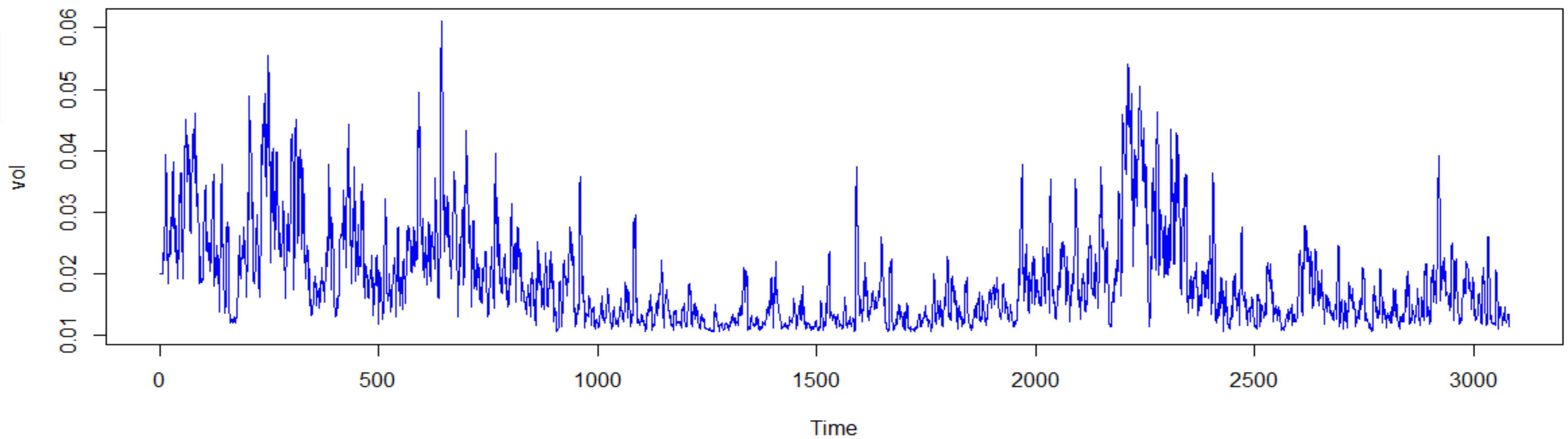
GARCH(1,1) has the best fit –
smallest values of info criteria

ARCH(5) vs. GARCH(1,1)

GARCH(1,1) conditional vol



ARCH(5) conditional vol



GARCH(1,1) Forecasts

```
# Compute h-step ahead forecasts for h=1,…,100
> MSFT.garch11.fcst = ugarchforecast(MSFT.garch11.fit,
+                                    n.ahead=100)
> class(MSFT.garch11.fcst)
[1] "uGARCHforecast"
attr(,"package")
[1] "rugarch"

> slotNames(MSFT.garch11.fcst)
[1] "forecast" "model"

> names(MSFT.garch11.fcst@forecast)
[1] "n.ahead"   "N"         "n.start"   "n.roll"    "forecasts"
[6] "fdates"
```

GARCH(1,1) Forecasts

```
> MSFT.garch11.fcst

*------------------------------------*
*       GARCH Model Forecast         *
*------------------------------------*
Model: sGARCH
Horizon: 100
Roll Steps: 0
Out of Sample: 0

0-roll forecast:
             sigma    series
2012-04-04 0.01136 0.0003397
2012-04-05 0.01151 0.0003397
2012-04-06 0.01166 0.0003397
2012-04-09 0.01180 0.0003397
2012-04-10 0.01194 0.0003397
…
```

