**Diversified Stock Portfolio Creation with K-Means Clustering**

Matthew Dyer

December 10, 2021

CU Boulder APPM CE

**Introduction**

The stock market is infamously esoteric. Financial experts dedicate their entire lives attempting to understand the stock market’s complexity and build the most profitable portfolios. Stocks are mercurial and difficult to predict. Many critics of the stock market lament that investing is not accessible to much of the general public due to lack of surplus funds and lack of financial expertise. In this statistical analysis, I hope to make investing more accessible to the layperson such as myself. Rather than financial domain expertise, I used unsupervised machine learning to create a practical stock portfolio.

The golden standard of selecting a stock portfolio is diversification and mitigating risk. The general idea is to invest in disparate stocks so that when one sector of connected stocks fails, the loss will be hedged by the disconnected stocks that are at least somewhat unaffected by that sector’s dip. Investment strategy is ideally risk-averse and low variance. In this analysis, I clustered stocks that perform similarly. This clustering formed groups that act as sectors of stock. Therefore, an investor could create a diversified portfolio simply by investing in a variety of these clusters. I used unsupervised machine learning rather than pre-labelled data for this analysis. In particular, I used K-means clustering for its flexibility and efficiency in clustering.

Clustering, rather than classification, will lead to groups of stocks that are not categorized by name and, instead, categorized purely by how similarly these stocks perform. In this way, unsupervised learning has a few ideal advantages. First of all, different sectors of stock are already well-established. According to the Global Industry Classification Standard, there are 11 stock market sectors (https://www.fool.com/investing/stock-market/market-sectors/). These sectors include energy, materials, industrials, utilities, healthcare, financials, consumer discretionary, consumer staples, information technology, communication services, and real estate. My objective is to find novel sectors that could be subsets of these established sectors, blends of multiple sectors, or even a disparate menagerie of stocks that cannot be neatly categorized into a sector. Unsupervised learning is not restricted by labels and clustering can discover interesting connections between seemingly disparate companies. For example, some biomedical companies could focus on medical robotics and be more connected to the materials or information technology sector for their shared reliance on microchips and the precious metals needed to develop them. However, these biomedical companies could be erroneously categorized as healthcare even though their financial performance is associated with the other sectors. Another advantage of unsupervised learning is that classification relies on labelled data. The user would need to have financial knowledge or access to domain experts to correctly categorize the data. However, this project is aimed for the layperson who has no financial expertise and just wants to know which stocks perform well together and not necessarily why those stocks perform well together. Finally, I am using unsupervised learning for its relative computational efficiency. I am using a dataset with millions of data points and thousands of features. Computational efficiency has paramount importance handling such a large dataset.

**Data**

The dataset I am using is the “AMEX, NYSE, NASDAQ Stock Histories” from user Jiun Yen on Kaggle (<https://www.kaggle.com/qks1lver/amex-nyse-nasdaq-stock-histories>). This dataset is a .csv file that is 3 GB unzipped and roughly 600 MB zipped. It contains the stock prices of 6335 different stocks scraped from the NYSE, NASDAQ, and AMEX on Yahoo Finance. These data include basic financial indicators on each stock over a roughly 5 year period with 1385 unique dates ranging from January 2nd, 2015 to July 2nd, 2020. The 6,852,038 observations in the dataset correspond to the stocks analyzed on different days. The 8 variables correspond to the date (year-month-day), stock symbol, volume, opening price, closing price, highest price of that day, lowest price of that day, and the adjusted closing price of that day.

**Data Wrangling**

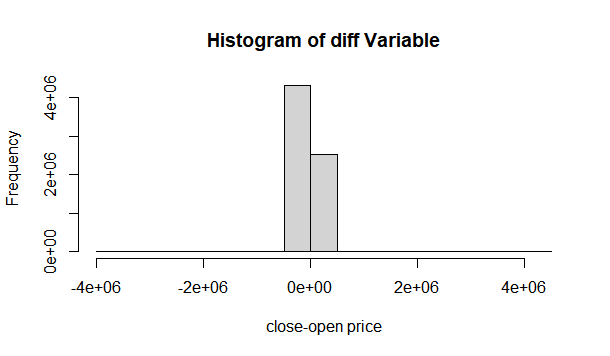
Several of these variables are closely related and pose an issue of multicollinearity. For example, ‘close’ and ‘adjusted close’ will always be close to the same. My analysis has a primary focus on how the price of the stock changes. Therefore, the volume variable is superfluous. Measuring the movement of the stock poses a challenge in concept validity. There are several different options to operationalize stock movement that end up interpreting ‘movement’ differently. The first option is the scope of the timeframe. The dataset includes individual days but I could aggregate the data to measure movement in weeks, months, etc. After all, two stocks could increase slightly each day while one stock falls drastically at the end of the week for a net loss for the week. These stocks shared 6/7 days of stock movement and seem to increase together on a daily scope. However, these stocks have opposite movement on a weekly scope. I am using a daily scope for my analysis to provide the flexibility for daytraders who operate on a daily scope and to make use as many datapoints as possible so that my algorithm is as accurate as possible. Another major issue in concept validity is how to measure the movement. I could use the difference between the stock’s opening and closing price or just indicate whether the stock increased. For example, two stocks could increase every day but one may increase drastically every day while the other has modest gains. Then the stock that increased drastically could decrease slightly the following days while the stock that originally increased drastically also decreases the following day, albeit drastically. These two stocks moved in the same direction each day but one stock ends much higher than the other stock.

