Spreads, Beta and Risks

```r
pepsi <- getSymbols('PEP', from = '2013-01-01',
                    to = '2014-01-01', adjust = T, auto.assign = F)
```

```
'getSymbols' currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
'loadSymbols' to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
Warning in read.table(file = file, header = header, sep = sep,
quote = quote, : incomplete final line found by readTableHeader
on 'https://query1.finance.yahoo.com/v7/finance/download/PEP?
period1=-2208988800&period2=1583798400&interval=1d&events=split&crumb=1y.QAbwvddB'

Warning in read.table(file = file, header = header, sep = sep,
quote = quote, : incomplete final line found by readTableHeader
on 'https://query1.finance.yahoo.com/v7/finance/download/PEP?
period1=-2208988800&period2=1583798400&interval=1d&events=split&crumb=1y.QAbwvddB'
```

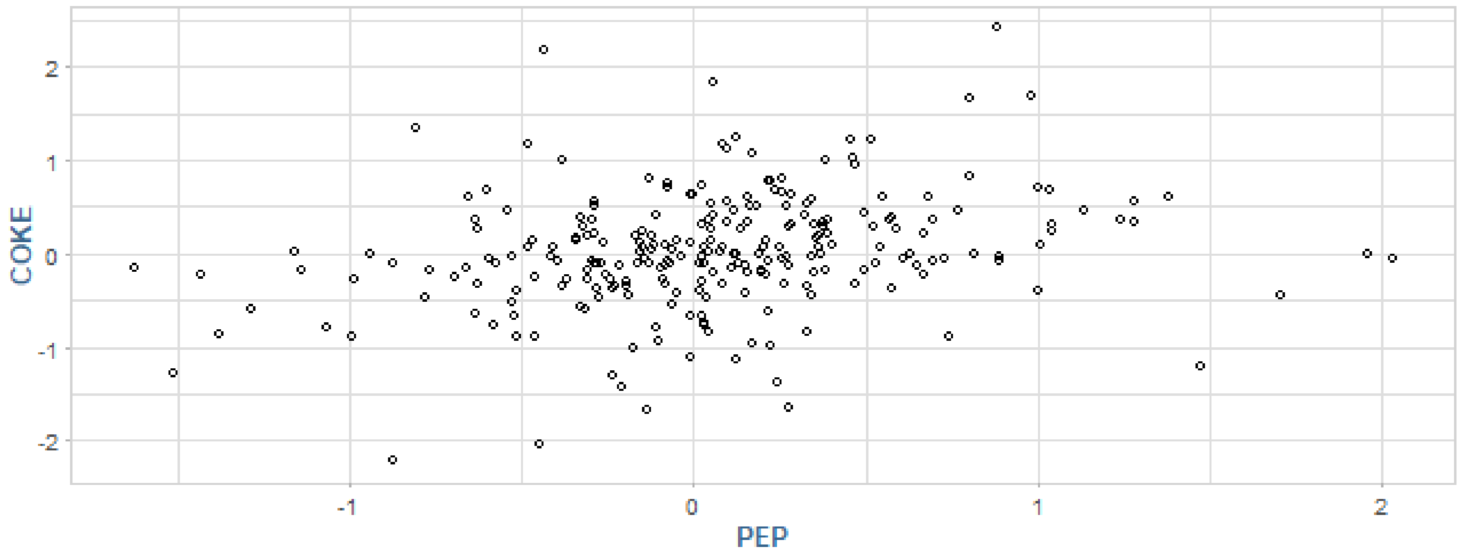
```r
coke <- getSymbols('COKE', from = '2013-01-01',
                    to = '2014-01-01', adjust = T, auto.assign = F)
```

```
Warning in read.table(file = file, header = header, sep = sep,
quote = quote, : incomplete final line found by readTableHeader
on 'https://query2.finance.yahoo.com/v7/finance/download/COKE?
period1=-2208988800&period2=1583798400&interval=1d&events=split&crumb=1y.QAbwvddB'
```

```
Warning in read.table(file = file, header = header, sep = sep,
quote = quote, : incomplete final line found by readTableHeader
on 'https://query1.finance.yahoo.com/v7/finance/download/COKE?
period1=-2208988800&period2=1583798400&interval=1d&events=split&crumb=1y.QAbwvddB'
```

```r
Sys.setenv(TZ = "UTC")
```

```r
prices <- cbind(pepsi$PEP.Adjusted, coke$COKE.Adjusted)
price_changes <- data.table(apply(prices, 2, diff), keep.rownames = T)
colnames(price_changes) <- c('Date', 'PEP', 'COKE')

ggplot(price_changes, aes(PEP, COKE)) +
  geom_point(shape = 1)
```

