Estimating Volatilities and Correlation

Ross Bennett

March 15, 2014

Abstract

The purpose of this vignette is to demonstrate methods for estimating volatility and correlation as outlined in Chapter 10 of Foundations of Risk Management.

Contents

1 Estimating Volatility

We define σ_n as the volatility on day n, as estimated at the end of day $n-1$. This section describes the standard approach to estimating σ_n from historical data.

First we define the continuously compounded return between the end of day $i-1$ and the end of day i.

$$u_i = \ln \frac{S_i}{S_{i-1}} \tag{1}$$

where:

S_i is the value of the market variable at the end of day i (e.g. asset prices)

An unbiased estimate of the variance rate per day on day n, σ_n^2, using the m most recent observations is

$$\sigma_n^2 = \frac{1}{m-1} \sum_{i=1}^{m} (u_{n-1} - \bar{u})^2 \tag{2}$$

where $\bar{u}$ is the mean of u_i for $i = 1, 2, ..., m$

$$\bar{u} = \frac{1}{m} \sum_{i=1}^{m} u_{n-1} \tag{3}$$

A few changes can be made to simplify the equation for monitoring daily volatility.

1. define u_i as the percentage change in the market variable between the end of day $i-1$ and the end of day i.

$$u_i = \frac{S_i - S_{i-1}}{S_i} \tag{4}$$

2. Assume $\bar{u}$ to be zero

3. Replace $m-1$ with m

These changes simplify the formula for the variance rate to

$$\sigma_n^2 = \frac{1}{m}\sum_{i=1}^{m} u_{n-1}^2 \tag{5}$$

This equation gives equal weight to each of the previous m observations. A model that allows one to assign weights to the previous observations is

$$\sigma_n^2 = \sum_{i=1}^{m} \alpha_i u_{n-1}^2 \tag{6}$$

Where α_i is the weight given to the observation i days ago.
A special case of equation is the Exponentially Weighted Moving Average (EWMA) Model.

2 Exponentially Weighted Moving Average Model

The Exponentially Weighted Moving Average (EWMA) Model is a special case of a weighted moving average where the weights α_i decrease exponentially as we move backwards through time. Greater weights are given to more recent observations.

This weighting scheme leads to simple formula for updating volatility estimates. The predictive version of the variance rate of day n is given as

$$\hat{\sigma}_n^2 = \lambda\hat{\sigma}_{n-1}^2 + (1-\lambda)u_{n-1}^2 \tag{7}$$

where:

$\hat{\sigma}_{n-1}^2$ is the estimated variance rate of period $n-1$

u_{n-1}^2 is the squared return of preiod $n-1$

λ is a constant between 0 and 1

The value for λ determines how responsive the volatility estimate is to the most recent percentage change, u_{n-1}. A lower (higher) value for λ leads to a greater (lesser) weight given to u_{n-1}. One way to think of this is that values of λ close to 1 produce volatility estimates that respond relatively slow to new information coming into the market provided by u_{n-1}.

Load the package and data. Unless noted otherwise, the weekly returns of Microsoft (MSFT) will be used as the asset return data.

```
suppressPackageStartupMessages(library(GARPFRM))
data(crsp_weekly)

# Use the weekly MSFT returns
R <- largecap_weekly[, "MSFT"]
```

Here we calculate the volatility estimates of the MSFT weekly returns using the EWMA model. We choose `lambda=0.94`. The RiskMetrics database, originally created by J.P. Morgan and made publicly available in 1994, uses the EWMA model with $\lambda = 0.94$ for updating daily volatility in its RiskMetrics database. An `initialWindow=15` is specified to use the first 15 periods to calculate the initial conditions, u_0 and σ_0.

```
lambda <- 0.94
initialWindow <- 15
volEst <- EWMA(R, lambda, initialWindow, type="volatility")
volEst

## EWMA Estimate
##
## Parameters
## lambda: 0.94
## initialWindow: 15
## type: volatility
##
## Final Period EWMA Estimate:
##               MSFT
## 2010-12-28 0.03364
```