First, I considered tracking the magnitude of each stock’s movement. I subtracted each stock’s closing price from its opening price and I used each day as a variable in my model. Since volume does not measure stock movement and the other variables are redundant and collinear, I discarded them from the model and only consider the new movement variable. Checking the summary statistics of this variable indicates a significant issue.

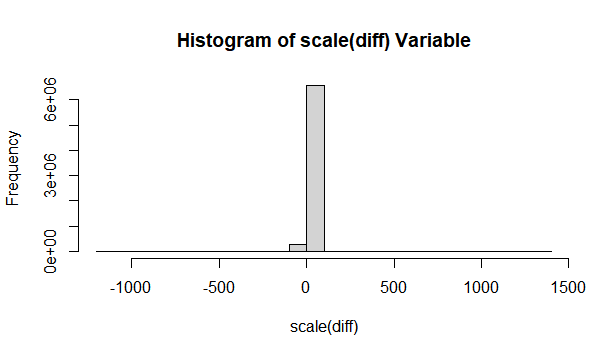
Min. 1st Qu. Median Mean 3rd Qu. Max.

-3583180 0 0 -1 0 4299812

These 0 values are actually rounded indicating most of the stock movements are small. The maximum and minimum indicate that there are massive outliers in the data. This is especially problematic in K-means clustering, which heavily weights these outliers and potentially produce clusters of individual stocks. The histogram further displays the issue in data spread.



The vast majority of the data hovers around zero but the exponential scale of the x-axis portrays a problematic overdispersion. This variable is not suitable for K-means clustering. A common practice in K-means clustering is to normalize the data to prevent unbalanced data from affecting the model unfairly. After normalizing this ‘diff’ variable, the data produce the following histogram:



The scale of the x-axis has dropped dramatically but the data does not look even close to normal. The maximum of the scale(diff) variable is 1334.439 which is still magnitudes larger than the majority of the data. These data simply do not work in K-means clustering. The best option appears to be creating a dummy variable that indicates whether the stock increases that day while disregarding the magnitude. I created a variable on movement that I assigned to 1 if the stock increases that day and assign to 0 if the stock decreases or stays the same that day. Therefore, this has a significant impact on concept validity. I grouped stocks based on the direction of their movement and not necessarily the magnitude of their movement. This variable transformation results in the following table:

0 1

3655136 3196902

There is a relatively balanced spread and there are no outliers by design. Next, I changed the data from long format to wide format by using each day as an individual column. This poses a new issue as each stock does not have data for each day in the dataset. As a result, this transformation leads to a massive introduction of missing data. In fact, there are 1,921,937 NAs in the new transformed dataset. This is too many datapoints to consider deleting the rows or columns containing missing data. Instead, I used the average stock movement for each day and replace that value at each NA. Since the stock market largely moves in the same direction, this is a reasonable handling of the NAs. Now that the proper variable is transformed and the missing data is replaced, the data are ready for analysis.

**K-Means Clustering**

I am using K-means clustering to group the data. The overall idea is to cluster the data based on intra-cluster similarity and inter-cluster dissimilarity. The main objective is to create clusters containing data similar to each other while different from the other clusters. The algorithm is initialized by determining the number of clusters (indicated as k). This step is arbitrary and the selection of the number of clusters will be assessed in the next section. Next, k centroids are determined randomly. I used the kmeans() function in r which splits the data randomly into k clusters and uses the average of each cluster as the starting centroid. Next, each datapoint is assigned to the closest cluster. The closest cluster is the cluster with the centroid closest to the datapoint. This measure of ‘closeness’ is the Euclidean distance in my analysis. This distance is calculated by taking the observed movement of that stock for each day and subtracting it from the mean movement of that day in the centroid. This value gets squared and then added to the resulting squared difference for each day. After each datapoint is assigned to a cluster, new centroids are calculated based on the average of each datapoint in that cluster. This algorithm then repeats until it reaches equilibrium where no datapoints change clusters with each iteration. I limited the number of iterations to 10 for significant calculations such as the elbow method in the next section. Overall, it is a simple and efficient algorithm that is effective with large intractable datasets. Therefore, it is an ideal unsupervised machine learning technique for my analysis.

