Gravity Market Proposal

Basic Info

The project title for this proposal is Gravity Market. This project will be completed 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

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.

Data

The data for this project could come from a variety of different places, which could require a considerable amount of data wrangling to format it to this project. For the core must-have features, there needs to be enough historical market data going back at least a couple decades. Considering what market data to represent, the three most widely followed indexes in the U.S. are the S&P 500, Dow Jones Industrial Average, and Nasdaq Composite. Using these make the most sense as they are widely known and contain the most recognizable companies by the general population. The must-have for this project is a working S&P 500 index, however additional indexes would optionally be interesting to include. The Yahoo Finance API (https://python-yahoofinance.readthedocs.io/en/latest/api.html) is a solid contender for these large amounts of historical data. These data points are not historically intraday, but do provide enough useful information on a day-by-day basis going back many years to populate our visualization with interesting historical trends.

Ideally this data will contain enough information about sectors to allow us to organize companies by sector. If not, sector data will need to be inferred through another data set in order to tie each company listed in the indexes to their relevant sector. The best dataset that comes to mind would be the Standard & Poor's indexes across 11 different sectors. Finding the company listings in these respective sectors will overcome this issue.

In the event the optional feature of real time streamed updates to facilitate visualization of intraday trading is deemed appropriate there are several options available to us. The Yahoo Finance API is one option, but it delays results by 15 minutes, making real-time representations difficult. If it seems like the delay can be dealt with for the sake of cost constraints, it should be possible to get minute-by-minute updates with the Yahoo Finance API. Otherwise Alpha Vantage API (https://www.alphavantage.co/documentation/) has an API that could serve real-time data, however utilizing it would require an upfront cost that may or may not seem appropriate.

Historical data would not contain intraday metrics beyond the prior two weeks, as these would likely take an incredible amount of time to populate depending on the historical range requested by the user. Whatever datasets we end up using will most likely be manipulated to a csv that can be queried directly for sake of speed.

Data Processing

For now, we don't want to have to perform substantial data processing in d3. We have yet to find a perfect dataset which has exactly the columns at the time series granularity that we need. As we start coding our project, we want to start with a pre-processed csv 'toy dataset' that we have prepared using other applications, perhaps extracting this toy dataset using pandas or with database operations. To obtain this dataset, we would first obtain a real world stock price dataset, which would be about 1-10GB. This dataset would contain stock ticker, price, and date and time. From this large dataset, we would extract the rows corresponding to the particular time-series granularities we wish to test in while developing our app, within a certain range of time. We expect we will have to add a column representing the sector. This extracted dataset should be in the tens to hundreds of megabytes.

Starting out, we want the code for our application to be focused on rendering a complete and cleaned dataset. We have some additional features described which elaborate on what additional data processing we could add and what features it would add to our application.

Select ticker to control bubbles size or sector to control bubble size

Visualization Design

Our data will be displayed via multiple graphs that highlight different aspects of the data. The first main visualization will be a bubble chart that displays the market cap of each company through the size of the bubbles. Depending on the filters selected, the bubble chart will organize the bubbles by sector by moving them and changing the color.

The second main visualization will be a line chart that displays the average change of all the companies within the bubble chart. For instance, for the S&P 500 the line chart would show the increase or decrease of the entire index, while the bubbles would represent the change within that index. With more filters, the line chart will split into the corresponding sectors, and color coordinate based on the colors of the bubbles in the bubble chart.

To show individual company data, when the bubble is hovered over by the cursor, a small box will appear with the individual stats. When the bubble is clicked on, the corresponding stock is selected. That click should then populate a new line or candlestick chart beneath the line chart showing the average change of the current index that shows the relevant stock's metrics.

The first image is the brainstorming page.

The second image is prototype #1, which incorporates the bubble chart, line chart and a secondary line chart to display the individual company data.

The third image is prototype #2. This prototype includes the bubble chart, the line chart that displays the average change for all the companies over the time period. Different from the prototype #1, the individual company data would be displayed with text and statistics, rather than a secondary line chart.

The fourth image is prototype #3. This prototype shows the effect of different mouse actions on the visualization. Hovering over beeswarm spheres reveals more information about the stock. Timeline controls are shown, and the playhead position is prototyped. No secondary line plot is represented in this visualization; information that would be shown from this line plot is supplemented by the popup hover window.

