Exercise 3. Introduction to Data Science

Part 2. *Using tidyquant Analyze Stock Performance*

Summary
The purpose of this assignment is to introduce you to `tidyquant`, a R package that integrates resources for collecting and analyzing financial data. As you did in part 1, you will retrieve stock price data, apply some of the core functions from `tidyquant` and perform analysis over a set of selected stocks. Finally, you will plot your data and attempt to allocate your portfolio in order to maximize profits over a set time period.

Objectives:

1. To obtain stock price data using a `tidyquant` function in R that retrieves data from an API.
2. To convert quantitative data from wide to long format including modifying periodicity for consistency.
3. To perform a quantitative analysis to maximize profits based on asset allocation throughout your selected portfolio of stocks.

Grading:
This is part two of lab 3 and is due on Saturday, March 16th by 10:00 PM. This lab is worth 10 points towards your final grade. In addition to the deliverables and criteria specified and enumerated within the following laboratory instructions, your grade will be based on the following criteria.

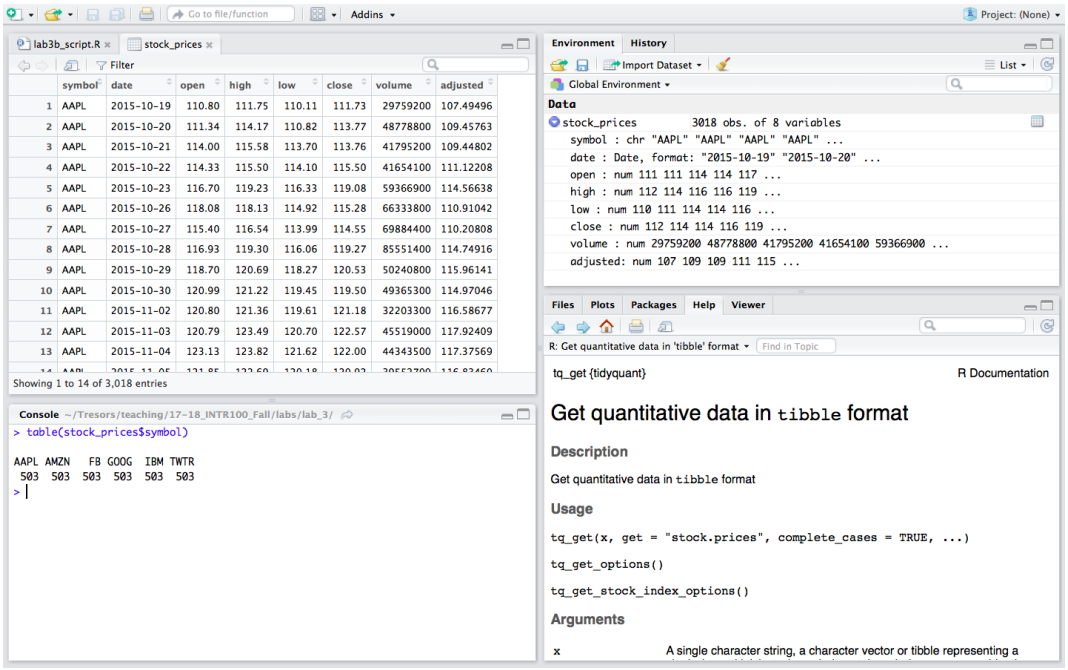
- Your ability to obtain quantitative data that describes a stock prices.
- Your ability to convert that data in order to determine monthly stock market returns.
- Your ability to allocate investments to maximize portfolio profits over a given time period.

You will need to upload the laboratory deliverables to blackboard prior to the deadline.

Instructions. *Step 1: Selecting and obtaining your stock data*

1. Open R Studio. Go to your William & Mary `H:\\` drive, and create a new `.R` script, name it `script_3b` or something similar and save it to your `H:\\lab_3` folder. Just as we have previously done, set your working directory using the `setwd()` command.
2. Once you have named your script and set your working directory, install the package `tidyquant` including all of its `dependencies`. Load the `tidyquant` library of functions.

