post01-jerry-chiang

Jerry Chiang

10/29/2017

Stock Market Data Analysis with R: Visualizing Stocks

Verizon vs. AT&T

By Jerry Chiang

Today, I will be applying the data manipulation and visualization techniques learned so far to demonstrate how stocks can begin to be analyzed.

The post will feature techniques such as data manipulation and visualization through packages such as `dplyr` and `ggplot2` that we have learned in Stat133, in addition to new stock-related packages from CRAN.

Since we are dealing with data such as stock prices, which are *highly dependent* on time, our `input` will be a **time series data.** From this data, we will create meaningful charts plotting prices, returns, and even moving averages, `outputs` that are all useful for traders.

This post was motivated by my personal interest in following the stock markets, and hopefully can serve as a reference for other aspiring traders, or the very least, demonstrate one practical use of R (in analyzing something that's not NBA-related for once, no shade). **We will be comparing the stocks of Verizon and AT&T in this example.**

Retrieving data with `quantmod` package

Before analyzing stock data, we will need to retrieve the data in a useful format from sources such as Yahoo!Finance. As I mentioned before, this post will feature new R packages: one of them, `quantmod`, will allow us to easily load stock data from Yahoo!Finance as a xts type, which you can read all about here.

So first we install and load the `quantmod` package, in addition to `dplyr` and `ggplot2`:

```
library(dplyr)
library(ggplot2)

#Get quantmod if you don't already have it
if (!require("quantmod")) {
    install.packages("quantmod")
    library(quantmod)
}
```

We will be working with stock data from Verizon first. To use the following function `getSymbols`, which is part of the package `quantmod`, we supply the stock abbreviation ('VZ'), data source ('yahoo'), and the starting and ending dates (year to date):

```
start = as.Date("2017-01-01")
end = as.Date("2017-10-29")
getSymbols("VZ", src = "yahoo", from = start, to = end)
```

```
## [1] "VZ"
```

```
class(VZ)
```

```
## [1] "xts" "zoo"
```

Notice that the output of this function is an xts object `VZ`. We will actually convert it to a data frame for easier manipulation; this package just allows us to easily pull data online. Xts is similar to a data frame but used for time series data, where each row corresponds to a time stamp, like so:

```
#Preview of VZ
head(VZ)
```

```
##            VZ.Open VZ.High VZ.Low VZ.Close VZ.Volume VZ.Adjusted
## 2017-01-03   53.96   54.67  53.73    54.58  22891700    52.05062
## 2017-01-04   54.55   54.81  54.33    54.52  18316100    51.99340
## 2017-01-05   54.78   54.83  54.30    54.64  13840600    52.10784
## 2017-01-06   53.67   53.78  53.10    53.26  15615000    51.33482
## 2017-01-09   53.22   53.27  52.67    52.68  14095000    50.77579
## 2017-01-10   52.69   53.16  52.02    52.76  11787400    50.85290
```

For an explanation of what the column titles mean, refer here for stock jargon.

Visualizing Verizon

Once we have gotten the xts object `VZ`, we can convert it to a data frame `dt_VZ` before plotting it using the `ggplot` function.

```r
#Convert xts to dataframe for VZ
dt_VZ = data.frame(date = index(VZ), coredata(VZ))
#Notice you access the time stamp through index() instead of row.names, and other columns with coredata()

#Preview of VZ as a data frame
head(dt_VZ)
```

```
##         date VZ.Open VZ.High VZ.Low VZ.Close VZ.Volume VZ.Adjusted
## 1 2017-01-03   53.96   54.67  53.73    54.58  22891700    52.05062
## 2 2017-01-04   54.55   54.81  54.33    54.52  18316100    51.99340
## 3 2017-01-05   54.78   54.83  54.30    54.64  13840600    52.10784
## 4 2017-01-06   53.67   53.78  53.10    53.26  15615000    51.33482
## 5 2017-01-09   53.22   53.27  52.67    52.68  14095000    50.77579
## 6 2017-01-10   52.69   53.16  52.02    52.76  11787400    50.85290
```

```r
#Graph closing VZ stock prices
ggplot(dt_VZ) + geom_line(aes(x = date, y = VZ.Close)) + labs(title = "VZ Closing Prices", x = "Date", y = "Price"
)
```

