Gravity Market - Process Book

Visualization created by: Adam Colton (u1377031@utah.edu), Courtney Holbrook (u1156298@utah.edu), and Lane Zaugg (u1375540@utah.edu).

The link to the private project repository can be found here:
https://github.com/1anza/dataviscourse-pr-gravitymarket

The link to the live project can be found here:
https://1anza.github.io/dataviscourse-pr-gravitymarket/

Overview & Motivation:

Background and Motivation

The primary motivation in choosing this project revolves around the idea that stock market data has been presented in a stale and uninteresting way for a very long time. It would be fascinating to tweak some of the assumptions of what this information should look like. Rather than focus purely on line and candlestick charts with numbers to demonstrate changes in the market, this project will include a broad bubble chart that will visually demonstrate broader market movements. Presenting data in this way will allow a user to quickly infer general trends over periods of time, allowing users to identify key factors that are moving the market in certain directions.

Project Objectives

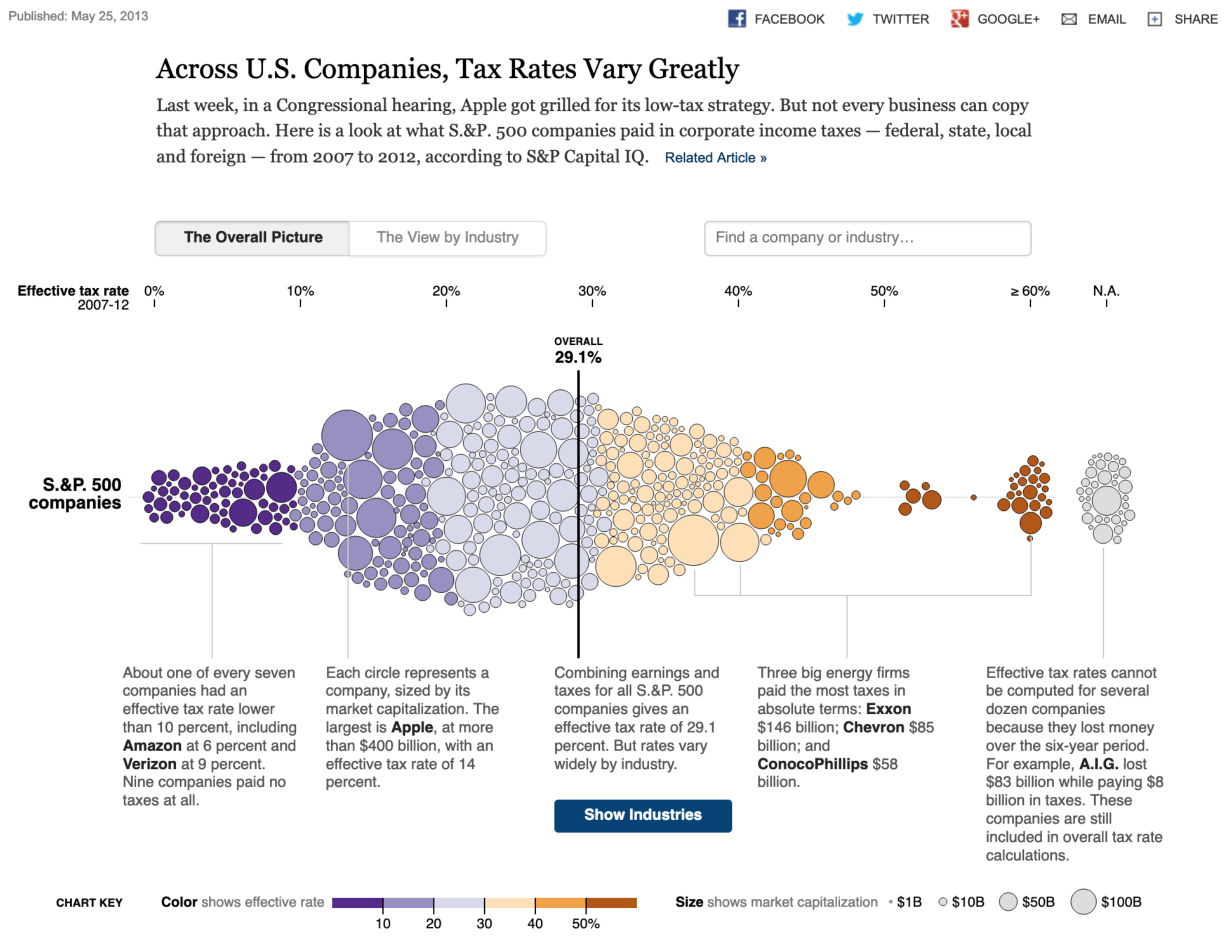
The primary questions this project and visualization is trying to answer are these.. What kind of outliers fuel or drag historical market movements? Do market sectors push or pull the market in specific directions? What does the market look like when its average is up? What about when it is down? Does representing the stock market in this way encourage more curiosity about why and how the market is moving the way it is?

It would be beneficial to learn how to link and utilize very large datasets to a visualization. It would be even more interesting to succeed at the optional task of populating this visualization with dynamic data from a minute-by-minute update. Academically, no project in any prior course has yet hit that dynamic mark and it seems as though most real-world applications utilize data in this way, making it a fun option to reach for.

The benefits of this visualization is the ability to immediately see outliers or underlying trends of a stock market index. Rather than be fixed to a numerical representation, this project's bubble chart will demonstrate these changes in a more intuitive way.

Related Work: Anything that inspired you, such as a paper, a website, visualizations we discussed in class, etc.

Works inspiring this project stem from class visualizations and other attempts at creating a stock-bubble visualization. The class-discussed visualization that comes to mind was the New York Times visualization on corporate tax rates: https://archive.nytimes.com/www.nytimes.com/interactive/2013/05/25/sunday-review/corporate-taxes.html.



Across U.S. Companies, Tax Rates Vary Greatly

Last week, in a Congressional hearing, Apple got grilled for its low-tax strategy. But not every business can copy that approach. Here is a look at what S.&P. 500 companies paid in corporate income taxes — federal, state, local and foreign — from 2007 to 2012, according to S&P Capital IQ. Related Article »

About one of every seven companies had an effective tax rate lower than 10 percent, including **Amazon** at 6 percent and **Verizon** at 9 percent. Nine companies paid no taxes at all.

Each circle represents a company, sized by its market capitalization. The largest is **Apple**, at more than $400 billion, with an effective tax rate of 14 percent.

