Quantifying Volatility in VaR Models

Ross Bennett

March 26, 2014

Abstract

The purpose of this vignette is to demonstrate methods for estimating and backtesting Value-at-Risk (VaR) using various methods as outlined in Chapter 11 of Foundations of Risk Management.

Contents

1 Stochastic Behavior of Returns

TODO: Add content on returns

1.1 The Distribution of Conoco Phillips Weekly Returns

TODO: Add content on returns

Here we consider the weekly returns of Conoco Phillips (COP).

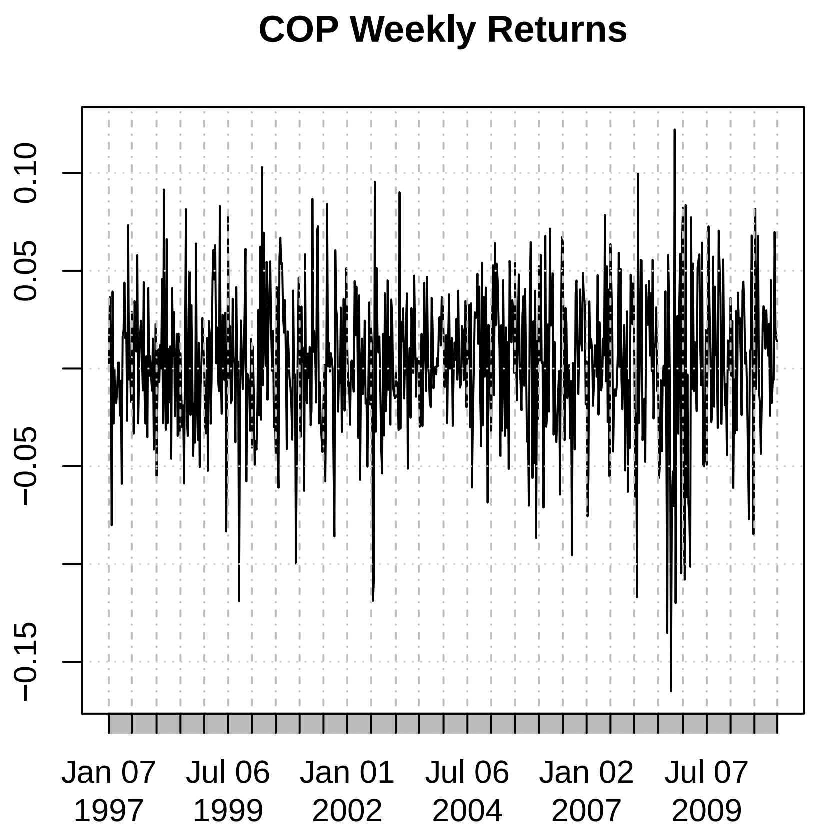
```
# Load the package and data
library(GARPFRM)

## Loading required package:  xts
## Loading required package:  zoo
##
## Attaching package:  'zoo'
```

```
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package:  PerformanceAnalytics
##
## Attaching package:  'PerformanceAnalytics'
##
## The following object is masked from 'package:graphics':
##
##     legend
```

```
data

            "COP"
```

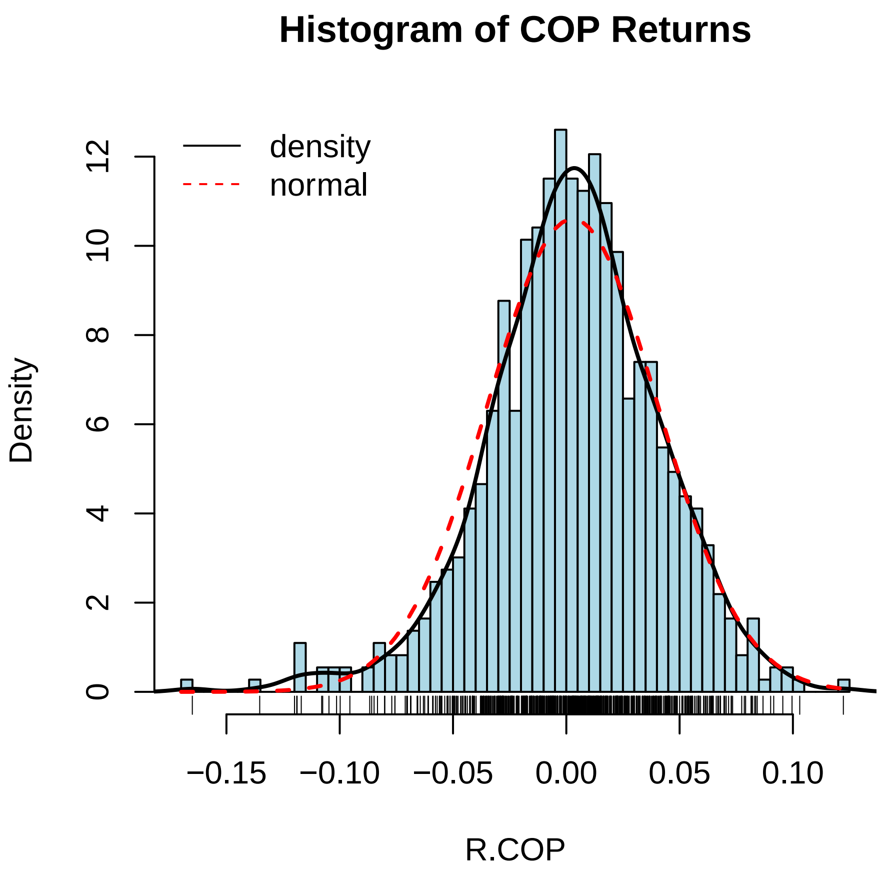
Plot the weekly returns

```
plot(R.COP, main="COP Weekly Returns")
```