**Model Selection**

As stated above, K-means clustering relies on the number of clusters to be specified. I could choose a k of 11 to match the number of sectors but I am more interested in finding novel clusters of stocks. I selected the number of clusters by using the Elbow Method. The ‘within sum of squares’ (WSS) is the sum of squares error of each datapoint with its respective centroid. The WSS decreases as the number of clusters increases. With one cluster, the centroid is simple the average of all the data and the WSS is the sum of squared errors with this mean. On the other extreme, the maximum number of clusters is the number of datapoints and each centroid would simply be each datapoint. Therefore, the center of each centroid would be the mean of the datapoint it represents exactly and the WSS would be exactly zero. The ideal number of clusters is the optimal compromise between model accuracy and model complexity. As the number of clusters increases, the WSS decreases with diminishing returns. Therefore, the Elbow Method is to plot the WSS as a function of k and identify where the diminishing returns are insignificant enough to warrant the increase in k. In essence, this is the location where it looks like it could be an elbow on an arm.

I evaluated the WSS for models with 2 clusters up to 40 and identify the ‘elbow’ of the graph:

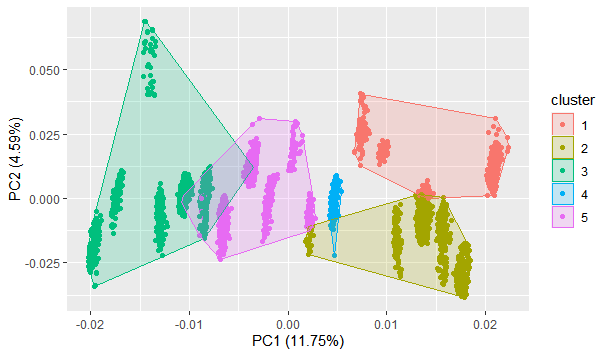


It is difficult to identify an elbow but there appears to be some diminishing returns at the red dot indicating 19 clusters. However, the steady trend in the graph suggests that a more suitable number of clusters may lie beyond 40.

**Methods**

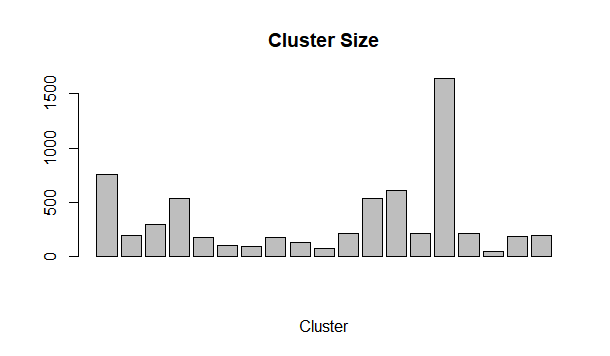
K=5 (illustrative example)

I used the kmeans() function in R to perform the K-means clustering. In order to illustrate K-means clustering I reduced the dimensionality of the data and number of clusters. The autoplot function of the ggfortify library uses principal components analysis to identify the two principal componets that had the highest impact in determining the clusters. This allows a 2D graph to be rendered to display high-dimensional data. It is important to note that I used over 1000 dimensions with each dimension being a day. Therefore, reducing dimensionality to the 2 principal components is mostly for visual purposes rather than practical purposes. Likewise, I used 5 clusters for the first model even though 5 clusters clearly produces an unreasonably high WSS as shown in the above elbow method plot. After all, reducing the stock market to 5 clusters seems dubious. However, for illustrative purposes, it is worthwhile to use 2 principal components and 5 clusters. The algorithm, at its basic level, produces literal clusters and draws boundaries around these clusters. The following plot is the results of a 5 cluster k-means algorithm on the data after being reduced to 2 principal components:

