Volatility Modeling: Case Study 4

Dr. Kempthorne

October 8, 2013

Contents

1 Volatility Modeling of Exchange Rate Returns

1.1 Load libraries and Federal Reserve FX Data

```
> # 0.1 Install/load libraries
> source(file="fm_casestudy_0_InstallOrLoadLibraries.r")
> library("zoo")
> # 0.2 Load R workspace created by script fm_casestudy_fx_1.r
> load(file="fm_casestudy_fx_1.Rdata")
> # 1.0 Extract time series matrix of exchange rates for symbols given by list.symbol0 ----
>
> list.symbol0<-c("DEXCHUS", "DEXJPUS", "DEXKOUS", "DEXMAUS",
+                 "DEXUSEU", "DEXUSUK", "DEXTHUS", "DEXSZUS")
> fxrates000<-fred.fxrates.00[,list.symbol0]
> dim(fxrates000)

[1] 3709    8

> head(fxrates000)

           DEXCHUS DEXJPUS DEXKOUS DEXMAUS DEXUSEU DEXUSUK DEXTHUS DEXSZUS
1999-01-04  8.2793  112.15  1187.5     3.8  1.1812  1.6581   36.20  1.3666
1999-01-05  8.2795  111.15  1166.0     3.8  1.1760  1.6566   36.18  1.3694
1999-01-06  8.2795  112.78  1160.0     3.8  1.1636  1.6547   36.50  1.3852
1999-01-07  8.2798  111.69  1151.0     3.8  1.1672  1.6495   36.30  1.3863
1999-01-08  8.2796  111.52  1174.0     3.8  1.1554  1.6405   36.45  1.3970
1999-01-11  8.2797  108.83  1175.0     3.8  1.1534  1.6375   36.28  1.3963

> tail(fxrates000)

           DEXCHUS DEXJPUS DEXKOUS DEXMAUS DEXUSEU DEXUSUK DEXTHUS DEXSZUS
2013-09-20  6.1210   99.38 1076.02  3.1640  1.3522  1.6021   31.04  0.9104
2013-09-23  6.2842   98.76 1073.90  3.1990  1.3520  1.6066   31.18  0.9100
2013-09-24  6.1208   98.76 1074.35  3.2150  1.3490  1.6006   31.27  0.9114
2013-09-25  6.1190   98.62 1076.42  3.2210  1.3536  1.6080   31.22  0.9082
2013-09-26  6.1194   98.95 1075.20  3.2145  1.3484  1.6012   31.16  0.9105
2013-09-27  6.1179   98.30 1074.38  3.2260  1.3537  1.6135   31.28  0.9050

>
> # Print symbol/description/units of these rates from data frame  fred.fxrates.doc

> options(width=120)
> print(fred.fxrates.doc[match(list.symbol0, fred.fxrates.doc$symbol),
+                        c("symbol0", "fx.desc", "fx.units")])

  symbol0                               fx.desc                      fx.units
3 DEXCHUS       China / U.S. Foreign Exchange Rate      Chinese Yuan to  1 U.S. $
7 DEXJPUS       Japan / U.S. Foreign Exchange Rate      Japanese Yen to  1 U.S. $
```

```
8  DEXKOUS South Korea / U.S. Foreign Exchange Rate  South Korean Won to  1 U.S. $
9  DEXMAUS    Malaysia / U.S. Foreign Exchange Rate Malaysian Ringgit to  1 U.S. $
20 DEXUSEU        U.S. / Euro Foreign Exchange Rate              U.S. $ to  1 Euro
22 DEXUSUK        U.S. / U.K. Foreign Exchange Rate     U.S. $ to  1 British Pound
18 DEXTHUS    Thailand / U.S. Foreign Exchange Rate         Thai Baht to  1 U.S. $
16 DEXSZUS Switzerland / U.S. Foreign Exchange Rate      Swiss Francs to  1 U.S. $
```

```
> source("test_vol1b.r")
>
```

1.2 Geometric Brownian Motion Model (two time scales)

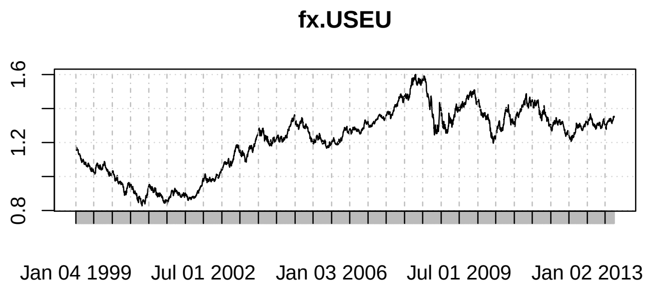
Case 1: EUR/USD Exchange Rate Returns

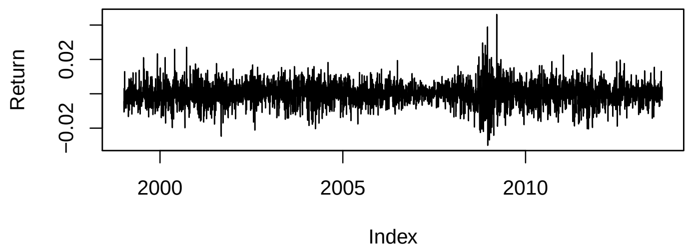
The object *fx.USEU* is a "zoo" time series object of daily US/Euro exchange rates from 1999 to 2013. The function *fcn.itsreturns* computes log returns of irregular daily time series and provides time information including the number of days in the return, the starting date of the return, the ending date of the return, and the day-of-week of the ending date.

```
> par(mfcol=c(2,1))
> plot(fx.USEU)
> fx.USEU.itsreturns<-fcn.itsreturns(fx.USEU)
> head(fx.USEU.itsreturns)
```

```
                    ret ndays date.end date.start dayofweek
1999-01-05 -0.004412021     1    10596      10595         3
1999-01-06 -0.010600202     1    10597      10596         4
1999-01-07  0.003089071     1    10598      10597         5
1999-01-08 -0.010161114     1    10599      10598         6
1999-01-11 -0.001732502     3    10602      10599         2
1999-01-12  0.001213067     1    10603      10602         3
```

```
> plot(zoo(fx.USEU.itsreturns[,"ret"], order.by=as.Date(fx.USEU.itsreturns[,"date.end"])),
+ ylab="Return")
```





The next page presents two plots:

- A histogram of the returns is created and the Gaussian/normal density corresponding to the maximum-likelihood fit is drawn.

  This graph may appear to display a reasonable fit, but other diagnostics are useful to evaluate whether the distribution is heavier-tailed than a Gaussian distribution.

- A normal qq-plot of the returns is created. The sample of returns is sorted from smallest to largest:

  $$y_{[1]} \leq y_{[2]} \leq \cdots \leq y_{[n]}.$$

  Consider a sample of size n from a $N(0,1)$ distribution, $X_1, X_2, \ldots, X_n$. Define the order statistics as the sorted sample elements:

  $$X_{[1]} \leq X_{[2]} \leq \cdots \leq X_{[n]}$$

  $X_{[1]}$ is the smallest value in the sample,

  $X_{[j]}$ is the j-th smallest value in the sample, and

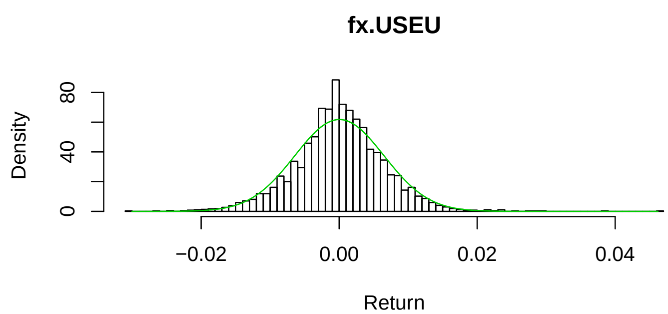
  $X_{[n]}$ is the largest value in the sample.

