Chapter 3

Working with Data

```
# Load the .csv file
aapl_2 <- read.csv(file = file.path(data.dir, "aapl.csv"), header = TRUE, stringsAsFactors = FA

# Reverse the entries
aapl_2 <- aapl_2[rev(rownames(aapl_2)), ]

aapl_close <- aapl_2[, "Close"]

summary(aapl_close)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.94   24.69   38.13   46.80   53.61  199.83
```

```
book <- loadWorkbook(file.path(data.dir, "/strategy.xlsx"))

# Convert it into a data frame
signals <- readWorksheet(book, sheet = "signals", header = TRUE)

signals
```

```
                time signal1 signal2
1 1904-01-01 08:30:00    0.43   -0.20
2 1904-01-01 08:31:00    0.54    0.33
3 1904-01-01 08:32:00    0.32   -0.21
```

```
strength <- readWorksheet(book, sheet = "strength", header = TRUE)

strength
```

```
  intensity score
1         2   7.5
2         3   8.4
3         6   5.4
```

```
# Setup a new spreadsheet
book <- loadWorkbook("demo_sheet.xlsx", create = TRUE)

# Create a sheet called stock1
createSheet(book, name = "stock1")

# Creating a sheet called stock2
createSheet(book, name = "stock2")
```

```
# Load data into workbook
df <- data.frame(a = c(1, 2, 3), b = c(4, 5, 6))
writeWorksheet(book, data=df, sheet="stock1", header = TRUE)

# Save the workbook
saveWorkbook(book, file = file.path(data.dir, "/demo_sheet.xlsx"))
```

```
# First, coerce the data into a data.table
flights_dt <- as_tibble(hflights)

# What type of object is this?
class(flights_dt)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

```
## [1] "tbl_dt"  "tbl"   "data.table"  "data.frame"

# Create a grouping by carrier
carrier_group <- group_by(flights_dt, UniqueCarrier)

# Now compute the summary statistics
summarise(carrier_group, avg_delay = mean(ArrDelay, na.rm = TRUE))
```

```
# A tibble: 15 x 2
   UniqueCarrier avg_delay
   <chr>             <dbl>
 1 AA                0.892
 2 AS                3.19
 3 B6                9.86
 4 CO                6.10
 5 DL                6.08
 6 EV                7.26
 7 F9                7.67
 8 FL                1.85
 9 MQ                7.15
10 OO                8.69
11 UA               10.5
12 US               -0.631
13 WN                7.59
14 XE                8.19
15 YV                4.01
```

```
flights_dt <- as_tibble(hflights)

class(flights_dt)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

```
# Load a small dataset that comes along with xts.
# We could have used our original .csv file as well.
data(sample_matrix)

# Look at the data
head(sample_matrix)
```

```
               Open     High      Low    Close
2007-01-02 50.03978 50.11778 49.95041 50.11778
2007-01-03 50.23050 50.42188 50.23050 50.39767
2007-01-04 50.42096 50.42096 50.26414 50.33236
2007-01-05 50.37347 50.37347 50.22103 50.33459
2007-01-06 50.24433 50.24433 50.11121 50.18112
2007-01-07 50.13211 50.21561 49.99185 49.99185
```

```
## [1] "matrix"

# What is the type of this object?
class(sample_matrix)
```

```
[1] "matrix"
```

```
## [1] "matrix"

# Use the str() command to get more details about this object.
str(sample_matrix)
```

```
 num [1:180, 1:4] 50 50.2 50.4 50.4 50.2 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:180] "2007-01-02" "2007-01-03" "2007-01-04" "2007-01-05" ...
  ..$ : chr [1:4] "Open" "High" "Low" "Close"
```

```
## num [1:180, 1:4] 50 50.2 50.4 50.4 50.2 ...
## - attr(*, "dimnames")=List of 2
## ..$ : chr [1:180] "2007-01-02" "2007-01-03"
## "2007-01-04" "2007-01-05" ...
## ..$ : chr [1:4] "Open" "High" "Low" "Close"

xts_matrix <- as.xts(sample_matrix, descr = 'my new xts object')

str(xts_matrix)
```