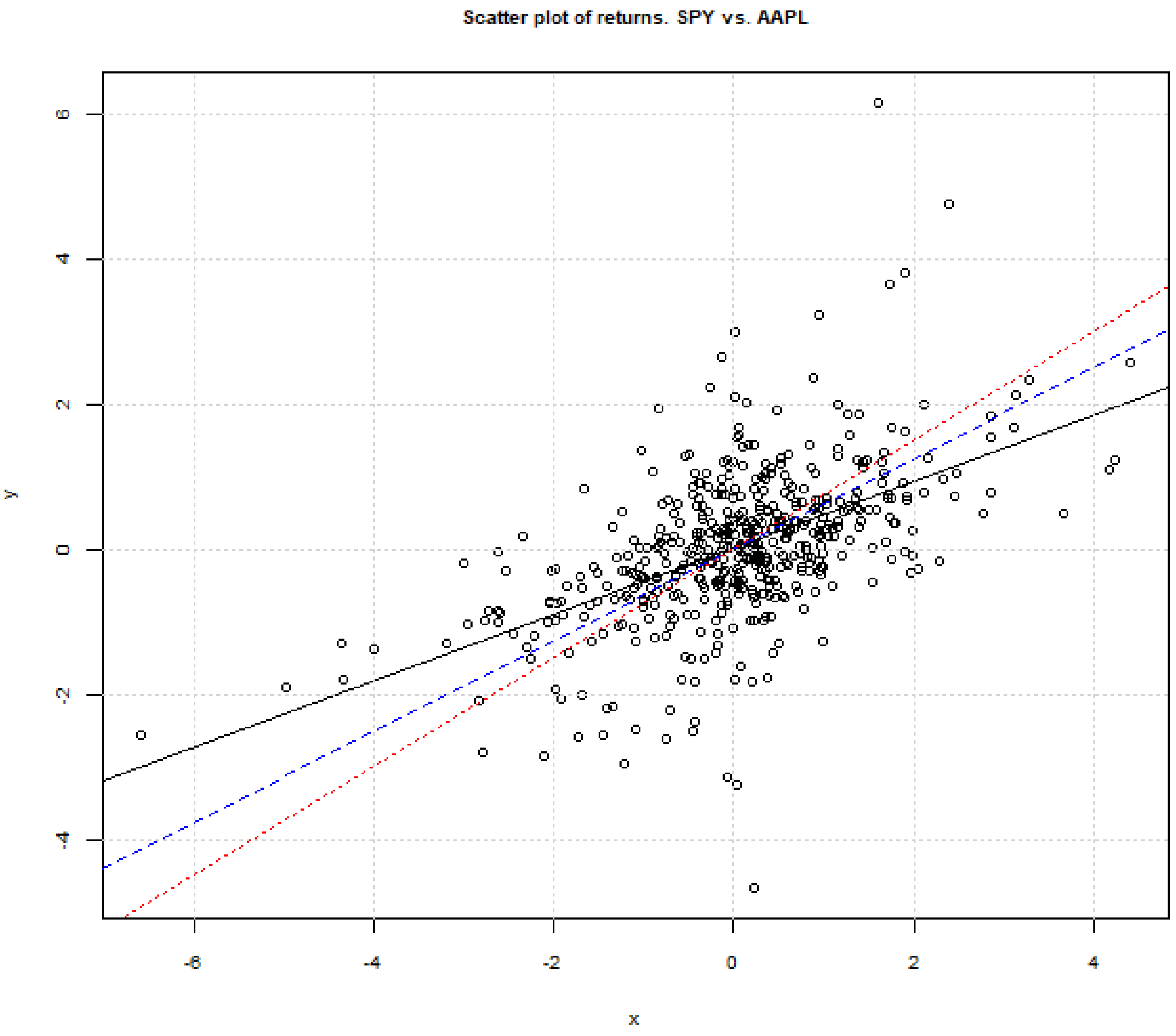


```r
ans <- lm(PEP ~ COKE, data = price_changes)
beta <- coef(ans)[2]
```

```r
ans2 <- lm(COKE ~ PEP, data = price_changes)
beta2 <- coef(ans2)[2]
```

Ordinary Least Squares (blue) approach vs Total Least Squares (red) approach:

```r
SPY <- getSymbols('SPY', from = '2011-01-01',
                  to = '2012-12-31', adjsut = T, auto.assign = F)

AAPL <- getSymbols('AAPL', from = '2011-01-01',
                  to = '2012-12-31', adjsut = T, auto.assign = F)

x <- diff(as.numeric(SPY$SPY.Adjusted))
y <- diff(as.numeric(AAPL$AAPL.Adjusted))

plot(x, y, main = "Scatter plot of returns. SPY vs. AAPL",
     cex.main = 0.8, cex.lab = 0.8, cex.axis = 0.8)
abline(lm(y ~ x))
abline(lm(x ~ y), lty = 2, col = "blue")
grid()

r <- prcomp(~ x + y)
slope <- r$rotation[2, 1] / r$rotation[1, 1]
intercept <- r$center[2] - slope * r$center[1]

# Show first principal component
abline(a = intercept, b = slope, lty = 3, col = "red")
```



Constructing the spread

```r
# Function to calculate the spread
calculate_spread <- function(x, y, beta) {
    return(y - beta * x)
}

# Function to calculate the beta and level
# given start and end dates
```

```r
calculate_beta_and_level <- function(x, y,
  start_date, end_date) {
  require(xts)

  time_range <- paste(start_date, "::",
    end_date, sep = "")

  x <- x[time_range]
  y <- y[time_range]

  dx <- diff(x[time_range])
  dy <- diff(y[time_range])
  r <- prcomp( ~ dx + dy)

  beta <- r$rotation[2, 1] / r$rotation[1, 1]
  spread <- calculate_spread(x, y, beta)
  names(spread) <- "spread"
  level <- mean(spread, na.rm = TRUE)

  outL <- list()
  outL$spread <- spread
  outL$beta <- beta
  outL$level <- level

  return(outL)
}

# Function to calculate buy and sell signals
# with upper and lower threshold
calculate_buy_sell_signals <- function(spread, beta,
  level, lower_threshold, upper_threshold) {

  buy_signals <- ifelse(spread <= level -
   lower_threshold, 1, 0)
  sell_signals <- ifelse(spread >= level +
    upper_threshold, 1, 0)

  # bind these vectors into a matrix
  output <- cbind(spread, buy_signals,
    sell_signals)
  colnames(output) <- c("spread", "buy_signals",
    "sell_signals")

  return(output)
}
```