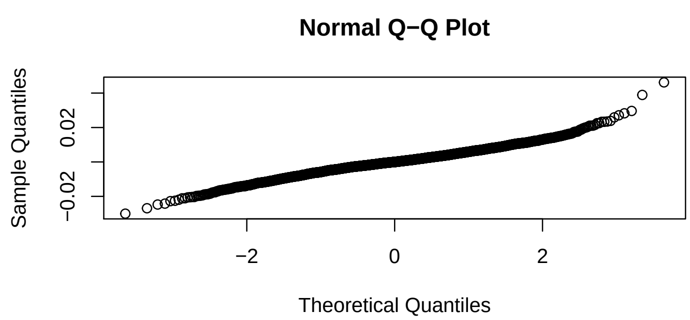
  Define $x_1, \ldots, x_n$ such that

  $$x_j = E[X_{[j]} \mid n], \text{ the expected value conditional on } n.$$

The qq-plot, plots the points (x_j, y_j). If the sample $\{y_j\}$ is consistent with the Gaussian assumption, then the points will fall close to a straight line (with slope equal to the Gaussian standard deviation, and y intercept equal to the Gaussian mean).

In this plot, it is apparent that the upper-tail of the sample has values larger than would be expected from a Gaussian distribution.





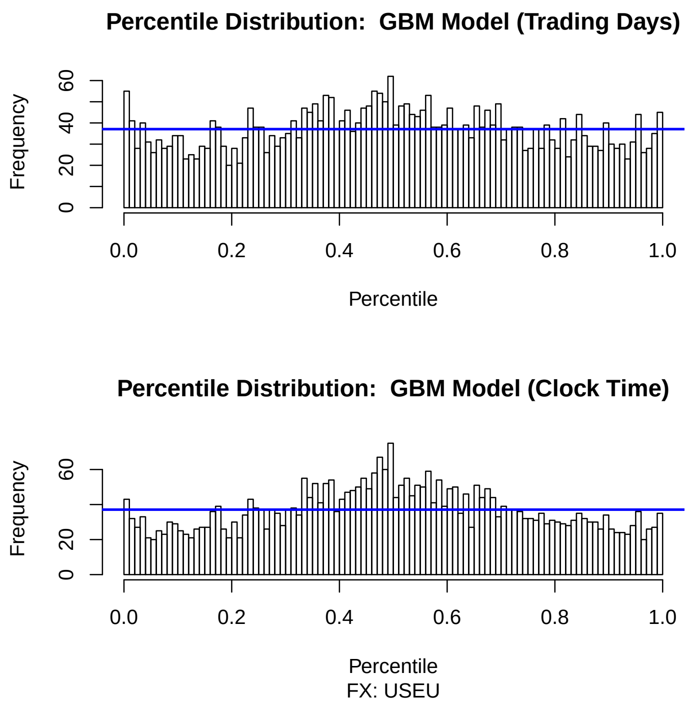
The Geometric Brownian Motion model can be fit on two time scales: trading days, and calendar days. Returns over weekends and holidays would have larger variances than returns overnight during the week, when the time scale is calendar days. The following function compares the fits under these two cases of time scale. The parameter estimates under the two cases are printed out.

Histograms of the fitted percentiles are also presented for the two models. If the data arise from the assumed model, then the fitted percentiles whould be uniformly distributed.

A horizontal line is drawn corresponding to the uniform distribution. Note that the 1% Percentile of the fitted Gaussian Distribution for Trading Days is exceeded negatively more than would be expected. This is consistent with a heavier down-side tail distribution than that of a Gaussian.

The same percetile when fitted on the Calendar Days time scale is exceeded about the same as would be expected.

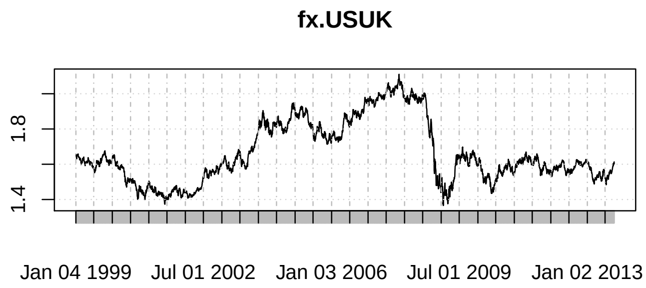
```
> fcn.gbm.compare(fx.USEU, sub="FX: USEU")

 Geometric Brownian Motion Model:

         Parameter Estimates under Scale 1 (Trading Days)

         mu=0.00926383  Sigma = 0.10251048


         Parameter Estimates under Scale 2 (Clock Time)

         mu=2.534e-05  Sigma = 0.12275173
```

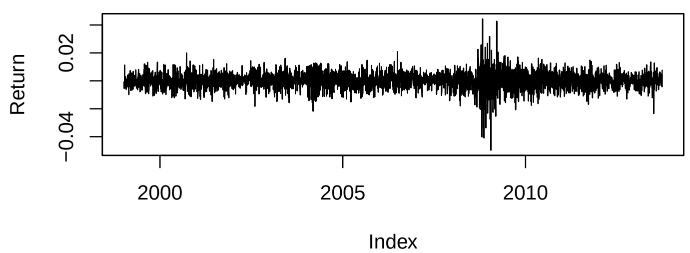


Case 2: GPB/USD Exchange Rate Returns

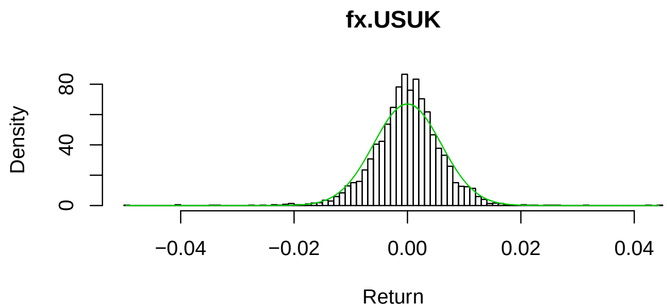
The same computations are applied to the the British Pound/US-Dollar exchange rate.

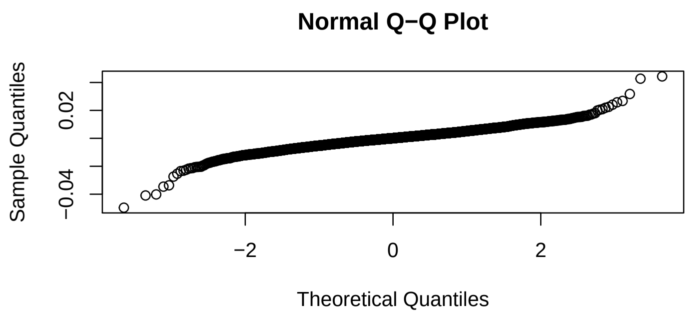
```
> par(mfcol=c(2,1))
> plot(fx.USUK)
> fx.USUK.itsreturns<-fcn.itsreturns(fx.USUK)
> #head(fx.USEU.itsreturns)
> plot(zoo(fx.USUK.itsreturns[,"ret"], order.by=as.Date(fx.USUK.itsreturns[,"date.end"])),
+       ylab="Return")
>
```