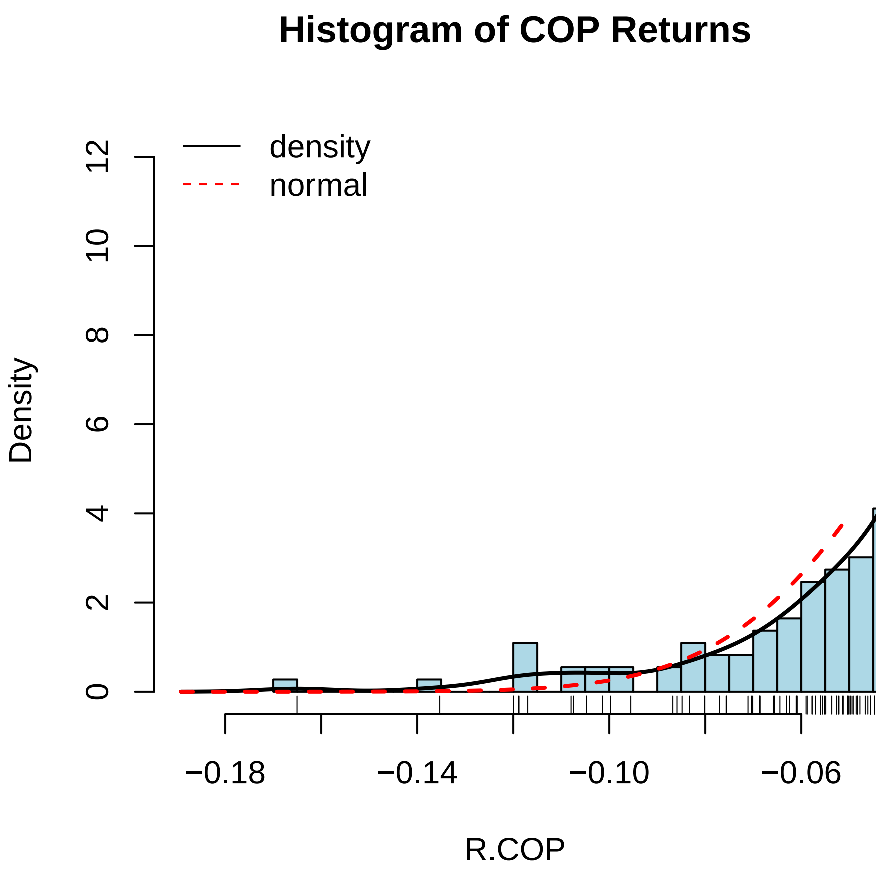
Here we plot the histogram with a kernel density estimate and normal curve overlayed to better understand the distribution of COP returns. Plotting the distribution of returns will give us a better idea if COP returns have ”fat tails” or is skewed relative to a normal distribution.

```
hist(R.COP, breaks=50, main="Histogram of COP Returns",
     col="lightblue", probability=TRUE)
lines(density(R.COP), lwd=2)
curve(dnorm(x, mean=mean(R.COP), sd=sd(R.COP)),
      add=TRUE, col="red", lty=2, lwd=2)
rug(R.COP)
legend("topleft", legend=c("density", "normal"),
       col=c("black", "red"), lty=c(1, 2), bty="n")
```



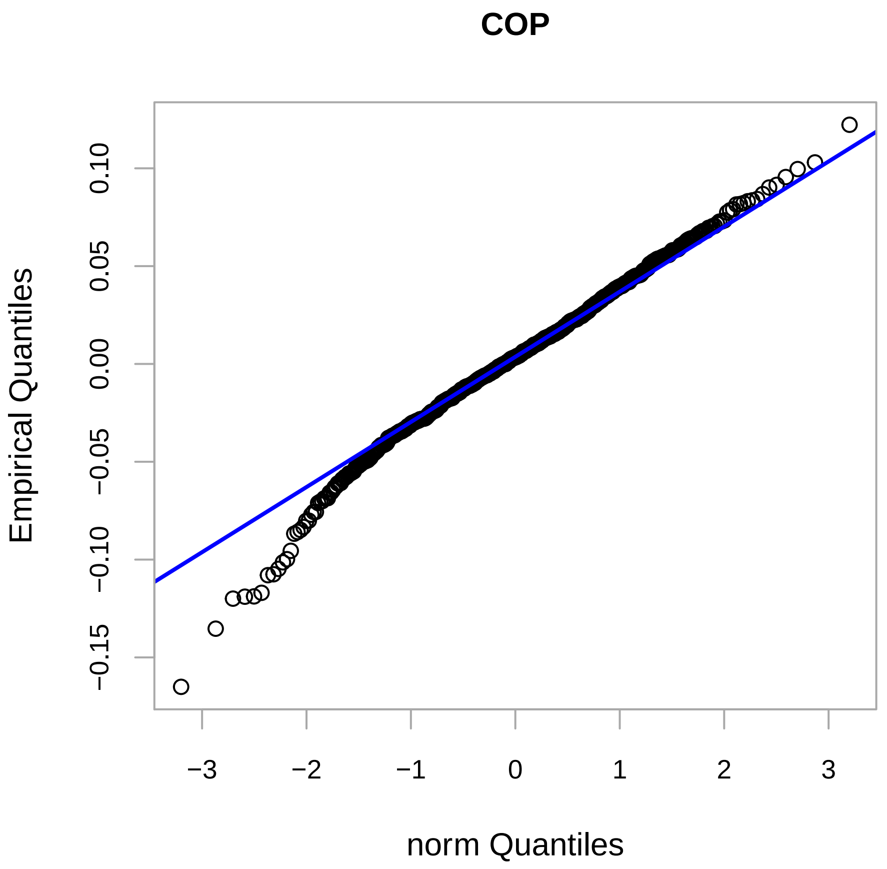
Here we zoom in on the left tail and see that indeed, there are more extreme left tail events than predicted by a normal distribution.

```r
hist(R.COP, breaks=50, main="Histogram of COP Returns",
     col="lightblue", probability=TRUE,
     xlim=c(min(density(R.COP)$x), -0.05))
lines(density(R.COP), lwd=2)
curve(dnorm(x, mean=mean(R.COP), sd=sd(R.COP)),
      add=TRUE, col="red", lty=2, lwd=2)
rug(R.COP)
legend("topleft", legend=c("density", "normal"),
       col=c("black", "red"), lty=c(1, 2), bty="n")
```