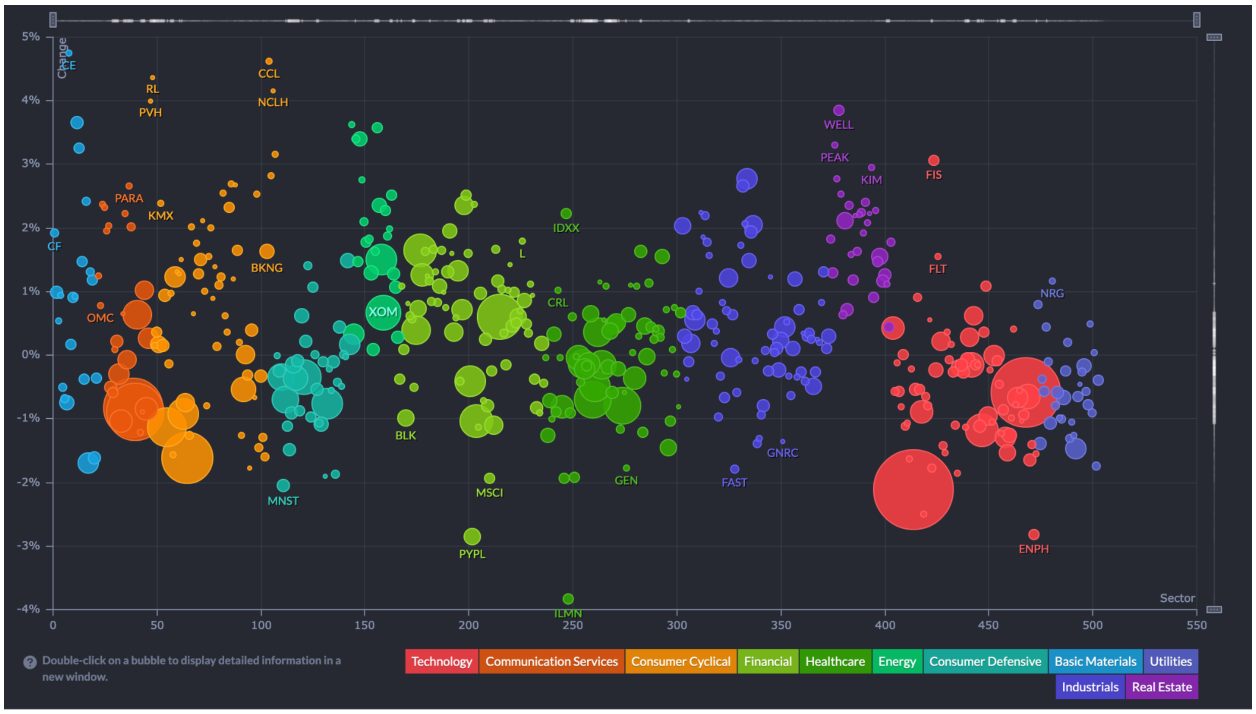
Combining earnings and taxes for all S.&P. 500 companies gives an effective tax rate of 29.1 percent. But rates vary widely by industry.

Three big energy firms paid the most taxes in absolute terms: **Exxon** $146 billion; **Chevron** $85 billion; and **ConocoPhillips** $58 billion.

Effective tax rates cannot be computed for several dozen companies because they lost money over the six-year period. For example, **A.I.G.** lost $83 billion while paying $8 billion in taxes. These companies are still included in overall tax rate calculations.

This visualization focuses more on taxes and doesn't dynamically update with new data, however it is significant in that it demonstrates a beeswarm with forces as well as grouping by sectors.

In addition, FinViz has a bubble visualization that is static with current market movements, but it also shows sector differentiations similar to the NYTimes article.



This FinViz visualization is static. It doesn't allow a user to manipulate time ranges to see trends. This is important, especially when considering multiple sectors across the market. Bubbles here also overlap, making the overall visualization appear a bit messy.

There also needed to be some type of physics to pull the stocks toward their respective destination. Similar to this bouncing ball:http://physicscodes.com/html5/bouncing_ball/

Here the ball is getting pulled to the bottom of the screen due to gravity. This project aims to accomplish something similar with the gravitational pull being toward the stock's relevant position regarding its percent change.

Questions

There are some questions this project wanted to explore and provide an answer to. One of the first questions was how can this project make interacting with stock market tools more interesting? As it stands, current interactions with this type of data remain mostly static and boring. Being able to change dates and see how individual companies in the broader market perform compared to their peers.

Another question that was considered was whether data could be directly queried from an API as a user interacted with the visualization. This proved to be far too slow for the type of snappy visualization this project is aiming for.

Additionally, the length of historical stock data greatly determined the amount of stress the DOM would have to handle. Ideally minute-by-minute granularity could be served for the last 30 years off of a server, however retrieving this kind of data for free is very challenging, both in the collection and in the serving of the data. Considering the goal is to provide as snappy a visualization as possible for the lowest cost, day-by-day granularity over the last year was the most sensible decision.

This goal of snappiness also drove the project to consider the question of using events in D3. In doing so, it would be possible to update all charts and data at the same time without any repeated calculations. The creation of these events were done using the GlobalAppState aka ‘gas’. Taking full advantage of the gas proved to be very effective in making sure everything tied together.

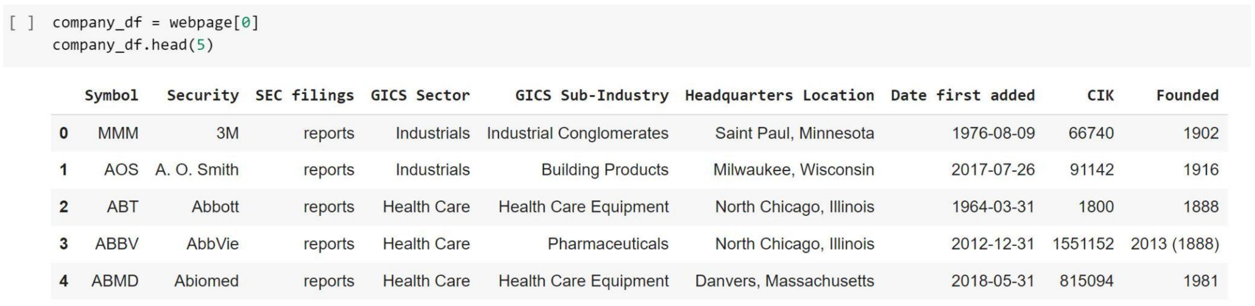
Considering the prototype images, the project's initial goal was to present the stock bubbles in a vertical format in order to utilize the most vertical space possible to show movements. However, working with so much data in the beeswarm it became clear that the best option would be to present the bubble horizontally with dynamically changing scales. This would allow a maximum amount of information to be displayed without losing out on individual stock movements.

Time intervals were another big question that the project faced. A very simple way to implement time intervals would be to allow a user to select a beginning date and an end date. While common and simple, this evolved into the project pursuing a more dynamic way to bind time. This was done similar to the way garage band uses its selection for music tracks. You have a beginning and an end, each being able to move across the ‘track’ in order to pinpoint where the data should be focused. The project accomplishes this in the same way, allowing a user to manipulate beginning and end points over a time bar.

Changing the workflow also came up through working on the project. It made more sense for the team to use node and npm in order to easily import more libraries and make regular builds of the project for debugging. Npm also provided better type-hints while programming, making that process much easier.