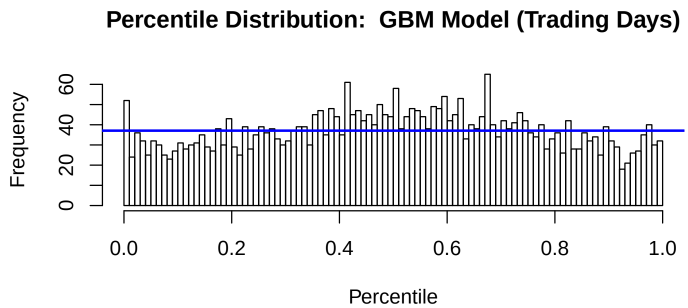


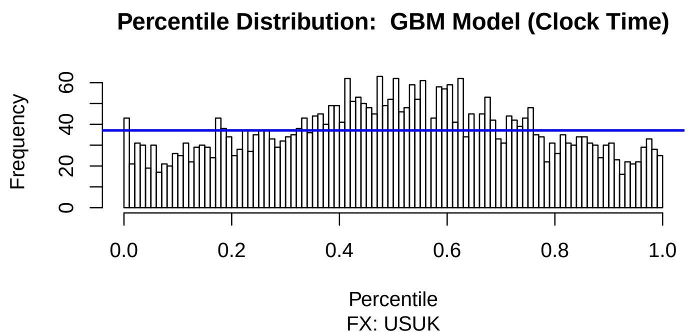

```
> par(mfcol=c(2,1))
> hist(fx.USUK.itsreturns[,"ret"], nclass=100, xlab="Return", main="fx.USUK",probability=TRU
> sy<-sort(fx.USUK.itsreturns[,"ret"])
> sy.dnorm<-dnorm(sy, mean=mean(sy), sd=sqrt(var(sy)))
> lines(sy, sy.dnorm, col=3)
> qqnorm(sy)
```






```
> fcn.gbm.compare(fx.USUK, sub="FX: USUK")

 Geometric Brownian Motion Model:

         Parameter Estimates under Scale 1 (Trading Days)

         mu=-0.00185307  Sigma = 0.09458791


         Parameter Estimates under Scale 2 (Clock Time)

         mu=-5.07e-06  Sigma = 0.11192779
```





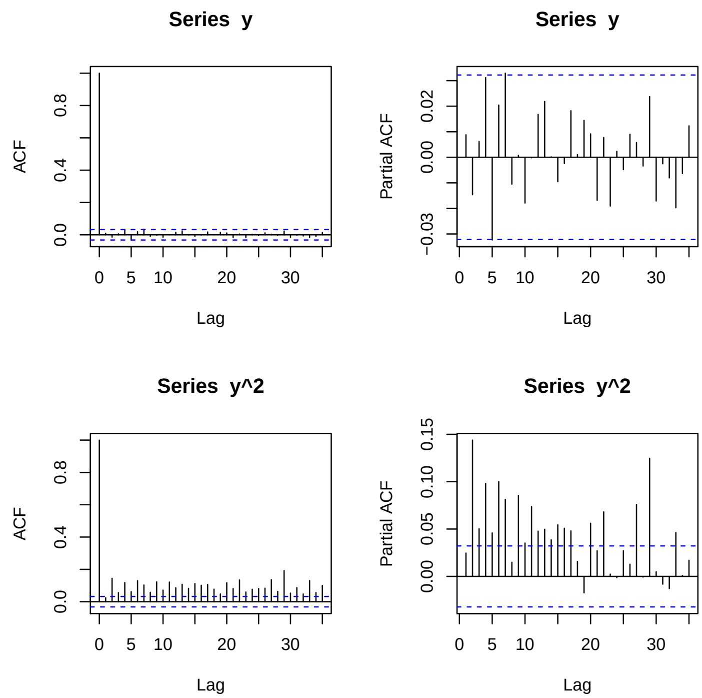
1.3 Time Dependence in Squared-Returns

Case: EUR/USD Exchange Rate Returns

Non-linear time dependence in the time series of exchange rate returns is exhibited with the time dependence of the squared returns.

The auto-correlation function (ACF) and the partial autocorrelation function (PACF) are computed for the exchange rate returns and for their squared values. Marginally significant time dependence is present in the returns, while highly significant time dependence is apparent in the squared returns. (The blue lines in the plots are at +/- two standard deviations for the sample correlation coefficients under the null hypothesis of no time-series dependence.)

```
> y<-fx.USEU.itsreturns[,"ret"]
> par(mfrow=c(2,2))
> acf(y)
> acf(y,type="partial")
> acf(y^2)
> acf(y^2,type="partial")
```



1.4 Gaussian ARCH and GARCH Models

Case: EUR/USD Exchange Rate Returns

The following models are fit the EUR/USD exchange rate:

- $ARCH(1)$
- $ARCH(2)$
- $ARCH(10)$
- $GARCH(1,1)$

The R function $garch()$ is fits both $ARCH(p)$ and $GARCH(p,q)$ models by maximum likelihood, assuming Gaussian distributions for the model innovations.

```
> y.arch1<-garch(y, order=c(0,1),trace=FALSE)
> y.arch2<-garch(y, order=c(0,2),trace=FALSE)
> y.arch10<-garch(y, order=c(0,10),trace=FALSE)
> y.garch11<-garch(y,order=c(1,1),trace=FALSE)
> options(show.signif.stars=FALSE)
> # print out the ARCH fitted model summaries
> summary(y.arch1)
```

```
Call:
garch(x = y, order = c(0, 1), trace = FALSE)

Model:
GARCH(0,1)

Residuals:
    Min      1Q  Median      3Q     Max
-4.6892 -0.5491  0.0000  0.5824  6.9893

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 4.052e-05   8.986e-07   45.086   <2e-16
a1 2.852e-02   1.182e-02    2.412   0.0159

Diagnostic Tests:
        Jarque Bera Test

data:  Residuals
X-squared = 626.6375, df = 2, p-value < 2.2e-16


        Box-Ljung test

data:  Squared.Residuals
X-squared = 0.0735, df = 1, p-value = 0.7863

> summary(y.arch2)

Call:
garch(x = y, order = c(0, 2), trace = FALSE)

Model:
GARCH(0,2)

Residuals:
    Min      1Q  Median      3Q     Max
-3.8419 -0.5550  0.0000  0.5858  7.3479

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 3.621e-05   9.741e-07   37.177  < 2e-16
a1 2.644e-02   1.171e-02    2.258    0.024
a2 1.001e-01   1.319e-02    7.586 3.29e-14

Diagnostic Tests:
```

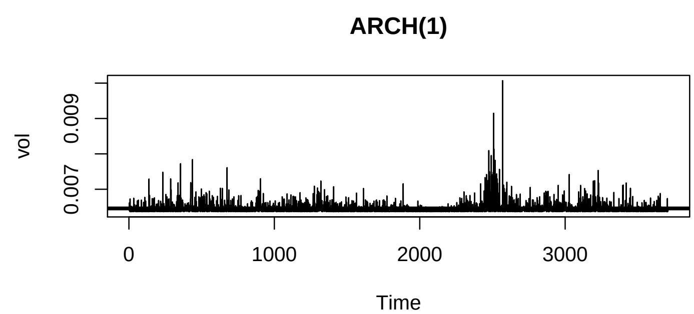
```
        Jarque Bera Test

data:  Residuals
X-squared = 459.1684, df = 2, p-value < 2.2e-16


        Box-Ljung test

data:  Squared.Residuals
X-squared = 0.0965, df = 1, p-value = 0.756

> summary(y.arch10)

Call:
garch(x = y, order = c(0, 10), trace = FALSE)

Model:
GARCH(0,10)

Residuals:
    Min      1Q  Median      3Q     Max
-4.4736 -0.5745  0.0000  0.6046  7.4972

Coefficient(s):
     Estimate  Std. Error  t value Pr(>|t|)
a0  2.062e-05   1.256e-06   16.413  < 2e-16
a1  3.505e-03   1.075e-02    0.326  0.74435
a2  3.950e-02   1.483e-02    2.664  0.00773
a3  4.425e-02   1.384e-02    3.197  0.00139
a4  7.499e-02   1.828e-02    4.102 4.10e-05
a5  5.440e-02   1.690e-02    3.219  0.00129
a6  1.038e-01   1.841e-02    5.640 1.70e-08
a7  6.330e-02   1.620e-02    3.906 9.37e-05
a8  5.129e-02   1.651e-02    3.108  0.00189
a9  4.702e-02   1.447e-02    3.250  0.00116
a10 2.206e-02   1.428e-02    1.545  0.12246

Diagnostic Tests:
        Jarque Bera Test

data:  Residuals
X-squared = 368.3515, df = 2, p-value < 2.2e-16


        Box-Ljung test
```