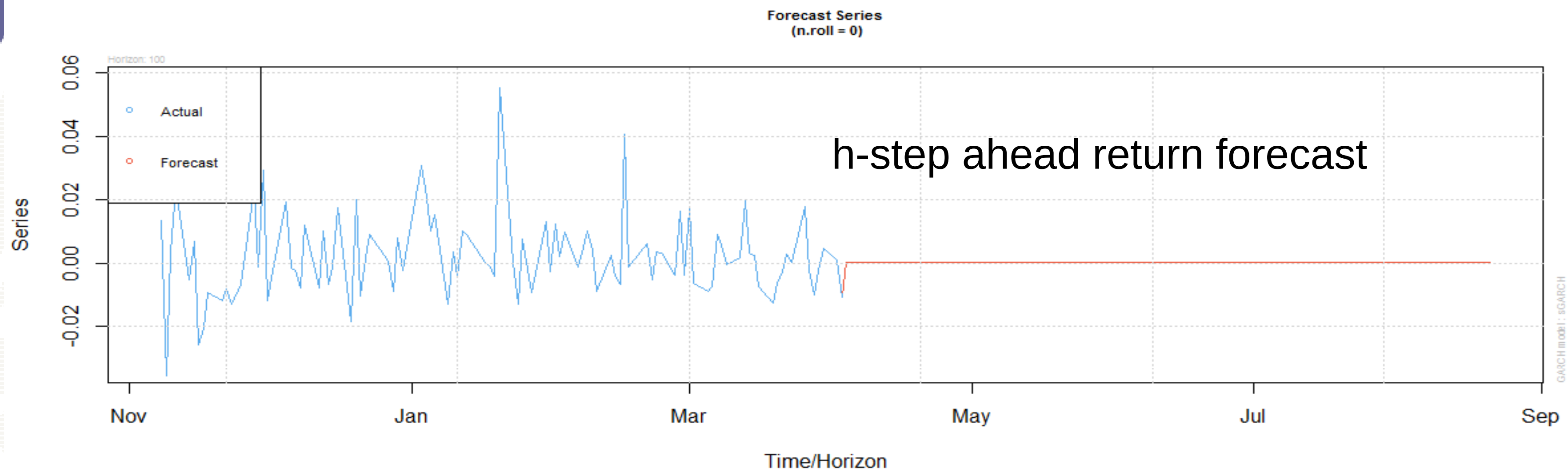
Forecast Object Method Functions

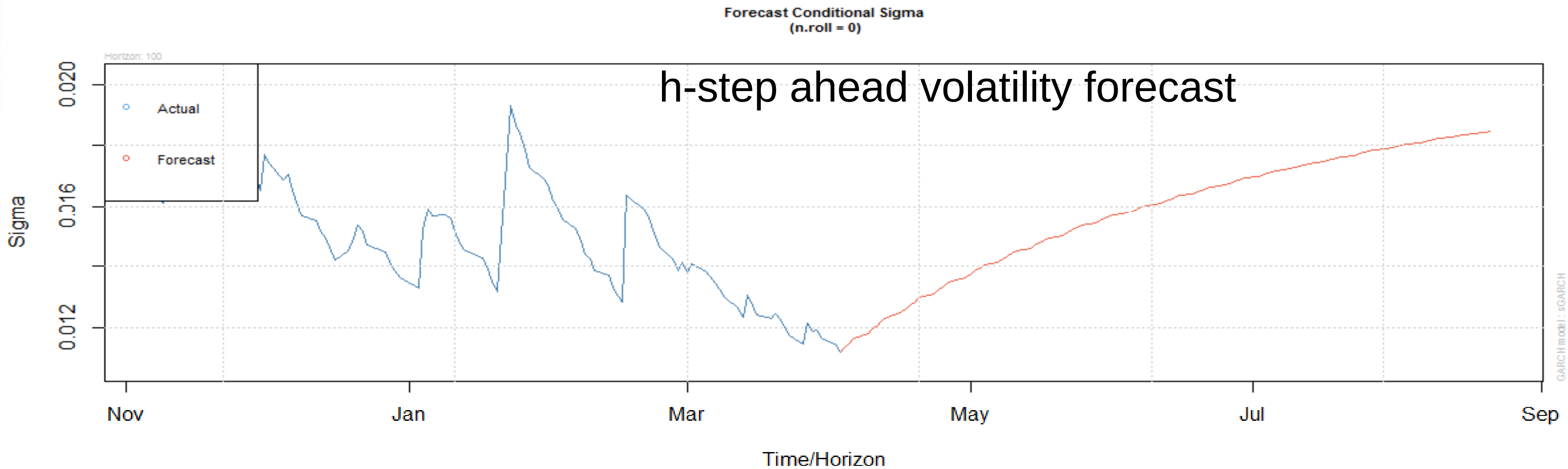
Function	Description
`as.array`	Extracts the forecast array
`as.data.frame`	Extracts the forecasts
`as.list`	Extracts the forecast list will all rollframes
`plot`	Forecasts plots
`fpm`	Forecast performance measures
`show`	Forecast summary

GARCH(1,1) Forecasts

```
> plot(MSFT.garch11.fcst)

Make a plot selection (or 0 to exit):

1:    Time Series Prediction (unconditional)
2:    Time Series Prediction (rolling)
3:    Conditional SD Prediction

Selection:
```