The fifth image is our final sketch. This final sketch incorporates the bubble chart and the two line charts from the prototypes. It includes the average change for all the companies over a time period. This sketch is our final goal to provide good visualizations both for the data as a whole, and include individual-based visualization.

Must-Have Features

A bubble chart that displays the market cap for each company by size. The chart will have an option that divides the bubbles by sector, with a y axis that displays the percent change of each company represented in the bubble chart. When a cursor hovers over a bubble (stock) in the chart, a small window will display the name of the company, its market cap, and its percent change.

There must also be a way to pick a historical date range and display the changes occurring over the stated range for the selected index. Once that range is selected, the chart should then animate these changes from the beginning of the date range to the end. While this animation is happening, the line chart displaying the average change of the entire index will update with the corresponding range as well. Should the bubble chart be sorted by sector instead of an index, the line chart will have a line to demonstrate the average change for each individual sector instead of a single line for the entire index.

Some type of back button will be needed in order to 'undo' actions in the chart. For example, if a user splits an index by sector, the back button would allow the user to go back to seeing the index without being split by sector. If split by sector, the bubble chart will change color to represent each sector as well as split along the x axis to clearly see which stocks belong to which sector. If the bubble chart is not split by sector, it will be color encoded in shades of green and red. Any stocks with a negative rate of change will be colored in shades of red, with the biggest losers displaying a deep red color. The same can be said for green when there is a positive rate of change, with the highest earners shaded in a deep green color.

Optional Features

Our data and our visualization could update in real time, streaming market changes as they happen. A way we could implement this as a feature which is intuitive to use, we could make the playhead 'stick' to the far right, the far right being the actual current moment in time. When this happens, the playback will enter a special unpaused state. When in this state, we could attempt to stream in real time price changes, append them to our data, and then update our d3 simulation forces. Some negatives to this; it would be complicated to stream in more data. It would be slightly difficult to make the playhead stick to the far right at a certain threshold.

A feature that would not just be flashy, but also might reveal trends in the beeswarm data would be smooth timeline scrubbing. Timeline scrubbing would allow users to sweep the playhead across the timeline, while updating the forces in the beeswarm visualization such that the bubbles will represent, (as close as possible while also maintaining smooth physics), the distribution of percentage point change based on market-cap. Ideally, our beeswarm should approximate a violin plot representation of this distribution at any point in time. While scrubbing, we must carefully update the global force scaling so that the bubbles aren't too 'floaty', because we want the beeswarm to approximate the actual market-cap percentage change distribution at that particular point in time. Higher global force scaling will reduce the 'floatiness', but also make the plot not interpolate changes to the distribution as smoothly. Updating the forces based on the current playhead location must be done at every tick, not at every mouse movement. If updating how d3 calculates forces is expensive, then the scrubbing will be expensive.

Project Schedule

Make sure that you plan your work so that you can avoid a big rush right before the final project deadline, and delegate different modules and responsibilities among your team members. Write this in terms of weekly deadlines.

By Friday, Oct. 21st:

Submit Final Project Proposal.

By Friday, Oct. 28th:

Collect and format data for javascript. Have basic starter code structure implemented. Have our plan for what will be in the global application state, what events our application needs, what shared state we need to have. Also develop basic HTML and CSS for populating various charts to the visualization, and styling elements of our visualization. Figure out how we want to partition different elements of the application between our group members.

By Friday, Nov. 4th:

By this point we want to have the beeswarm correctly calculating physics for a hardcoded video playback. We will probably hardcode the time series playback to loop at a set speed. The playhead should be correctly rendered on the line plot. Ideally we will want the line plot to also be rendering correctly. Scales on the beeswarm and on the line plot should be dynamically adjusted appropriately. Performance concerns that we may have to work on should be apparent by this point; we are not adding many features to the project after this point which increase the load significantly.

By Friday, Nov. 11th:

By this point we want to implement the functionality of the 'group by' feature that we want. By interaction with the main line plot, users can adjust which stocks are displayed in the beeswarm. Grouping by sector correctly splits up the beeswarm into multiple x axes. This also should expand the line plot into multiple lines, each representing a sector. By clicking on the main aggregate line in the line plot, no matter the depth of the current group by, the visualization should return to plotting as one large beeswarm and one main aggregate line. Clicking on a sector line should expand to showing only that sector and the main aggregate line, the beeswarm only showing that sector.