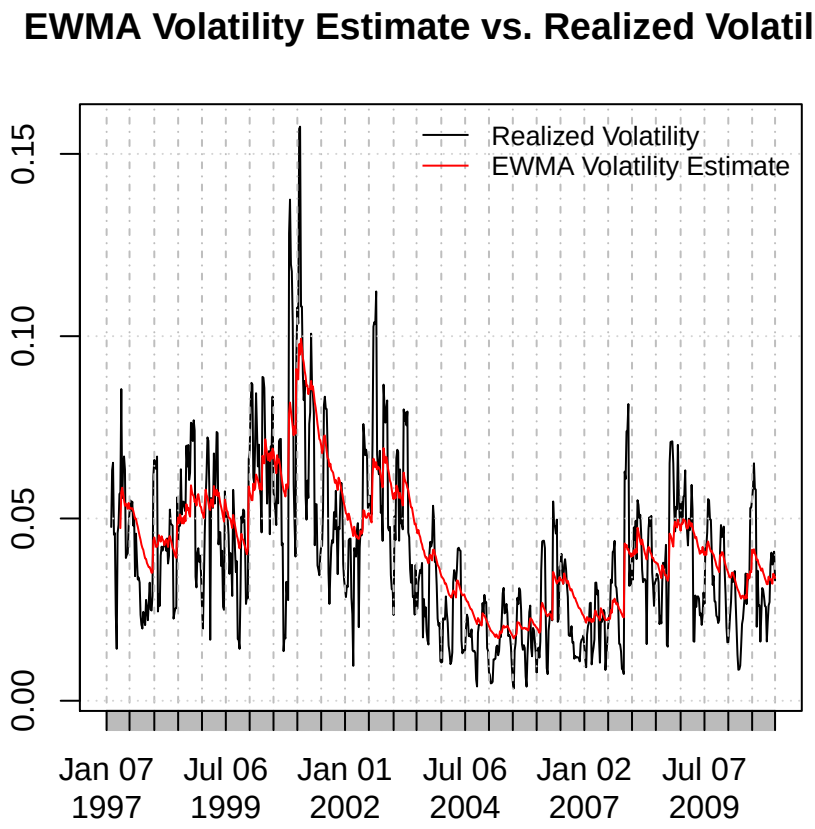
An important point to note is that we are using weekly returns to estimate weekly volatility while the lambda value used in the RiskMetrics database is for daily volatility estimates. A data driven approach for selecting a value for λ is to determine the λ that minimizes the mean squared error between the realized volatility and the estimated volatility from the EWMA model.

Here we calculate the realized volatility defined as the equally weighted average of the standard deviation of the subsequent n periods.

```
vol <- realizedVol(R, n=5)
```

Now we plot the estimated volatility from the EWMA model and the realized volatility.

```
plot(vol, main="EWMA Volatility Estimate vs. Realized Volatility")
lines(volEst$estimate, col="red")
legend("topright", legend=c("Realized Volatility", "EWMA Volatility Estimate"),
       col=c("black", "red"), lty=c(1,1), cex=0.8, bty="n")
```