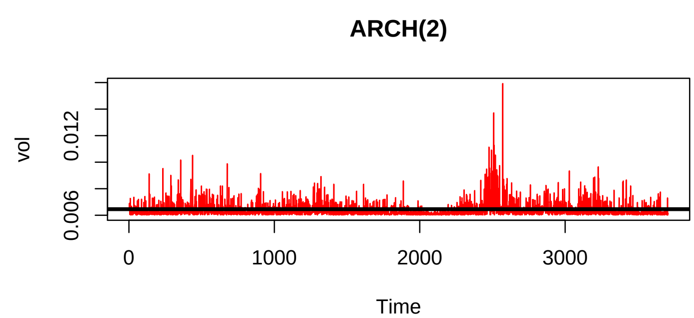
```
data:  Squared.Residuals
X-squared = 0.0367, df = 1, p-value = 0.8482

> # Note the high significance of high-order arch terms.
>
> # Print out the GARCH(1,1) model summary
> summary(y.garch11)

Call:
garch(x = y, order = c(1, 1), trace = FALSE)

Model:
GARCH(1,1)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1119 -0.5772  0.0000  0.6396  4.6832

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 1.310e-07   5.150e-08    2.544    0.011
a1 2.778e-02   3.358e-03    8.273 2.22e-16
b1 9.692e-01   3.546e-03  273.348  < 2e-16

Diagnostic Tests:
        Jarque Bera Test

data:  Residuals
X-squared = 77.0983, df = 2, p-value < 2.2e-16


        Box-Ljung test

data:  Squared.Residuals
X-squared = 8.1329, df = 1, p-value = 0.004347

>
>
```

The next pages display the fitted volatilities of the EUR/USD rate from these four models.

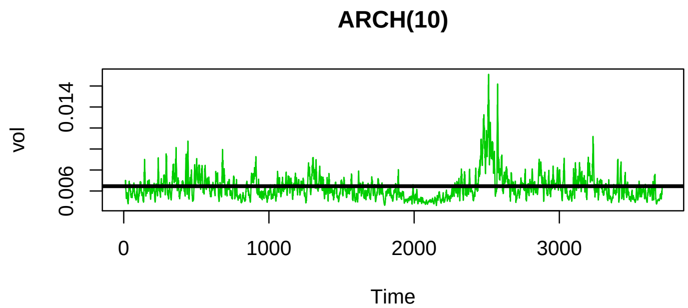
```
> names(y.garch11)

 [1] "order"         "coef"          "n.likeli"      "n.used"
 [5] "residuals"     "fitted.values" "series"        "frequency"
 [9] "call"          "vcov"
```

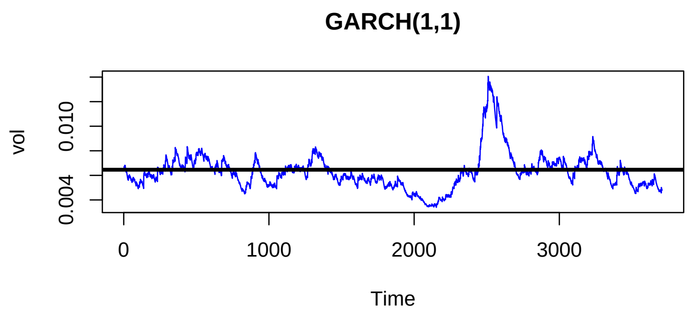
```
> vol.estmat<-cbind(
+   y.arch1$fitted.values[,1],
+   y.arch2$fitted.values[,1],
+   y.arch10$fitted.values[,1],
+   y.garch11$fitted.values[,1]
+ )
> par(mfcol=c(2,1))
> ts.plot(vol.estmat[,1],col=c(1,2,3,4), main="ARCH(1)",ylab="vol")
> abline(h=sqrt(var(y)),col=1, lwd=3)
> ts.plot(vol.estmat[,2],col=c(2,2,3,4), main="ARCH(2)",ylab="vol")
> abline(h=sqrt(var(y)),col=1, lwd=3)
```

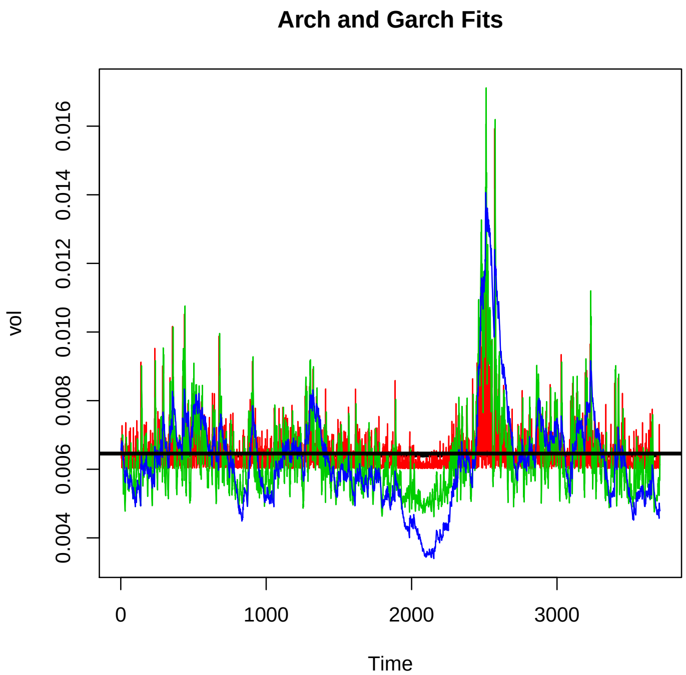





```
> par(mfcol=c(2,1))
> ts.plot(vol.estmat[,3],col=c(3,2,3,4), main="ARCH(10)",ylab="vol")
> abline(h=sqrt(var(y)),col=1, lwd=3)
> ts.plot(vol.estmat[,4],col=c(4,2,3,4), main="GARCH(1,1)",ylab="vol")
> abline(h=sqrt(var(y)),col=1, lwd=3)
```


ARCH(10)
vol
0.006
0.014
0
1000
2000
3000
Time


GARCH(1,1)
vol
0.004
0.010
0
1000
2000
3000
Time



Note:

- The ARCH models have a hard lower bound ($\hat{\alpha}_0$) which gets lower with higher-order p values.
- The $GARCH(1, 1)$ model provides an extremely parsimonious model compared to that of the $ARCH(10)$ model.
- The $GARCH(1, 1)$ model is quite smooth when compared to every ARCH model.
- The $GARCH(1, 1)$ model is very close to being non-stationary

  $$\alpha_1 + \beta_1 = 0.9970$$

  This near-non stationarity is consistent with there being no long-term mean volatility. Instead, the volatility evolves slowly over time (i.e., with high value β_1) with no tendency to revert toward any specific mean volatility level.