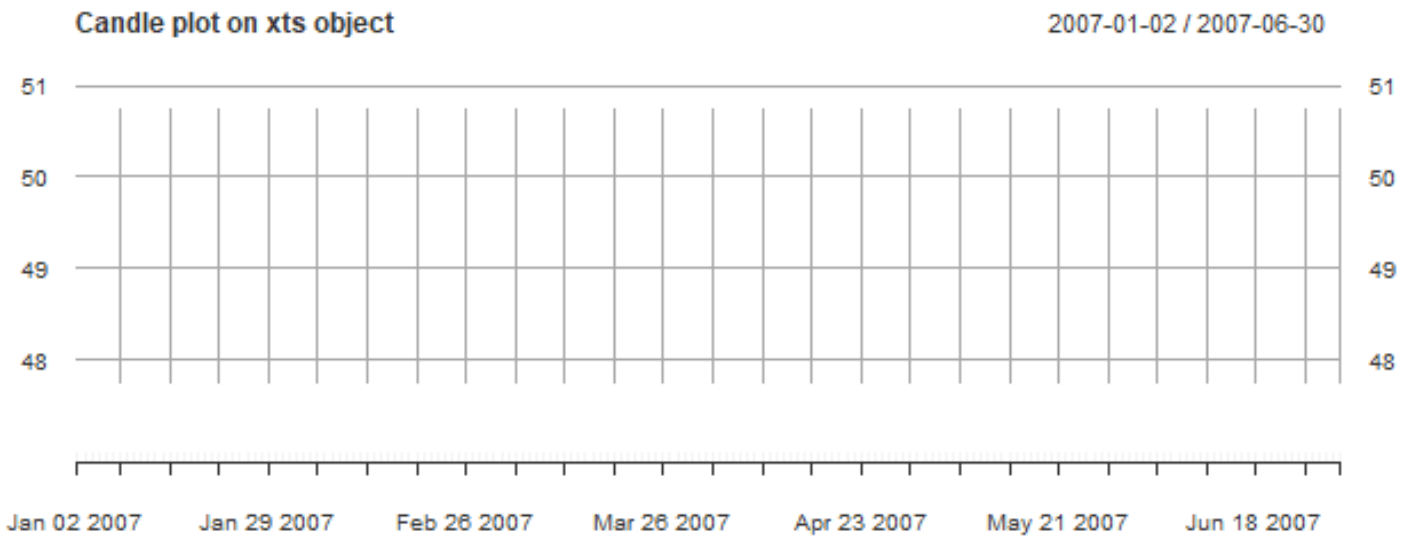
```
An 'xts' object on 2007-01-02/2007-06-30 containing:
  Data: num [1:180, 1:4] 50 50.2 50.4 50.4 50.2 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
```

```
  ..$ : chr [1:4] "Open" "High" "Low" "Close"
  Indexed by objects of class: [POSIXct,POSIXt] TZ:
  xts Attributes:
List of 1
 $ descr: chr "my new xts object"
```

```
# Simple plot
plot(xts_matrix[,1], main = "Our first xts plot",
 cex.main = 0.8)
```

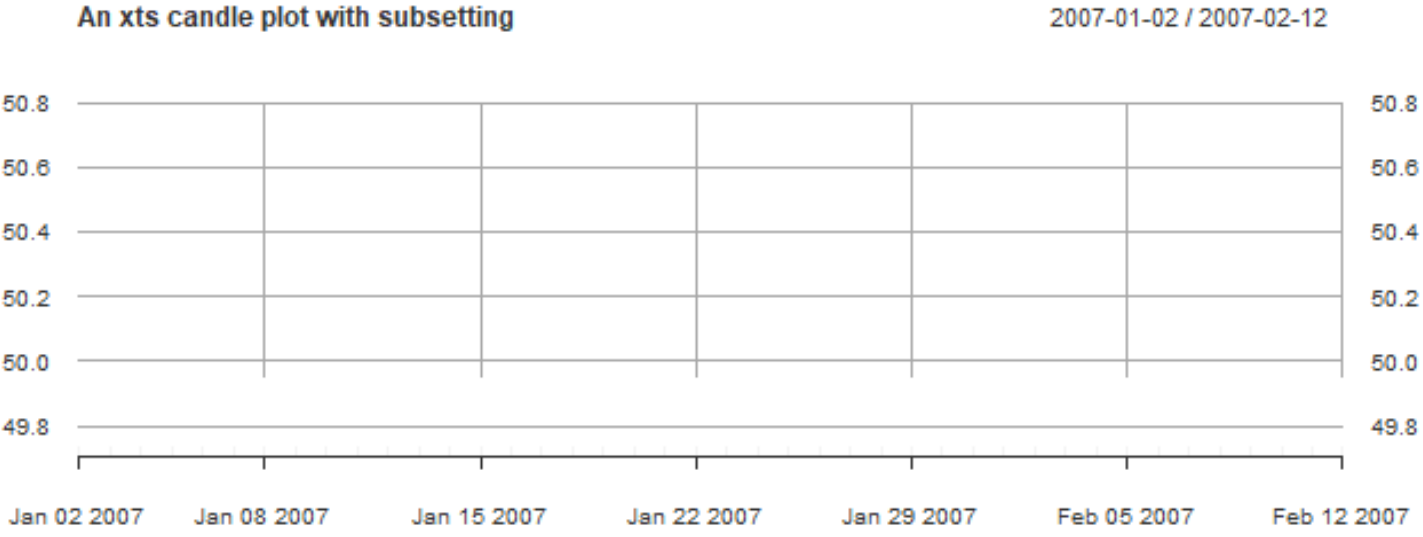

```
# Or we can try something fancier.
plot(xts_matrix, main = "Candle plot on xts object",
 cex.main = 0.8, type = "candles")
```

```
Warning in chart.lines(xdata[xsubset], type = type, lty = lty, lwd = lwd, : candles not recogni
                        'p', 'l', 'b, 'c', 'o', 'h', 's', 'S', 'n'.
                        plot.xts supports the same types as plot.default,
                        see ?plot for valid arguments for type
```