Data

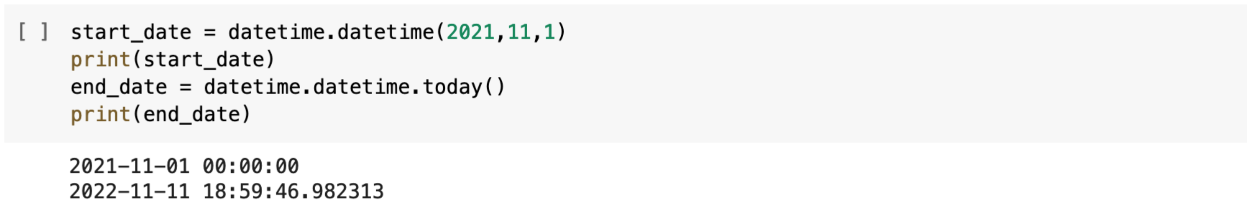
The data source for this project is the Yahoo Finance API. The Yahoo Finance API (https://python-yahoofinance.readthedocs.io/en/latest/api.html) provides a simple framework to query historical data. These data points are not historically intraday, but do provide enough useful information on a day-by-day granularity to demonstrate interesting historical trends over the last year. The data collected using the API came from the S&P 500 stock index. In order to determine which stocks are in the S&P 500, a quick scrape of wikipedia provided a comprehensive list of all stocks in the index. In addition to the S&P 500 stocks, the additional information scraped from Wikipedia included economic sectors to which a stock belongs. This sector information, ranging from health care to real estate, provides an interesting division amongst the S&P 500 companies.



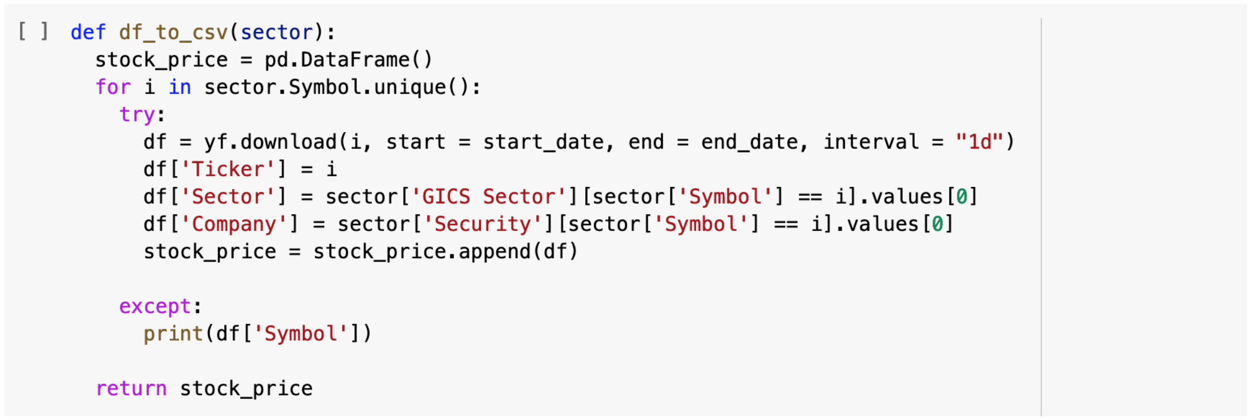
```
[ ]  company_df = webpage[0]
     company_df.head(5)
```

	Symbol	Security	SEC filings	GICS Sector	GICS Sub-Industry	Headquarters Location	Date first added	CIK	Founded
0	MMM	3M	reports	Industrials	Industrial Conglomerates	Saint Paul, Minnesota	1976-08-09	66740	1902
1	AOS	A. O. Smith	reports	Industrials	Building Products	Milwaukee, Wisconsin	2017-07-26	91142	1916
2	ABT	Abbott	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	1800	1888
3	ABBV	AbbVie	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	1551152	2013 (1888)
4	ABMD	Abiomed	reports	Health Care	Health Care Equipment	Danvers, Massachusetts	2018-05-31	815094	1981

Using the API, it was then possible to use this information to pull the open, high, low, close, and volume data for each stock ranging from now to the past year.

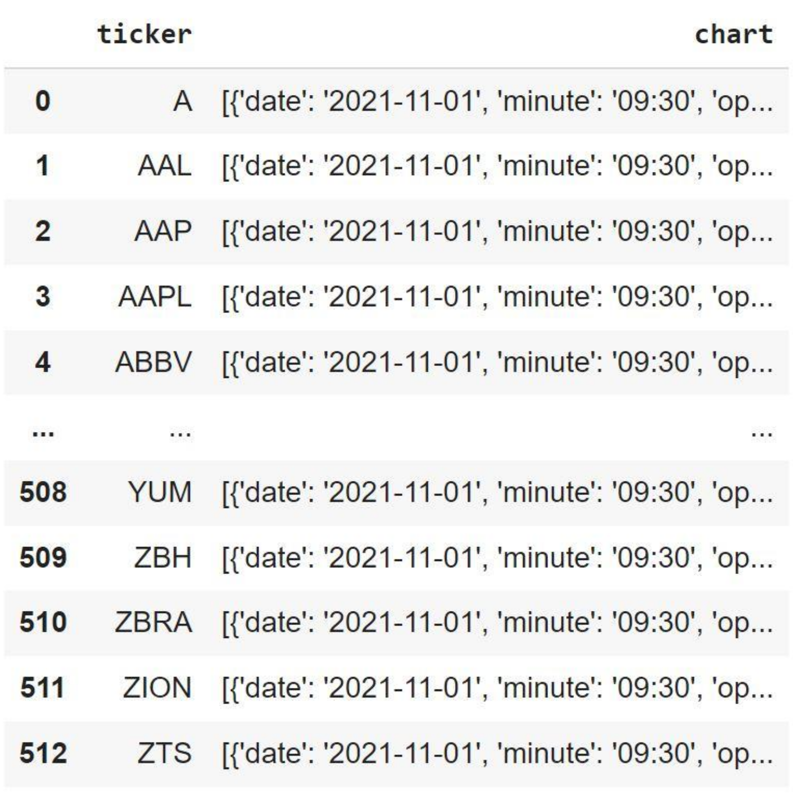


```
[ ]  start_date = datetime.datetime(2021,11,1)
     print(start_date)
     end_date = datetime.datetime.today()
     print(end_date)

     2021-11-01 00:00:00
     2022-11-11 18:59:46.982313
```



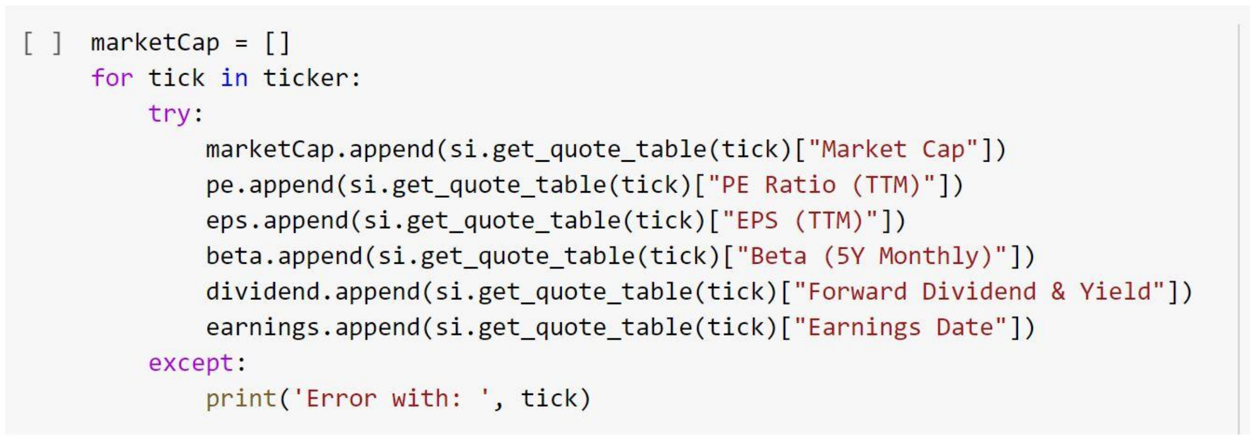
```
[ ]  def df_to_csv(sector):
       stock_price = pd.DataFrame()
       for i in sector.Symbol.unique():
         try:
           df = yf.download(i, start = start_date, end = end_date, interval = "1d")
           df['Ticker'] = i
           df['Sector'] = sector['GICS Sector'][sector['Symbol'] == i].values[0]
           df['Company'] = sector['Security'][sector['Symbol'] == i].values[0]
           stock_price = stock_price.append(df)

         except:
           print(df['Symbol'])

       return stock_price
```

Pulling the relevant data for the selected stocks, it then became important to format the data in a way that the visualization could easily use. The data points that were able to be collected throughout the entire interval of time were a stock's open, high, low, close, and volume values. Considering this time series data, it made the most sense to bind them all under the same data field of 'chart' in order to make it easy to iterate through the time data.