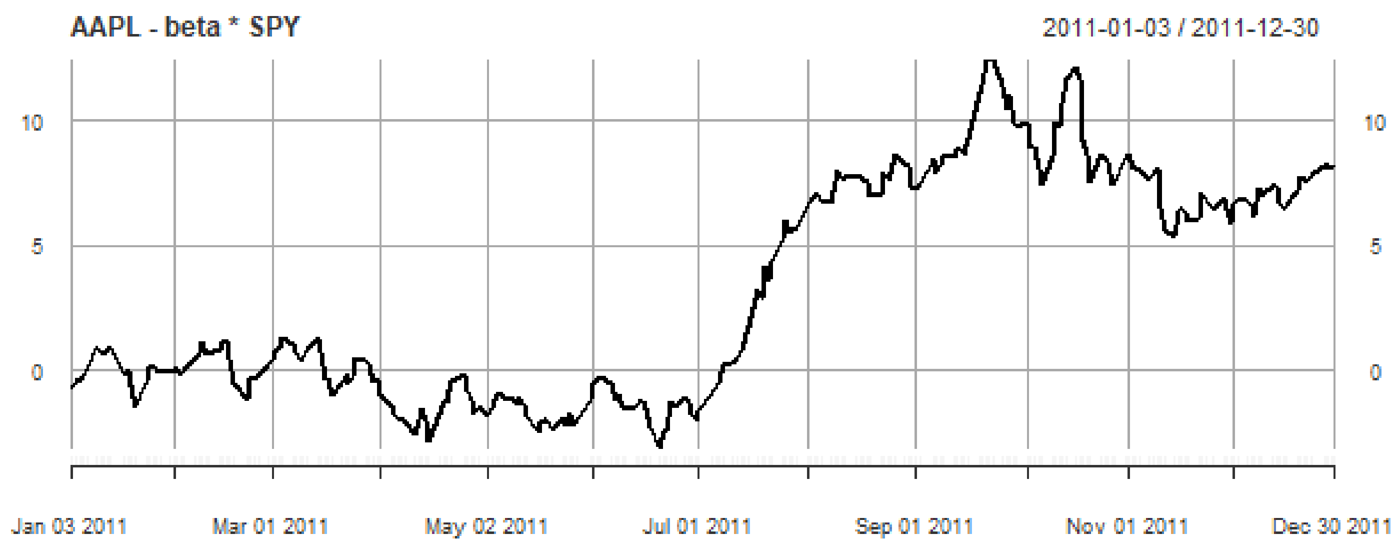
```
# Implementation
# Pick an in-sample date range
start_date <- "2009-01-01"
end_date <- "2011-12-31"
x <- SPY[, 6]
y <- AAPL[, 6]

results <- calculate_beta_and_level(x, y,
  start_date, end_date)

results$beta
```

```
[1] 0.3918035
```

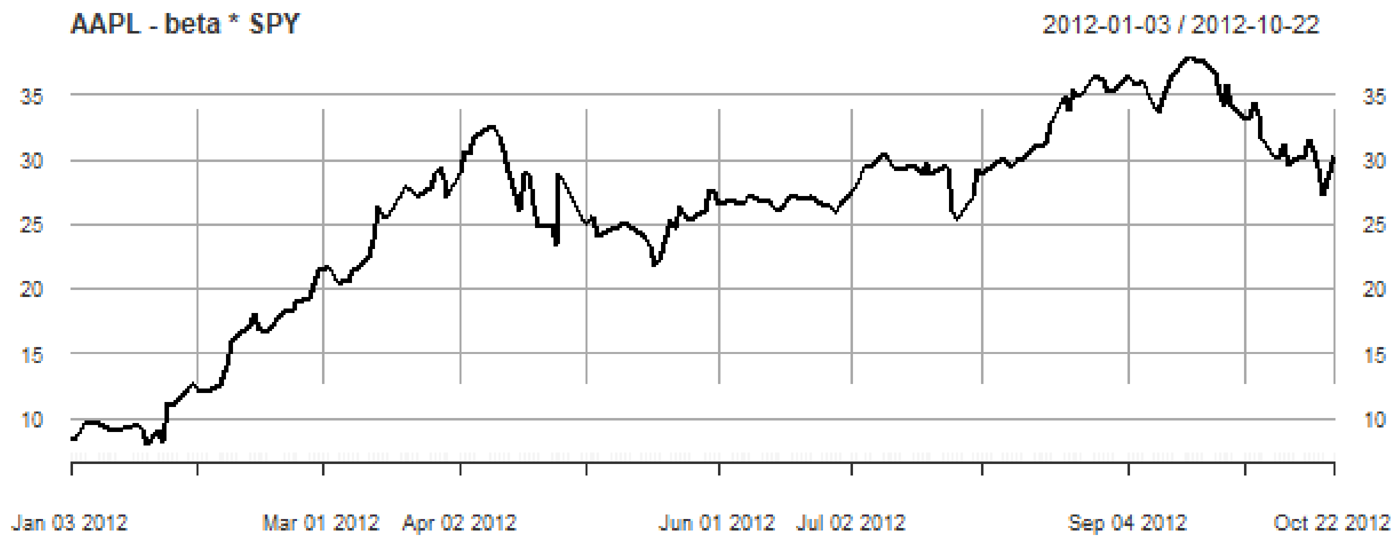
```
## [1] 4.923278

results$level
```

```
[1] 3.297486
```

```
## [1] -239.0602

plot(results$spread, ylab = "Spread Value",
  main = "AAPL - beta * SPY",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
```



```r
# Out of sample start and end dates
start_date_out_sample <- "2012-01-01"
end_date_out_sample <- "2012-10-22"
range <- paste(start_date_out_sample, "::",
  end_date_out_sample, sep = "")

# Out of sample analysis
spread_out_of_sample <- calculate_spread(x[range],
  y[range], results$beta)

plot(spread_out_of_sample, main = "AAPL - beta * SPY",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
abline(h = results$level, lwd = 2)
```



Signal generation and validation

```r
# Rolling window of trading days
window_length <- 10

# Time range
start_date <- "2011-01-01"
end_date <- "2011-12-31"
range <- paste(start_date, "::",
  end_date, sep = "")

# Our stock pair
```

```r
x <- SPY[range, 6]
y <- AAPL[range, 6]

dF <- cbind(x, y)
names(dF) <- c("x", "y")

# Function that we will use to calculate betas
run_regression <- function(dF) {
  return(coef(lm(y ~ x - 1, data = as.data.frame(dF))))
}

rolling_beta <- function(z, width) {
  rollapply(z, width = width, FUN = run_regression,
  by.column = FALSE, align = "right")
}

betas <- rolling_beta(diff(dF), 10)

data <- merge(betas, dF)
data$spread <- data$y - lag(betas, 1) * data$x

returns <- diff(dF) / dF
return_beta <- rolling_beta(returns, 10)
data$spreadR <- diff(data$y) / data$y -
  return_beta * diff(data$x) / data$x

tail(data)
```

```
               betas        x        y    spread      spreadR
2011-12-22 0.4011826 106.7717 49.42252 17.856072 -0.002343066
2011-12-23 0.4479140 107.7263 50.01527  6.797341  0.003314795
2011-12-27 0.4993652 107.8115 50.41209  2.121792  0.007022997
2011-12-28 0.4866673 106.3967 49.92971 -3.201096  0.004251588
2011-12-29 0.4345634 107.4962 50.23725 -2.077628 -0.003371047
2011-12-30 0.4348485 106.9677 50.22237  3.738103  0.004297964
```

