tidyquant Tutorial

This markdown will serve as a brief tutorial for reading in finance data. Luckily, the package `tidyquant` offers the ability to interact directly with serval APIs in order to access up to date stock info.

Goals:

1. Acquire finance data
2. Make a tidy data set
3. Write it out

At the beginning of every script (code only) or markdown (code and room for plain text), it is customary to load necessary libraries and set the working directory. Think of the working directory as where the neighborhood of the file. It uses the working directory when reading in and writing out files. The file itself can "live" somewhere else, but you must specify where it needs to look if that is the case. Be cognizant of your file structure (i.e. don't keep everything in your Downloads folder).

The 'tidyquant' library was developed to turn the `quantmod` and `xts` package outputs into a tidier format. The `tidyverse` is a collection of packages built by Hadley Wickham [http://tidyverse.org/ (http://tidyverse.org/)] that share a common philosophy of shaping and interacting with data.

```
#install.packages("tidyquant") # to install the package if you don't have it; only ne
ed to run this once
#serwd() #insert you own working directory here
library(tidyquant)
library(ggplot2)
```

Let's pull Yahoo Finance data from a single stock. You must know the correct abbreviation for that stock.

```
google <- tq_get(x = "GOOG")
names(google)
```

```
## [1] "date"     "open"     "high"     "low"      "close"    "volume"
## [7] "adjusted"
```

```
head(google)
```

```
## # A tibble: 6 x 7
##         date    open    high     low    close   volume adjusted
##       <date>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 2007-01-03 231.494 236.790 229.065 232.2842 15513200 232.2842
## 2 2007-01-04 232.985 240.411 232.662 240.0686 15877700 240.0686
## 3 2007-01-05 239.691 242.175 237.510 242.0209 13833500 242.0209
## 4 2007-01-08 242.269 243.352 239.542 240.2276  9570600 240.2276
## 5 2007-01-09 241.157 242.547 239.045 241.1814 10832700 241.1814
## 6 2007-01-10 240.650 245.180 239.463 243.1486 12014600 243.1486
```

The default of `tq_get()` grabs the date, volume, opening, highest, lowest, closing, and adjusted price. These data come from Yahoo Finanace. This line is the same as `quantmod::getSymbols()`, but data are returned in tibble (tbl_df) format.

There's many more options for the types of finance data to read in. All options accessed by the following function.

```
tq_get_options()
```

```
##  [1] "stock.prices"       "stock.prices.japan" "financials"
##  [4] "key.stats"          "key.ratios"         "dividends"
##  [7] "splits"             "economic.data"      "exchange.rates"
## [10] "metal.prices"       "quandl"             "quandl.datatable"
```

The description for each of these can be found here:

```
?tq_get
```

Putting a question mark in front of a function will bring up the help file in the bottom right tile. Take a look at the other types of data to read in. The options and sources for the “get =” argument are outlined in the help file. For example, “stock.prices”, “stock.prices.japan”, “key.stats”, and “dividends” are sourced from Yahoo Finance.

More potentially helpful arguments are “from = YYYY-MM-DD” and “to = YYYY-MM-DD”, which allow the user to customize the range of data. Each row read in represents a day of stats.

`tq-exchange()` allows users to read in all the names for a given exhange.

```
tq_exchange_options()
```

```
## [1] "AMEX"   "NASDAQ" "NYSE"
```

```
nyse <- tq_exchange("NYSE")
```

```
## Getting data...
```

We can combine these past two functions to read in longitudinal data for a set of stock symbols.

```
full <- tq_get(x = nyse$symbol[1], get = "stock.prices") %>% add_column(symbol = nyse
$symbol[1])

for (s in nyse$symbol[2:20]){
  single <- try(tq_get(x = s, get = "stock.prices") %>% add_column(symbol = s))

  full <- try(rbind(full, single))
}
```

This loop got us 10 yrs of daily price data from the first 20 listed names in the NYSE.

Let’s look at the structure of this data set

```
str(full)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    45302 obs. of  8 variables:
##  $ date    : Date, format: "2007-01-03" "2007-01-04" ...
##  $ open    : num  5.35 5.07 4.89 4.81 4.82 ...
##  $ high    : num  5.35 5.07 4.9 4.86 4.89 ...
##  $ low     : num  4.99 4.79 4.83 4.77 4.76 ...
##  $ close   : num  5.07 4.93 4.83 4.85 4.86 ...
##  $ volume  : num  381600 394800 329400 369900 312600 ...
##  $ adjusted: num  5.07 4.93 4.83 4.85 4.86 ...
##  $ symbol  : chr  "DDD" "DDD" "DDD" "DDD" ...
```

Dates are their own class, and the labels for symbols are character strings. All numeric information is of type numeric.