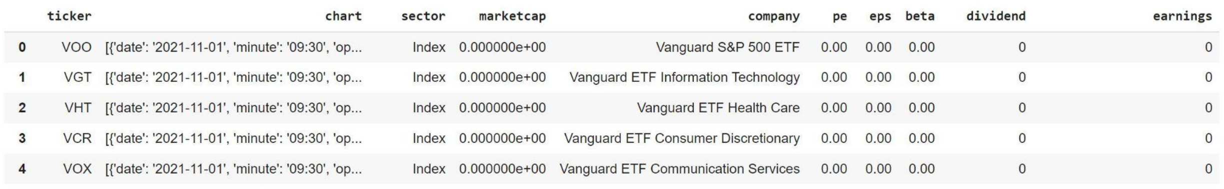
	ticker	chart
0	A	[{'date': '2021-11-01', 'minute': '09:30', 'op...
1	AAL	[{'date': '2021-11-01', 'minute': '09:30', 'op...
2	AAP	[{'date': '2021-11-01', 'minute': '09:30', 'op...
3	AAPL	[{'date': '2021-11-01', 'minute': '09:30', 'op...
4	ABBV	[{'date': '2021-11-01', 'minute': '09:30', 'op...
...	...	...
508	YUM	[{'date': '2021-11-01', 'minute': '09:30', 'op...
509	ZBH	[{'date': '2021-11-01', 'minute': '09:30', 'op...
510	ZBRA	[{'date': '2021-11-01', 'minute': '09:30', 'op...
511	ZION	[{'date': '2021-11-01', 'minute': '09:30', 'op...
512	ZTS	[{'date': '2021-11-01', 'minute': '09:30', 'op...

All other data fields had to be statically obtained using the most recent data for each stock. This was done by performing an additional scrape using a different element of the Yahoo Finance API. This allowed information to be pulled regarding a stock's PE ratio, market cap, earnings per share, beta, next earnings data, and whether or not the company provided a dividend.



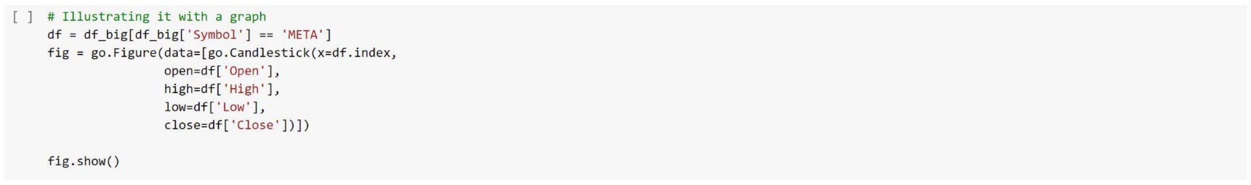
```
[ ]  marketCap = []
     for tick in ticker:
         try:
             marketCap.append(si.get_quote_table(tick)["Market Cap"])
             pe.append(si.get_quote_table(tick)["PE Ratio (TTM)"])
             eps.append(si.get_quote_table(tick)["EPS (TTM)"])
             beta.append(si.get_quote_table(tick)["Beta (5Y Monthly)"])
             dividend.append(si.get_quote_table(tick)["Forward Dividend & Yield"])
             earnings.append(si.get_quote_table(tick)["Earnings Date"])
         except:
             print('Error with: ', tick)
```

Some stocks did not have any of this additional information. These were stocks indexes that were added to the data in order to easily extrapolate and see overall index performance in each stock sector. Since there were a total of 11 sectors in all, 11 stock indexes were queried which displayed the overall performance of each sector. This information was important because it saved calculations in the DOM and allowed the project to get an average of sector performance in a dedicated line chart designed to show this information.

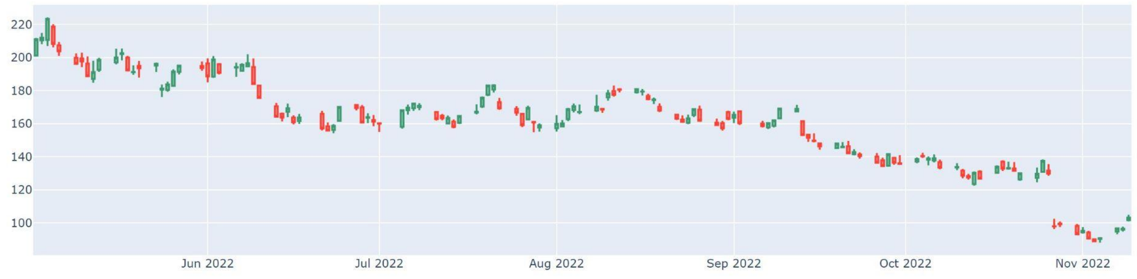


	ticker	chart	sector	marketcap	company	pe	eps	beta	dividend	earnings
0	VOO	[{'date': '2021-11-01', 'minute': '09:30', 'op...	Index	0.000000e+00	Vanguard S&P 500 ETF	0.00	0.00	0.00	0	0
1	VGT	[{'date': '2021-11-01', 'minute': '09:30', 'op...	Index	0.000000e+00	Vanguard ETF Information Technology	0.00	0.00	0.00	0	0
2	VHT	[{'date': '2021-11-01', 'minute': '09:30', 'op...	Index	0.000000e+00	Vanguard ETF Health Care	0.00	0.00	0.00	0	0
3	VCR	[{'date': '2021-11-01', 'minute': '09:30', 'op...	Index	0.000000e+00	Vanguard ETF Consumer Discretionary	0.00	0.00	0.00	0	0
4	VOX	[{'date': '2021-11-01', 'minute': '09:30', 'op...	Index	0.000000e+00	Vanguard ETF Communication Services	0.00	0.00	0.00	0	0

As the data was pulled and fitted to the format needed, it was easy to check the data points within python to see that the information was querying correctly. Most exploratory data analysis was done this way.



```
[ ] # Illustrating it with a graph
    df = df_big[df_big['Symbol'] == 'META']
    fig = go.Figure(data=[go.Candlestick(x=df.index,
                    open=df['Open'],
                    high=df['High'],
                    low=df['Low'],
                    close=df['Close'])])

    fig.show()
```



Since nearly all the data processing for this project was done in python, it saved a considerable amount of work that would have otherwise needed to be done in D3. Most of this work was done using pandas dataframe to fit everything the way it needed to be fitted. Overall, the goal became to get as much reasonable data as possible in the smallest file size possible to ensure the DOM could handle the data it was given on most machines. The only primary data calculations the project handled in javascript were those to calculate percent change relative to the time interval.