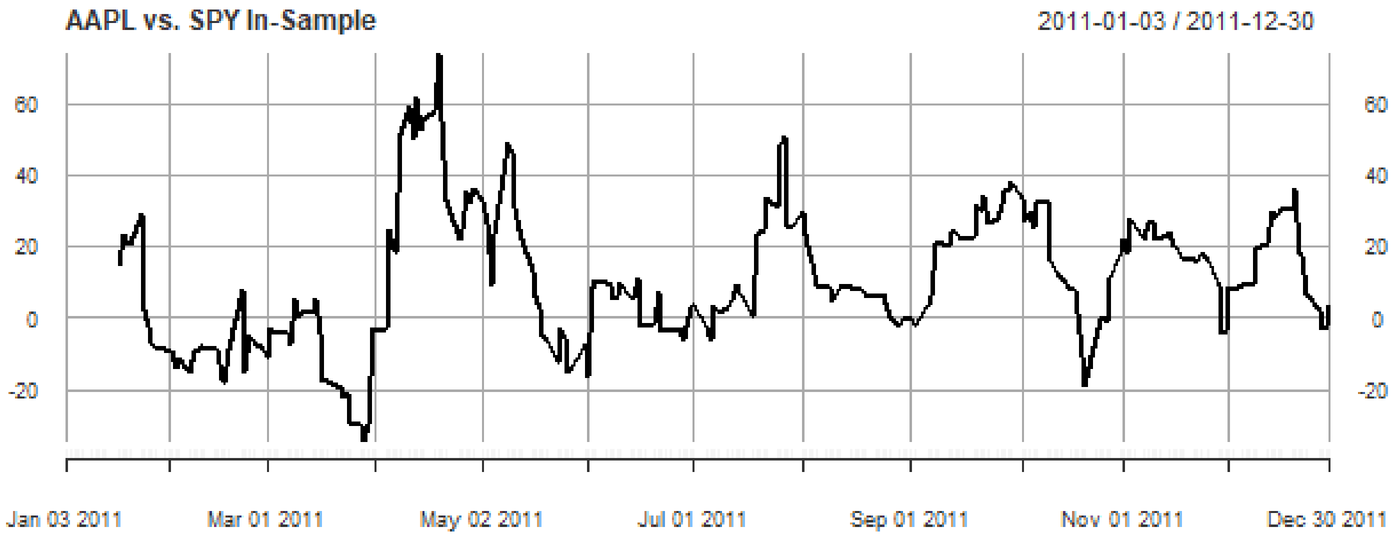
```r
##               betas      x      y    spread      spreadR
## 2011-12-22 2.770586 119.60 383.07 138.70795 -0.002322110
## 2011-12-23 3.094533 120.67 387.66  53.33343  0.003311904
## 2011-12-27 3.450416 120.76 390.74  17.04417  0.007083611
## 2011-12-28 3.364819 119.18 387.00 -24.22055  0.004194527
## 2011-12-29 3.004804 120.41 389.38 -15.77781 -0.003361064
```

```r
threshold <- sd(data$spread, na.rm = TRUE)
```

```r
threshold
```

```
[1] 18.54186
```

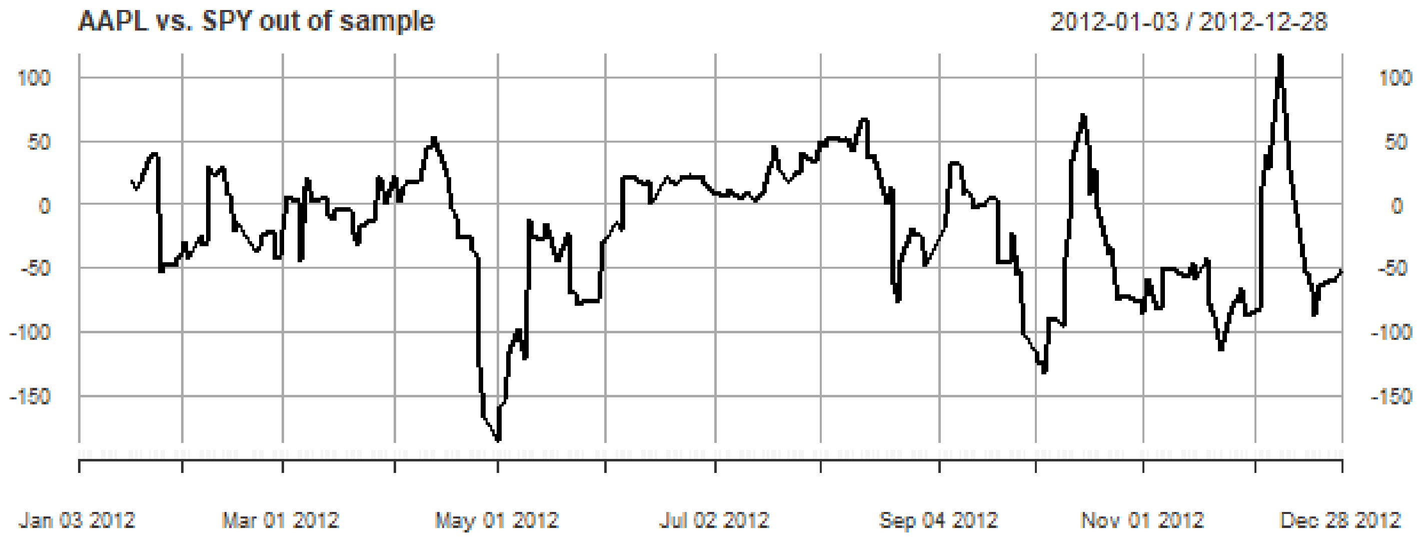
```r
## [1] 143.7734

plot(data$spread, main = "AAPL vs. SPY In-Sample",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
abline(h = threshold, lty = 2)
abline(h = -threshold, lty = 2)
```