3. In part 2 of this lab, imagine you are managing a portfolio of stocks. You will need to make decisions about allocating an investment across your portfolio in order to maximize profit for your client. Create an object that contains your portfolio of stocks by using the same 6 symbols you used in part 1 of this lab (the 5 predictors and 1 response stocks). Using the `c()` command from base R, write your first line of code as though you are assigning a single character vector of length 6 that contains all of your selected stocks. For example in part 1 of this lab, I used `AAPL`, `GOOG`, `IBM`, `AMZN`, `TWTR` and `FB`. Now instead of actually creating a vector object, use the pipe operator `%>%` to *pipe* that *object* forward into a function (found on the next line of code) and populate it with the selected stock data for a given period of time. The pipe operator is the `%>%` series of characters, and it is placed at the end of a line of code to indicate that the preceded object will be *piped* into the following *function*. Its specification is similar to the + symbol and its manner of use with the `ggplot()` library of functions. Follow the pipe operator with the `tq_get()` function to output the retrieved stock data as a new kind of data frame called a `tibble`.




```
tibble_name <- baseR_function("SYM1", "SYM2", "SYM3", "SYM4", "SYM5", "SYM6") %>%
  tidyquant_function(get = "stock.prices",
         from = "yyyy-mm-dd",
         to = "yyyy-mm-dd")
```

A `tibble` is a new version of a data frame that keeps important features while dropping less convenient ones (i.e. converting character vectors to factors). Within the argument of your function that the data is being piped into, set the `from =` and `to =` arguments to capture stock price data for the past two years. See the image on the previous page to review the resulting `tibble` created by piping data into a `tidyquant` function.

4. You have created a `tibble` that contains prices over the past 2 years for each of your 6 stocks. Now you will transform the data in order to describe each stocks monthly returns over the past 25 months. To do that, name a new `tibble` object that will be created by *grouping* your stock data and adjusting its periodicity so it is consistent across the time horizon. In this second function, you will again use the `%>%` to *pipe* your stock price data into the `group_by()` function and then you will use the `tidyquant` function `tq_transmute()` in order to set the period rate of return to monthly.

```
new_tibble <- existing_tibble %>%
  group_by(variable_containing_symbols) %>%
  tidyquant_function(select = variable_with_adjusted_values,
                     mutate_fun = periodReturn,
                     period = "set_period",
                     col_rename = "name_of_col_for_month_returns")
```

   You are using the `periodReturn` option in the `mutate_fun =` argument. If you type `tq_mutate_fun_options()` into the console, you can review all of the possible functions available. Use the `periodReturn` function from the `quantmod` package, and then set the period to `"monthly"` with the following argument within the `tq_transmute()` function. Open your new tibble that was created by the above command and you should see a data frame like object that contains 150 rows (6 stocks x 25 months) and 3 columns (symbol, month and monthly return).

5. Now imagine you have been given $10,000.00 and have been asked to allocate that investment in your portfolio of 6 stocks. You will distribute a percentage of that money to each of the stocks that has performed best over the past 24 months with the goal of maximizing the total return on your client's investment. To do this, first create a numeric vector object that is a combination of 6 numbers where the sum of those 6 numbers is equal to 1.00 and all of those numbers are positive (no negative numbers). Each of these numbers will represent the percentage of the $10,000.00 that will be invested in each corresponding stock. For example, if `"AAPL"` is your first stock, and you set the first value in your numeric vector to `0.10` then that means $1,000.00 will be invested in Apple. Each of these 6 numbers represents the weight that its corresponding stock holds of the total initial investment.

```
weights <- function(0.10, 0.20, 0.00, 0.30, 0.00, 0.40)
```

   Since my original tibble object that contains my stock prices for the past two years was ordered `AAPL`, `GOOG`, `IBM`, `AMZN`, `TWTR` and then `FB`, the above investment strategy would allocate $1000.00 to Apple, $2000.00 to Google, $3000.00 to Amazon and $4000.00 to Facebook. Go back to your candlestick plots from part 1 and review the data to estimate allocating your client's investment across your portfolio with the intent to optimize the return on investment.

6. Now create a new tibble object that is a transformation of the monthly stock returns aggregated into total portfolio returns, also on a monthly basis. Again, start off by as-

signing your new tibble object from your old one (in this case from the tibble you created of monthly stock returns) and follow it with your `%>%` pipe operator. Then use the `tq_portfolio()` command to receive the *piped* data and transform it using the arguments from that `tidyquant` function.

```
new_tibble <- stock_returns_tibble %>%
  tidyquant_function(assets_col = variable_containing_symbols,
                     returns_col = name_of_col_for_month_returns,
                     weights = name_of_weights_object,
                     col_rename = "name_of_col_for_portfolio_returns")
```

Your resulting `tibble` object should have 25 rows [1], and two columns indicating the month and total return for your portfolio.