Besides ggplot, the `quantmod` package also features a function called `candleChart` that allows us to plot a candlestick chart, which is commonly used by traders for analyses. It essentially combines the various columns of our data frame into one graph.

```r
#Notice that it takes in our original xts object VZ as an input
candleChart(VZ)
```

The `candleChart` function also allows us to show moving averages. In stock analyses, **k-day moving averages** are essentially the averages of the past *k* days. As you can see from the graph below, moving averages help traders "smooth" out a series and better identify trends underlying price movements of a stock:

```
#Add moving average line with the addSMA() function as part of this package

#For 20-day moving average
addSMA(n = 20)
```

Notice that there is a general downwards movement (in stock jargon: “bearish” trend) up until August before an upwards movement (or “bullish”).

```
#For 50-day moving average
addSMA(n = 50)
```

When we apply a 50-day moving average, we see the line is more flat with less fluctuations, but with a similar trend.

```
#For 100-day and 200-day moving averages
addSMA(n = c(100, 200))
```

The same is true for our 100-day moving average (in blue), while we don't have enough days to get a clear trend from the 200-day moving average (in cyan). But depending on your position as a trader (how long you decide to hold onto `VZ` stock), you can see how various k-day moving averages can be helpful in understanding trends of past price movements.

Also notice that similar to how `ggplot` works, we can "overlay" these lines on top of the original candlestick chart!

Comparing stock returns between Verizon vs AT&T

Now that we understand the `quantmod` package and how we can use it to plot graphs, let's begin to compare stocks for Verizon (`VZ`) and AT&T (`T`). For useful comparison between stocks, we have to look at a *relative* metric such as **stock returns** because as you can imagine, using an absolute metric (i.e. plotting the prices of Verizon and AT&T) isn't all that meaningful.

To begin, we must first create a combined data frame of closing prices for `VZ` and `T` . We can do so by creating and appending the data frame for `T` to `dt_VZ` , which we previously created.

```
#Pulling data for T
getSymbols("T", src = "yahoo", from = start, to = end)
```

```
## [1] "T"
```

```
#Creating T data frame
dt_T = data.frame(date = index(T), coredata(T))

#Combining data frames and keeping only date and closing prices
dt_combo = left_join(dt_VZ, dt_T, by = "date")
dt_combo = select(dt_combo, c("date", "VZ.Close", "T.Close"))

head(dt_combo)
```

```
##         date VZ.Close T.Close
## 1 2017-01-03    54.58   43.02
## 2 2017-01-04    54.52   42.77
## 3 2017-01-05    54.64   42.65
## 4 2017-01-06    53.26   41.32
## 5 2017-01-09    52.68   40.80
## 6 2017-01-10    52.76   40.81
```

To calculate returns between the prior period and the next, we will first create a data frame that has all the closing prices moved up 1 year from the original so we can divide it by the original prices to get annual returns.

```
#Select VZ closing prices
VZ_close = dt_combo[2]

#Shift values up a year (shift rows down 1)
VZ_close1 = VZ_close %>% mutate_each(funs = funs(lag))

#Before and after:
head(VZ_close)
```

```
##   VZ.Close
## 1    54.58
## 2    54.52
## 3    54.64
## 4    53.26
## 5    52.68
## 6    52.76
```

```
head(VZ_close1)
```

```
##   VZ.Close
## 1       NA
## 2    54.58
## 3    54.52
## 4    54.64
## 5    53.26
## 6    52.68
```

Now we do the same for AT&T:

```
T_close = dt_combo[3]

#Shift values up a year (shift rows down 1)
T_close1 = T_close %>% mutate_each(funs = funs(lag))

#Before and after:
head(T_close)
```

```
##   T.Close
## 1   43.02
## 2   42.77
## 3   42.65
## 4   41.32
## 5   40.80
## 6   40.81
```