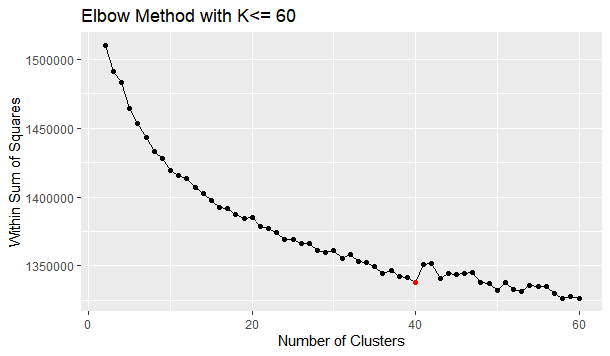


K=19

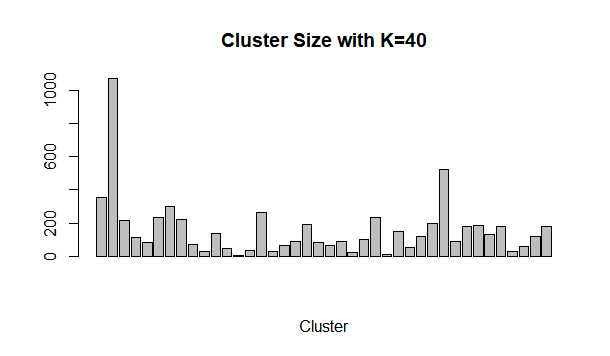
The above elbow method plot indicated that 19 clusters may be suitable but looked like the optimal cluster count is potentially beyond 40. Ideally, k-means clustering separated the data into equally-sized clusters. If k is too small, there will be a wide variation in clusters as the optimal clusters will, themselves, be clustered together in order to satisfy the restraints on k. I performed K-means clustering with 19 clusters and also indicate the nstart argument as 10. This argument performs the algorithm 10 times with different randomly generated initial centroids on each iteration and then selects the resulting model with the lowest WSS. The cluster size distribution resulted in the following bar graph:



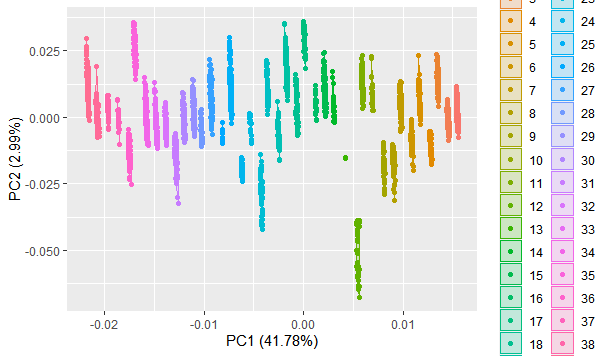
The distribution of cluster sizes further suggests that increasing k is optimal. There is one cluster with 1,640 stocks and three clusters with fewer than 100 stocks. I repeated the elbow method with k up to 60 to reassess a more proper number of clusters.



The graph appears to level out more beyond 40 clusters. In fact, the plot even briefly increases in WSS after 40 clusters. The actual elbow point appears to be at this 40 point cluster point and is indicated in red in the plot above. I repeated K-means clustering with 40 clusters, while keeping the maximum number of iterations at the default 10, and using the best of 10 clusters for each k. This resulted in the following bar graph of cluster size.



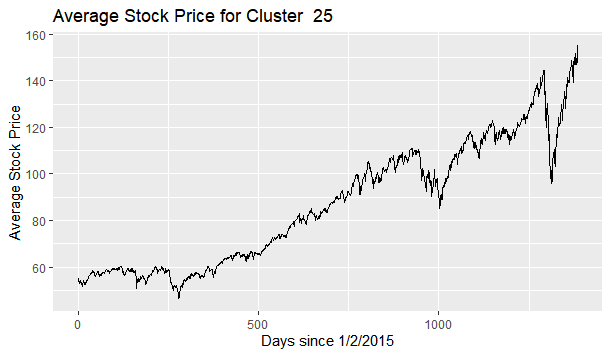
This bar graph appears better but still not ideal. There is still one predominant cluster and several tiny clusters. The biggest cluster contains 1071 stocks and the two smallest clusters contain 2 and 9 stocks respectively. Partitioning into more clusters does not appear necessary with only marginal improvements in WSS. 40 clusters appears optimal despite the distribution of cluster sizes. The resulting principal components clustering plot is shown below.