We can also inspect a Quantile-Quantile plot and again we see that the COP weekly returns exhibit a fat left tail.

```
chart.QQPlot(R.COP)
```



2 VaR Estimation Approaches

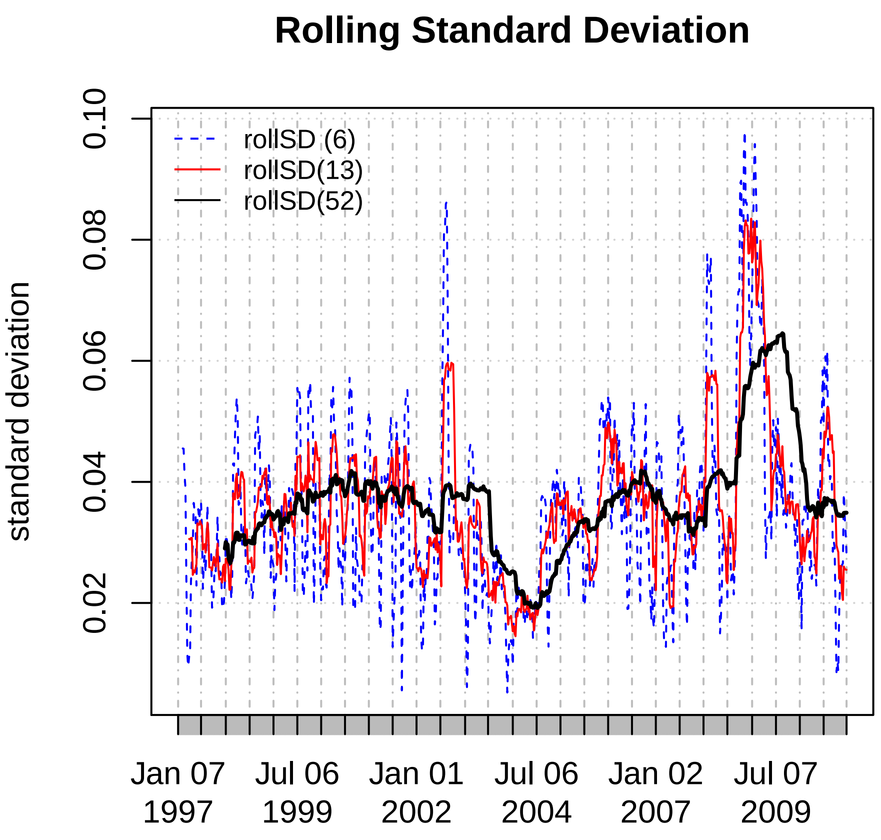
2.1 Historical Standard Deviation

TODO: Content on how volatility changes over time.

Here we estimate historical standard deviation over a moving window using the most recent 6, 13, and 52 periods.

```
# Compute rolling standard deviation estimate
SD6 <- rollSD(R.COP, 6)
SD13 <- rollSD(R.COP, 13)
SD52 <- rollSD(R.COP, 52)
# Plot rolling standard deviation estimates
plot(SD6, type="n", main="Rolling Standard Deviation",
     ylab="standard deviation")
lines(SD6, col="blue", lty=2)
lines(SD13, col="red")
```

```r
lines(SD52, lwd=2)
legend("topleft", legend=c("rollSD (6)", "rollSD(13)", "rollSD(52)"),
       bty="n", lty=c(2, 1, 1), col=c("blue", "red", "black"), cex=0.8)
```



We can see that volatility varies over time and using different widths of a moving window result in different estimates.

2.2 Exponential Smoothing Volatility

Exponential smoothing is an approach of estimating volatility that gives more weight to more recent information and less weight to distant information. In the Exponentially Weighted Moving Average (EWMA) Model, the weights decrease exponentially as we move backwards through time.

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

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

where:

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

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

λ is a constant between 0 and 1

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. The realized volatility defined as the equally weighted average of the standard deviation of the previous n periods.