7. In your next step, take the new tibble you just created that aggregated monthly stock returns for all stocks in your portfolio follow it with the `%>%` pipe operator to send the data to a command. But this time instead of *piping* the data to a `quantmod` function, use the *grammar of graphics* to plot the returns from your portfolio of selected stocks. Start with the `ggplot()` function, and use the aesthetics argument to specify that the x axis will be time (or the month) and the y axis will be the return on investment from the portfolio. Then use the `geom_bar()` function to create a bar plot of all monthly values. In order to just plot values, use the `stat = "identity"` argument within the `geom_bar()` function. Also add the `fill = palette_light()[1]` argument to set the fill color of each bar. You can change the number inside the `[]` subscripting operator in order to modify the color. You can also just type `palette_light()` into the console to see all of the options.

   After adding the data as a bar plot, convert the y axis from absolute values in dollars to the percentage of total returns by using the `scale_y_continuous(labels = scales::percent)` command. Continue your quantitative description of the portfolio returns by adding a geometric line that describes the trend of returns using a linear regression model as its method to *smooth* the data. Geometric smoothing uses the `geom_smooth()` function and in order to plot the line using a linear model add the argument `method = "lm"`. Following adding the trend line to your plot, use the `labs()` command to add a `title`, `subtitle`, `caption` describing your plot and a `y` axis label. You can leave the `x` axis label blank. Finally, add the tidyquant theme to your barplot.

```
portfolio_returns_monthly %>%
  ggplot(aes(x = variable, y = variable)) +
  geom_bar(stat = "identity", fill = palette_light()[col_number]) +
  scale_y_continuous(labels = scales::percent) +
  geom_smooth(method = "lm") +
  labs(title = "add a title",
```

[1] I have both the first and last month in my time series, thus adding one extra month

```
        subtitle = "add sub title",
        caption = "describe the plot",
        x = "",
        y = "y axis label") +
  theme_tq() +
  scale_color_tq()
```

8. Now repeat step 6, creating a new tibble using your `%>%` pipe operator to first *pipe* the data into the `tidyquant` function `tq_portfolio()` and then use your `%>%` pipe operator a second time to *pipe* the results from the first function into the `mutate()` command.

```
portfolio_growth_monthly <- stock_returns_monthly %>%
  tidyquant_function(assets_col = variable_containing_symbols,
                     returns_col = name_of_col_for_month_returns,
                     weights = name_of_weights_object,
                     col_rename = "name_of_col_for_portfolio_returns")
                     wealth.index = TRUE) %>%
  mutate(investment.growth = renamed_column_from_above * amount_of_investment)
```

9. Finally, plot a line that describes the growth of your portfolio. Go back to step 7 above and replace `y =` with your object that describes investment growth. Instead of `geom_bar()` use `geom_line()` and specify arguments for `size =` and again `color =`. Add a title, subtitle, caption as well as a label for the y axis. Instead of using percentage, use `scale_y_continuous(labels = scales::dollar)` to measure absolute value.

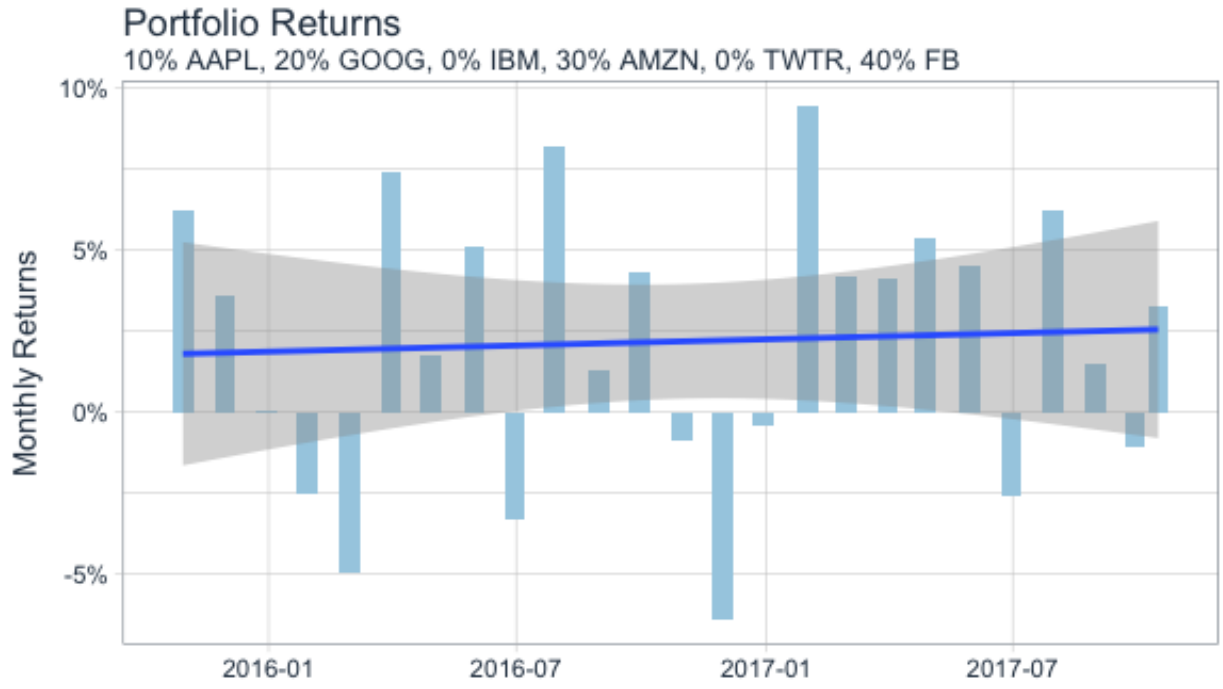
10. Go back and reallocate your investments across your 6 stocks and optimize the distribution in order to maximize profits. Use your optimal investment strategy to produce your two deliverables.