1.5 GARCH(1,1) Models with t Distributions

In this section, the R package "rugarch" is used to specify GARCH models with t distributions for the innovations.

We build a volatility model for the EUR/USD exchange rate returns in three steps

- Specify a Gaussian AR(p) model for the returns.

  The model residuals are heavy-tailed, relative to the Gaussian distribution and exhibit non-linear dependence, i.e., autocorrelations in the squared residuals series.

- Specify a Gaussian AR(p) - GARCH(1,1) model for the returns.

  This model accommodates the non-linear dependence and reduces the severity of the heavy-tailed distribution of the residuals, relative to a Gaussian distribution.

- Specify t-Distribution AR(p) - GARCH(1,1) models for the returns, using Maximum Likelihood to specify the degrees-of -freedom parameter for the t-distribution.

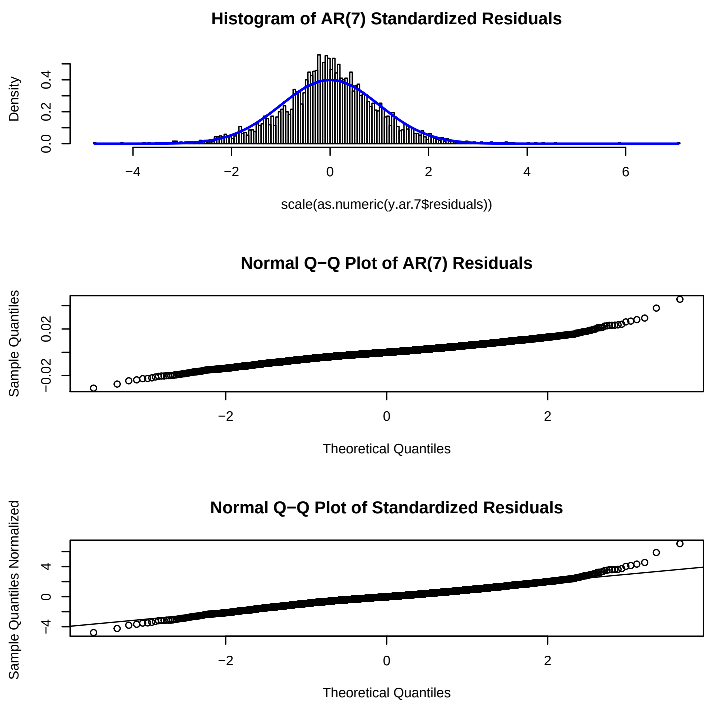
  This model explicitly incorporates non-linear dependence in the residuals (i.e., volatility) and provides a specific distribution alternative to the Gaussian with excess kurtosis (i.e., heavier tails).

```
> # 1.  Load rugarch library
> library("rugarch")
> # 2.  For times series y, specify an AR(p) autoregressive model
> #     (implicit Gaussian/Normal assumption for errors)
>
> # plot(y,type="l")
> y.ar<-ar(y)
> print(y.ar$order)

[1] 7

> y.ar.0<-ar(y, order.max=y.ar$order, aic=FALSE)
> summary(y.ar.0)

             Length Class  Mode
order         1     -none- numeric
ar            7     -none- numeric
var.pred      1     -none- numeric
x.mean        1     -none- numeric
aic           8     -none- numeric
n.used        1     -none- numeric
order.max     1     -none- numeric
partialacf    7     -none- numeric
```

```
resid         3708   -none- numeric
method           1   -none- character
series           1   -none- character
frequency        1   -none- numeric
call             4   -none- call
asy.var.coef    49   -none- numeric

> y.lag1<-fcn.lag0(y,lag=1)
> y.lag2<-fcn.lag0(y,lag=2)
> y.lag3<-fcn.lag0(y,lag=3)
> y.lag4<-fcn.lag0(y,lag=4)
> y.lag5<-fcn.lag0(y,lag=5)
> y.lag6<-fcn.lag0(y,lag=6)
> y.lag7<-fcn.lag0(y,lag=7)
> y.ar.7<-lm(y ~ y.lag1 + y.lag2 + y.lag3 + y.lag4 + y.lag5 + y.lag6 + y.lag7)
> summary(y.ar.7)

Call:
lm(formula = y ~ y.lag1 + y.lag2 + y.lag3 + y.lag4 + y.lag5 +
    y.lag6 + y.lag7)

Residuals:
      Min        1Q    Median        3Q       Max
-0.030775 -0.003578 -0.000062  0.003766  0.045501

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.761e-05  1.060e-04   0.355   0.7228
y.lag1       9.783e-03  1.644e-02   0.595   0.5518
y.lag2      -1.403e-02  1.644e-02  -0.853   0.3935
y.lag3       4.720e-03  1.643e-02   0.287   0.7739
y.lag4       3.135e-02  1.642e-02   1.910   0.0562
y.lag5      -3.118e-02  1.642e-02  -1.898   0.0577
y.lag6       2.047e-02  1.643e-02   1.246   0.2128
y.lag7       3.293e-02  1.644e-02   2.003   0.0452

Residual standard error: 0.006449 on 3693 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.003813,        Adjusted R-squared: 0.001924
F-statistic: 2.019 on 7 and 3693 DF,  p-value: 0.04916

> #
> # Note the t statistic for the order-7 parameter exceeding 2.
>
> #
> # The r function arma() provides an alternative specification of the AR(7) model
> # The function applies a different estimation algorithm (numerical optimization with
```

```
> # the r function optim() using finite-differences for gradients).
> #
> # Comparison of the output shows different results numerically, but
> # the fitted models are consistent with each other.
> y.ar.00<-arma(y,order=c(y.ar$order,0))
> summary(y.ar.00)

Call:
arma(x = y, order = c(y.ar$order, 0))

Model:
ARMA(7,0)

Residuals:
       Min         1Q     Median         3Q        Max
-3.079e-02 -3.580e-03 -6.255e-05  3.767e-03  4.550e-02

Coefficient(s):
            Estimate  Std. Error  t value Pr(>|t|)
ar1        9.923e-03   1.641e-02    0.605   0.5453
ar2       -1.383e-02   1.640e-02   -0.843   0.3993
ar3        4.781e-03   1.640e-02    0.292   0.7706
ar4        3.149e-02   1.638e-02    1.922   0.0546
ar5       -3.111e-02   1.639e-02   -1.898   0.0577
ar6        2.064e-02   1.639e-02    1.259   0.2080
ar7        3.294e-02   1.640e-02    2.008   0.0446
intercept  3.788e-05   1.058e-04    0.358   0.7203

Fit:
sigma^2 estimated as 4.151e-05,  Conditional Sum-of-Squares = 0.15,  AIC = -26873.52

> par(mfcol=c(3,1))
> # Plot residuals histogram
>
> # Plot[1,1]
> hist0<-hist(scale(as.numeric(y.ar.7$residuals)), freq=FALSE, nclass=200,
+              main="Histogram of AR(7) Standardized Residuals")
> x.density<-sort(scale(y.ar.7$residuals))
> y.density<-dnorm(x.density)
> #help(dnorm)
> lines(x.density, y.density, col=4, lwd=2)
> ####
>
> # Plot[2,1]
> qqnorm(y.ar.7$residuals, main="Normal Q-Q Plot of AR(7) Residuals")
> # Plot[3,1]
```