Milestone 1 due

By Friday, Nov. 18th:

By this point we want to have refined the UI playback controls. Through the user interface, we want to have the time range selection working, and the play, pause, and speed selections implemented. Also, we should have the detailed sub line plot window correctly showing select details of the selected stocks/groups.

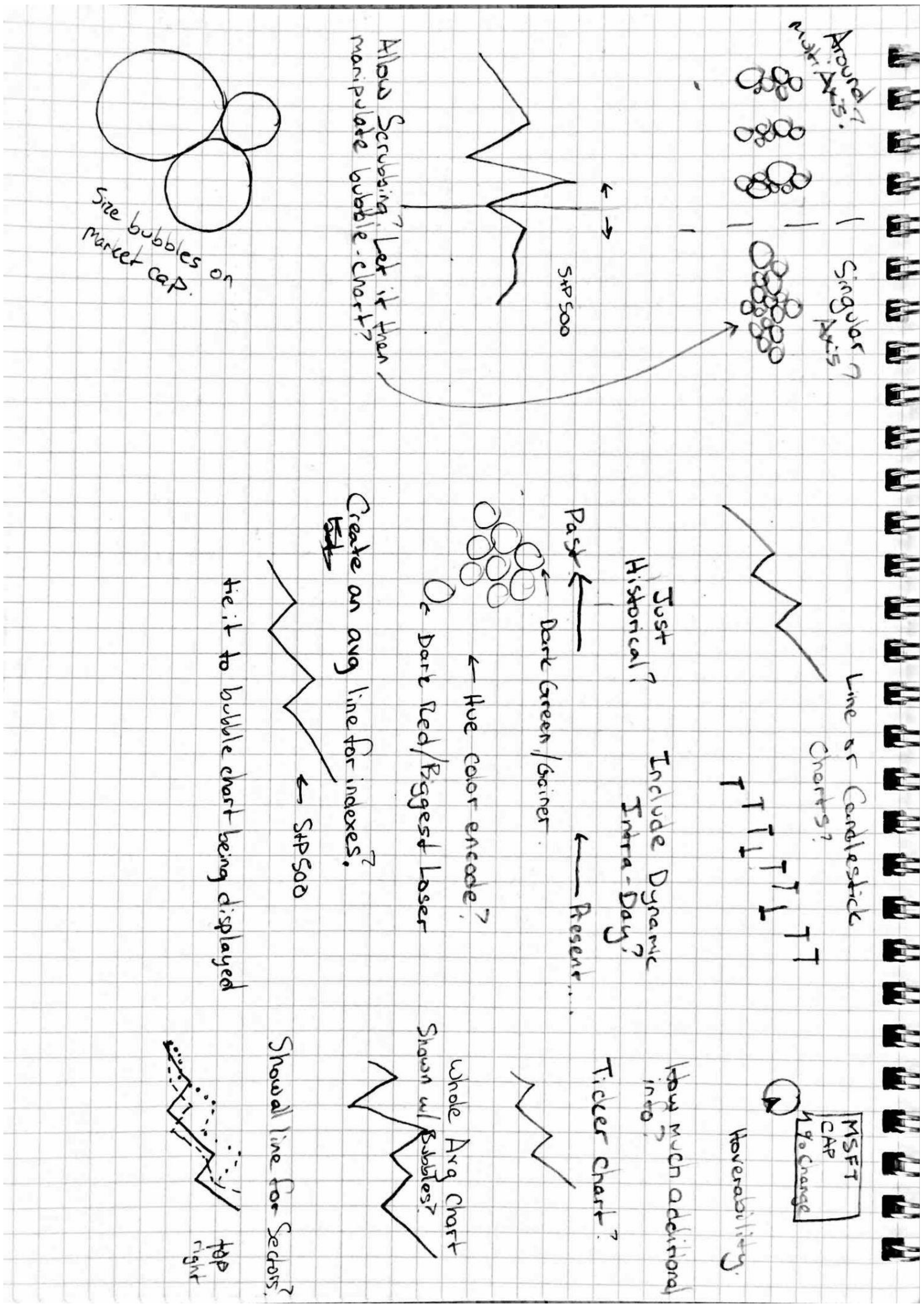
By Wed, Nov. 23rd:

By this point, we should work on smoothing out current controls. We want to refine how we load data into the application, perhaps we can work on one of our extra features.