Deliverables

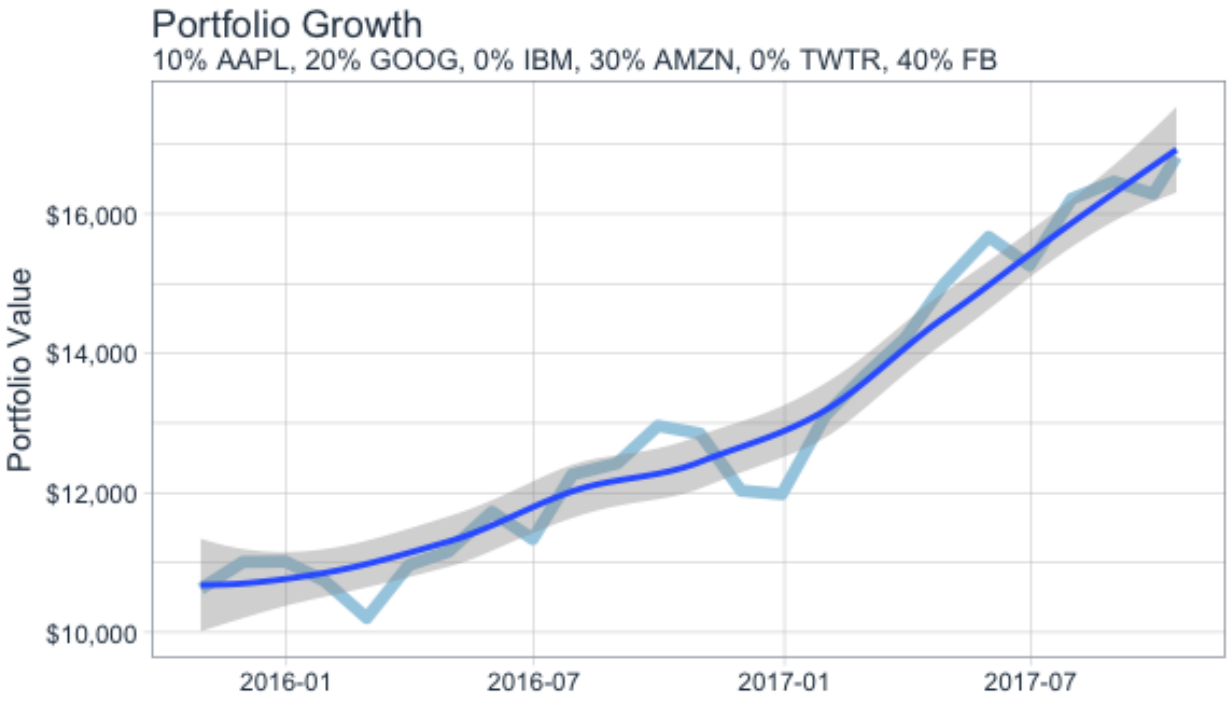
1. A barplot that describes portfolio returns for your six stocks over the past 2 years.
2. A line plot that describes portfolio growth for your six stocks over the past 2 years.

Stretch Goals

1. Choose six new stocks and allocate your funds in order to maximize returns.
2. Compare growth of different investment allocations as line plots on the same graphic.



Shows an above-zero trend meaning positive returns



Visualize Performance