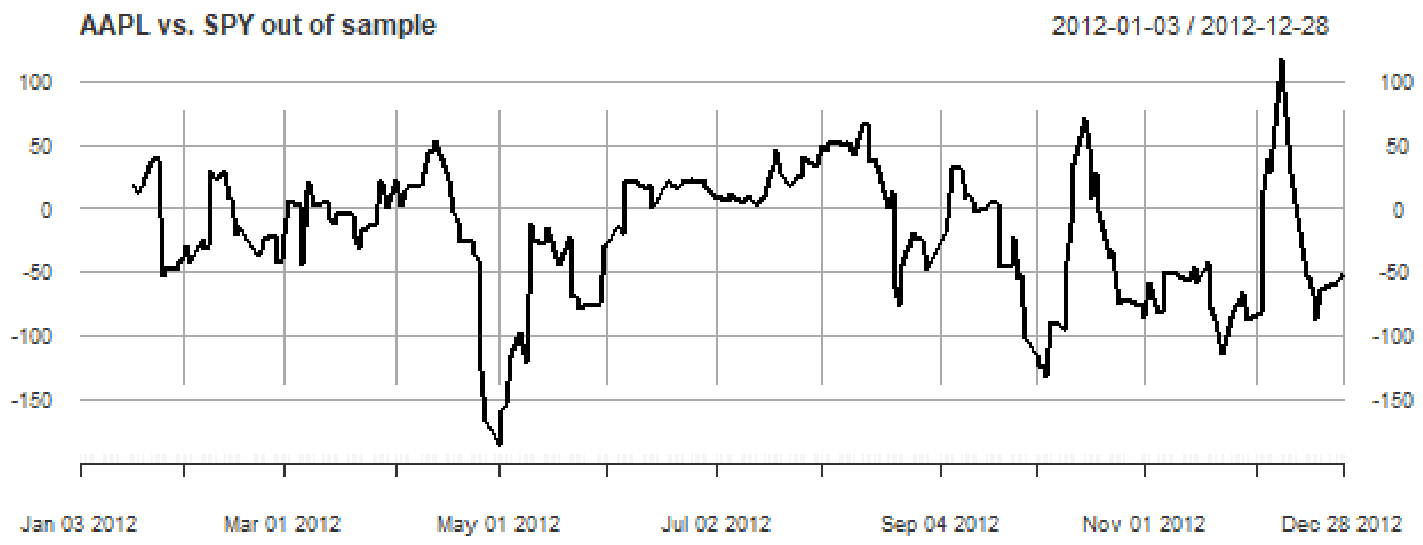
```r
# Construct the out of sample spread
# Keep the same 10 day rolling window
window_length <- 10

# Time range
start_date <- "2012-01-01"
end_date <- "2013-12-31"
range <- paste(start_date, "::",
  end_date, sep = "")

# Our stock pair
x <- SPY[range, 6]
y <- AAPL[range, 6]

# Bind these together into a matrix
dF <- cbind(x, y)
names(dF) <- c("x", "y")
```

```r
# Calculate the out of sample rolling beta
beta_out_of_sample <- rolling_beta(diff(dF), 10)
```

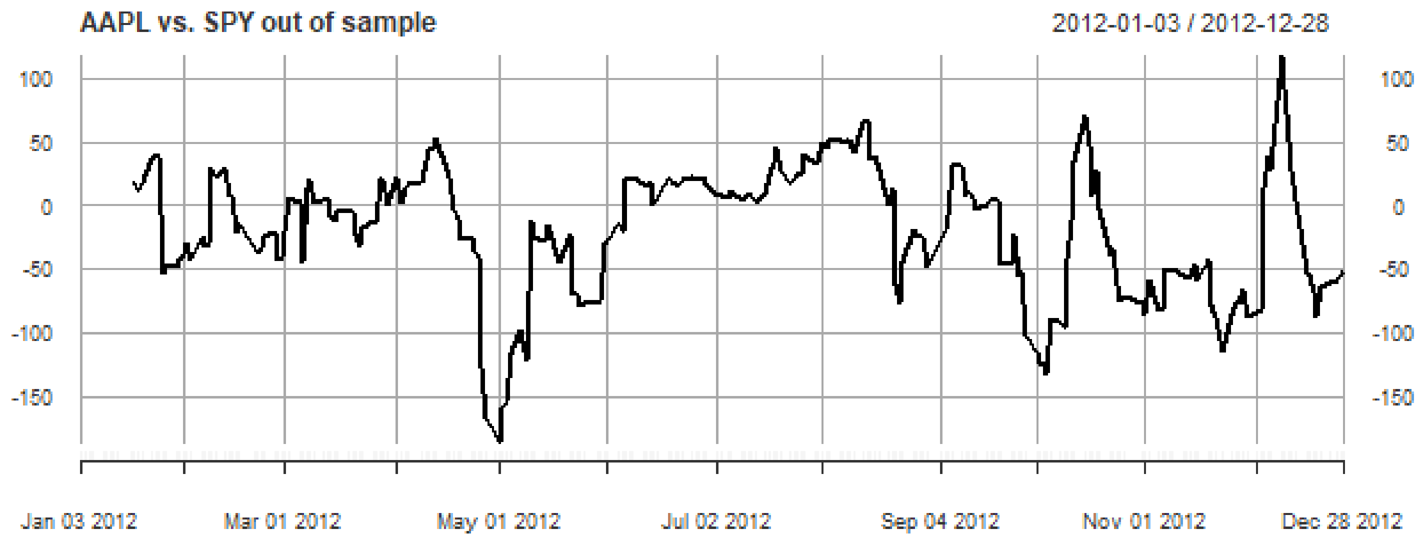
```r
# Buy and sell threshold
data_out <- merge(beta_out_of_sample, dF)
data_out$spread <- data_out$y -
  lag(beta_out_of_sample, 1) * data_out$x

# Plot the spread with in-sample bands
plot(data_out$spread, main = "AAPL vs. SPY out of sample",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
abline(h = threshold, lwd = 2)
abline(h = -threshold, lwd = 2)
```