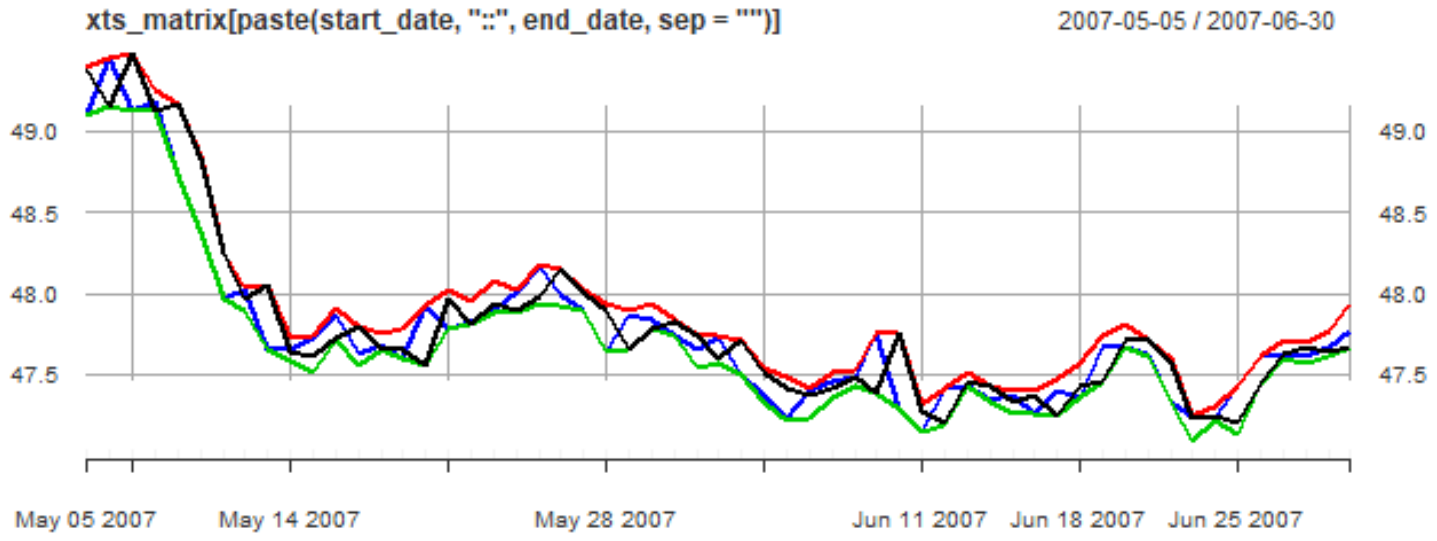



```
plot(xts_matrix["2007-01-01::2007-02-12"],
  main = "An xts candle plot with subsetting",
  cex.main = 0.8, type = "candles")
```

```
Warning in chart.lines(xdata[xsubset], type = type, lty = lty, lwd = lwd, : candles not recogni
                       'p', 'l', 'b, 'c', 'o', 'h', 's', 'S', 'n'.
                       plot.xts supports the same types as plot.default,
                       see ?plot for valid arguments for type
```