Design Evolution:

Design decisions influenced by the perceptual & design principles learned in the course:

Scale: In order to maintain graphical integrity with our scales, we decided to have the scales dynamically update when a new sector is selected or a single company for the OHLC chart. This ensures that our data is properly represented, and not skewed by a static scale.

Avoid Chart Junk/Small Multiples: We also wanted our visualization to have clean and clear graphics, rather than something overly-embellished that would interfere with the overall message of the data. We also wanted to ensure the user didn't have to memorize one visual in order to compare it with others. As a variation of small multiples, we utilized many different types of chart to represent the stock market data. The line chart displays the average percent change for the sector, weighted by market cap, while the beeswarm shows the percent change for each individual company all at once, and the OHLC chart displays the open and close prices for the individual company.

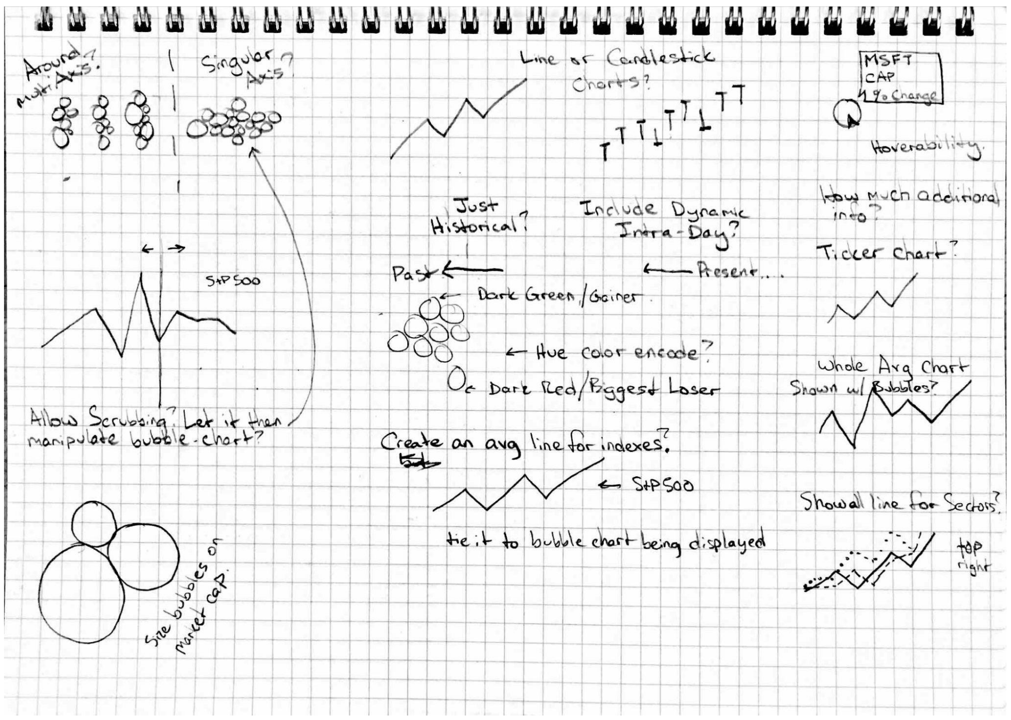
Interaction/Animated Transitions/Change over time: Our visualization utilizes many interactable features to allow the user to gain more insight from the data. The beeswarm, line chart and OHLC allow the user to select what sectors they want to see/compare with animated transitions throughout the selection process. Since our data focuses mostly on the change over time, our visualization implements a playback option that displays how the stock market data changes over time, allowing the user to see how the stock market has been affected over the past year.

Dashboard: Even though our data is historical, it can become a visualization for real-time stock data, so we chose to implement a type of "dashboard" design that gives a lot of information about the stocks. The beeswarm provides the percent change and market cap. The line chart provides the average percent change, weighted by the market cap. The OHLC provides the open and closing prices, colored depending on either an increase or decrease in price. The table provides individual statistical data for a selected company.

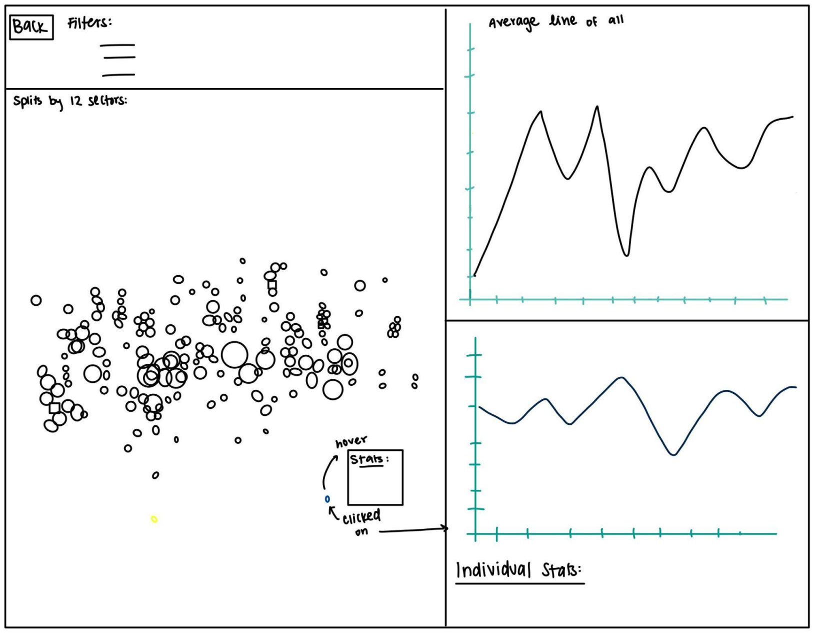
Throughout the prototyping process, we applied these concepts to the overall visualization. Through prototypes #1 & #2 we brainstormed how to avoid chart junk and keep the visualizations informative, but not overly complex. Through prototypes #3 & #4, we discussed a dashboard view for the main page, and how to apply interactions to the visualization. Once the final design was reached, we decided on the dashboard drawn, with each visualization connected through an interaction from the user. Once the implementation process began, we added the proper scales to protect the integrity of our data.