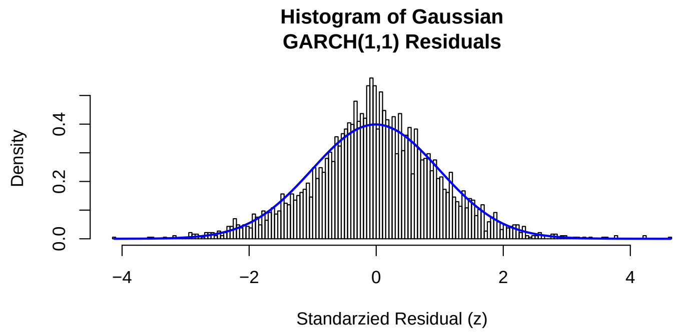
```
> #   Scale the residuals to have mean 0 and variance 1
> #   (subtracting their mean (0) and dividing by their standard deviation)
>
> qqnorm(scale(y.ar.7$residuals),ylab="Sample Quantiles Normalized ",
+        main="Normal Q-Q Plot of Standardized Residuals")
> abline(a=0,b=1)
> # Note that the realized residuals include values 6 st deviations from the mean
>
>
```

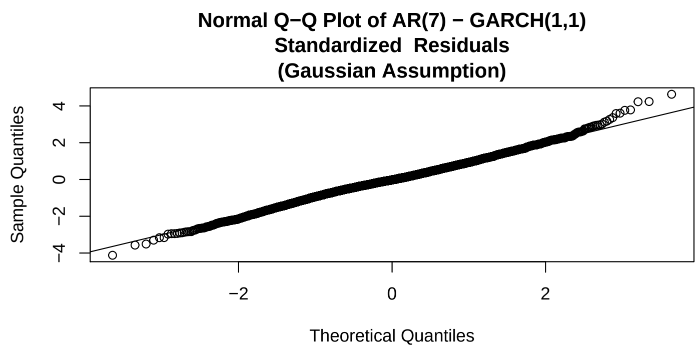



```
> # Load the library rugarch
> #
> library("rugarch")
> # Fit Gaussian GARCH(1,1) using the functions ugarchspec() and ugarchfit() from
> # the library "rugarch"
>
> spec=ugarchspec(mean.model=list(armaOrder = c(7,0)))
> fit.garch11.gaussian=ugarchfit(spec=spec,data=y)
> fit.garch11.gaussian.fit<-attributes(fit.garch11.gaussian)$fit
> names(fit.garch11.gaussian.fit)
```

```
 [1] "hessian"         "cvar"            "var"             "sigma"
 [5] "condH"           "z"               "LLH"             "log.likelihoods"
```

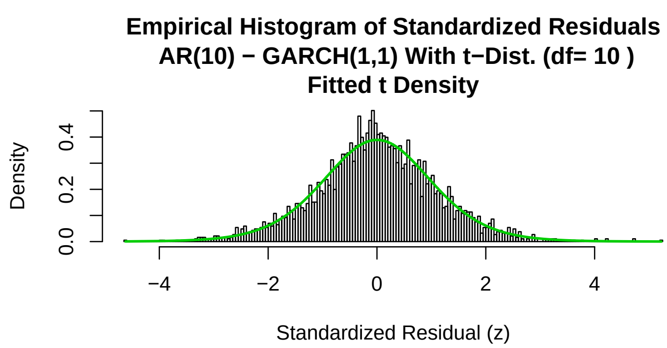
```
 [9] "residuals"       "coef"            "robust.cvar"     "scores"
[13] "se.coef"         "tval"            "matcoef"         "robust.se.coef"
[17] "robust.tval"     "robust.matcoef"  "fitted.values"   "convergence"
[21] "kappa"           "persistence"     "timer"           "ipars"
[25] "solver"

> par(mfcol=c(2,1))
> # plot[1,1]
> hist(fit.garch11.gaussian.fit$z, nclass=200,probability=TRUE,
+   main="Histogram of Gaussian\nGARCH(1,1) Residuals",
+      xlab="Standarzied Residual (z)")
> x.density<-sort(fit.garch11.gaussian.fit$z)
> y.density<-dnorm(scale(x.density))
> #help(dnorm)
> lines(x.density, y.density, col=4, lwd=2)
> # plot[2,1]
> qqnorm(x.density,
+        main=paste("Normal Q-Q Plot of AR(7) - GARCH(1,1)\n",
+                   "Standardized  Residuals\n",
+                   "(Gaussian Assumption)", sep=""))
> abline(a=0,b=1)
>
> # These results show how the AR(7)  - GARCH(1,1) model with Gaussian residuals
> # improves the fit in terms of the QQ plot when compared to the AR(7) model with
> # no GARCH structure
>
```

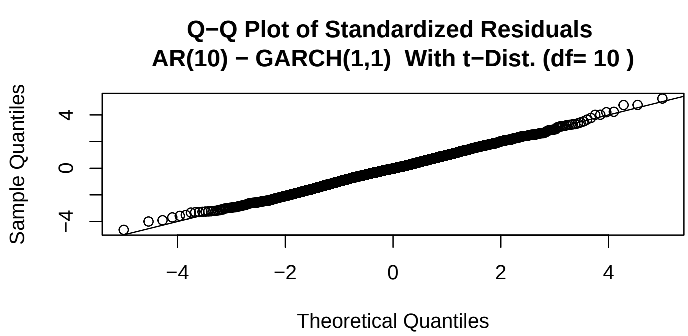





```
> #
> # Fit t-Distribution GARCH(1,1) model ----
> # Use t-dist (df0=10)
>
> df0<-10
> fixed.pars.df0=list(shape=df0)
> spec.B.df0<-ugarchspec(distribution.model="std",
+                        fixed.pars=fixed.pars.df0,
+                        mean.model=list(armaOrder = c(7,0)))
> fit.B.df0<-ugarchfit(spec=spec.B.df0,data=y)
> #
> fit.B.df0.attributes.fit<-attributes(fit.B.df0)$fit
> # Plot histogram of residuals and fitted distribution
> # Adjust standardized residuals (z) which have variance 1
> #   to t residuals
> x.density0<-sort(fit.B.df0.attributes.fit$z)*sqrt(df0/(df0-2))
> y.density0<-dt(x.density0, df=df0)
> y.quantile0<-qt(c(1:length(x.density0))/(length(x.density0)+1), df=df0)
> ###############

> par(mfcol=c(2,1))
> hist(x.density0, nclass=200, probability=TRUE,
```