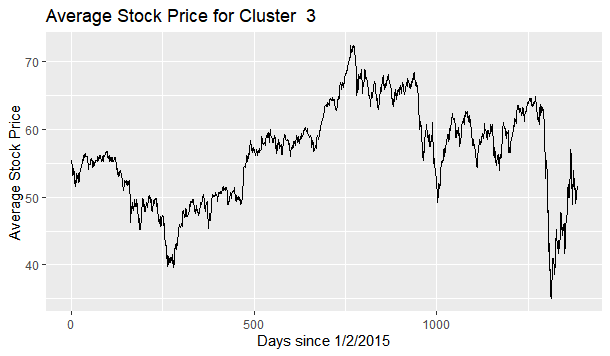
**Results**

In conclusion, K-means clustering created 40 groups of stocks based on their daily movement. However, since there is no labelled data, it is difficult to assess whether the model is useful or accurate. One heuristic approach is to check stocks in the same established sectors and see if they exist in the same clusters. The optimal result would have most stocks in the same sector to appear in the same cluster to show the algorithm captured true relationships. However, it is also ideal not have every stock in a single sector appear in the same cluster since one of the original goals was to discover novel clusters. First, I checked what clusters contained 'AAPL','MSFT','AMZN','INTC','CSCO','NFLX','FB', and 'GOOG'. These are some of the most popular stocks in the information technology sector. These stocks all appeared in cluster 25 indicating that this clustering did capture accurate relationships. Next, I checked popular stocks in the discretionary consumer sector. These stocks included 'AMZN','MCD','NKE','SBUX','TJX', and 'DIS'. The respective clusters are 25,16,27,27,27, and 5. One of the benefits of these results is how best to classify stocks that are in multiple sectors. Amazon commonly appears in information technology and discretionary spending but, according to these results, it behaves more similarly to the information technology cluster. Interestingly, NKE, SBX, and TJX belong to the same cluster 27 but DIS belongs to cluster 5 and MCD belongs to cluster 16. These results show several stocks in the same sector appearing in the same clusters but not all the stocks. These results may indicate that, although McDonald’s and Starbucks are considered fast food, Starbucks stocks behave more similarly to Nike than McDonald’s. This is not necessarily an obvious relationship that K-means clustering uncovered. I also looked at the materials sector stocks 'SHW','DD','RIO','LYB', and 'IP' as well as the industrial sector stocks 'BA','UNP','WM','FDX', and 'MMM'. In both sectors, cluster 3 is predominant. This suggests that the materials and industrial sectors may behave more similarly as a single sector rather than separate sectors. Of course, the clustering is simply based on similar stock movements and does not explain why these stocks move together. However, interesting relationships can be gleaned and further investigated.

With these data, it is also possible to group the stocks in each cluster and average their values. Then a time series can be created to assess the trends in that cluster as a whole. It is important to note, however, that some stocks are worth magnitudes higher than other stocks and can greatly influence the average. However, a general picture can be formed. For example, cluster 25 seems to be the information technology cluster. With these data, we can assess a general trend of these stocks in the past 5 years.



It appears these stocks have been overall increasing and this plot suggests that it is a good idea to buy stock in this cluster. Cluster 3 seems to contain several of the industrial and materials stocks. We can also assess the recent trends in these stocks as a group.



This cluster appears to be more stagnant over the 5 years of data but still appears to be increasing most recently after a drastic dip. Overall, K-means clustering allows us to look at the behavior of similar stocks rather than trying to assess trends individually.

Finally, the ultimate objective is to build a diversified portfolio. I found that an optimal amount of clusters appeared to be 40, so a diversified portfolio can simply be investing in a stock from each cluster. In the code, there is a function that selects a specified number of stocks from each cluster. One sample diversified portfolio using this function is:

"HTHT" "IMAC" "ECHO" "SMMT" "UA" "UEUR" "ONEV" "KTH" "GDX" "XHB" "KOIN"

"SDS" "UGAZ" "ARI" "KAR" "CHTR" "XSLV" "JRI" "VT" "MMT" "CERN" "SPIP"

"UGA" "PNI" "ATVI" "PAM" "KBWP" "EXC" "RSXJ" "HALL" "LINC" "JKS" "MCRB"

"HLX" "PHO" "FDTS" "BAK" "MASI" "RPT" "SCHW"

The ultimate result is a simple method to create a diversified portfolio without expert knowledge in finances. To further evaluate the results, I could create a mock portfolio investing in these stocks and compare performance to a standard index fund. If an investor specified an exact number of stocks to invest in, this K-means clustering could also be run again setting that number of stocks to k.

In conclusion, these results allow novel relationships to be found between stocks, allow analysis of the behavior of these clusters of stocks rather than the individual stocks, and also allow an easy creation of a diversified portfolio without domain knowledge in finances. Next, this research can be expanded by comparing performance of this diversified portfolio with industry standards and a random selection of stocks. The time series plots can also be analyzed to assess trends of the clusters in order to determine which clusters are more lucrative. Finally, this cluster analysis was evaluated using the direction of stock movement. There exists a plethora of stock indicators that can be used to improve the model and cluster these stocks even more accurately.