Modeling Financial Time Series Data in R

Peter Grantcharov

11/4/2018

OVERVIEW

This tutorial will walk you through making visualizations of financial time series data. It will cover basic graphs, interactive graphs, and add-ons that may be pertinent for both types when modelling stock price data.

First, we have to import the necessary packages and data.

```
library(tidyquant)
library(lubridate)
aapl <- tq_get("AAPL")
```

Basic stock information can be accessed from the *tidyquant* package, while date manipulations can be performed with the package *lubridate*. In this demonstration, I will showcase how these graphs were created for Apple (symbol: AAPL), given the high degree of activity of its stock over the years. Apple's stock data can easily be obtained by using the *tidyquant* package function called *tq_get()*. We simply have to provide this function with the stock ticker/symbol, written as a simple string, as the only parameter. The information is imported as a tibble. The head of the tibble for Apple is shown below:

```
head(aapl)
```

```
## # A tibble: 6 x 7
##   date        open  high   low close    volume adjusted
##   <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 2009-01-02  12.3  13.0  12.2  13.0 186503800     8.68
## 2 2009-01-05  13.3  13.7  13.2  13.5 295402100     9.05
## 3 2009-01-06  13.7  13.9  13.2  13.3 322327600     8.90
## 4 2009-01-07  13.1  13.2  12.9  13.0 188262200     8.71
## 5 2009-01-08  12.9  13.3  12.9  13.2 168375200     8.87
## 6 2009-01-09  13.3  13.3  12.9  12.9 136711400     8.67
```

BASIC LINE GRAPH

If we want to construct a simple line graph of the stock's closing prices over the time period contained in the imported tibble, we can make a standard line graph of the closing stock prices using *ggplot*, accessed through the *tidyverse* collection of packages. Since the *date* column already comes with values of the "Date" object, no further transformations are necessary.

```
library(tidyverse)
ggplot(aapl, aes(x = date, y = close)) +
    geom_line(col = "darkblue") +
    ggtitle("Closing Prices of AAPL since January 2, 2008") +
    theme_bw() +
    labs(x = "Date", y = "Closing Prices ($)")
```

This is a standard graph that'll give a good overview of the stock's movement over full time period.

PRICE AND VOLUME COMBINED

To make the graph more exciting and informative, however, I will make three feature additions:

1. Specify a time range
2. Add volume bars to the same graph (pardon the double y-axis!)
3. Manipulate the volume bars to be the the net volume over an arbitrary time period. In this case, it is the net volume for each calendar month.

Change 1) can be performed inside the ggplot definition by using the *scale_x_date* function. We need to ensure that our limits are of the *Date* class, so I have set a lower limit of Jan 1, 2010, and an upper limit of the most recent data that was given to us by the tidyquant import.

Change 2) requires the addition of another ggplot function. By typical finance standards, we will model the trading volume using bars with *geom_bar*. Here, it is critical to properly manipulate the scales such that both the closing price and the volume data can be plotted on the same graph. This requires careful attention, as in this case, volume supersedes price by a factor of several hundred million. To do this:

- Pick a denominator to divide our volume with so it appears on the graph at a height that suits us. In this case, I have picked 150,000,000 so that it doesn't cross the closing price line graph.
- Add the secondary y-axis, which also has to be scaled accordingly. This is done with the *sec.axis* argument, given to *scale_y_continuous*. To properly scale the it such that it accurately depicts the volume scale, the fraction that can be seen in this argument should be: volume divisor divided by desired volume units. In this case, our divisor was 150,000,000, and we wish to show the volume in billions (1 billion = 1000 million). Some zeros have been removed to keep it clean while maintaining the same fraction.

Lastly, for such a large time scale, the daily volume would clearly provide more information than one would be able to digest (about 4000 bars for this dataset!). As such, it is desirable to take the total volume, for example, over a monthly period. This requires some manipulations, which can be seen in the upper region of the coding block. We have to group our data by month, take the volume sum over this period, and store these volume sums over the correct date periods in a new data frame (*monthvol*). As long as the *date* column for monthvol contains dates within the range given by the date column in *aapl*, it will plot it on the same x-axis without issues.