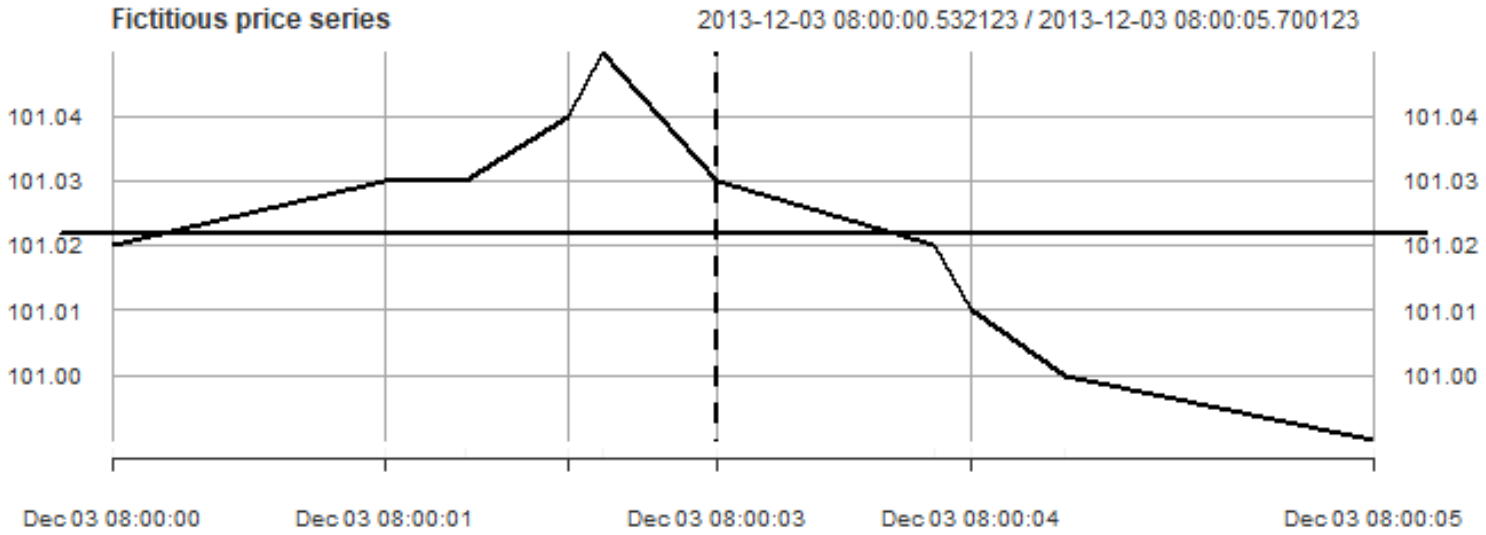

```
start_date <- "2007-05-05"
end_date <- "2007-12-31"

plot(xts_matrix[paste(start_date, "::",
  end_date, sep = "")])
```




```
# Create a vector of 10 fictitious stock prices along with
# a time index in microsecond resolution.
price_vector <- c(101.02, 101.03, 101.03, 101.04, 101.05,
  101.03, 101.02, 101.01, 101.00, 100.99)

dates <- c("03/12/2013 08:00:00.532123",
  "03/12/2013 08:00:01.982333",
  "03/12/2013 08:00:01.650321",
  "03/12/2013 08:00:02.402321",
  "03/12/2013 08:00:02.540432",
  "03/12/2013 08:00:03.004554",
  "03/12/2013 08:00:03.900213",
  "03/12/2013 08:00:04.050323",
  "03/12/2013 08:00:04.430345",
  "03/12/2013 08:00:05.700123")
```

```
# Allow the R console to display the microsecond field
options(digits.secs = 6)

# Create the time index with the correct format
time_index <- strptime(dates, format = "%d/%m/%Y %H:%M:%OS")

# Pass the time index into the its object
xts_price_vector <- xts(price_vector, time_index)

# Plot the price of the fictitious stock
plot(xts_price_vector, main = "Fictitious price series",
  cex.main = 0.8)
```

```
# Add a horizontal line where the mean value is
abline(h = mean(xts_price_vector), lwd = 2)

# Add a vertical blue line at a specified time stamp
my_time <- as.POSIXct("03/12/2013 08:00:03.004554",
  format = "%d/%m/%Y %H:%M:%OS")

abline(v = my_time, lwd = 2, lty = 2)
```




```
es_price <- c(1700.00, 1700.25, 1700.50, 1700.00, 1700.75,
    1701.25, 1701.25, 1701.25, 1700.75, 1700.50)

es_time  <- c("09/12/2013 08:00:00.532123",
  "09/12/2013 08:00:01.982333",
  "09/12/2013 08:00:05.650321",
  "09/12/2013 08:10:02.402321",
  "09/12/2013 08:12:02.540432",
  "09/12/2013 08:12:03.004554",
  "09/12/2013 08:14:03.900213",
  "09/12/2013 08:15:07.090323",
  "09/12/2013 08:16:04.430345",
  "09/12/2013 08:18:05.700123")

# create an xts time series object
xts_es <- xts(es_price, as.POSIXct(es_time,
  format = "%d/%m/%Y %H:%M:%OS"))

names(xts_es) <- c("price")
```

```
time_diff <- difftime(index(xts_es)[2], index(xts_es)[1],
  units = "secs")
```

```
time_diff
```

```
Time difference of 1.45021 secs
```

```
## Time difference of 1.45021 secs

diffs <- c()
for(i in 2:length(index(xts_es))) {
  diffs[i] <- difftime(index(xts_es)[i], index(xts_es)[i - 1],
    units = "secs")
}
```