Here we fit an EWMA model to the COP weekly returns to estimate VaR. 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 volatility and realized volatility. Here we compute realized volatility as the equally weighted average of the standard deviation of the previous 13 periods.

```
# Estimating volatility
# EWMA Model
initialWindow <- 100
n <- 13
type <- "volatility"

# Fit an EWMA model to estimate volatility
# Choose an optimal lambda parameter that minimizes the mean squared error
# betwen realized volatility and the EWMA model volatility estimate
ewmaModel <- EWMA(R.COP, lambda=NULL, initialWindow, n, type)
ewmaModel

## EWMA Estimate
##
## Parameters
## lambda: 0.8872
## initialWindow: 100
## type: volatility
##
## Final Period EWMA Estimate:
##                  COP
## 2010-12-28 0.03188


# One period ahead forecast
ewmaVolForecast <- forecast(ewmaModel)

# VaR using EWMA volatility forecast
# Here we assume the expected value of returns is simply the mean of returns
mean(R.COP) + as.numeric(ewmaVolForecast) * qnorm(0.05)

## [1] -0.04719
```

2.3 GARCH Model Volatility

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

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

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

$$\gamma + \alpha + \beta = 1 \tag{3}$$

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

$$\sigma_n^2 = \omega + \alpha u_{n-1}^2 + \beta \sigma_{n-1}^2 \tag{4}$$

Here we specify and fit a GARCH model to the COP weekly returns to estimate VaR.

```r
# Specify and fit a GARCH Model
garchModel <- uvGARCH(R.COP, armaOrder=c(0,0))

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


# One period ahead forecast of GARCH model
garchForecast <- forecast(garchModel, 1)

# VaR forecast using GARCH Model
fitted(garchForecast) + sigma(garchForecast) * qnorm(0.05)

##     2010-12-28
## T+1   -0.05074
```

3 Historic Simulation

Now we move to estimating VaR using historical data. Using historical data to estimate VaR is a simple and convenient method that has the advantage of requiring zero parameters to estimate except for the moving window width. Another advantage is that no distributional assumptions need to be made. If the distribution of returns is skewed or fat-tailed, this will will be captured in the estimate.

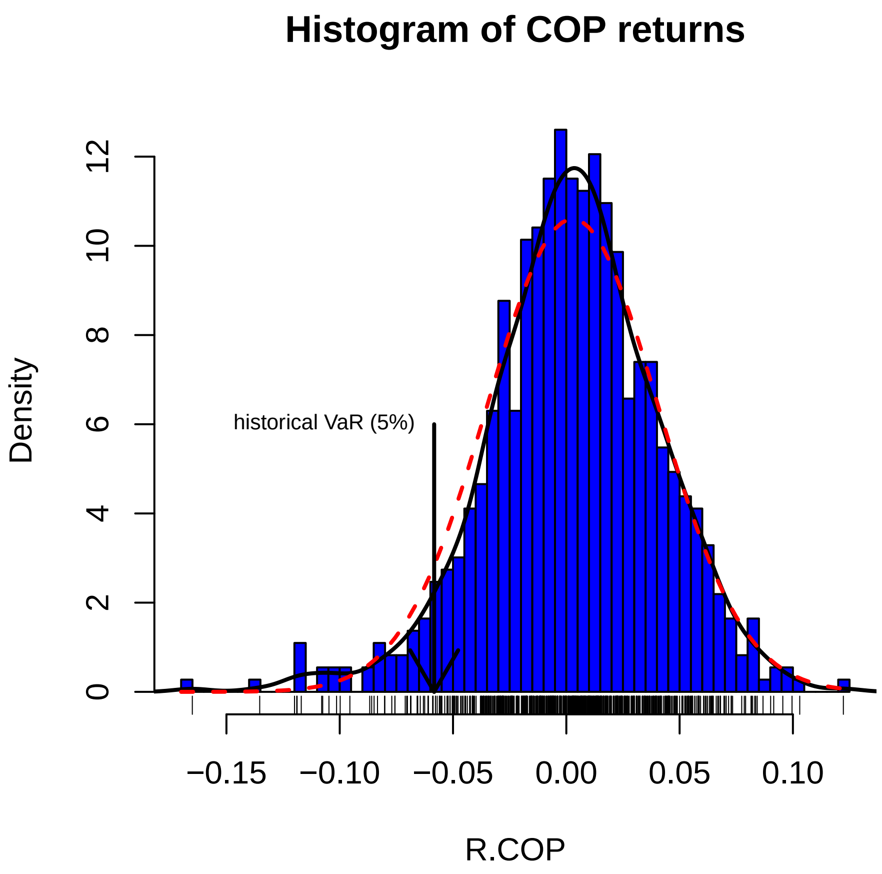
In theory, we could accurately estimate percentiles of a distribution given an infinite amount of historical data. This is data intensive and unrealistic. In reality, we only have a sample of the population of asset returns. One method to address this is the bootstrap.

Bootstrapping is a statistical method for estimating the sampling distribution of an estimator by sampling with replacement from the original sample. Bootstrap resampling generates data by sampling with replacement from the original observed data. One key assumption is that returns are independent. By random resampling, we break any pattern of time variation in returns. Another drawback is that resampling requires large sample sizes and is relatively computationally intensive.