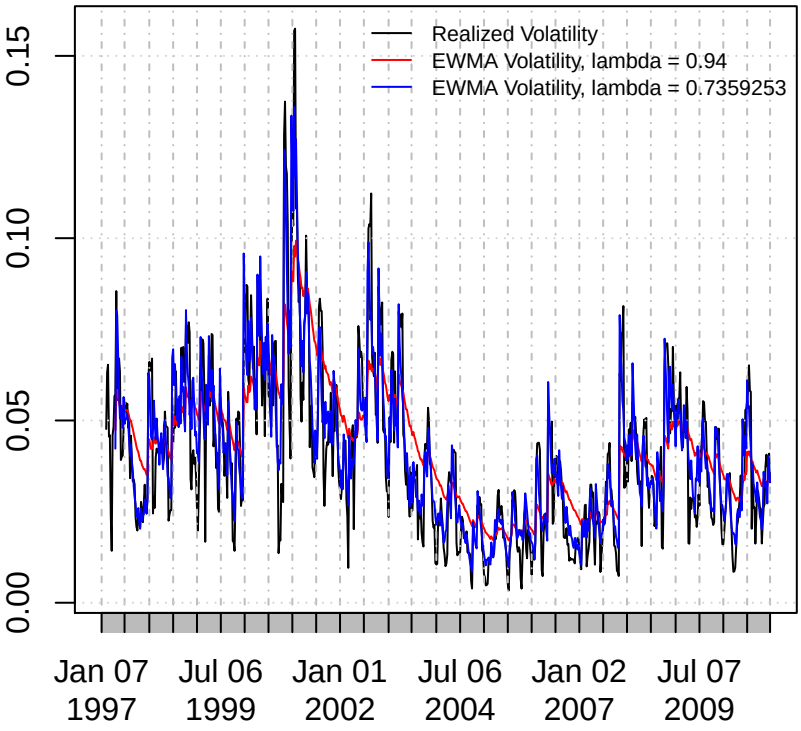
The `estimateLambdaVol` function estimates the value for λ by minimizing the mean squared error between the realized volatility and the EWMA model estimated volatility.

```
# Estimate lambda
# Use initialWindow = 15 for the EWMA volatility estimate and
# n = 5 to calculate the realized volatility
lambda <- estimateLambdaVol(R, initialWindow, n=5)
lambda

## [1] 0.7359


volEst2 <- EWMA(R, lambda, initialWindow, type="volatility")
volEst2

## EWMA Estimate
##
## Parameters
## lambda: 0.7359
```

```
## initialWindow: 15
## type: volatility
##
## Final Period EWMA Estimate:
##                MSFT
## 2010-12-28 0.03317
```

Here we plot the realized volatility along with the EWMA estimated volatility with $\lambda = 0.94$ and $\lambda = 0.7359253$ to gain intuition through visualization of the EWMA volatility estimates.

```
# Realized volatility
plot(vol, main="EWMA Volatility Estimate vs. Realized Volatility")
# EWMA volatility estimate, lambda = 0.94
lines(volEst$estimate, col="red")
# EWMA volatility estimate, lambda = 0.0.7359253
lines(volEst2$estimate, col="blue")
legend("topright", legend=c("Realized Volatility",
                            "EWMA Volatility, lambda = 0.94",
                            "EWMA Volatility, lambda = 0.7359253"),
       col=c("black", "red", "blue"), lty=c(1, 1, 1), cex=0.7, bty="n")
```



Now we move to using the EWMA model to calculate the covariance between the returns of two assets. Note that we set `lambda=NULL` in the `EWMA` function. If `lambda = NULL`, the optimal λ value is estimated by minimizing the mean squared error between the estimated covariance and realized covariance.