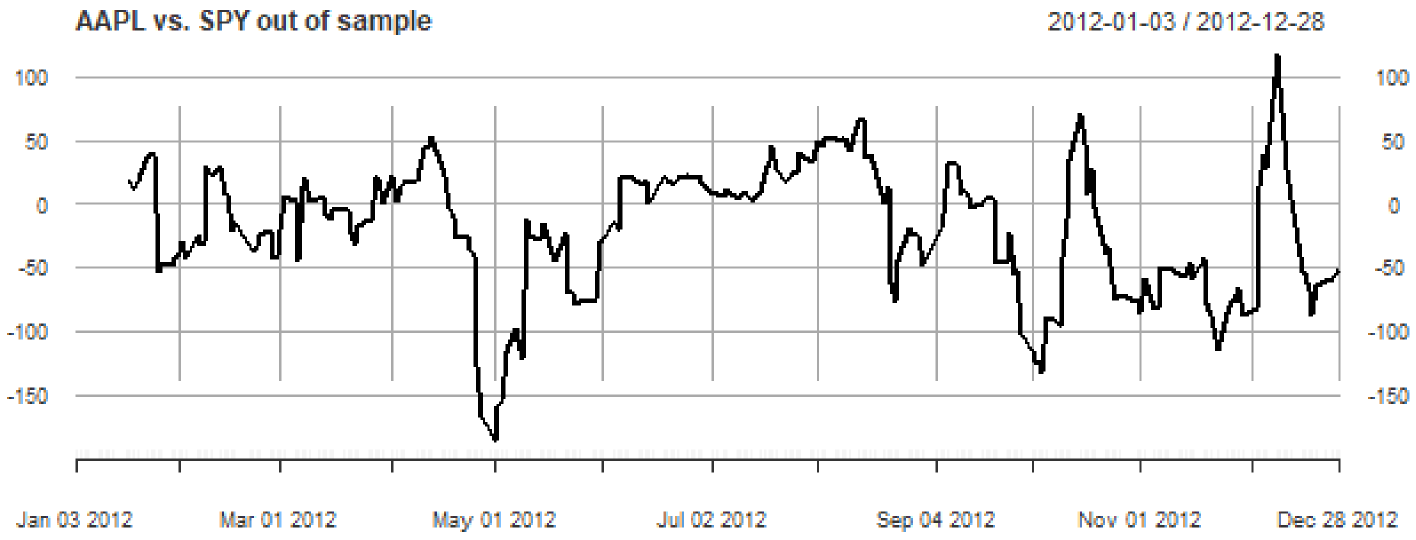
```r
# Generate sell and buy signals
buys <- ifelse(data_out$spread > threshold, 1, 0)
sells <- ifelse(data_out$spread < -threshold, -1, 0)
data_out$signal <- buys + sells

plot(data_out$spread, main = "AAPL vs. SPY out of sample",
 cex.main = 0.8,
 cex.lab = 0.8,
 cex.axis = 0.8)
abline(h = threshold, lty = 2)
abline(h = -threshold, lty = 2)
```



```r
# Buy and sell threshold
data_out <- merge(beta_out_of_sample, dF)
data_out$spread <- data_out$y -
  lag(beta_out_of_sample, 1) * data_out$x

# Plot the spread with in-sample bands
plot(data_out$spread, main = "AAPL vs. SPY out of sample",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
abline(h = threshold, lwd = 2)
abline(h = -threshold, lwd = 2)
```



```
# Generate sell and buy signals
buys <- ifelse(data_out$spread > threshold, 1, 0)
sells <- ifelse(data_out$spread < -threshold, -1, 0)
data_out$signal <- buys + sells

plot(data_out$spread, main = "AAPL vs. SPY out of sample",
 cex.main = 0.8,
 cex.lab = 0.8,
 cex.axis = 0.8)
abline(h = threshold, lty = 2)
abline(h = -threshold, lty = 2)
```



Trading the Spread

```
prev_x_qty <- 0
position <- 0
trade_size <- 100
signal <- as.numeric(data_out$signal)
signal[is.na(signal)] <- 0
beta <- as.numeric(data_out$beta_out_of_sample)

qty_x <- rep(0, length(signal))
qty_y <- rep(0, length(signal))

for(i in 1:length(signal)) {
  if(signal[i] == 1 && position == 0) {
    # buy the spread
    prev_x_qty <- round(beta[i] * trade_size)
    qty_x[i] <- -prev_x_qty
    qty_y[i] <- trade_size
    position <- 1
  }

  if(signal[i] == -1 && position == 0) {
    # sell the spread initially
    prev_x_qty <- round(beta[i] * trade_size)
    qty_x[i] <- prev_x_qty
    qty_y[i] <- -trade_size
    position <- -1
  }

  if(signal[i] == 1 && position == -1) {
  # we are short the spread and need to buy
  qty_x[i] <- -(round(beta[i] * trade_size) +
    prev_x_qty)
  prev_x_qty <- round(beta[i] * trade_size)
  qty_y[i] <- 2 * trade_size
  position <- 1
  }

  if(signal[i] == -1 && position == 1) {
    # we are long the spread and need to sell
    qty_x[i] <- round(beta[i] * trade_size) + prev_x_qty
    prev_x_qty <- round(beta[i] * trade_size)
    qty_y[i] <- -2 * trade_size
    position <- -1
  }
}

qty_x[length(qty_x)] <- -sum(qty_x)
qty_y[length(qty_y)] <- -sum(qty_y)

data_out$qty_x <- qty_x
data_out$qty_y <- qty_y

data_out[1:3, ]
```