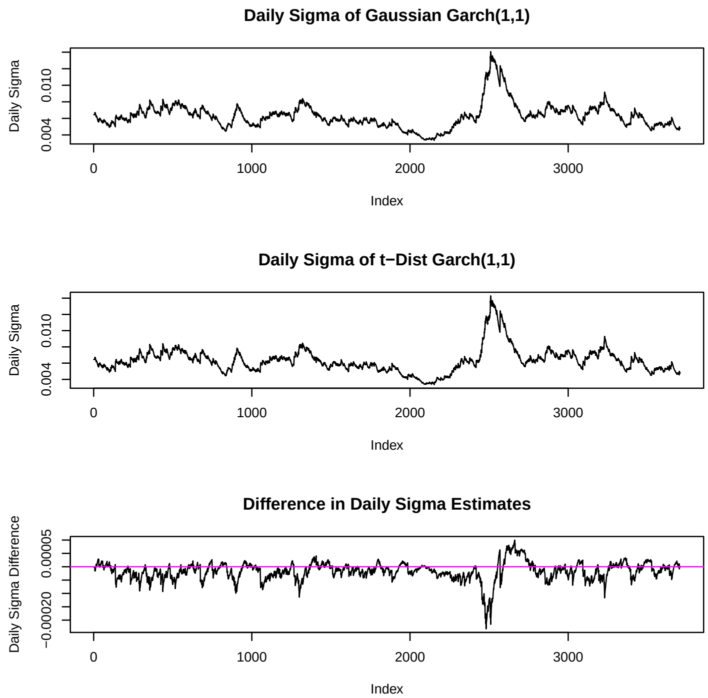
```
+       xlab="Standardized Residual (z)",
+       main=paste("Empirical Histogram of Standardized Residuals\n",
+        "AR(10) - GARCH(1,1) With t-Dist. (df=",
+                   as.character(df0),")\nFitted t Density",collapse="")
+      )
> lines(x.density0,y.density0,col=3,lwd=2)
> plot(y.quantile0,x.density0,
+       xlab="Theoretical Quantiles", ylab="Sample Quantiles",
+        main=paste(
+          "Q-Q Plot of Standardized Residuals \nAR(10) - GARCH(1,1) ",
+          "With t-Dist. (df=",as.character(df0),")",collapse=""))
> abline(a=0,b=1)
>
> ###
>
> # These results show that adjusting the AR(7)-GARCH(1,1) model to assume
> #  t-distribution (df=10) for the residuals improves the fit.
> #
> # The theoretical quantiles of the t distribution are larger in magnitude
> # at the extremes of the data
>
```



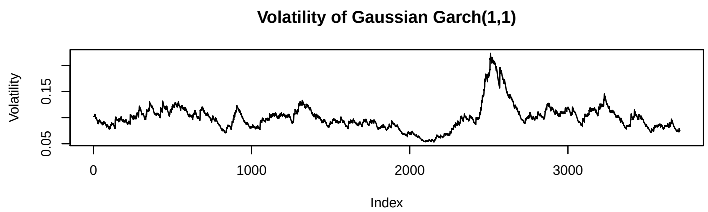


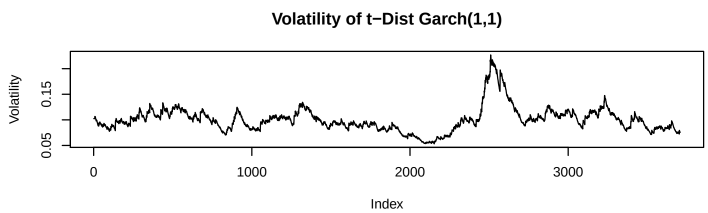
```
> # Compare the table of AR(7)-GARCH(1,1)  parameters from the two fits
> # assuming Gaussian and t-dist (df=10)
>
> fit.garch11.gaussian.fit$matcoef

            Estimate   Std. Error     t value     Pr(>|t|)
mu      1.454593e-04 9.811590e-05   1.4825248 1.382007e-01
ar1     4.527705e-03 1.631896e-02   0.2774506 7.814341e-01
ar2     7.704445e-04 8.765550e-03   0.0878946 9.299604e-01
ar3     2.591618e-03 1.629200e-02   0.1590731 8.736113e-01
ar4     1.667420e-02 1.674066e-02   0.9960297 3.192357e-01
ar5    -2.603436e-02 1.667282e-02  -1.5614853 1.184093e-01
ar6     2.174976e-02 1.664443e-02   1.3067294 1.913046e-01
ar7     1.404655e-02 1.674616e-02   0.8387919 4.015861e-01
omega   1.191858e-07 5.757056e-08   2.0702568 3.842830e-02
alpha1  2.776813e-02 3.677317e-03   7.5511936 4.307665e-14
beta1   9.695577e-01 3.980207e-03 243.5947936 0.000000e+00

> fit.B.df0.attributes.fit$matcoef

            Estimate   Std. Error     t value     Pr(>|t|)
mu      1.437787e-04 9.653462e-05   1.4894001 1.363820e-01
ar1    -6.806758e-03 1.641722e-02  -0.4146110 6.784267e-01
ar2     4.026815e-03 1.723819e-02   0.2335984 8.152967e-01
ar3     4.572084e-03 1.683483e-02   0.2715848 7.859413e-01
ar4     1.136456e-02 1.647312e-02   0.6898851 4.902664e-01
ar5    -2.525213e-02 1.634806e-02  -1.5446561 1.224295e-01
ar6     2.893609e-02 1.643198e-02   1.7609626 7.824474e-02
ar7     1.702675e-02 1.637892e-02   1.0395530 2.985476e-01
omega   1.284640e-07 6.714875e-08   1.9131260 5.573193e-02
alpha1  2.919647e-02 4.512527e-03   6.4700936 9.794232e-11
beta1   9.681037e-01 4.828606e-03 200.4934236 0.000000e+00
shape   1.000000e+01           NA          NA           NA

> # Note that the persistence parameter is virtually the same
> fit.garch11.gaussian.fit$persistence

[1] 0.9973259

> fit.B.df0.attributes.fit$persistence

[1] 0.9973002

> par(mfcol=c(3,1))
> plot(y001<-fit.garch11.gaussian.fit$sigma,type="l",
+      main="Daily Sigma of Gaussian Garch(1,1)", ylab="Daily Sigma")
> plot(y002<-fit.B.df0.attributes.fit$sigma,type="l",
```

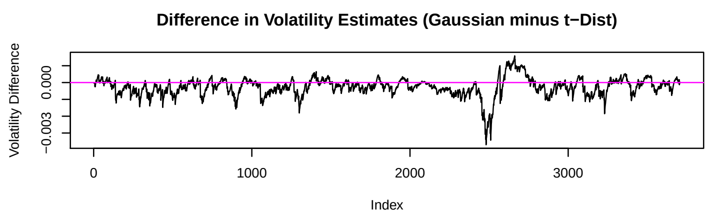
```
+       main="Daily Sigma of t-Dist Garch(1,1)", ylab="Daily Sigma")
> plot(y001 - y002, main="Difference in Daily Sigma Estimates",
+       ylab="Daily Sigma Difference",type="l")
>   abline(h=0, col=6)
```




```
> # Re-do plot on volatility (annualized st. dev.) scale
> par(mfcol=c(3,1))
> plot(y001<-sqrt(252)*fit.garch11.gaussian.fit$sigma,type="l",
+       main="Volatility of Gaussian Garch(1,1)", ylab="Volatility")
> plot(y002<-sqrt(252)*fit.B.df0.attributes.fit$sigma,type="l",
+       main="Volatility of t-Dist Garch(1,1)",ylab="Volatility")
> plot(y001 - y002,
+       main="Difference in Volatility Estimates (Gaussian minus t-Dist)",
+       ylab="Volatility Difference",type="l")
> abline(h=0, col=6)
>
```