The covariance between two variables, X and Y, is defined as

$$\sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)] \tag{8}$$

where

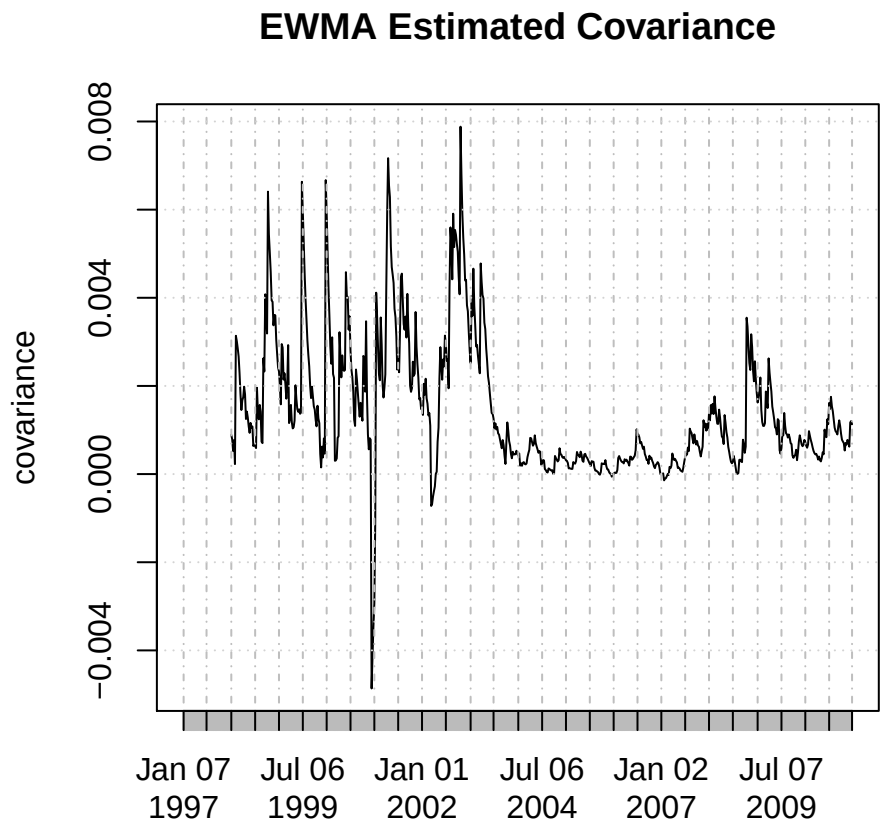
μ_X is the sample mean of X

μ_Y is the sample mean of Y

An EWMA model for the updating the covariance estimate between X and Y on day n is

$$cov(X, Y) = \sigma_{XY,n} = \lambda\sigma_{XY,n-1} + (1 - \lambda)X_{n-1}Y_{n-1} \tag{9}$$

```r
# Use the first 2 columns of the large cap weekly returns
R <- largecap_weekly[,1:2]
initialWindow <- 52
covEst <- EWMA(R, lambda=NULL, initialWindow, n=10, "covariance")
covEst

## EWMA Estimate
##
## Parameters
## lambda: 0.8665
## initialWindow: 52
## type: covariance
##
## Final Period EWMA Estimate:
##            ORCL.MSFT
## 2010-12-28   0.00113

plot(covEst, main="EWMA Estimated Covariance")
```



In a similar fashion, we can also use the EWMA model to calculate the correlation between the returns of two assets.

The correlation between two variables, X and Y, is defined as

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{10}$$

where

$cov(X,Y)$ is the covariance between X and Y

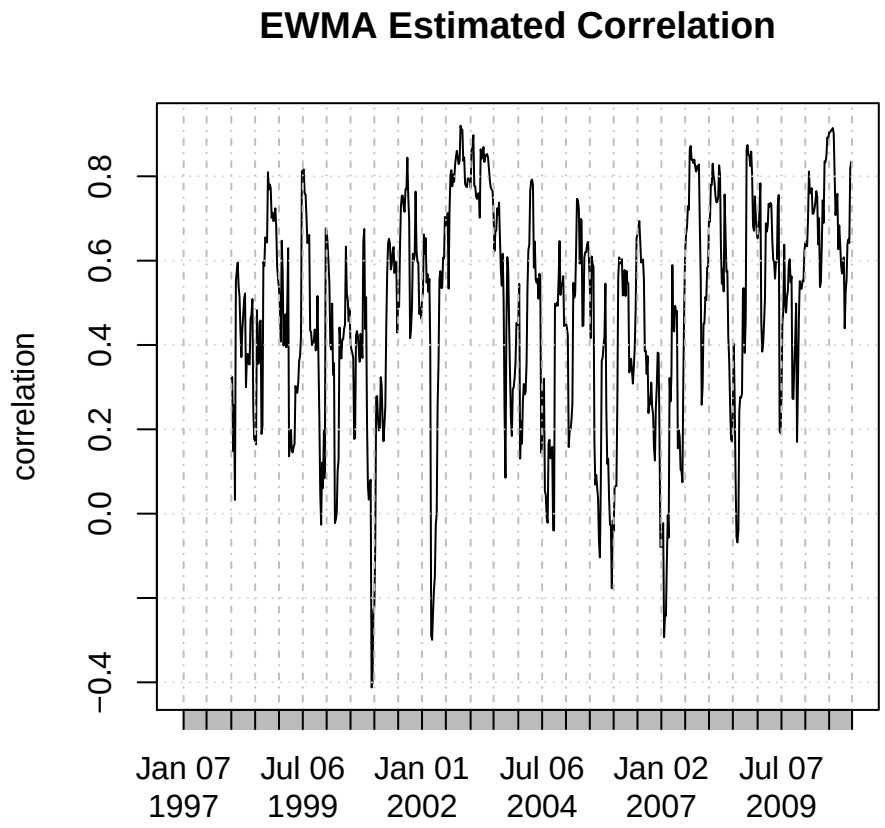
σ_X is the standard deviation of X

σ_Y is the standard deviation of Y

The EWMA mdoel for correlation is calculated using an EWMA model for the estimated covariance between X and Y, estimated volatility of X, and estimated volatility of Y.

```
corEst <- EWMA(R, lambda=NULL, initialWindow, n=10, "correlation")
corEst
```

```
## EWMA Estimate
##
## Parameters
## lambda: 0.8404
## initialWindow: 52
## type: correlation
##
## Final Period EWMA Estimate:
##            ORCL.MSFT
## 2010-12-28     0.831

plot(corEst, main="EWMA Estimated Correlation")
```