Putting all of this togher, we get the following code and graph:

```
monthvol <- data_frame(aapl$date, aapl$volume)
names(monthvol) <- c("date", "volume")
monthvol$date <- as.Date(monthvol$date)
monthvol$date <- format(monthvol$date, "%Y-%m")
monthvol <- monthvol %>% group_by(date) %>%
  summarize(totalvol = sum(volume))
monthvol$date <- as.Date(as.yearmon(monthvol$date))

ggplot() +
    geom_line(data = aapl, aes(x = date, y = close, col = "Price")) +
    geom_bar(data = monthvol, aes(x = date, y = totalvol/150000000, fill = "Volume"),
             stat = 'identity', alpha = 0.8, width = 20) +
    ggtitle("Closing Prices of AAPL since January 1, 2010") +
    scale_y_continuous(breaks = c(seq(0, 260, by = 20)),
                       sec.axis = sec_axis(~.*(150/1000),
                                           name = "Volume (Billion Shares Traded)",
                                           breaks = c(seq(0, 36, by = 3)))) +
    scale_x_date(limits = c(as.Date("2010-01-01"), max(aapl$date))) +
    scale_fill_manual("", values = "darkred") +
    scale_color_manual(" ", values = c('Price' = 'darkblue',
                                       'Volume' = 'darkred')) +
    theme_bw() +
    labs(x = "Date", y = "Closing Prices ($)", color = "Legend:")
```

Volume data is very insightful, and has a close relationship to the stock price, as this gives you insight into the which prices the majority of people are holding the stock at. As such, this is a case where it is rather beneficial having the two data sources graphed in very distinct ways on the same graph.

MULTIPLE STOCKS ON INTERACTIVE GRAPH

To spice it up even further, we will make the following adjustments for the next graph:

1. Add a second stock to our observations on the same graph: Expedia (symbol: EXPE)
2. Make our line graph interactive

Feature 1) will simply require importing data for Expedia in the same way as how it was done for Apple. We will combine data form both stocks into a single data frame, *combo*, that has a date column, and seperate columns for the closing stock prices, Apple and Expedia.

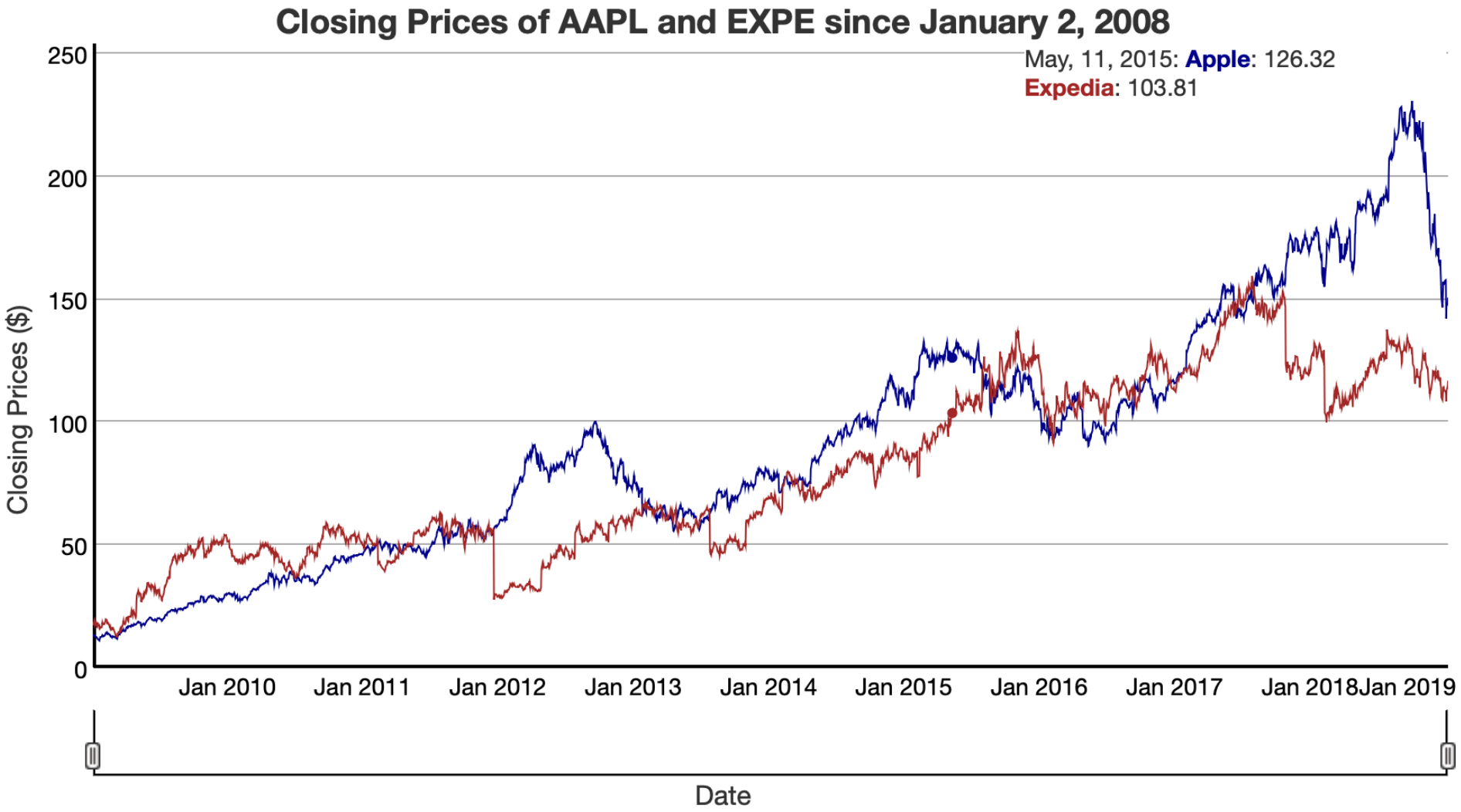
To make an interactive graph, we will take advantage of the *dygraphs* package. Since the required time series type for this package to plot our data has to be xts (extensible time series), we will also import the *xts* package. This just requires that we process the combo data frame using the *xts* function, with the necessary argument *order.by* set equal to the date column in our data frame.

Then we plot our graph using *dygraphs* with some self-explanatory arguments and functions passed and piped to it. The code and output is shown below:

```
library(dygraphs)
library(xts)
expe <- tq_get("EXPE")

combo <- data_frame(aapl$date, aapl$close, expe$close)
names(combo) <- c("Date", "Apple", "Expedia")
combo$Date <- as.Date(combo$Date)
combo <- xts(combo, order.by = combo$Date)

dygraph(combo, main = "Closing Prices of AAPL and EXPE since January 2, 2008",
        xlab = "Date", ylab = "Closing Prices ($)") %>%
  dyLegend(hideOnMouseOut = FALSE, ) %>%
  dyAxis(name = "x", axisLineWidth = 3,  drawGrid = FALSE) %>%
  dyAxis(name = "y", axisLineWidth = 3) %>%
  dySeries("Apple", color = "darkblue") %>%
  dySeries("Expedia", color = "brown") %>%
  dyRangeSelector()
```



As we can see, we can hover our cursor over the graph and see the exact prices and dates for both stocks written in the legend in the top right of the graph. This interactive graph is therefore able to cleanly plot an enormous amount of data (all closing prices of multiple stocks for the last 8 years) on a single graph, which is very beneficial to get a complete picture of a stocks history.

Another interesting feature is the time range selector (specified by *dyRangeSelector()* function) at the bottom of the graph. The entire selector represent the full time scale of data that is fed into the *dygraph*, and can be changed to focus in on the stock closing price to observe its movement in a very specific time period.

TRANSFORMING PRICE TO % CHANGE

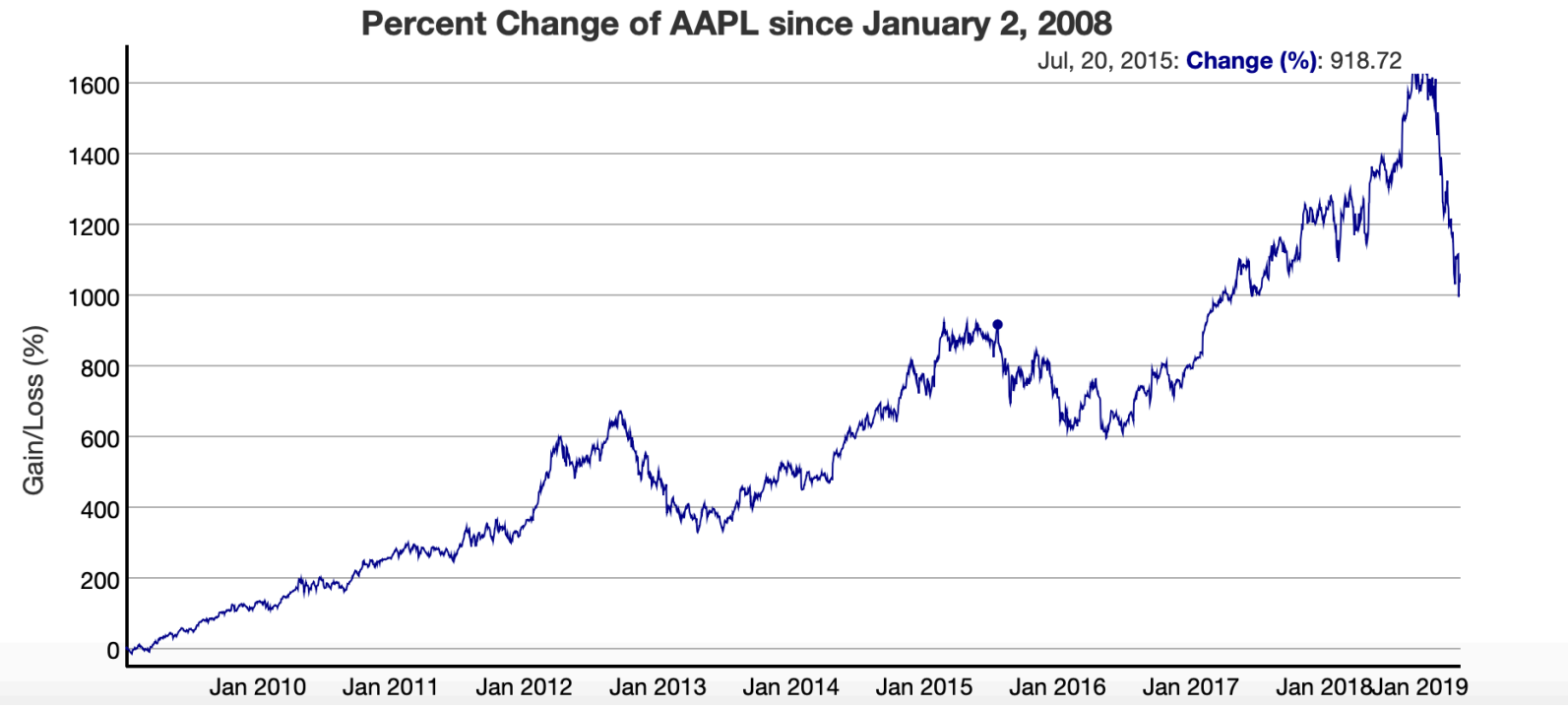
For the next graph, we will transform the stock price so that it represents the the percentage gained/lost since day 1. Since we are back to using the full dataset, this will be since Jan 2, 2008, but can easily be altered by making the transformed data frame, in this case *xts_df*, start at a different date.

It should also be noted that when we only plot a single stock, the *dyRangeSelector* will have its own visualization that represents the relative value of the stock that is being graphed.

```
xts_df <- data_frame(aapl$date, aapl$close)
names(xts_df) <- c("date", "Change (%)")
xts_df$date <- as.Date(xts_df$date)
xts_df$`Change (%)` = (xts_df$`Change (%)`/xts_df$`Change (%)`[1] - 1)*100
minimum = min(xts_df$`Change (%)`)
maximum = max(xts_df$`Change (%)`)

xts_df <- xts(xts_df, order.by = xts_df$date)

dygraph(xts_df, main = "Percent Change of AAPL since January 2, 2008",
        xlab = "Date", ylab = "Gain/Loss (%)") %>%
  dyOptions(colors = "darkblue") %>%
  dyLegend(hideOnMouseOut = FALSE) %>%
  dyAxis(name = "x", axisLineWidth = 3, drawGrid = FALSE) %>%
  dyAxis(name = "y", axisLineWidth = 3,
         valueRange = c(minimum - 40, maximum+ 20)) %>%
  dyRangeSelector()
```