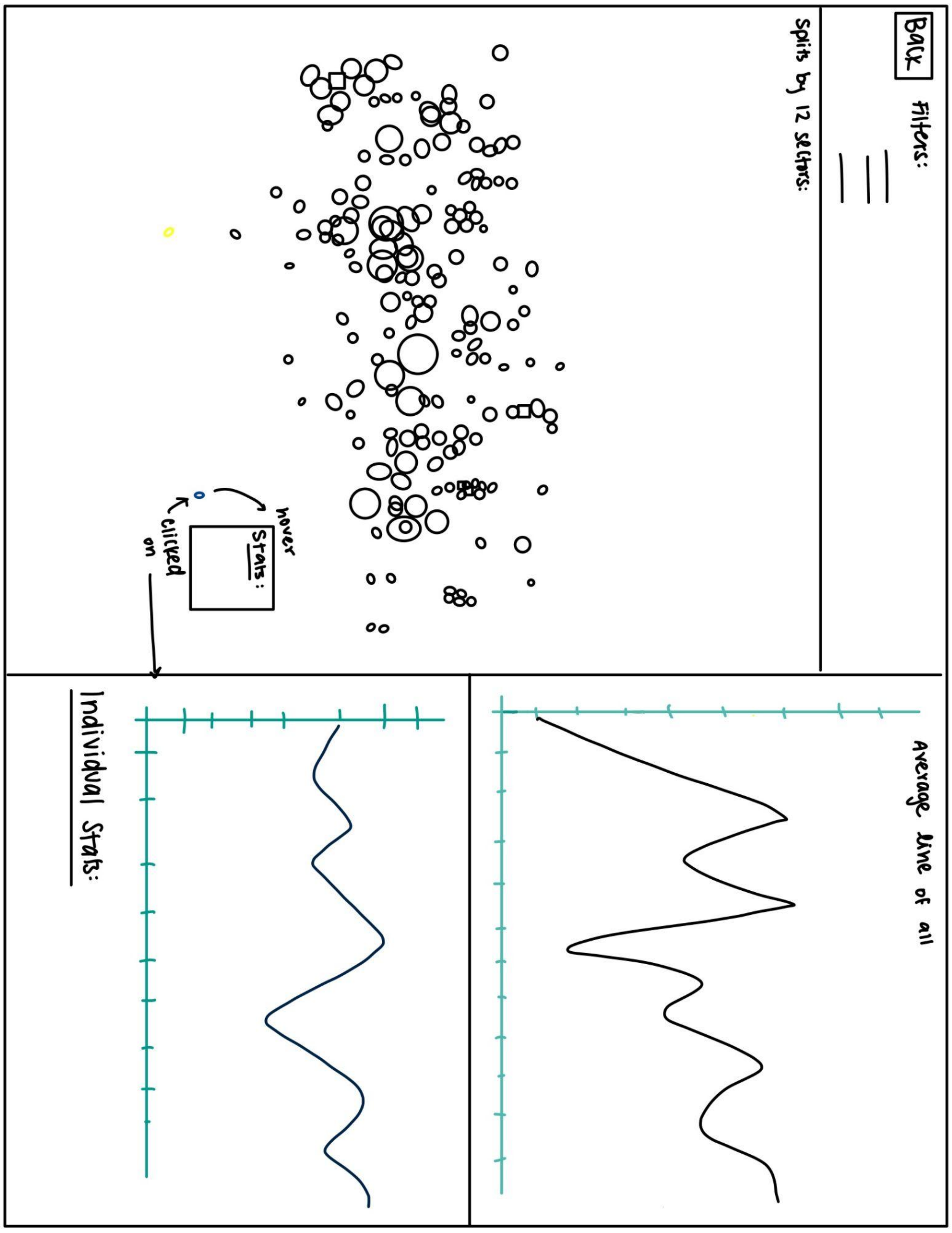
By Friday, Dec 2nd:

Figure out how to host our application, and turn in the Final Project.