The previous examples demonstrated using an EWMA model to estimate the volatility of the returns of a single asset, and the correlation and volatility between the returns of two assets. Now we move to using an EWMA model to estimated the covariance and correlation of a multivariate data set.

```r
# Use the first 4 columns of the largecap_weekly dataset
R <- largecap_weekly[,1:4]

# calculate the sample covariance matrix
sample_cov <- cov(R)
sample_cov
```

```
##          ORCL      MSFT       HON      EMC
## ORCL 0.004500 0.0013887 0.0010673 0.002696
## MSFT 0.001389 0.0021176 0.0008716 0.001614
## HON  0.001067 0.0008716 0.0025923 0.001225
## EMC  0.002696 0.0016142 0.0012249 0.005008
```

```r
# EWMA covariance matrix estimate
lambda <- 0.94
initialWindow <- 52
covEst <- EWMA(R, lambda, initialWindow, type="covariance")
covEst
```

```
## EWMA Estimate
##
## Parameters
## lambda: 0.94
## initialWindow: 52
## type: covariance
##
## Final Period EWMA Estimate:
##           ORCL      MSFT       HON       EMC
## ORCL 0.0016265 0.0010008 0.0008490 0.0008698
## MSFT 0.0010008 0.0011319 0.0006191 0.0005775
## HON  0.0008490 0.0006191 0.0009928 0.0007313
## EMC  0.0008698 0.0005775 0.0007313 0.0010230
```

In a similar fashion, we can also use the EWMA model to estimate the correlation matrix.

```r
# calculate the sample covariance matrix
sample_cor <- cor(R)
sample_cor
```

```
##        ORCL   MSFT    HON    EMC
## ORCL 1.0000 0.4499 0.3125 0.5678
## MSFT 0.4499 1.0000 0.3720 0.4957
## HON  0.3125 0.3720 1.0000 0.3400
## EMC  0.5678 0.4957 0.3400 1.0000
```

```r
# EWMA covariance matrix estimate
```

```
lambda <- 0.94
initialWindow <- 52
corEst <- EWMA(R, lambda, initialWindow, type="correlation")
corEst

## EWMA Estimate
##
## Parameters
## lambda: 0.94
## initialWindow: 52
## type: correlation
##
## Final Period EWMA Estimate:
##        ORCL   MSFT    HON    EMC
## ORCL 1.0000 0.7376 0.6682 0.6743
## MSFT 0.7376 1.0000 0.5841 0.5367
## HON  0.6682 0.5841 1.0000 0.7257
## EMC  0.6743 0.5367 0.7257 1.0000
```

3 The GARCH(1,1) Model

We now demonstrate the generalized autoregressive conditional heteroskedasticity (GARCH) as presented by Bollerslev in 1986 as a way to estimate volatility. The general GARCH(p,q) model calculates σ_n^2 from the most recent p observations of u^2 and the most recent q estimates of σ_n^2. The GARCH(1,1) model refers to the most recent observation of u^2 and the most recent estimate of σ_n^2. The GARCH(1,1) is a popular model and the one we will focus on. The equation for the GARCH(1,1) model is

$$\sigma_n^2 = \gamma V_L + \alpha u_{n-1}^2 + \beta \sigma_{n-1}^2 \tag{11}$$

where:

γ is the weight assigned to V_L

V_L is the long-run average variance rate

$alpha$ is the weight assigned to u_{n-1}^2

u_{n-1} is the squared returns of preiod $n-1$

β is the weight assigned to σ_{n-1}^2

σ_{n-1}^2 is the estimated variance rate of period $n-1$

The weights must sum to 1 such that

$$\gamma + \alpha + \beta = 1 \tag{12}$$

It should be noted that the EWMA model discussed in the previous section is a special case of the GARCH(1,1) model where $\gamma = 0$, $\alpha = 1 - \lambda$, and $\beta = \lambda$.