(Prototype sketches pictured below)

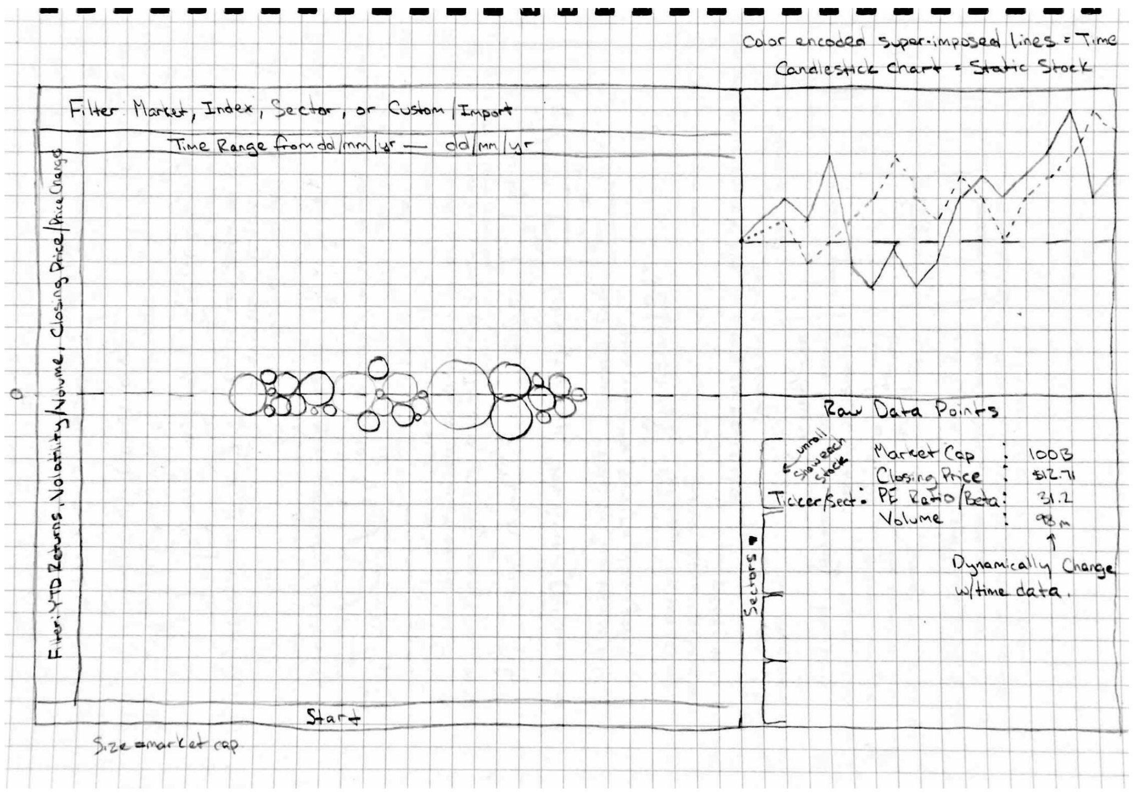
Prototype #1:


Around multi Axis?
Singular Axis?
Line or Candlestick Charts?
MSFT
CAP
% Change
Hoverability
Just Historical?
Include Dynamic Intra-Day?
How much additional info?
Past
Present...
Ticker Chart?
S+P500
Dark Green/Gainer
Hue color encode?
Dark Red/Biggest Loser
Whole Avg Chart Shown w/ Bubbles?
Allow Scrubbing? Let it then manipulate bubble-chart?
Create an avg line for indexes?
S+P500
Show all line for Sectors?
tie it to bubble chart being displayed
top right
Size bubbles on market cap.

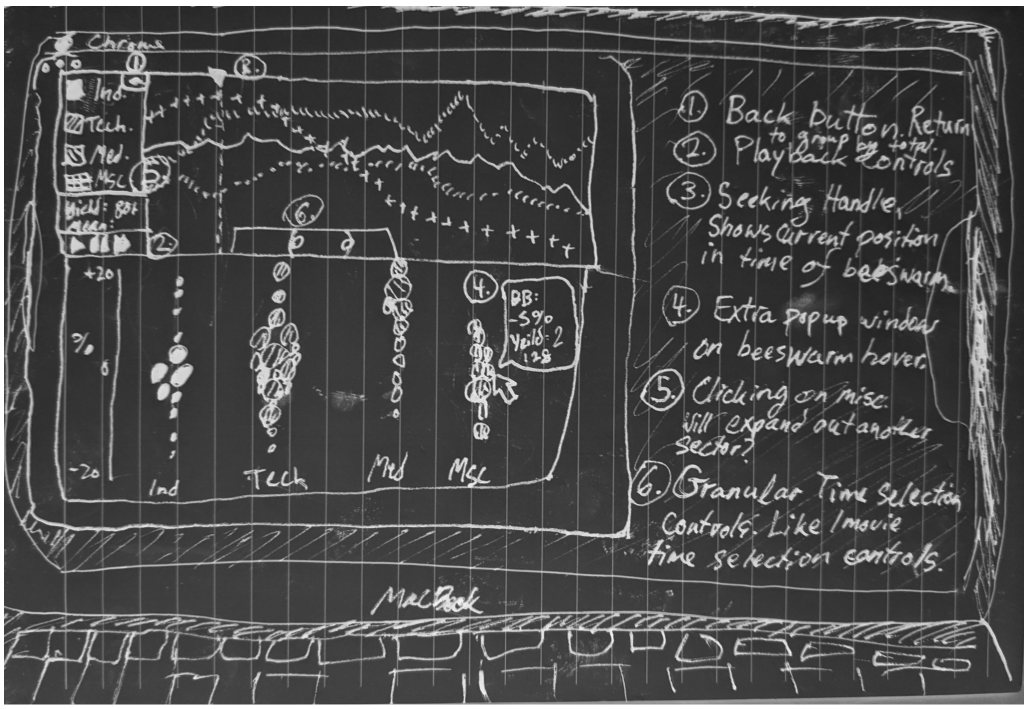
Prototype #2:



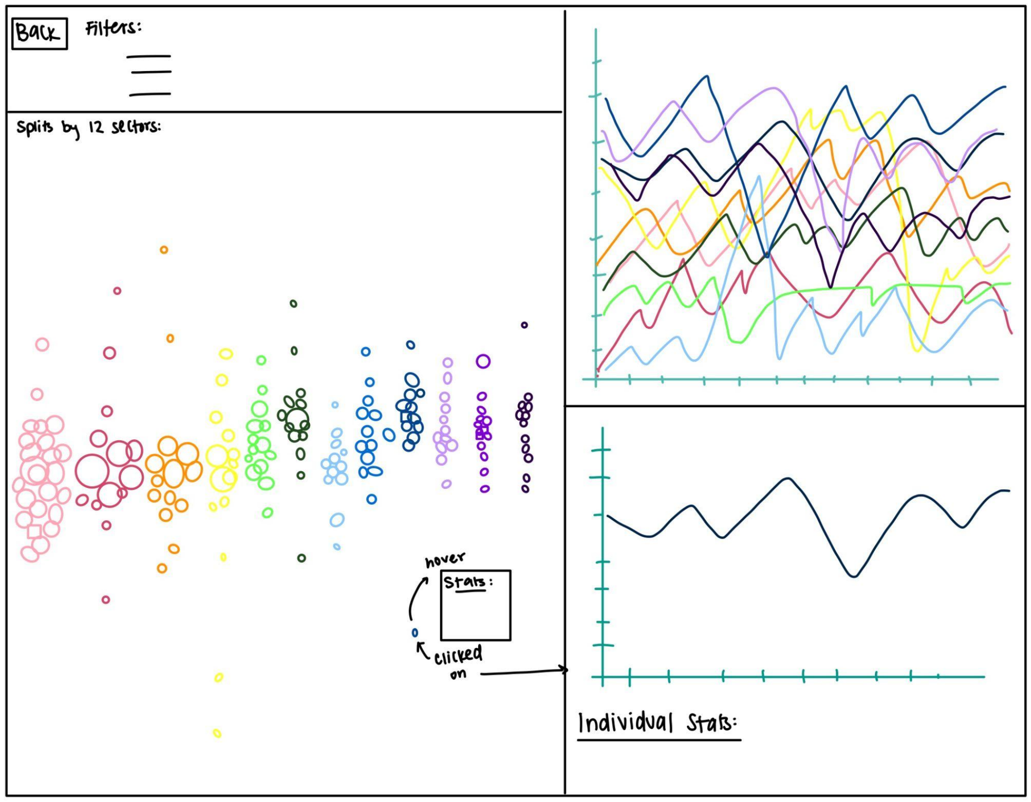
Prototype #3:


Color encoded super-imposed lines = Time
Candlestick Chart = Static Stock
Filter: Market, Index, Sector, or Custom/Import
Time Range from dd/mm/yr — dd/mm/yr
Filter: YTD Returns, Volatility, Volume, Closing Price/Price Change
0
Raw Data Points
unroll show each stock
Ticker/Sect:
Market Cap : 100B
Closing Price : $12.71
PE Ratio/Beta : 31.2
Volume : 98m
Dynamically Change w/time data.
Sectors
Start
Size = market cap

Prototype #4:


Chrome
1
2
3
5
6
4
Ind.
Tech.
Med.
Msc.
+20
%
0
-20
Ind
Tech
Med
Msc
BB: -5% Yeild: 2 128
1. Back button. Return to group by total.
2. Playback controls
3. Seeking Handle, Shows current position in time of beeswarm.
4. Extra popup window on beeswarm hover.
5. Clicking on misc. will expand out another sector?
6. Granular Time Selection Controls. Like Imovie time selection controls.
MacBook

Final proposal sketch:



Deviations

The project did deviate from its initial proposal in a few instances.

It became clear that processing queries directly from the Yahoo Finance API through the visualization would take an incredible amount of time. Rather than have a buggy visualization, the team decided to focus on having a stable dataset over the last year to demonstrate instead. Another option was to load 20 years worth of information at once, but this would require a server and database to implement effectively, costing more time and money than the scope of this project warranted.

Another deviation was in the layout of the charts and tables within the visualization. Initially, the goal was to have the beeswarm in a portrait aspect ratio on the left side of the page with the remaining charts on the right, one above the other. However, due to the amount of information present in the beeswarm, the best course of action was to allow the beeswarm to use the full width of the screen with the other charts above it.

The project was also going to display all stocks individually without sorting by sector. When displayed this way, the goal was to show a gradient between dark red and dark green to represent percent change amongst all the stocks in the beeswarm. This proved to be too messy and hard to parse as a user. Instead, the project now forces selections of sectors in order to display data. In order to see all stocks on the beeswarm, all sectors will now have to be selected.

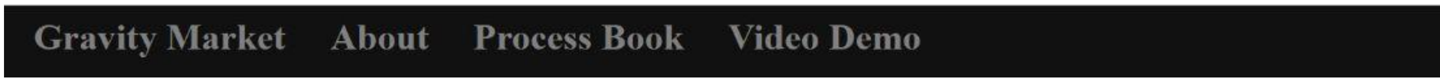
It was not intended that bubbles would be resized based on sector selections, but it became clear that the visualization would present information better if stock bubbles were dynamically resized based on the amount of stocks present on the screen at any given time. This was done by drawing circles based on area instead of by radius. This makes for a much cleaner and intuitive visualization when manipulating the data.

Selecting time intervals by resizing a time bar became a much more intuitive way to select time intervals instead of relying on date drop-downs in order to accomplish the same task. This gives the visualization a more fluid design overall.

Implementation

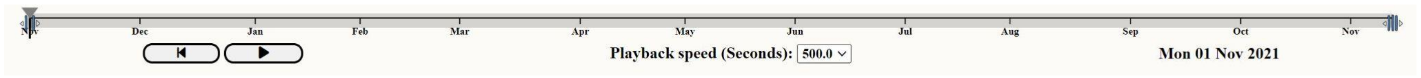
Functionality of the menu:

Along the top of the website, there is an interactive menu bar. If the user clicks on “Gravity Market” the main visualization is displayed. When “About” is clicked, a page appears that has a short bio for the creators of the visualization. When “Process Book” is clicked, a page appears with the process book and the design of the visualization. When “Video Demo” is clicked, a page will appear with a Youtube link for the video demo of the visualization.



Functionality of the playback bar:

The playback bar allows the user to see the data over time, without having to manually scrub. The user can set their playback speed in seconds by selecting from the drop down menu at the bottom-center of the screen. The user can then press play, and the simulation will start. The user will see movement in all three graphs as time goes on. To pause the simulations, the user can click the pause button. The playback bar also has a scrubbing feature, where the user can drag the bar across the timeline to observe a specific day.

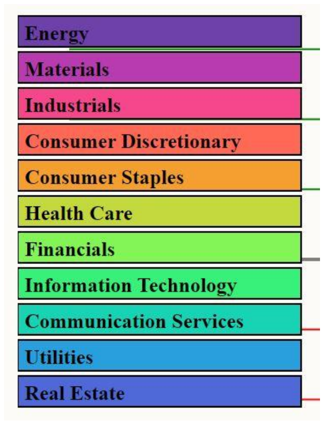


Playback speed (Seconds): 500.0

Mon 01 Nov 2021

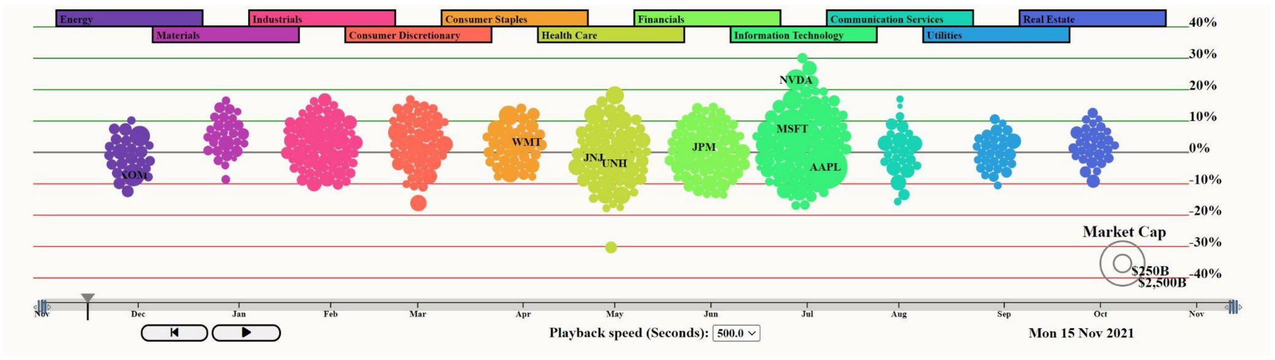
Functionality of the Sector selection bars:

The sector selection bars on the left side of the screen allow the user to click on which sectors they want to see in the visualization. The color of the bar represents the color the sector will be across all the different graphs. When a bar is clicked, the sector is added to the beeswarm and line chart, with the button moving over into the beeswarm above the corresponding circles. To deselect a sector, the user must click the bar again, and it will go back into the list on the left.