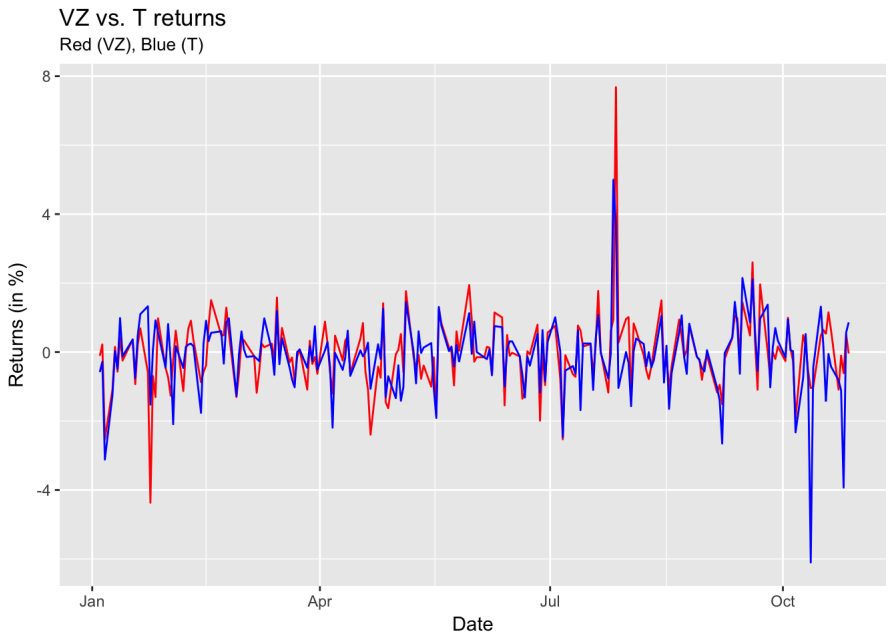
```
head(T_close1)
```

```
##   T.Close
## 1      NA
## 2   43.02
## 3   42.77
## 4   42.65
## 5   41.32
## 6   40.80
```

Notice how the prices have moved down one row in the second table? That's so when we divide the tables, we can get the percent change. Date 1 is NA because you won't have a percent change since that is the very first date!

We can now add the returns (in %) for Verizon and AT&T to our combined data frame `dt_combo` and plot it with `ggplot`.

```
#Add new column
dt_combo$VZ.return = ((VZ_close/VZ_close1-1)*100)

#Add new column
dt_combo$T.return = ((T_close/T_close1-1)*100)

#Rename columns
names(dt_combo) = c("date", "VZ_close", "T_close", "VZ_r", "T_r")

ggplot(dt_combo) + geom_line(aes(x = date, y = VZ_r), color = "red") + geom_line(aes(x = date, y = T_r), color = "
blue") +
  labs(title = "VZ vs. T returns", y = "Returns (in %)", x = "Date", subtitle = "Red (VZ), Blue (T)")
```



From the graph, we can see that the blue line for `T` appears to have sharper and more frequent highs and lows (more volatile), or deviations from its average return. We summarize the expected return (average) and risk (standard deviation) as follows:

```
#Calculate averages
VZ_mean = mean((dt_combo[[4]][-1,])) #VZ
T_mean = mean((dt_combo[[5]][-1,])) #T

#Calculate standard deviations
VZ_sd = sd(dt_combo[[4]][-1,]) #VZ
T_sd = sd((dt_combo[[5]][-1,])) #T

r_summary = data.frame("Verizon" = c(VZ_mean, VZ_sd), "AT&T" = c(T_mean, T_sd))
row.names(r_summary) = c("Expected return", "Risk")
r_summary
```

```
##                    Verizon       AT.T
## Expected return -0.0479483 -0.1082855
## Risk             1.0486658  1.0718786
```

Indeed as we discover, Verizon `VZ` provides a higher expected return for lower risk, but not by much. However, this is still an important consideration for a trader in choosing between the two stocks.