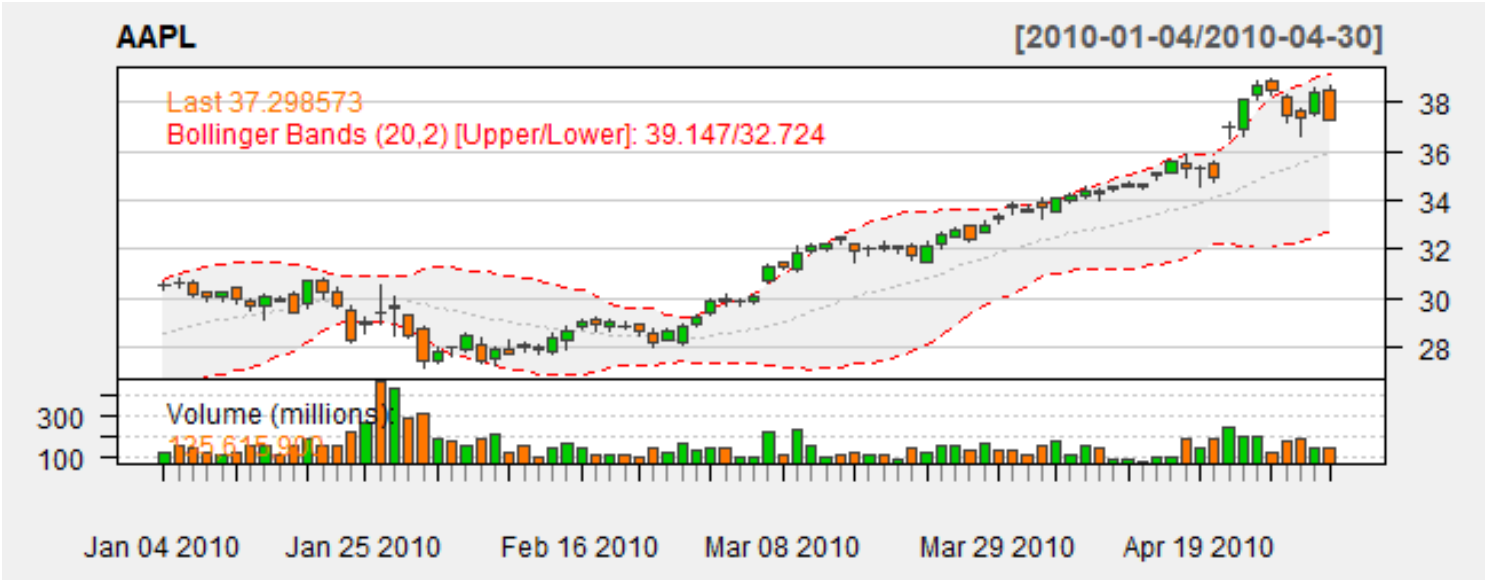
```
############################
# Charting with quantmod #
############################

AAPL <- getSymbols("AAPL", auto.assign=FALSE)
```

```
'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.
```

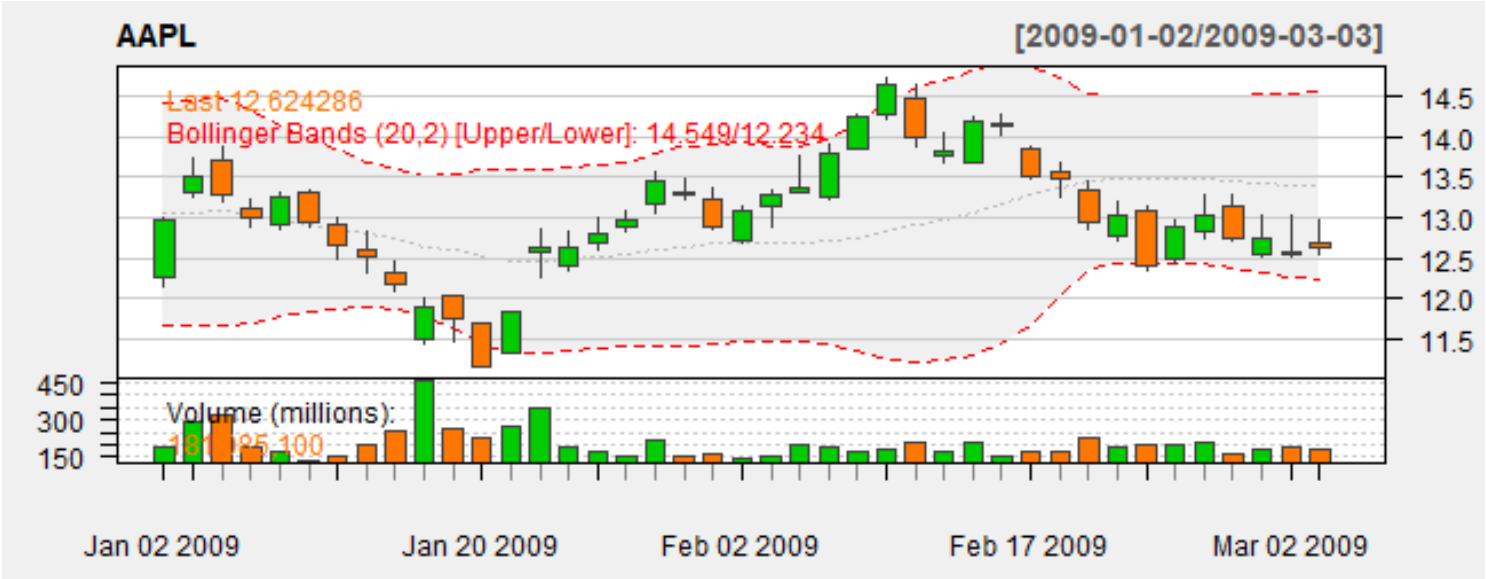
```
head(AAPL)
```

```
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2007-01-03  12.32714  12.36857 11.70000   11.97143   309579900      10.39169
2007-01-04  12.00714  12.27857 11.97429   12.23714   211815100      10.62234
2007-01-05  12.25286  12.31428 12.05714   12.15000   208685400      10.54669
2007-01-08  12.28000  12.36143 12.18286   12.21000   199276700      10.59878
2007-01-09  12.35000  13.28286 12.16429   13.22429   837324600      11.47922
2007-01-10  13.53571  13.97143 13.35000   13.85714   738220000      12.02857
```

```
# Adding some technical indicators on top of the original plot
chartSeries(AAPL, subset='2010::2010-04',
  theme = chartTheme('white'),
  TA = "addVo(); addBBands()")
```