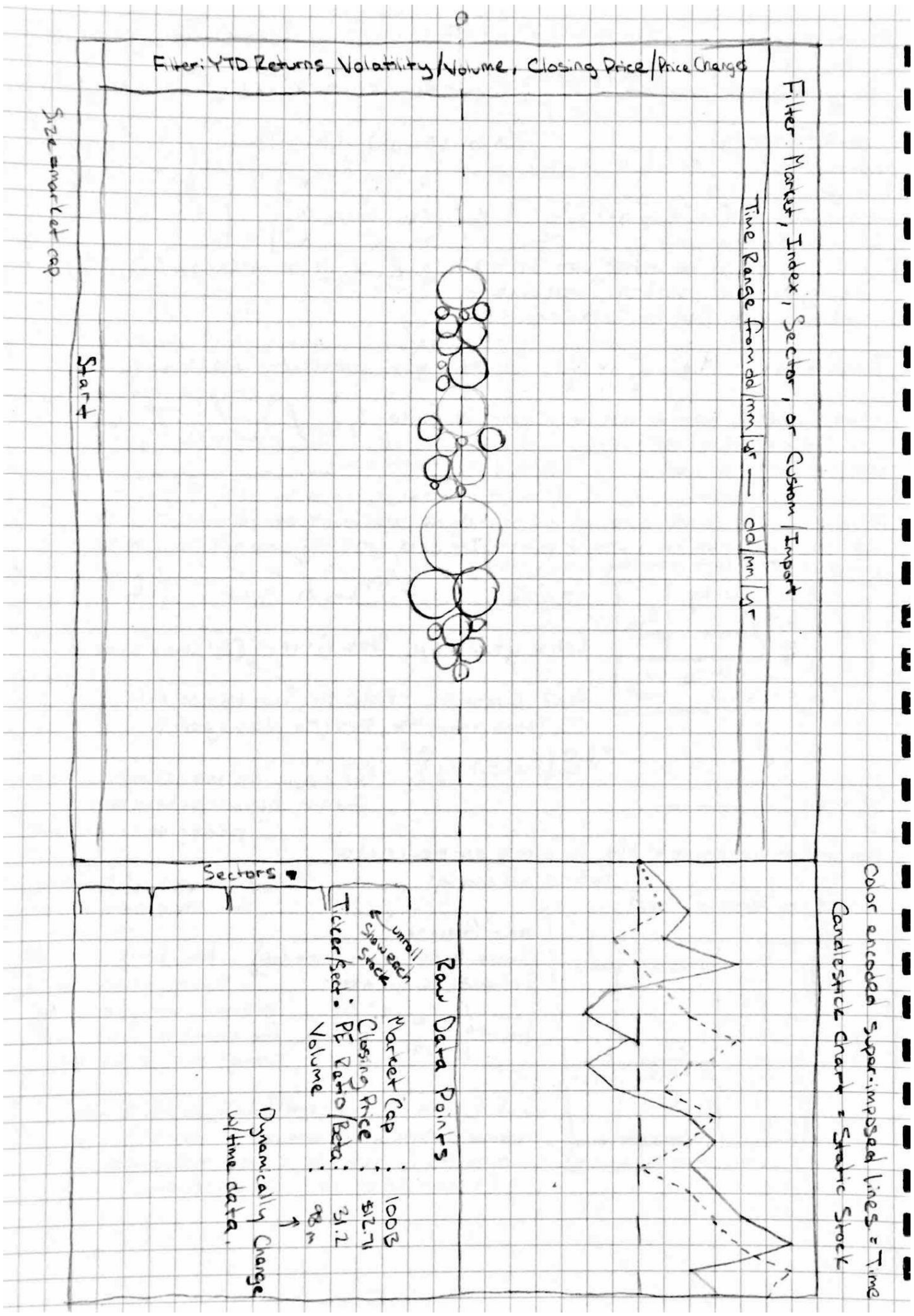
Brainstorm:


Around? multi Axis.
Singular Axis?
Allow Scrubbing? Let it then manipulate bubble chart?
S&P500
Size bubbles on market cap.
Line or Candlestick Charts?
Just Historical?
Include Dynamic Intra-Day?
Past
Present...
Dark Green/Gainer
Hue color encode?
Or Dark Red/Biggest Loser
Create an avg line for indexes?
S&P500
tie it to bubble chart being displayed
MSFT
CAP
1% Change
Hoverability
How much additional info?
Ticker Chart?
Whole Avg Chart Shown w/ Bubbles?
Show all line for Sectors?
top right