Comparing stock returns with a benchmark

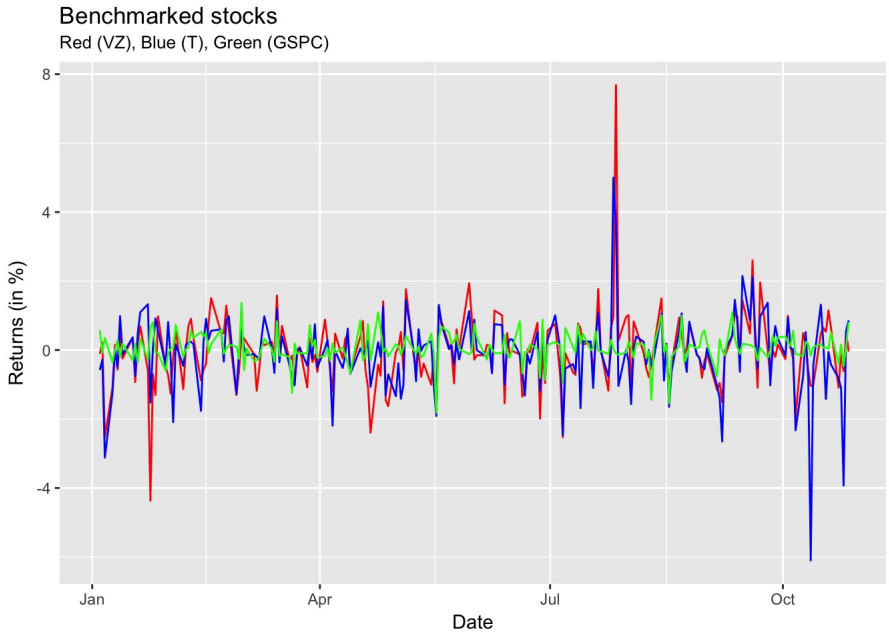
To finish up this comparison, let's add a benchmark to these two stocks by comparing them to the stock market as a whole, which can be represented by the S&P500 index. We do this because as a trader, we are always considering how our particular investment compares with investment opportunities elsewhere (i.e. the market)!

We will repeat what we did before.

```
#Note: The S&P500 index is listed on Yahoo!Finance as "^GSPC"
getSymbols("^GSPC", src = "yahoo", from = start, to = end)
```

```
## [1] "GSPC"
```

```
#Create data frame
dt_SP = data.frame(date = index(GSPC), coredata(GSPC))

#Select SP closing prices
SP_close = dt_SP[5]

#Shift values up a year (shift rows down 1)
SP_close1 = SP_close %>% mutate_each(funs = funs(lag))

#Add new column
dt_SP$return = ((SP_close/SP_close1-1)*100)

#Create new data frame of all three returns
dt_bench = select(dt_combo, c(1,4,5))
dt_bench$GSPC_r = dt_SP$return

#Add to plot of VZ and T
ggplot(dt_bench) + geom_line(aes(x = date, y = VZ_r), color = "red") +
  geom_line(aes(x = date, y = T_r), color = "blue") +
  geom_line(aes(x = date, y = GSPC_r), color = "green") +
  labs(title = "Benchmarked stocks", y = "Returns (in %)", x = "Date", subtitle = "Red (VZ), Blue (T), Green (GSPC
)")
```




```
#Calculate average and SD and summarize
GSPC_mean = mean((dt_bench[[4]][-1,])) #GSPC
GSPC_sd = sd(dt_bench[[4]][-1,]) #GSPC

bench_summary = r_summary
bench_summary$GSPC = c(GSPC_mean, GSPC_sd)

bench_summary
```

```
##                   Verizon       AT.T       GSPC
## Expected return -0.0479483 -0.1082855 0.06557532
## Risk             1.0486658  1.0718786 0.42901469
```

Conclusion

As both our graph and summary table demonstrates, the S&P500 `GSPC` offers the highest returns of the three for the lowest volatility. However, this should only be thought of as a benchmark exercise, meaning that in real life, we cannot *actually* invest into the entire market which is what the `GSPC` index represents. Rather, this is a useful framework for investors to consider their investments: *"Do the expected returns of my investment relative to the market's justify its volatility, and vice versa?"*

Takeaway: Although the answer to this is ultimately up to the individual investor (and that's what makes investing so interesting!), we can always perform analyses, like above, to refine our judgement.

Hopefully you've found my post easy to follow along, and interesting, if not practical.

I've tried to combine what we were taught in class (`dplyr` , `ggplot2`) with some new packages and functions (such as `quantmod`) to address a topic I've been interested in for a while.

All my references are hyperlinked throughout the post but can also be found below for ease of access:

Thanks for reading!

-Jerry

References

- Yahoo!Finance
- More on Xts
- Stock jargon
- Candlestick charts
- Moving averages
- Risk and returns
- S&P500 index