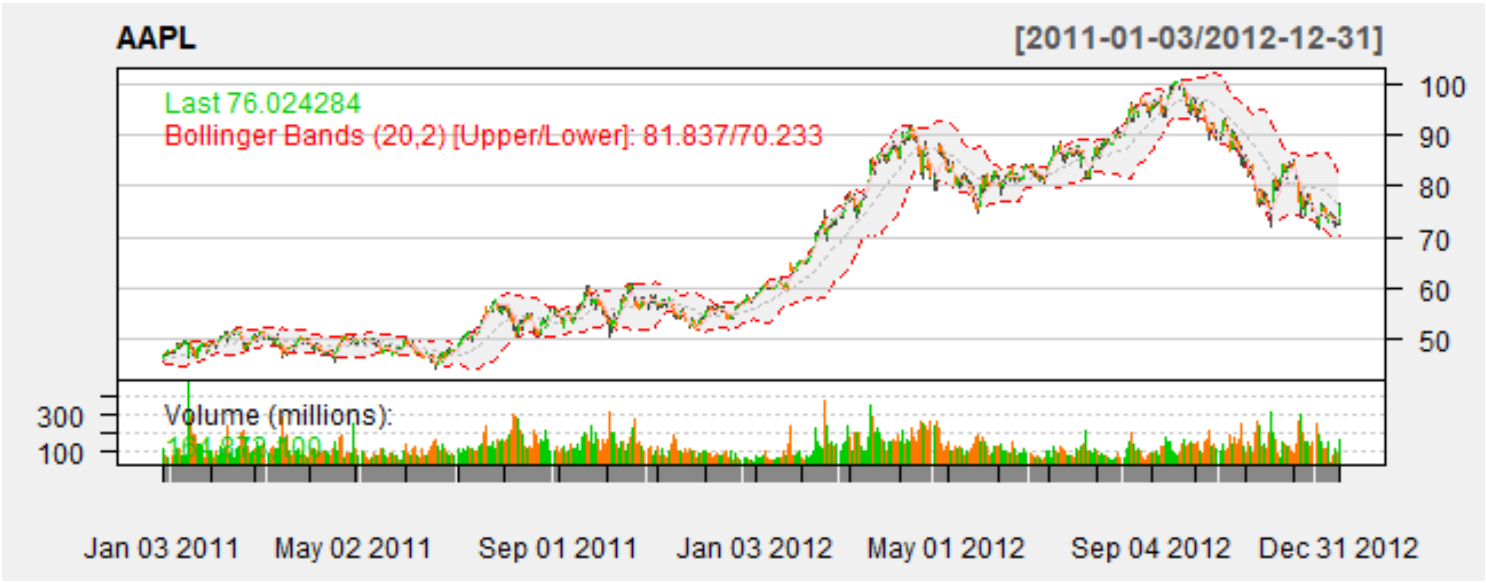



```
reChart(subset='2009-01-01::2009-03-03')
```