Let's look at all the different stocks in here. Obviously, you can change which stocks you pull from the NYSE depending on your personal portfolio.

```
unique(full$symbol)
```

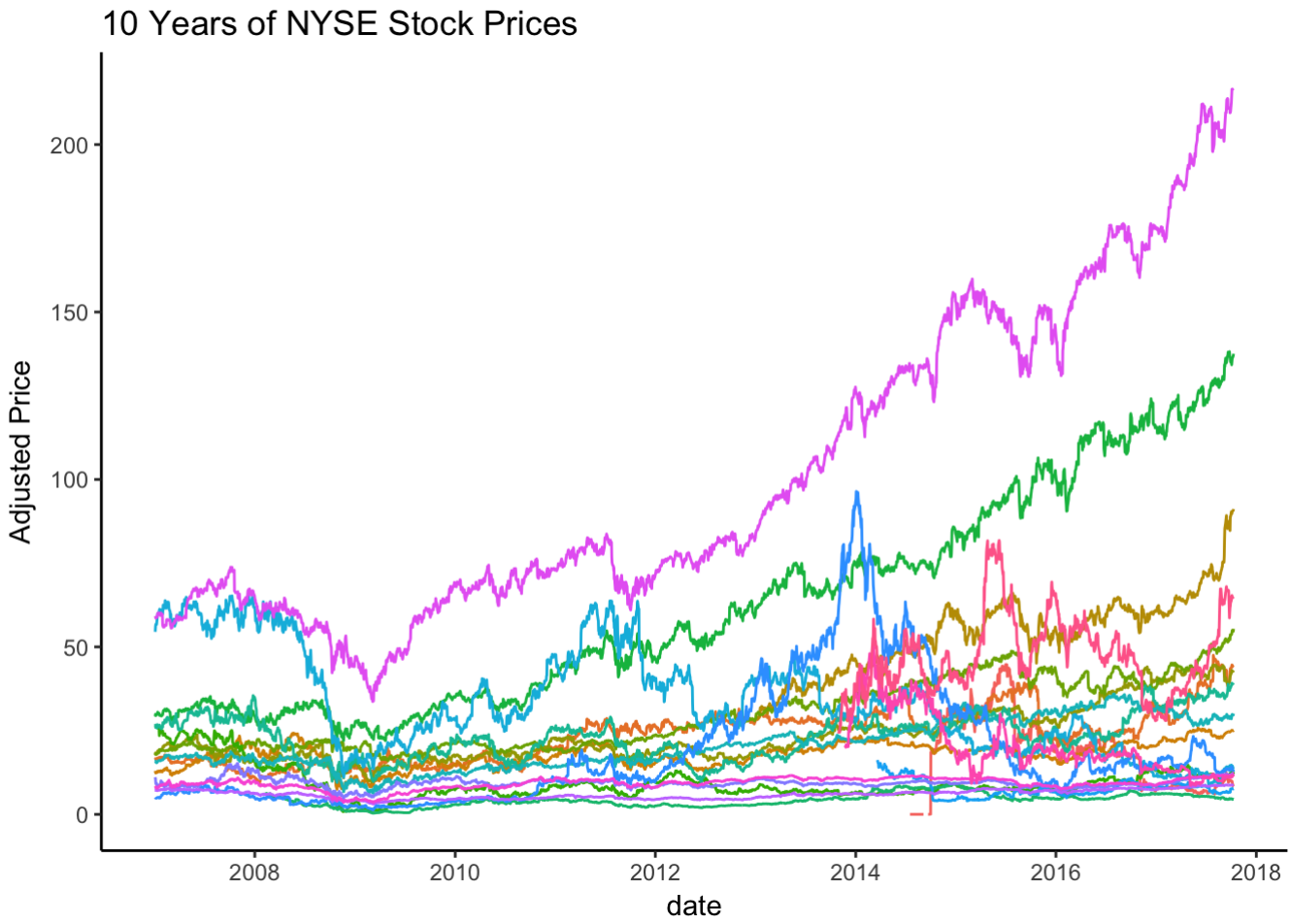
```
##  [1] "DDD"  "MMM"  "WBAI" "WUBA" "AHC"  "ATEN" "AAC"  "AIR"  "AAN"  "ABB"
## [11] "ABT"  "ABBV" "ANF"  "GCH"  "JEQ"  "SGF"  "ABM"  "AKR"  "ACN"  "ACCO"
```

Here's our company names.

The way in which this data set is structured reflects the main principle of tidy data: each row is a unique observation. In our case, each row refers to a day's worth of stock prices for a single company.

Given the tidy structure, we can easily plot these data

```
ggplot(data = full, aes(x = date, y = adjusted, color = symbol)) +
  geom_line(aes(group = symbol)) +
  theme_classic() +
  theme(legend.position = "none") +
  ylab("Adjusted Price") +
  ggtitle("10 Years of NYSE Stock Prices")
```



Lastly, we will write out this data set in two different formats. The first is csv, which can be opened with Excel and is compatible with many programs. Before writing any files, it's a good idea to check the working directory again just to make sure.

```
getwd()
write.csv(full, file = "nyse_stock_prices.csv")
```

The second format is ".RData" which takes up less space in comparison to csv.

```
save(full, file = "nyse_stock_prices.RData")
```

Helpful links

- [http://www.business-science.io/code-tools/2017/01/01/tidyquant-introduction.html (http://www.business-science.io/code-tools/2017/01/01/tidyquant-introduction.html)]
- [https://blog.exploratory.io/introduction-to-tidyquant-quantitative-financial-analysis-for-tidyverse-habitats-e5f72a023ce2 (https://blog.exploratory.io/introduction-to-tidyquant-quantitative-financial-analysis-for-tidyverse-habitats-e5f72a023ce2)]
- [https://business-science.github.io/tidyquant/articles/TQ01-core-functions-in-tidyquant.html (https://business-science.github.io/tidyquant/articles/TQ01-core-functions-in-tidyquant.html)]