Here we estimate VaR of COP weekly returns at the 5% α level.

```r
# Historical VaR estimate at the 5% level
historicalVaR5 <- VaR(R.COP, p=0.95, method="historical")

# VaR estimate assuming a normal distribution
normalVaR5 <- VaR(R.COP, p=0.95, method="gaussian")

# Bootstrapped historical VaR estimate
bootHistVaR5 <- bootVaR(R.COP, p=0.95, method="historical")

rnames <- c("Historical", "Normal", "Bootstrap Historical")
matrix(c(historicalVaR5, normalVaR5, bootHistVaR5[1,]),
       nrow=3, ncol=1, dimnames=list(rnames, "VaR (5%)"))

##                        VaR (5%)
## Historical             -0.05832
## Normal                 -0.05905
## Bootstrap Historical -0.05865
```

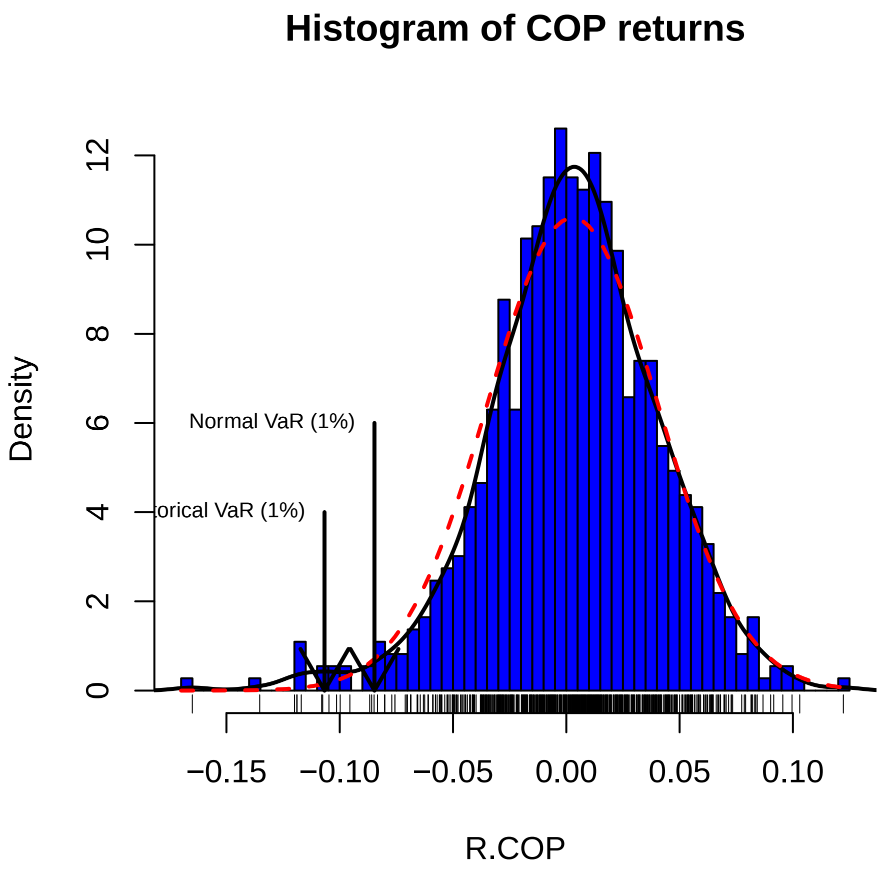
```r
hist(R.COP, main="Histogram of COP returns", breaks=50,
     col="blue", probability=TRUE)
lines(density(R.COP), lwd=2)
curve(dnorm(x, mean=mean(R.COP), sd=sd(R.COP)),
      add=TRUE, col="red", lty=2, lwd=2)
rug(R.COP)
arrows(historicalVaR5, 0, historicalVaR5, 6, code=1, lwd=2)
text(historicalVaR5, 6, labels="historical VaR (5%)", pos=2, cex=0.7)
```



We now estimate VaR of COP weekly returns at the 1% α level. As we move further in the tail, we see that the VaR estimate assuming normally distributed returns differes from the historical VaR estimate.

```
# Estimating VaR at the 1% level
historicalVaR1 <- VaR(R.COP, p=0.99, method="historical")
normalVaR1 <- VaR(R.COP, p=0.99, method="gaussian")
bootHistVaR1 <- bootVaR(R.COP, p=0.99, method="historical")

matrix(c(historicalVaR1, normalVaR1, bootHistVaR1[1,]),
       nrow=3, ncol=1, dimnames=list(rnames, "VaR (1%)"))

##                        VaR (1%)
## Historical             -0.10671
## Normal                 -0.08468
## Bootstrap Historical -0.10552
```

```r
hist(R.COP, main="Histogram of COP returns", breaks=50,
     col="blue", probability=TRUE)
lines(density(R.COP), lwd=2)
curve(dnorm(x, mean=mean(R.COP), sd=sd(R.COP)),
      add=TRUE, col="red", lty=2, lwd=2)
rug(R.COP)
arrows(historicalVaR1, 0, historicalVaR1, 4, code=1, lwd=2)
text(historicalVaR1, 4, labels="Historical VaR (1%)", pos=2, cex=0.7)
arrows(normalVaR1, 0, normalVaR1, 6, code=1, lwd=2)
text(normalVaR1, 6, labels="Normal VaR (1%)", pos=2, cex=0.7)
```



4 Return Aggregation

Here we consider a portfolio of the first 10 assets in the largecap_weekly dataset.

Now consider computing VaR for a portfolio with a number of positions. One method is to use the covariance matrix of asset returns to compute the volatility and VaR. Another approach is return aggregation where the historical returns of each asset are weighted by the relative position size and aggregated.

The VarCov approach assumes that asset returns are jointly normal and the return on the portfolio is also normally distributed. The RiskMetrics^{TM} approach is commonly termed the Variance-Covariance (VarCov) approach. Using this approach, a portfolio of N positions requres N volatility estimates and $N(N-1)/2$ correlation estimates. For large portfolios with a large number of positions, this is potentially a very large number which exposes the model to estimation error.

A key concern with the VarCov approach is the correlation estimate. As we saw with volatility in a previous section, correlation changes over time. For example, if correlations increase when markets fall, the VaR estimate of the position may be understated.