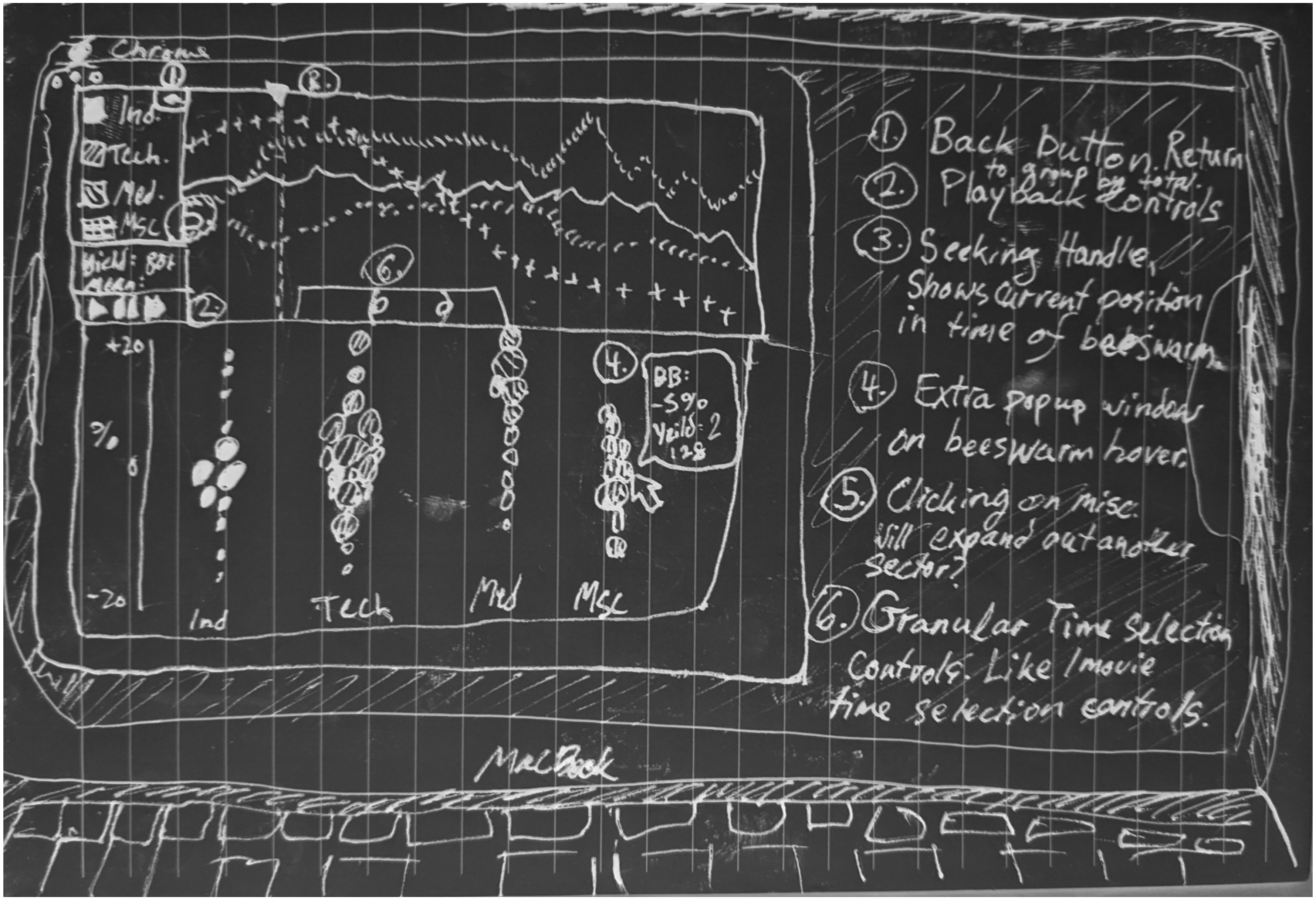
Prototype #1:


Back
Filters:
Splits by 12 sectors:
hover
Stats:
clicked on
Individual Stats:
Average line of all