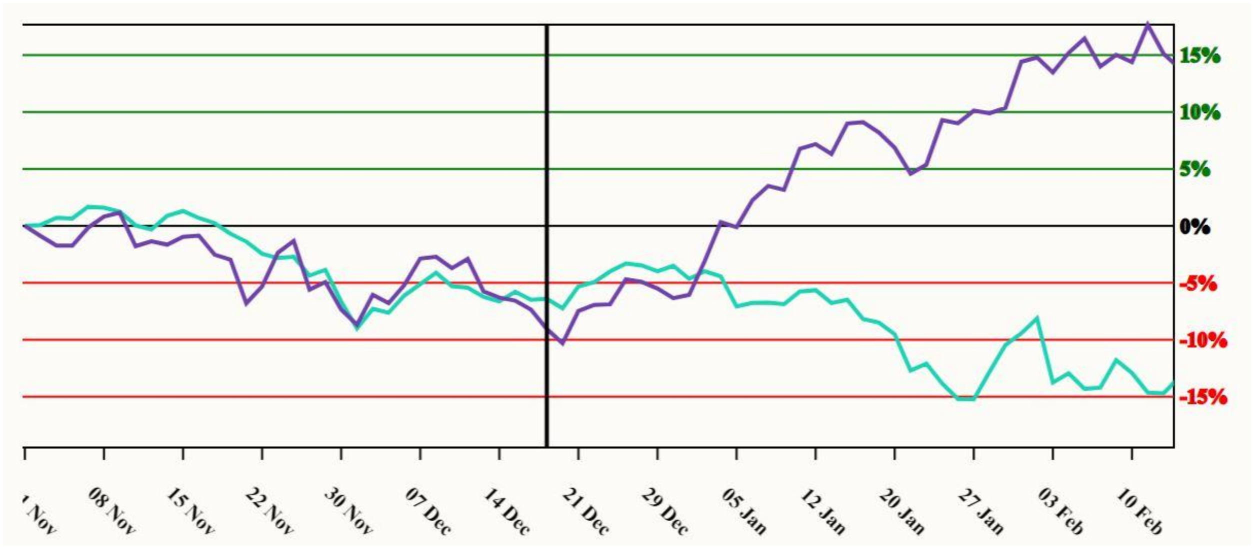
Functionality of Beeswarm:

The beeswarm contains a bubble for each company in our dataset, and is the main feature of our visualization. When a sector is selected, the bubbles for the companies within that sector appear. The size of the bubble represents the market cap for the individual company. The y-axis of the beeswarm chart represents the percent change of the stock for the given time, both positive and negative. Multiple sectors can be chosen so the user can compare them. When the playback is started, the bubbles move to the percent change for the given day, and continue to change as time goes on. Hitting the spacebar starts and stops this visualization from playing.



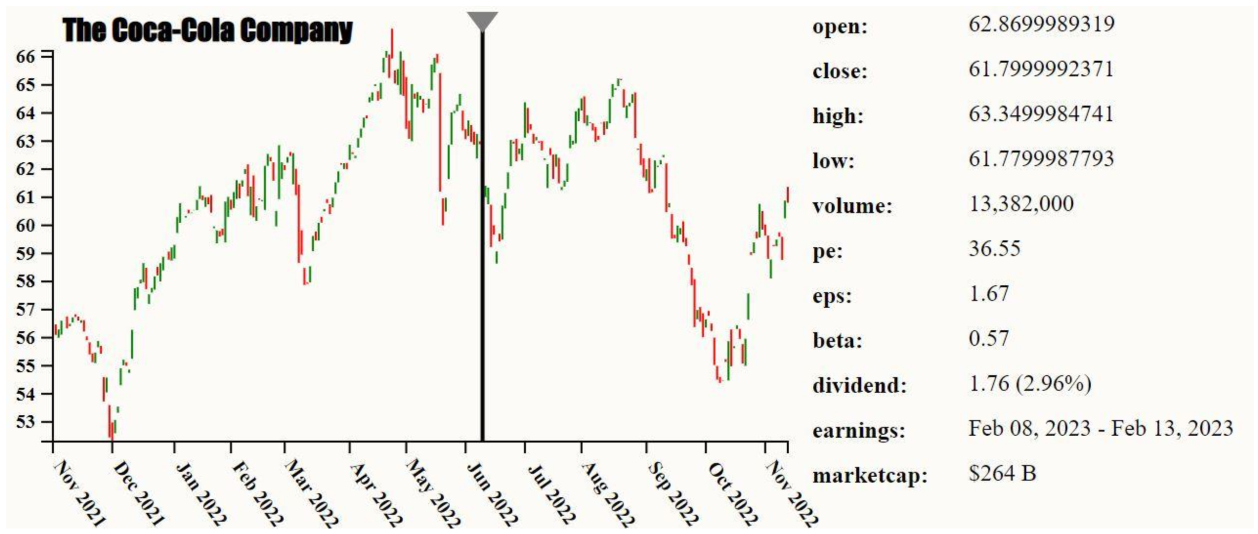
Functionality of Line Chart:

The lines on the line chart represent the weighted average by market of the percent change for the entire sector. The x-axis represents the date, and the y-axis represents the percent change, (same as the beeswarm). Once a sector is selected from the menu, the line chart displays a line for that sector. Along with the beeswarm chart, multiple sectors can be selected and deselected. When the playback is started, a scrubbing bar appears on the line chart to indicate what day the bubbles are representing.



Functionality of OHLC and Table:

The implementation of our version of an OHLC chart allows the user to see the opening and closing prices of each day for the specified company. When a bubble is selected in the beeswarm chart, that company's data is displayed in the OHLC chart. Each day is represented by a line, with the endpoints representing the open and close prices of the day. The line is red if the stock for the company went down in price at the end of the day, and green if the price went up. When the playback is started, a scrubbing bar appears on the OHLC chart to indicate what day the bubbles are representing. The table on the top right of the webpage displays individual statistics for a single company. Similar to the OHLC chart, when a single company bubble is selected from the beeswarm chart, the individual stats for that company is displayed in the table.



open:	62.8699989319
close:	61.7999992371
high:	63.3499984741
low:	61.7799987793
volume:	13,382,000
pe:	36.55
eps:	1.67
beta:	0.57
dividend:	1.76 (2.96%)
earnings:	Feb 08, 2023 - Feb 13, 2023
marketcap:	$264 B

Evaluation:

What we learned: Some of the sectors in the data showed really poor performance a year ago, but throughout 2022 they did so much better than everyone else, (for example, energy) while other sectors flipped (for example, Information Technology). Overall, the market wasn't doing very well for the year of 2022, and it was interesting to see which sectors played a heavy part in that.

Our previous questions answered:

What kind of outliers fuel or drag historical market movements?/Do market sectors push or pull the market in specific directions?

-The biggest outlier was the energy sector, however it wasn't enough to keep the market afloat. A drag in the market was the IT companies, due to their high market cap, the negative percent change had a larger impact on the market overall, dragging it downwards.

Does representing the stock market in this way encourage more curiosity about why and how the market is moving the way it is?

-Yes, this visualization allows the user to look deeper into the values of the stock market and how the market is affected over time. The individual charts give the user the ability to compare different sectors and point out outliers over time.

Our visualization works very well, and effortlessly shows the change over time of the stock market. Each chart updates dynamically depending on the user's interaction. Our visuals are clean and evenly spaced, with the page compact enough that everything can be seen, while still spaced nicely.

One way we would improve our visualization is the data itself. We would want to have a server to allow real-time data, while allowing a user to go back 30+ years to truly see how the stock market has changed over time. Due to the storage and server required for the real time and older data, this would be a difficult improvement, but worth the effort.