GARCH(1,1) Forecasts

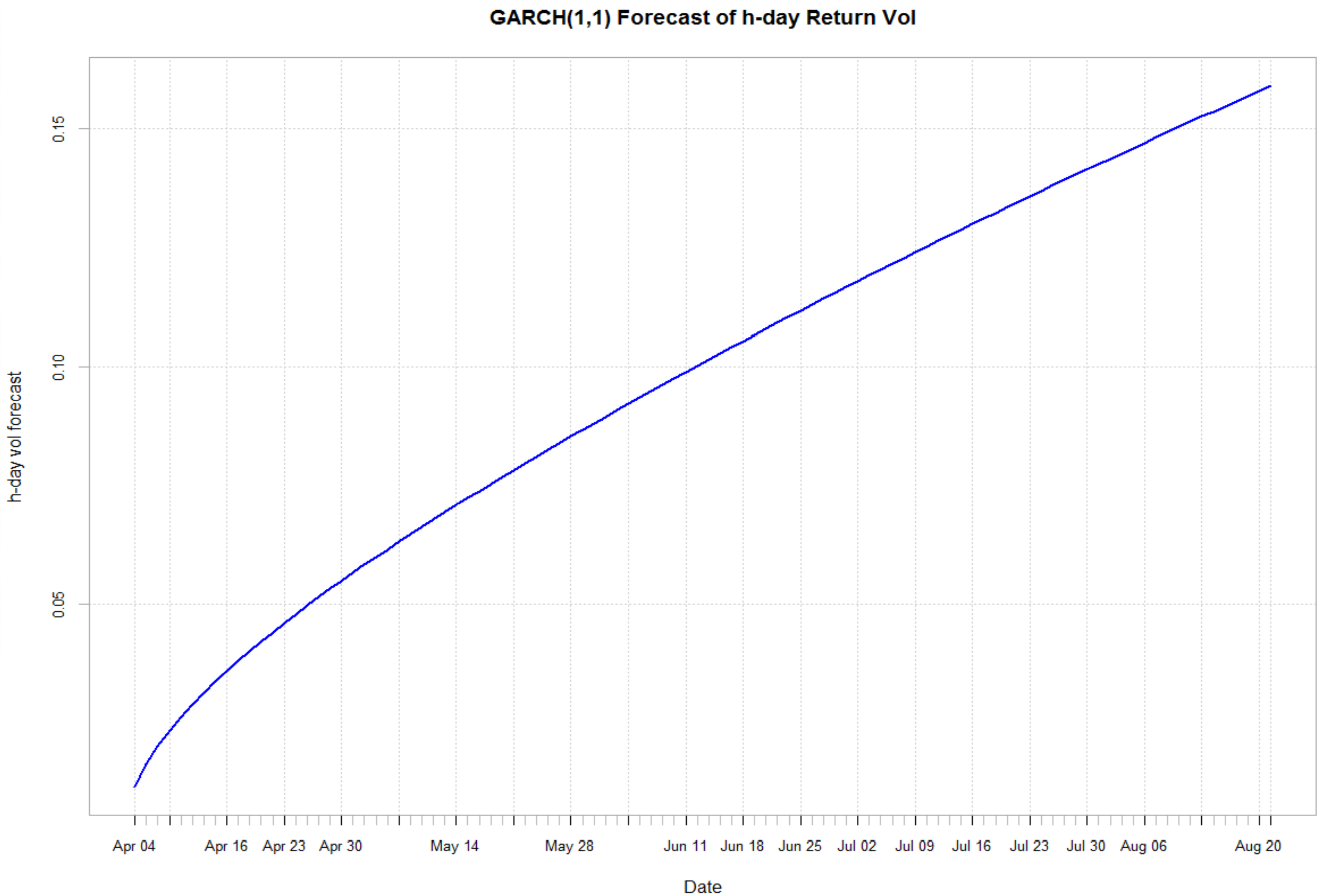




Forecasts of h-day Return Vol

```
# Extract forecasts into data.frame
> MSFT.fcst.df = as.data.frame(MSFT.garch11.fcst)
> head(MSFT.fcst.df)
             sigma    series
2012-04-04 0.01136 0.0003397
2012-04-05 0.01151 0.0003397
2012-04-06 0.01166 0.0003397
2012-04-09 0.01180 0.0003397
2012-04-10 0.01194 0.0003397
2012-04-11 0.01208 0.0003397

# h-day return variance forecast = sum of h-day ahead
# variance forecasts
> fcst.var.hDay = cumsum(MSFT.fcst.df$sigma^2)
> fcst.vol.hDay = sqrt(MSFT.fcst.var.hDay)
```

Forecasts of h-day Return Vol



Conditional VaR Forecasts

```
# h step-ahead conditional normal GARCH(1,1) VaR
> VaR.95.garch11 = MSFT.fcst.df$series[1] +
+                  MSFT.fcst.df$sigma[1]*qnorm(0.05)
> VaR.95.garch11
[1] -0.01835

# compute 20-day vol forecast from fitted GARCH(1,1)
> sigma.20day = sqrt(sum(MSFT.fcst.df$sigma[1:20]^2))
> VaR.95.garch11.20day = 20*MSFT.fcst.df$series[1] +
+                        sigma.20day*qnorm(0.05)
> VaR.95.garch11.20day
[1] -0.08621
```