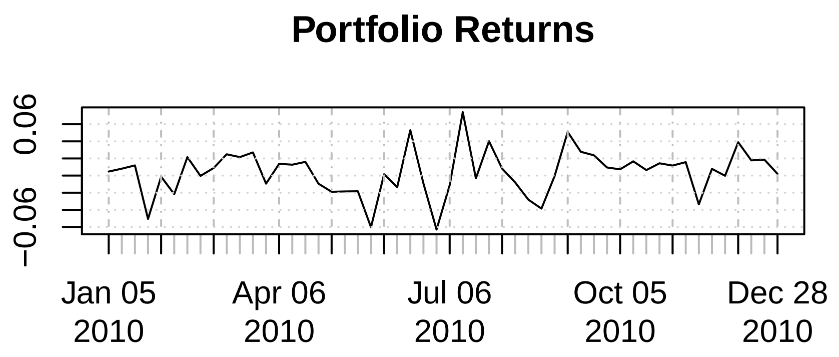
The return aggregation approach is a ”simulation” method where the returns are calculated using historical data, but the weights of each position today. For example, we calculate the returns we would have earned over the most recent K periods by pretending the relative postions we hold today are the same positions we held K days ago. This approach has the advantage that no parameters need to be estimated (not considering the K lookback period). This means we do not have to estimate correlation. If markets fall and move together, this will be captured by the return aggregation approach. This approach will also capture fatter tails relative to a normal distribution.

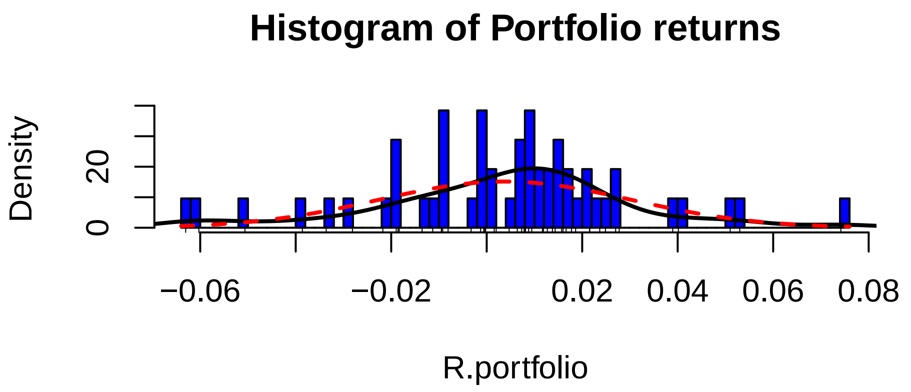
Suppose we have an equally weighted portfolio consisting of the first 10 assets in the largecap_weekly dataset and use a lookback period of $K = 52$.

```r
# Asset returns
R <- largecap_weekly[, 1:10]

# Lookback period
K <- 52

# Set the weights K periods ago
weights <- xts(matrix(rep(1 / 10, 10), nrow=1), index(R)[nrow(R) - K])

# Calculate the portfolio returns for the most recent K periods
R.portfolio <- Return.rebalancing(R, weights)
```

Here we plot the aggregated portfolio returns and histogram of returns.

```r
par(mfrow=c(2,1))
# Portfolio returns
plot(R.portfolio, main="Portfolio Returns")

# Histogram of portfolio returns
hist(R.portfolio, main="Histogram of Portfolio returns", breaks=50,
     col="blue", probability=TRUE)
lines(density(R.portfolio), lwd=2)
curve(dnorm(x, mean=mean(R.portfolio), sd=sd(R.portfolio)),
      add=TRUE, col="red", lty=2, lwd=2)
rug(R.portfolio)
```





```
par(mfrow=c(1,1))
```

Here we estimate the VaR of the portfolio using the aggregated returns. We can see that the normal assumption underestimates the historical VaR estimate.

```
# Estimate the VaR of the portfolio using the last 52 periods
# Historical VaR estimate
portfVaR.HS <- VaR(R.portfolio, p=0.95, method="historical")

# Bootstrapped VaR estimate
portfVaR.BootHS <- bootVaR(R.portfolio, p=0.95, method="historical")

# Normal VaR estimate
portfVaR.normal <- VaR(R.portfolio, p=0.95, method="gaussian")
```

Now we estimate the portfolio VaR using the VarCov approach.

```r
# Use the most recent 52 periods to compute the sample covariance matrix
sampleCov <- cov(tail(R, 52))
# Convert the xts object of weights to a matrix
weights <- matrix(weights, ncol=1)

# Compute the portfolio VaR estimate using the VarCov approach with sample
# covariance matrix
portfVaR.cov <- sqrt(t(weights) %*% sampleCov %*% weights) * qnorm(0.05)

# Use EWMA model to compute variance covariance matrix
# EWMA model to compute variance covariance matrix
ewmaCov <- EWMA(tail(R, 52), lambda=0.9, initialWindow=10,
                type="covariance")$estimate

# Compute the portfolio VaR estimate using the VarCov approach with EWMA
# model estimated covariance matrix
portfVaR.ewmaCov <- sqrt(t(weights) %*% ewmaCov %*% weights) * qnorm(0.05)
```