CHANGING COLORS TO MODEL STOCK PRICE PERFORMANCE

Since this is a visualization tutorial regarding financial stock data, it would be appropriate to have a feature that distinguishes between when we're in the green and when we're in the black. To do this, we'll have seperate columns (mutually exclusive) for the price when it is in these two distinct regions. When a stock is not in the region of interest for a given column, the price will be assigned to be *NA*. The first few lines creating this data frame requires a bit of gymnastics as it is originally populated with two identical columns. The "break-even" value is then given by the first cell, and the gain and loss columns are populated accordingly, by comparing their values to this "break-even" value. The colors are changed to green and red to appropriately allow viewers to distinguish between the sections, by using *dyOptions(colors = c("red", "green"))*. The loop is included to ensure that there is continuity between the points as the graph switches for green to red.

```
xts_df <- data_frame(aapl$date, aapl$close)
names(xts_df) <- c("date", "Gain")
xts_df$Loss <- aapl$close
xts_df$date <- as.Date(xts_df$date)
xts_df$Gain[xts_df$Gain < xts_df$Gain[1]] <- NA
xts_df$Loss[xts_df$Loss >= xts_df$Gain[1]] <- NA

for (i in 2:(nrow(xts_df))) {
  if (is.na(xts_df$Gain[i]) && is.na(xts_df$Loss[i-1])) {
    xts_df$Gain[i] <- xts_df$Loss[i]
  }
  else if (is.na(xts_df$Loss[i]) && is.na(xts_df$Gain[i-1])) {
    xts_df$Loss[i] <- xts_df$Gain[i]
  }
}

xts_df <- xts(xts_df, order.by = xts_df$date)

dygraph(xts_df, main = "Closing Prices of AAPL since January 2, 2008", xlab = "Date", ylab = "Stock Price ($)") %
>%
  dySeries("Loss", fillGraph = TRUE, color = "red") %>%
  dySeries("Gain", fillGraph = TRUE, color = "green") %>%
  dyLegend(hideOnMouseOut = FALSE) %>%
  dyAxis(name = "x", axisLineWidth = 3,  drawGrid = FALSE) %>%
  dyAxis(name = "y", axisLineWidth = 3, valueRange = c(min(xts_df$Loss), max(xts_df$gain))) %>%
  dyRangeSelector()
```


Closing Prices of AAPL since January 2, 2008
— date — Loss — Gain
250
200
150
100
50
0
Stock Price ($)
Jan 2010
Jan 2011
Jan 2012
Jan 2013
Jan 2014
Jan 2015
Jan 2016
Jan 2017
Jan 2018
Jan 2019
Date

CANDLESTICK!

And for the grand finale, we will construct a candlestick graph, which is one of the most popular ways to visualize stock prices. First, we will need to find a time period to compress the data. Since I would like to visualize the full 10 year period for Apple's stock, I have arbitrarily selected this time period to be months. To do this, I have formatted the *Date* column to remove the day. This will lump all days in a month into a single group.

The *dyCandlestick()* graph requires four columns to construct this graph: the open price, the high price, the low price, and the close price. To accomodate, I will group by date, and calculate these values for each given group, which in this case is by month. This five column data frame (date, open, high, low, close) is stored in *xts_df_candle*, and the date column is then converted back to a proper date object before being given to the *xts()* function to convert the entire dataframe into a format acceptable by the *dygraph* function.