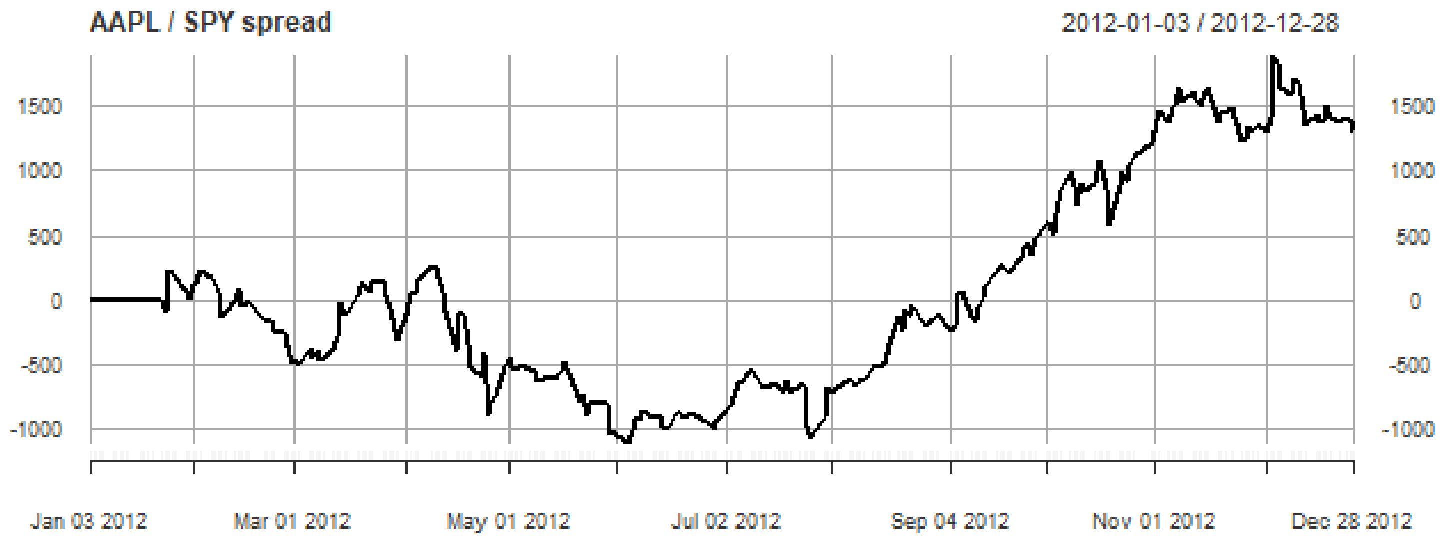
```
           beta_out_of_sample        x        y spread signal qty_x qty_y
2012-01-03                 NA 108.6724 50.99491     NA     NA     0     0
2012-01-04                 NA 108.8429 51.26897     NA     NA     0     0
2012-01-05                 NA 109.1327 51.83817     NA     NA     0     0
```

```
##  beta_out_of_sample        x      y  spread signal qty_x qty_y
## 2012-01-17  2.1511279 123.48 408.20     NA     NA     0     0
## 2012-01-18  2.5890817 124.85 412.44  143.87168  1 -259   100
## 2012-01-19  2.0711505 125.51 411.13   86.17435  0     0     0

tail(data_out, 3)
```

```
           beta_out_of_sample        x        y    spread signal qty_x qty_y
2012-12-26          0.9489434 123.4973 64.17994 -58.00313     -1     0     0
2012-12-27          0.9430080 123.3317 64.43764 -52.59717     -1     0     0
2012-12-28          0.8220693 121.9987 63.75330 -51.29246     -1   -97   100
```

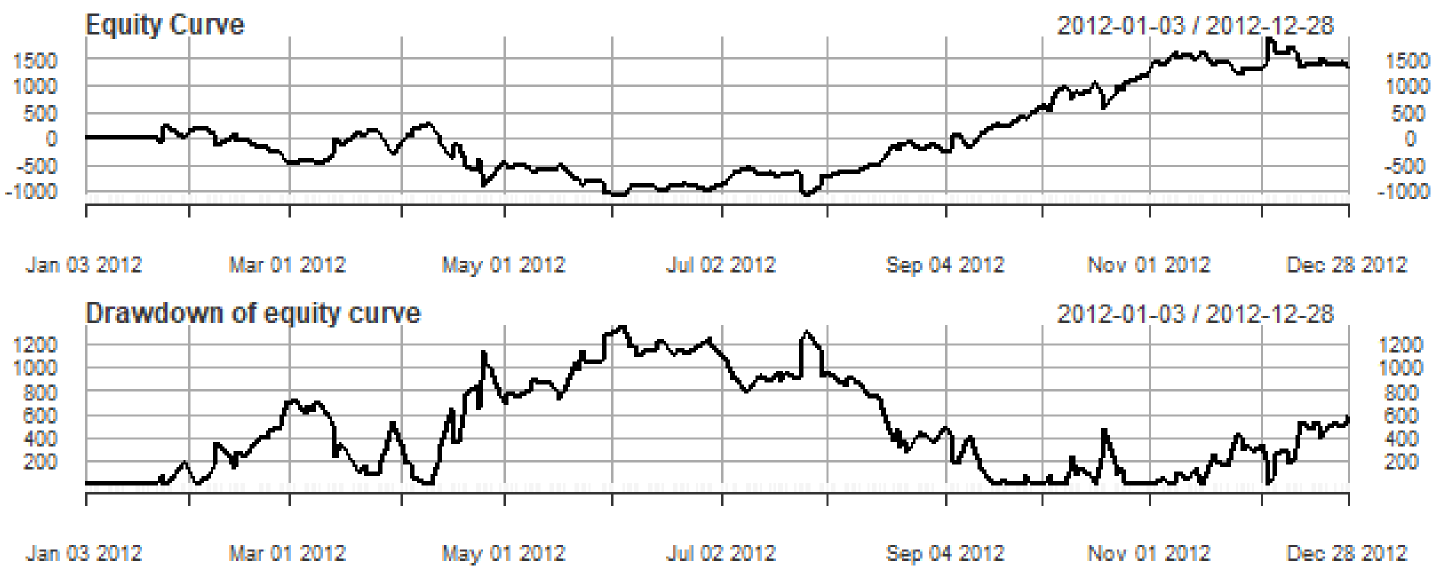
```
##   beta_out_of_sample        x      y  spread signal qty_x qty_y
## 2012-12-27  6.5051194 138.15 499.45 -404.90307 -1     0     0
## 2012-12-28  5.6770827 136.66 494.14 -394.84962 -1     0     0
## 2012-12-31  6.3934172 138.98 516.04 -272.96095 -1 -668   100
```

```
# function for computing the equity curve
compute_equity_curve <- function(qty, price) {

  cash_buy <- ifelse(sign(qty) == 1,
    qty * price, 0)
  cash_sell <- ifelse(sign(qty) == -1,
    -qty * price, 0)
  position <- cumsum(qty)
  cumulative_buy <- cumsum(cash_buy)
  cumulative_sell <- cumsum(cash_sell)

  equity <- cumulative_sell - cumulative_buy +
    position * price
  return(equity)
}

# Add the equity curve columns to the data_out table
data_out$equity_curve_x <- compute_equity_curve(
  data_out$qty_x, data_out$x)
data_out$equity_curve_y <- compute_equity_curve(
  data_out$qty_y, data_out$y)

plot(data_out$equity_curve_x +
  data_out$equity_curve_y, type = 'l',
  main = "AAPL / SPY spread", ylab = "P&L",
  cex.main = 0.8,
  cex.axis = 0.8,
  cex.lab = 0.8)
```