Returning our focus back to the return aggregation approach. We can also compute the portfolio VaR using methods we covered earlier when computing the VaR estimate of Conoco Phillips (COP) returns.

```r
# GARCH Model
garchModel <- uvGARCH(R.portfolio, armaOrder=c(0,0))

## Error:
## ugarchfit-->error:  function requires at least 100 data
##  points to run

garchForecast <- forecast(garchModel, 1)

# VaR using GARCH Model
portfVaR.GARCH <- fitted(garchForecast) + sigma(garchForecast) * qnorm(0.05)

# Portfolio VaR estimate with EWMA model
ewmaModel <- EWMA(R.portfolio, lambda=NULL, initialWindow=10, n, type)
# One period ahead forecast
ewmaVolForecast <- forecast(ewmaModel)

# VaR using EWMA volatility forecast
# Here we assume the expected value of returns is simply the mean of returns
portfVaR.EWMA <- mean(R.portfolio) + as.numeric(ewmaVolForecast) * qnorm(0.05)
```

Here we compare each portfolio VaR estimate.

```r
dfVaR <- t(data.frame(portfVaR.HS, portfVaR.BootHS[1,], portfVaR.normal,
                      portfVaR.cov, portfVaR.ewmaCov, portfVaR.GARCH,
                      portfVaR.EWMA))
rownames(dfVaR) <- c("portfVaR.HS", "portfVaR.BootHS", "portfVaR.normal",
                      "portfVaR.cov", "portfVaR.ewmaCov", "portfVaR.GARCH",
                      "portfVaR.EWMA")
dfVaR
```

```
##                      VaR
## portfVaR.HS      -0.04402
## portfVaR.BootHS  -0.04298
## portfVaR.normal  -0.03911
## portfVaR.cov     -0.04319
## portfVaR.ewmaCov -0.03936
## portfVaR.GARCH   -0.05074
## portfVaR.EWMA    -0.03117
```

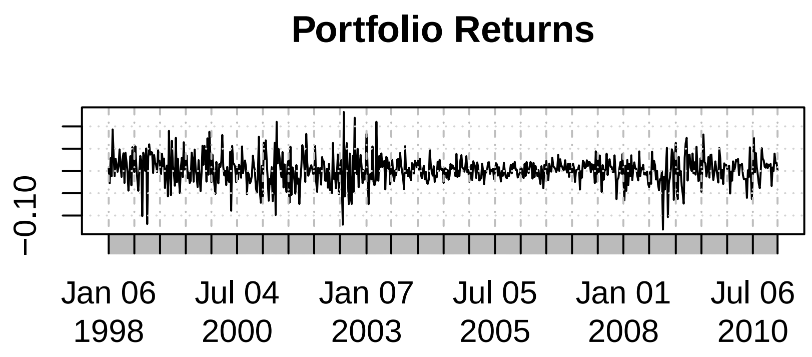
5 VaR Backtesting

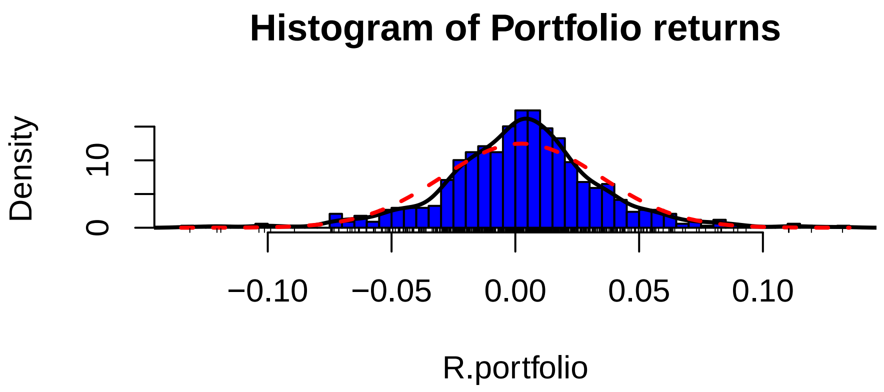
TODO: Comments on VaR backtesting

Here we consider an equal weight portfolio, rebalanced annually, of the first 10 assets of the largecap_weekly dataset.

```
R <- largecap_weekly[, 1:10]

# Annual rebalance dates
rebalanceDates <- index(R)[endpoints(index(R), on="years")]

# Create an xts object of weights at the specified rebalance dates
weights <- xts(matrix(1 / 10, nrow=length(rebalanceDates),
                      ncol=10), rebalanceDates)

# Calculate the aggregate portfolio return
R.portfolio <- Return.rebalancing(R, weights)
```

```
par(mfrow=c(2,1))
# Portfolio returns
plot(R.portfolio, main="Portfolio Returns")

# Histogram of portfolio returns
hist(R.portfolio, main="Histogram of Portfolio returns", breaks=50,
     col="blue", probability=TRUE)
lines(density(R.portfolio), lwd=2)
curve(dnorm(x, mean=mean(R.portfolio), sd=sd(R.portfolio)),
      add=TRUE, col="red", lty=2, lwd=2)
rug(R.portfolio)
```