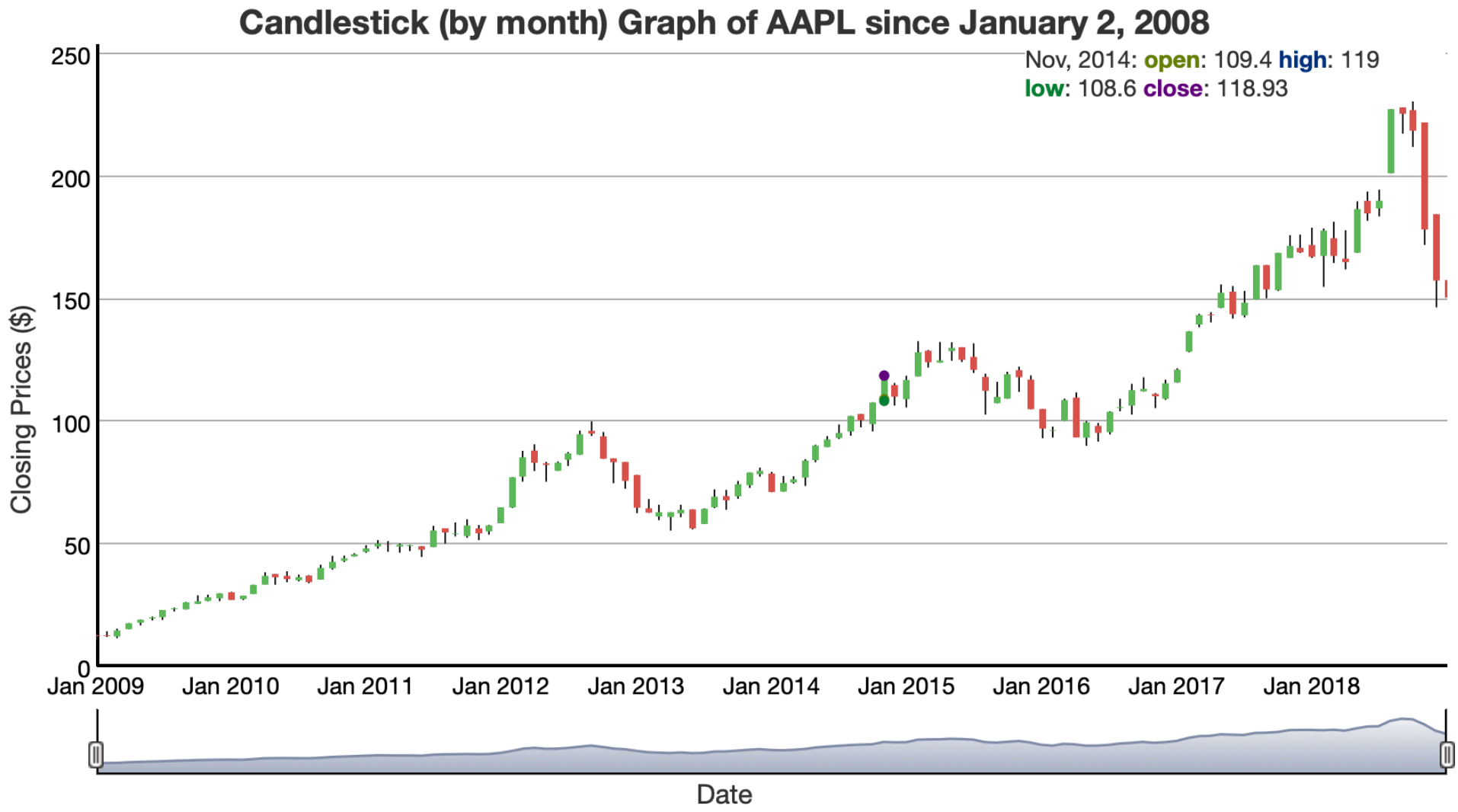
With these specifications, we get the following beauty:

```
# Create data (needs 4 data points per date stamp)
xts_df <- data_frame(aapl$date, aapl$close)
names(xts_df) <- c("date", "close")
xts_df$date <- as.Date(xts_df$date)
xts_df$date <- format(xts_df$date, "%Y-%m")

xts_df_candle <- xts_df %>% group_by(date) %>%
  summarize(open = close[1], high = max(close), low = min(close), close = close[length(close)])

xts_df_candle$date <- as.yearmon(xts_df_candle$date)
xts_df_candle = xts(xts_df_candle[,-1], order.by = xts_df_candle$date)

dygraph(xts_df_candle, main = "Candlestick (by month) Graph of AAPL since January 2, 2008",
        xlab = "Date", ylab = "Closing Prices ($)") %>%
  dyCandlestick() %>%
  dyAxis(name = "x", axisLineWidth = 3,  drawGrid = FALSE) %>%
  dyAxis(name = "y", axisLineWidth = 3) %>%
  dyRangeSelector()
```



Thank you for reading!

THE END