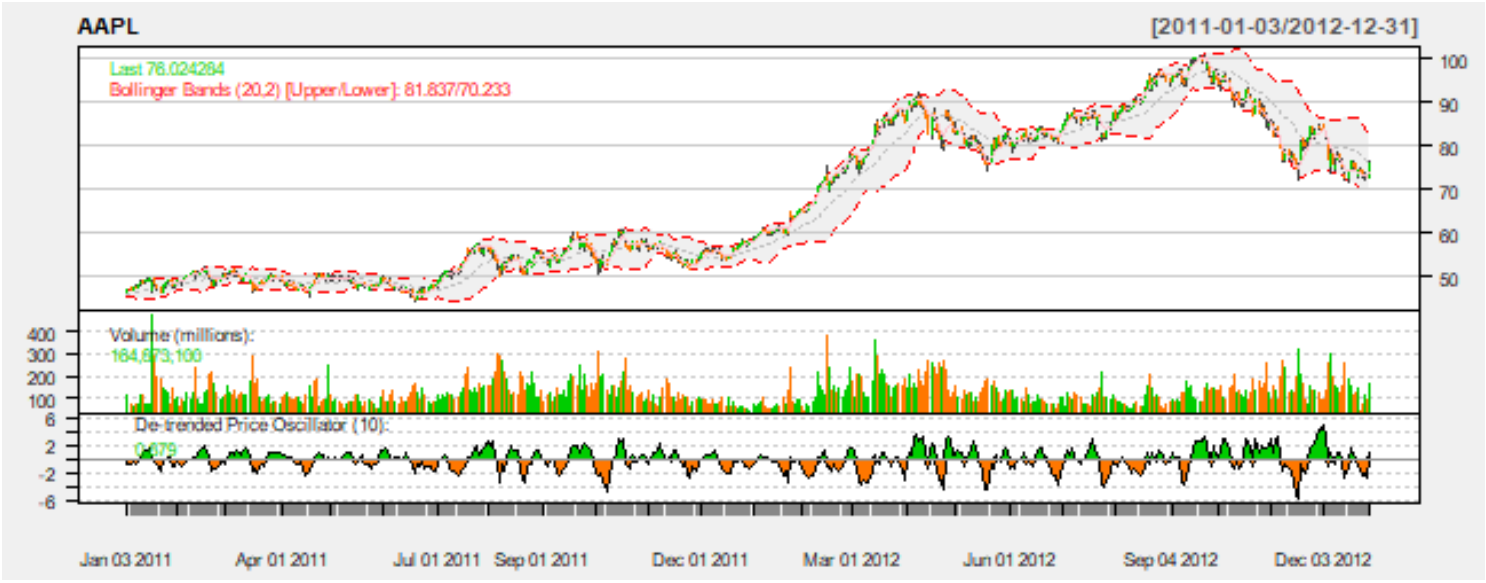



```
chartSeries(AAPL, subset='2011::2012',
  theme = chartTheme('white'),
  TA = "addBBands(); addDEMA()")
```