More on the Equity Curve

```r
# Calculates the Sharpe ratio
sharpe_ratio <- function(x, rf) {
  sharpe <- (mean(x, na.rm = TRUE) - rf) /
    sd(x, na.rm = TRUE)
  return(sharpe)
}

# Calculates the maximum drawdown profile
drawdown <- function(x) {
  cummax(x) - x
}

par(mfrow = c(2, 1), mar = c(1, 1, 1, 1))
equity_curve <- data_out$equity_curve_x + data_out$equity_curve_y

plot(equity_curve, main = "Equity Curve",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)

plot(drawdown(equity_curve), main = "Drawdown of equity curve",
 cex.main = 0.8,
 cex.lab = 0.8,
 cex.axis = 0.8)
```



```r
equity <- as.numeric(equity_curve[, 1])
equity_curve_returns <- diff(equity) / equity[-length(equity)]

# Remove any infinities and NaN
invalid_values <- is.infinite(equity_curve_returns) | is.nan(equity_curve_returns)

sharpe_ratio(equity_curve_returns[!invalid_values], 0.03)
```

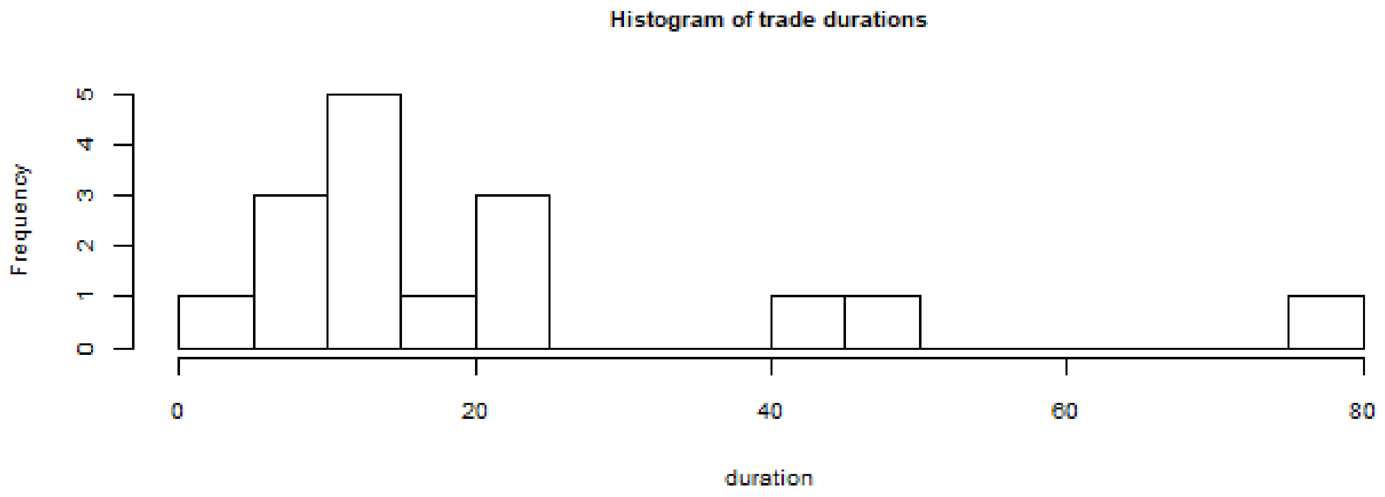
```
[1] 0.0003634364
```

Strategy Attributes

```r
omega_ratio <- function(r, T) {
  omega <- mean(pmax(r - T, 0)) / mean(pmax(T - r, 0))
  return(omega)
}

# Find out where the trades occur
trade_dates <- data_out$qty_x[data_out$qty_x != 0]

# The trade_dates object is an xts object whose index
# contains the necessary time information
duration <- as.numeric(diff(index(trade_dates)))

# Summary statistics
summary(duration)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   3.00   10.75   14.00   21.25   21.50   77.00 
```

```
## Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1.00   13.50   21.00   31.84   44.00  128.00

# Histogram of trade duration
hist(duration, breaks = 20,
  main = "Histogram of trade durations",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
```


Histogram of trade durations
Frequency
duration