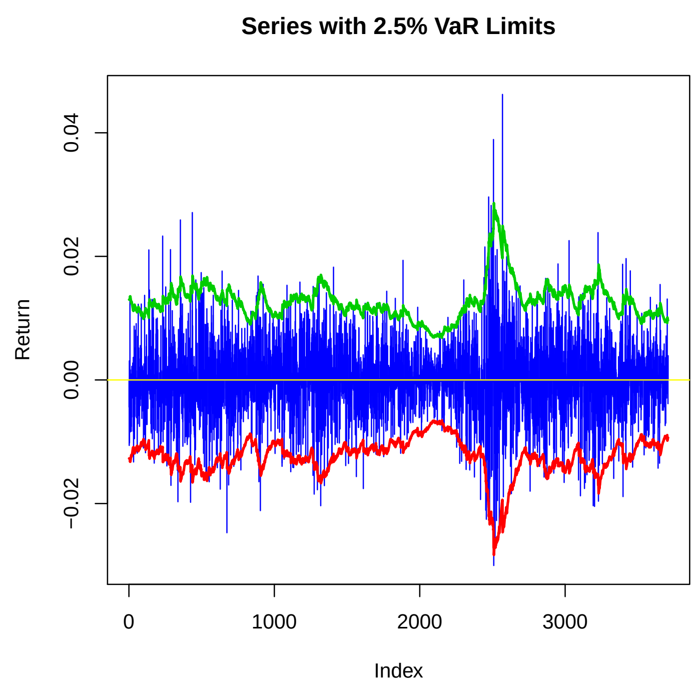







```
> # Value at Risk Plot for AR(7)-GARCH(1,1) Model with
> #   t distribution (df=10) innovations
>
> fit000<-fit.B.df0.attributes.fit
> fit000.longtermmean<-as.numeric(fit000$coef[1]/(1-sum(fit000$coef[2:8])))
> quantile.level0<-0.025
> t.df0<-df0
> tquantile.lo<-qt(quantile.level0,df=10)
> tquantile.hi<-qt(1-quantile.level0,df=10)
> fit.varlimit.hi<-fit000$fitted.values +
+   tquantile.hi*fit000$sigma*sqrt((t.df0-2.)/(t.df0))
> fit.varlimit.lo<-fit000$fitted.values +
+   tquantile.lo*fit000$sigma*sqrt((t.df0-2.)/(t.df0))
> fit.varlimit.hi<-fit000.longtermmean +
+   tquantile.hi*fit000$sigma*sqrt((t.df0-2.)/(t.df0))
> fit.varlimit.lo<-fit000.longtermmean +
+   tquantile.lo*fit000$sigma*sqrt((t.df0-2.)/(t.df0))
> fit000.0<-0*fit.varlimit.hi + fit000.longtermmean
> mean(y < fit.varlimit.lo)

[1] 0.02885653

> mean(y > fit.varlimit.hi)
```

```
[1] 0.02481122

> ylim000=c(min(c(fit.varlimit.hi,fit.varlimit.lo,y)),
+         max(c(fit.varlimit.hi,fit.varlimit.lo,y)))
> plot(y,type="h", col=4,
+       ylim=ylim000,
+       ylab="Return",
+       main=paste(c("Series with ",
+                   round(100*quantile.level0,digits=1),
+                   "% VaR Limits"),collapse=""))
> lines(c(1:length(fit.varlimit.hi)), fit.varlimit.hi, col=3,lwd=2)
> lines(c(1:length(fit.varlimit.lo)), fit.varlimit.lo, col=2,lwd=2)
> abline(h=0,col=7)
> ######################################3
```



The choice of 10 degrees of freedom for the t distribution in the AR(7)-GARCH(1,1) model is based upon maximum likelihood.

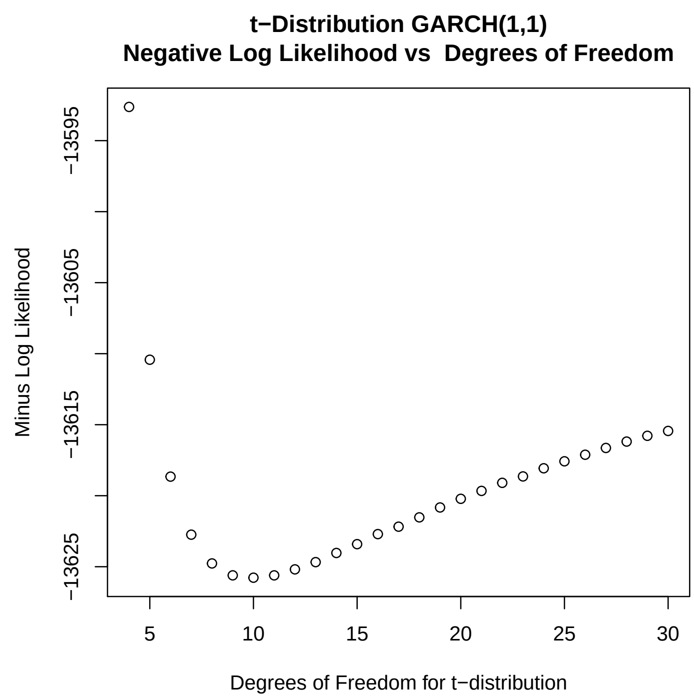
The code below computes the case-wise likelihoods of the returns under different cases of the AR(7)-GARCH(1,1) model in terms of the degrees of freedom parameter. The (conditional) maximum-likelihood value df=10 (conditioning on first 8 observations) is determined.

```
> ##############################################3
>
```

```
> # Determination of maximum-likelihood estimate of t distribution
> # degrees of freedom
>
> # Fix the AR(7) model for the mean process
> spec=ugarchspec(mean.model=list(armaOrder = c(7,0)))
> # Consider degrees of freedom ranging from 4 to 30
> list.df0<-c(4:30)
> # Create matrices for the log likelihoods and fitted sigmas
> # where each column corresonds to one of the t distribution fitted models
>
> mat.log.likelihoods<-matrix(0,nrow=length(y),ncol=length(list.df0))
> mat.fitted.sigma<-matrix(0,nrow=length(y),ncol=length(list.df0))
> for (j.df0 in c(1:length(list.df0))){
+   df0<-list.df0[j.df0]
+     fixed.pars.df0=list(shape=df0)
+     spec.B.df0<-ugarchspec(distribution.model="std",
+                            fixed.pars=fixed.pars.df0,
+                            mean.model=list(armaOrder = c(7,0)))
+     fit.B.df0<-ugarchfit(spec=spec.B.df0,data=y)
+
+   fit.B.df0.attributes.fit<-attributes(fit.B.df0)$fit
+   mat.log.likelihoods[,j.df0]<- fit.B.df0.attributes.fit$log.likelihoods
+   mat.fitted.sigma[,j.df0]<-fit.B.df0.attributes.fit$sigma
+ }

> ### Likelihood Plot  vs Degrees of freedom
> # (Condition on first 31 observations so likelihood of every model
> #  is based on same sample of cases).
> par(mfcol=c(1,1))
> mat.log.likelihoods.tot<-apply(mat.log.likelihoods[-c(1:8),], 2, sum)
> print(cbind(list.df0, mat.log.likelihoods.tot))

      list.df0 mat.log.likelihoods.tot
 [1,]        4               -13592.63
 [2,]        5               -13610.43
 [3,]        6               -13618.66
 [4,]        7               -13622.74
 [5,]        8               -13624.77
 [6,]        9               -13625.61
 [7,]       10               -13625.77
 [8,]       11               -13625.61
 [9,]       12               -13625.19
[10,]       13               -13624.68
[11,]       14               -13624.03
[12,]       15               -13623.41
[13,]       16               -13622.70
```

```
[14,]       17              -13622.18
[15,]       18              -13621.53
[16,]       19              -13620.83
[17,]       20              -13620.22
[18,]       21              -13619.66
[19,]       22              -13619.10
[20,]       23              -13618.64
[21,]       24              -13618.07
[22,]       25              -13617.58
[23,]       26              -13617.12
[24,]       27              -13616.64
[25,]       28              -13616.19
[26,]       29              -13615.78
[27,]       30              -13615.44
```

```
> plot(list.df0, mat.log.likelihoods.tot,
+      ylab="Minus Log Likelihood",
+      xlab="Degrees of Freedom for t-distribution",
+      main="t-Distribution GARCH(1,1)\nNegative Log Likelihood vs  Degrees of Freedom"
+ )
>
```



References

R Core Team (2012) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Alexios Ghalanos (2013) rugarch: Univariate GARCH models. R package version 1.0-16.