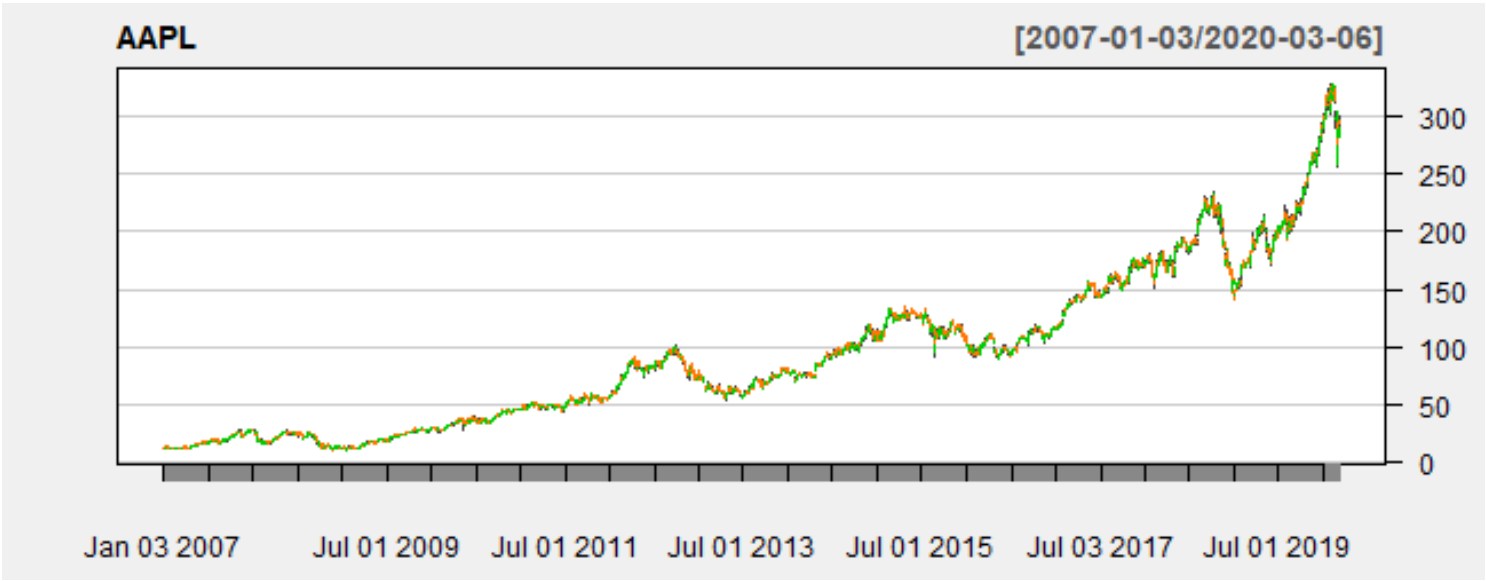

```
addVo()
```



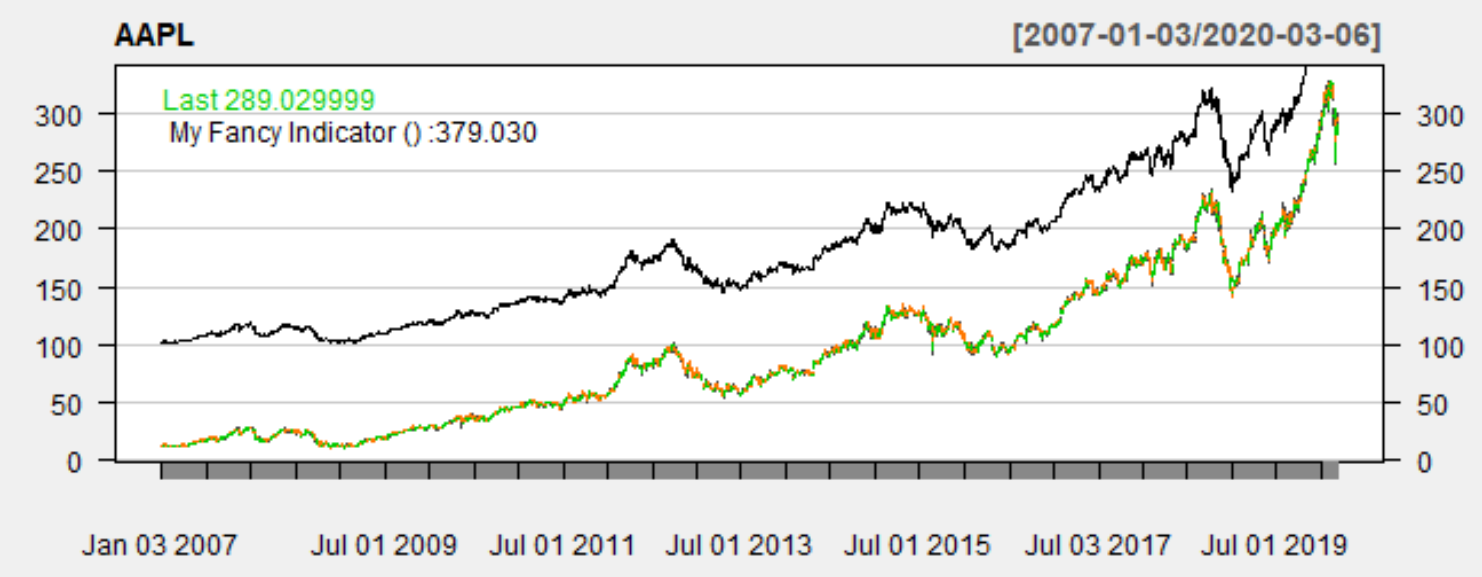

```
addDPO()
```




```
# Initial chart plot with no indicators
chartSeries(AAPL, theme = chartTheme('white'), TA = NULL)
```




```
# Custom function creation
my_indicator <- function(x) {
   return(x + 90)
}

add_my_indicator <- newTA(FUN = my_indicator, preFUN=Cl,
  legend.name = "My Fancy Indicator", on = 1)

add_my_indicator()
```




```
##########################
# Graphing wiht ggplot2 #
##########################

# Create a matrix with price and volume
df <- AAPL[, c("AAPL.Adjusted", "AAPL.Volume")]
names(df) <- c("price", "volume")

# Create
df$return <- diff(log(df[, 1]))
df <- df[-1, ]

df$cuts <- cut(abs(df$return),
  breaks = c(0, 0.02, 0.04, 0.25),
  include.lowest = TRUE)

# Create another column for the mean
df$means <- NA
for(i in 1:3) {
  group <- which(df$cuts == i)
  if(length(group) > 0) {
    df$means[group] <- mean(df$volume[group])
  }
}

ggplot(df) +
   geom_histogram(aes(x=volume)) +
   facet_grid(cuts ~ .) +
   geom_vline(aes(xintercept=means), linetype="dashed", size=1)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```