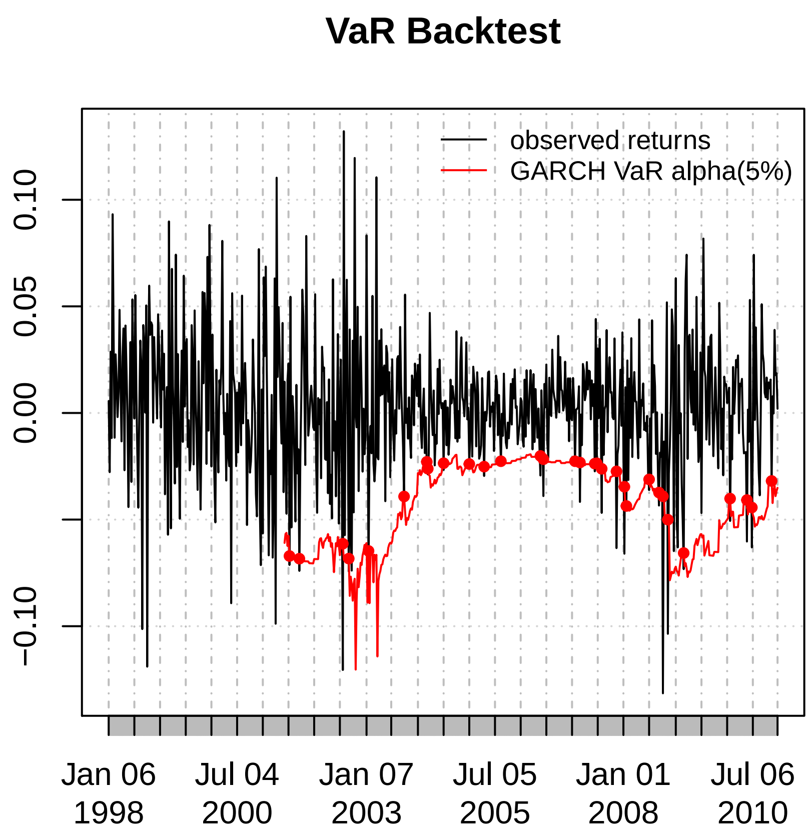
```r
par(mfrow=c(1,1))
```

Here we fit a GARCH(1,1) model to the portfolio returns and run a backtest on the VaR estimates to test for the number of violations.

```r
garchModel <- uvGARCH(R.portfolio, armaOrder=c(0,0))
btVaR.GARCH <- backtestVaR.GARCH(garchModel, p=0.95, refitEvery=5, window=100)
btVaR.GARCH

## Value-at-Risk Backtest
##
## Returns Data:
## NULL
##
## 1 - p tail quantile:
## [1] 0.05
##
```

```
## Number of Violations:
## GARCH VaR alpha(5%)
##                  31
##
## Violations (%):
## GARCH VaR alpha(5%)
##                 6.2
```

```
# Plot the GARCH VaR backtest
plot(btVaR.GARCH, pch=20, legendLoc="topright")
```

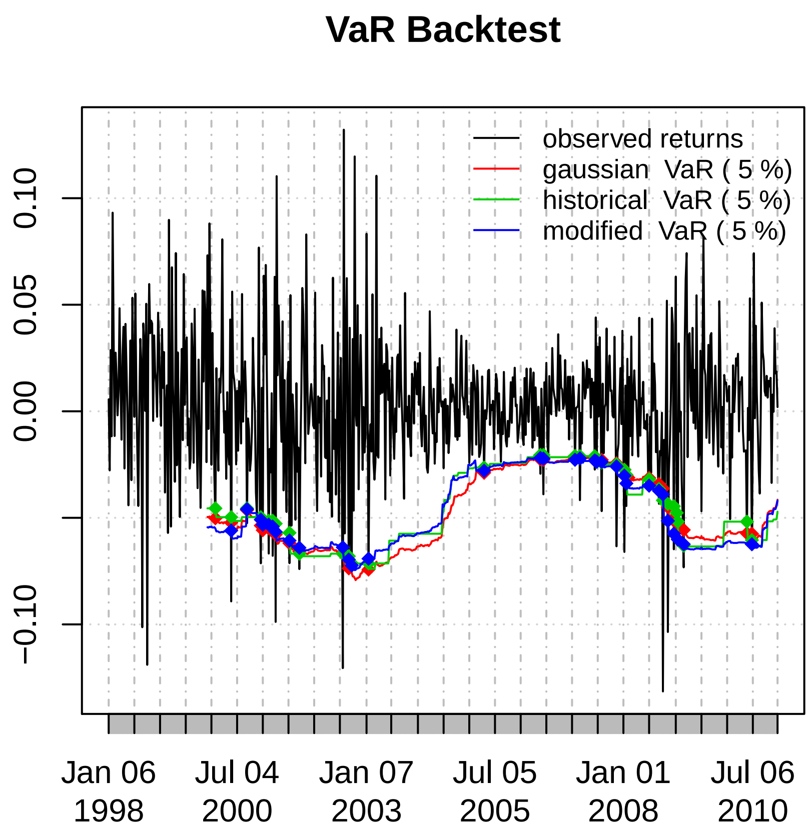


Now we perform a VaR backtest using historical VaR estimates.

```
# Run a VaR backtest on portfolio returns
# Compute VaR estimate using gaussian, historical, and modified methods
backtest <- backtestVaR(R.portfolio, window=100, p=0.95,
                        method=c("gaussian", "historical", "modified"))
backtest
```

```
## Value-at-Risk Backtest
##
## Returns Data:
## [1] "portfolio.returns"
##
## 1 - p tail quantile:
## [1] 0.05
##
## Number of Violations:
##   gaussian  VaR ( 5 %) historical  VaR ( 5 %)   modified  VaR ( 5 %)
##                     34                      35                      32
##
## Violations (%):
##   gaussian  VaR ( 5 %) historical  VaR ( 5 %)   modified  VaR ( 5 %)
##                  5.882                   6.055                   5.536
```

```
# plot the VaR backtest
plot(backtest, pch=18, legendLoc="topright")
```


VaR Backtest
observed returns
gaussian VaR (5 %)
historical VaR (5 %)
modified VaR (5 %)
0.10
0.05
0.00
−0.10
Jan 06 1998
Jul 04 2000
Jan 07 2003
Jul 05 2005
Jan 01 2008
Jul 06 2010