Hosted File: https://x3zr3z-jon-karanezi.shinyapps.io/US_Securities_Monitoring/

R files: https://uwmadison.box.com/s/dbdxfykiajztqmc9gn18jlx4p5yrdgl7

Data: library.install(quantmod)

Introduction

As part of this homework I wanted to create something of a financial monitor for securities in the US stock markets. As an avid enthusiast for all things financial and investing, this was a very fun project to work on. I took a lot of inspiration from **Google Finance and Yahoo Finance**, especially trying to mimic the Google Finance chart using ggplot. The goal of this Shiny app is to give someone an ability to easily screen various stocks and see gain/loss metrics over a certain amount of time. I do plan to continue working on this, as I would like to add some more advanced functionalities, such as stock chart comparisons and various tabs for things such as financial statements and company overview.

Interesting Findings

This project differs from others in that **the dataset is not static** (new stock data is added/changed daily), so it works more so as an API call to retrieve a new data copy of a particular stock each time we need it. Besides that, here's some interesting findings I had during the project.

- For many stocks, the **Covid-19 pandemic** shifted and caused massive economic implications, which tended to decrease the share prices. In essence, investing in almost any sensible stock during that time would have made money.
- Stock data only goes back to 2007, so for many companies that have been well established for some time, we only had a **limited scope**.

Data Preparation and Interface Setup

1. **Data preparation** was one of the trickiest parts of this project since the data supplied by the quantmod library, which was scraped from yahoo, arrived in xts format. Each xts dataframe showed data for 1 stock, according to a getSymbols() call and for better compatibility with ggplot, I transformed the data into a tsibble and only keeping the 2 columns (Date, Close Price)
2. On a more simple flow of data, here's how it went:

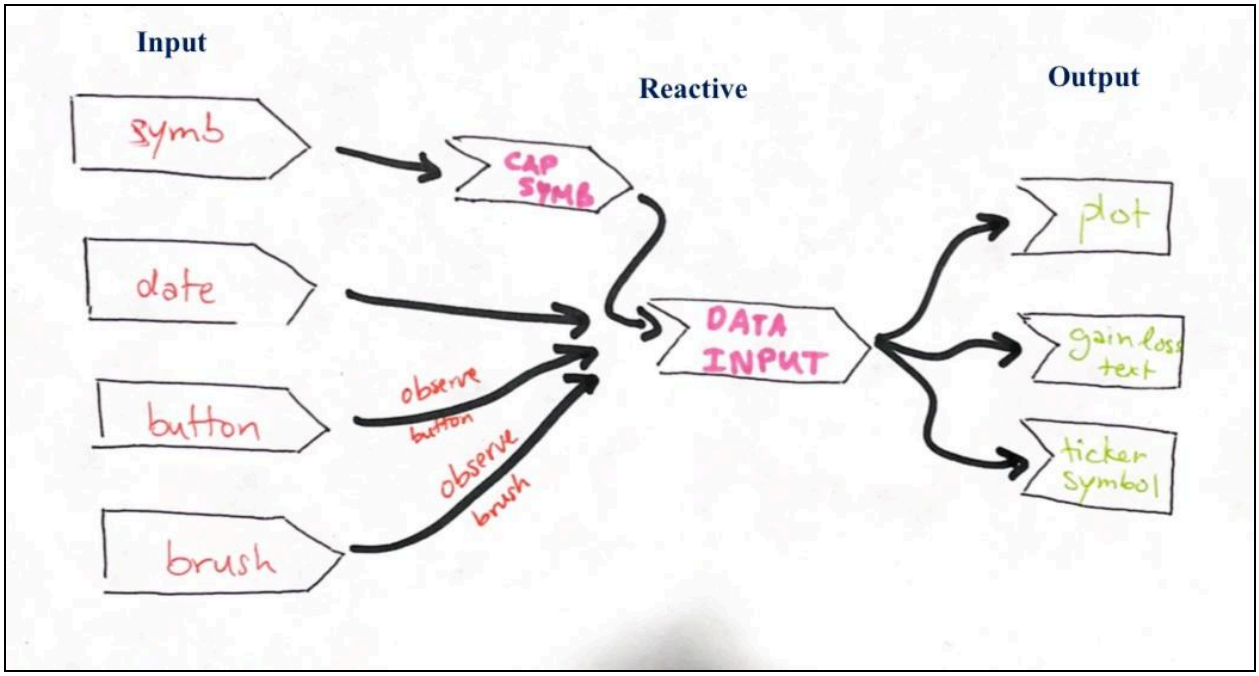
a. Based on stock symbol input, gather data from yahoo finance using getSymbols() function in the quantmod package.
b. Feed input to our reactive dataInput() to get, transform, clean and reshape the data to tsibble.
c. Plot our outputs using ggplot, geom_ribbon is added if there is a brush selection, and outputting our texts (ticker symbol and gain loss metric if there is a brush selection)

3. **Interface setup** was rather simple, but instead of using fluidRow I explored some other options and went with a page_sidebar, which I felt better reflected the functionality of this app. One of the more difficult areas was changing fonts and customizing the theme, which I used some stackOverflow and html/css customization to achieve.

Reactive Graph

Thing to note:

- The way the app works is by fetching a dataset for the stock ticker symbol that's entered. Each time a new ticker symbol is typed in, a new dataset is pulled for that particular stock and goes through the same process of filtering, transforming, etc. Since the data is dynamic, I used observeEvent(s) for both the brush and button. This was more practical than trying to manually set row selections to true or false, given that the data isn't static.



Inputs:

- symb
  - Input text for the ticker symbol, default set to SPY
  - Is routed to cap_symb which goes to dataInput()
- date
  - Date range input which filters our data we get
- button
  - Includes **ObserveEvent** to
    - Reset chart if there is brush selection and removes gain loss statistics, does not affect date or ticker symbol selection.
- brush
  - Includes **ObserveEvent** to
    - to select a certain date range across the x-axis on our stock chart. Accordingly adds another plot **(geom_ribbon)** with limited date range above the one already shown.
    - Adds gain loss metric

Reactive:

- cap_symb

    - Takes the symb and capitalizes so we can use accordingly for quantmod API call, routes to dataInput
- dataInput
    - Creates the tsibble of data, transforming and filtering if needed. **Note:** At this point only one dataset copy of the entire date range for a single stock is created. If we filter by date range, this does not call dataInput directly, but only filters the plot figure. Perhaps in the reactive graph it would make more sense to route data straight to plot, but I wasn't sure so I kept it routed to dataInput.

Outputs:

- plot (stock chart)
    - Plots the stock chart that we see in the middle, is layered using **geom_ribbon** if a brush is used to select an area.
- ticker_symbol
    - Prints the ticker symbol on top left for the accompanying stock ticker that was inputted in symb.
- gain/loss text
    - If the chart has been brushed through this prints a gain/loss percentage on the top right corner.