A more common form of the model is obtained by setting $\omega = \gamma V_L$ such that the equation for the model is

$$\sigma_n^2 = \omega + \alpha u_{n-1}^2 + \beta \sigma_{n-1}^2 \tag{13}$$

With the estimated parameters for ω, α, and β, we can calculate γ and V_L as

$$\gamma = 1 - \alpha - \beta \tag{14}$$

$$V_L = \omega / \gamma \tag{15}$$

A key characteristic of the GARCH(1,1) model is mean reversion, i.e. the variance rate is pulled back to the long-run average variance rate over time. In contrast, the EWMA model is not mean reverting.

3.1 Estimating GARCH(1,1) Parameters

Estimating the parameters for the GARCH model requires an optimization routine to maximize the likelihood. The FRM text describes an example of using a spreadsheet and a solver, e.g. the Microsoft Excel Solver. The implementation of GARCH in the GARPFRM(citation) package utilizes the rugarch(citation) package. The rugarch package uses C code for a fast and efficient algorithm for the main part of the likelihood calculation.

Here we demonstrate how to specify and fit a GARCH(1,1) model using weekly returns for Microsoft.

```
# Use the weekly MSFT returns
R <- largecap_weekly[,"MSFT"]

# Specify and fit the MSFT returns to a standard ARMA(0,0)-GARCH(1,1) model
# Note that the default is ARMA(1,1)-GARCH(1,1) so we only need to change
# the ARMA order. The default arguments were chosen to be consistent with the
# default arguments in rugarch.
model <- uvGARCH(R, armaOrder=c(0,0))

## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009


# Get the fitted GARCH model
fit <- getFit(model)

# Get the coefficients
coef(fit)

##        mu     omega    alpha1     beta1
## 2.792e-03 1.691e-05 5.030e-02 9.411e-01


# Show the summary results of the fit
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      0.002792    0.001384   2.0171 0.043688
## omega   0.000017    0.000007   2.4984 0.012475
## alpha1  0.050301    0.010768   4.6712 0.000003
## beta1   0.941126    0.010779  87.3098 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.002792    0.001274   2.1921 0.028374
## omega   0.000017    0.000009   1.7973 0.072288
## alpha1  0.050301    0.012004   4.1904 0.000028
## beta1   0.941126    0.010143  92.7891 0.000000
##
## LogLikelihood : 1279
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.4941
## Bayes        -3.4689
## Shibata      -3.4941
## Hannan-Quinn -3.4844
##
## Q-Statistics on Standardized Residuals
## ------------------------------------
##               statistic p-value
## Lag[1]          0.02799  0.8671
## Lag[p+q+1][1]   0.02799  0.8671
## Lag[p+q+5][5]   0.82000  0.9757
## d.o.f=0
## H0 : No serial correlation
##
## Q-Statistics on Standardized Squared Residuals
## ------------------------------------
##               statistic p-value
```

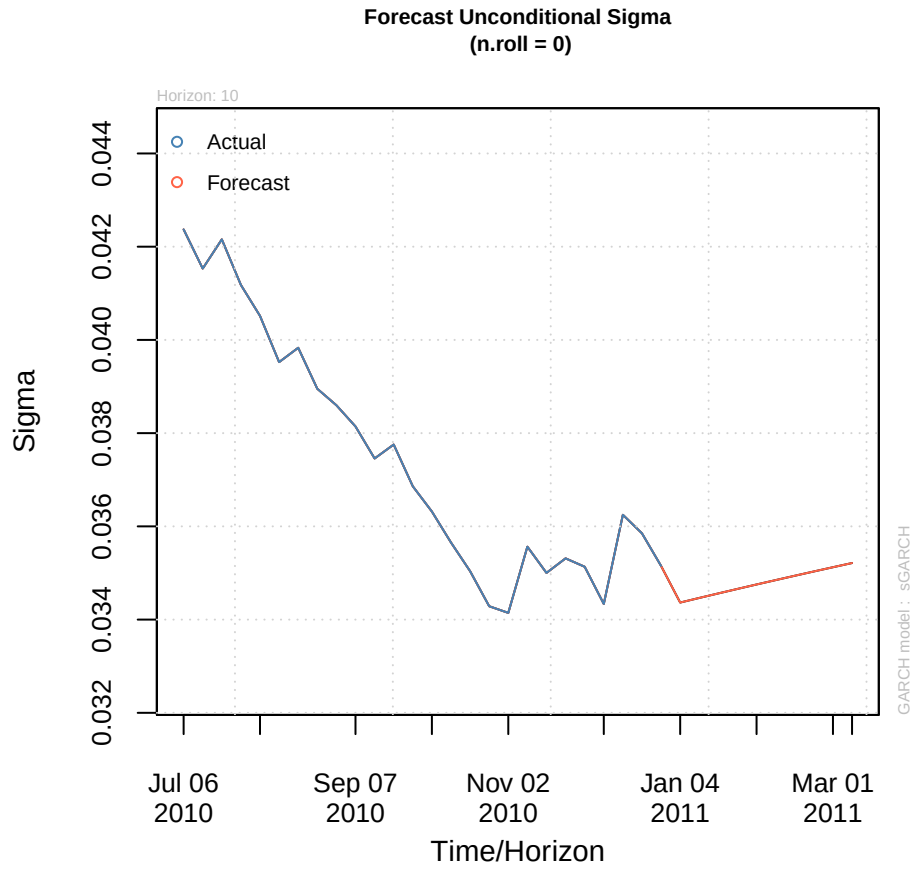
```
## Lag[1]           0.4305  0.5117
## Lag[p+q+1][3]    1.1187  0.2902
## Lag[p+q+5][7]    3.9181  0.5613
## d.o.f=2
##
## ARCH LM Tests
## ------------------------------------
##              Statistic DoF P-Value
## ARCH Lag[2]     0.7077   2  0.7020
## ARCH Lag[5]     2.8449   5  0.7239
## ARCH Lag[10]    6.2428  10  0.7945
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.8881
## Individual Statistics:
## mu     0.20386
## omega  0.09303
## alpha1 0.19960
## beta1  0.18777
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:       1.07 1.24 1.6
## Individual Statistic:  0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.1752 0.8610
## Negative Sign Bias  0.4044 0.6860
## Positive Sign Bias  0.7120 0.4767
## Joint Effect        0.8782 0.8307
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     36.19     0.009997
## 2    30     46.96     0.018797
## 3    40     55.75     0.039956
## 4    50     60.14     0.132261
##
##
## Elapsed time : 0.1991
```

3.2 Using GARCH(1,1) to Forecast Future Volatility

```r
# n period ahead forecast
forecast10 <- forecast(model, nAhead=10)
forecast10

##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 10
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=2010-12-28]:
##        Series   Sigma
## T+1  0.002792 0.03437
## T+2  0.002792 0.03446
## T+3  0.002792 0.03456
## T+4  0.002792 0.03466
## T+5  0.002792 0.03475
## T+6  0.002792 0.03485
## T+7  0.002792 0.03494
## T+8  0.002792 0.03503
## T+9  0.002792 0.03512
## T+10 0.002792 0.03521
```

Plot of the forecast.

```r
plot(forecast10, which=3)
```



Now we specify and fit a model with `outSample=100` so that we can use the last 100 data points for out of sample testing and do a rolling forecast.

```
model11 <- uvGARCH(R, armaOrder=c(0,0), outSample=100)
forecast2 <- forecast(model11, nRoll=10)
forecast2

##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 10
## Roll Steps: 10
## Out of Sample: 10
##
## 0-roll forecast [T0=2009-01-27]:
##        Series   Sigma
## T+1  0.002454 0.05028
```

```
## T+2  0.002454 0.05028
## T+3  0.002454 0.05028
## T+4  0.002454 0.05028
## T+5  0.002454 0.05029
## T+6  0.002454 0.05029
## T+7  0.002454 0.05029
## T+8  0.002454 0.05029
## T+9  0.002454 0.05030
## T+10 0.002454 0.05030
```

Plot the rolling forecast.

```
plot(forecast2, which=4)
```