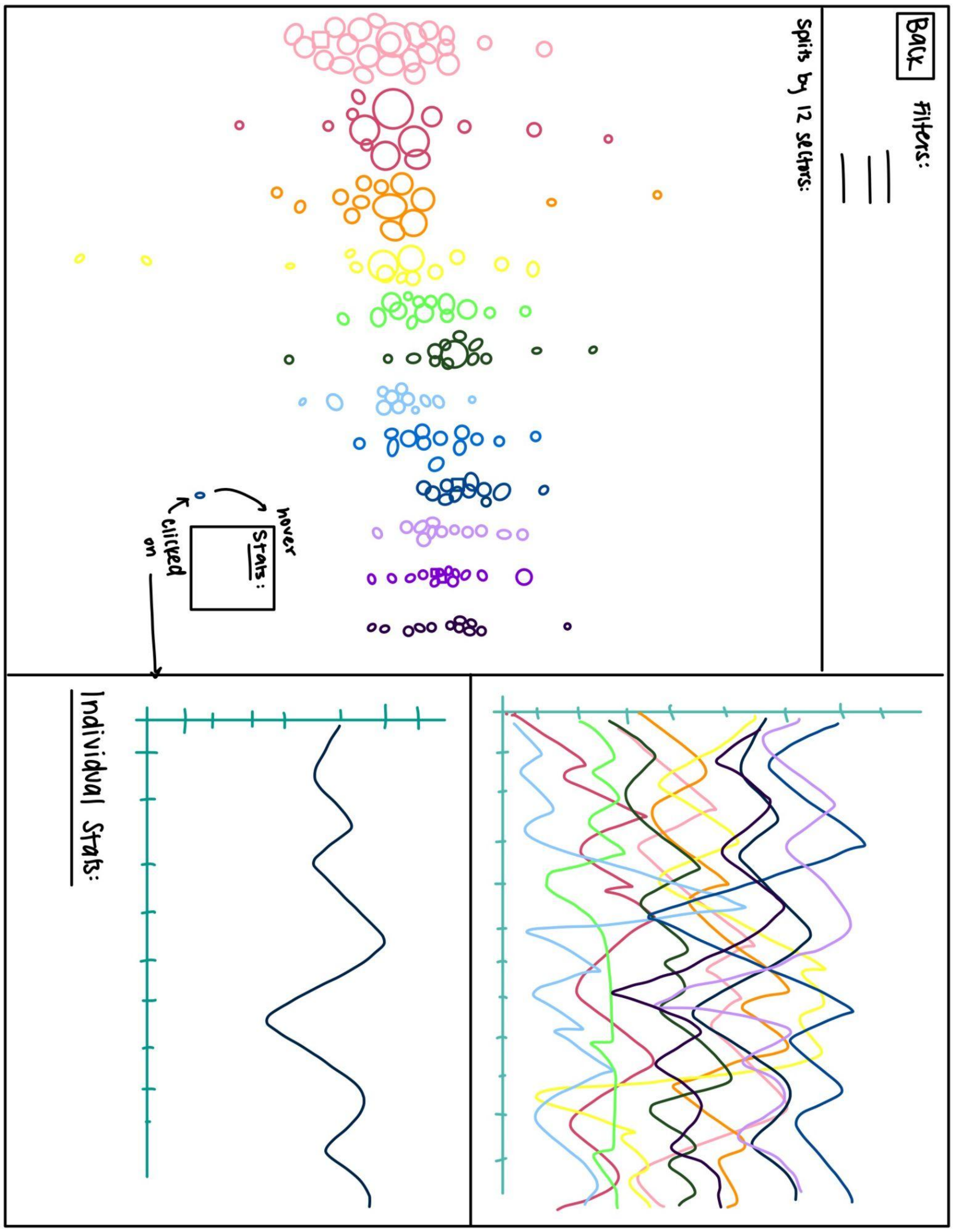
Prototype #2:


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

Prototype #3:


Chrome
Ind.
Tech.
Med.
MSC
Yield: 80%
+20
%
0
-20
Ind
Tech
Med
MSC
DB:
-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 Sketch:


Back
Filters:
Splits by 12 sectors:
hover
Stats:
clicked on
Individual Stats: