Principal Component Analysis for Clustering Stock Portfolios

Taylor Agarwal, Henk Quelle, Cooper Ryan

Abstract

Motivation: Due to the sheer size of the stock market and its dependencies on several factors, a catch-all method of determining stock performance continues to elude the public. Recent developments in machine learning have opened the door to new possibilities for a predictive algorithm. Principal Component Analysis (PCA) is one such tool that has allowed for the discovery of hidden interconnectivity in large data sets. We use PCA alongside *k*-means clustering to obtain groups of stocks with similar historical structure with the potential to assist in predictive stock management.

Results: In just over five seconds, our algorithm groups a hundred stocks from the New York Stock Exchange with mean correlation 0.2483 into fifty portfolios with mean correlation 0.4299. On average, the stocks in these fifty portfolios experienced price increases and decreases on the same day 65.34% of the time in the sample time frame of 4/17/2000 to 11/10/2017. The algorithm can be extended to encompass more stocks.

Implications: This algorithm provides a means to identify stocks with similar structure in both the short and long term. Stocks belonging to the same portfolio after clustering are shown to have positive and negative returns at the same time within the user defined periods of time.

Introduction

Principal Component Analysis, or PCA, is a feature detection and data reduction technique that has gained popularity in recent years for analyzing large data sets. PCA has an extensive history in numerical linear algebra dating back to 1901, when Karl Pearson devised the method before it was practically feasible on a computer [4]. Computational advancements since then has allowed PCA to become one of the most widely used data analysis techniques. PCA discovers the *principal components* of the stock data, which are eigenvectors that explain large portions of the variance in the data. The original data is then expressed in terms of these lower dimension features, revealing its hidden structure and cluster points.

From here, *k*-means clustering groups the lower dimensional stock data in portfolios with very high correlation between stocks and a high frequency of identical daily increases and decreases. The first mention of *k*-means clustering was by Hugo Steinhaus in 1956, but a formal algorithm was not presented until 1965 by Edward Forgy [1]. The apparent simplicity of *k*-means clustering conceals its ability to group complex data in meaningful ways. We will take advantage of these two powerful tools to form stock portfolios.

Since the founding of the New York Stock Exchange (NYSE) in 1817, investors have tried to analyze its behavior for financial gain. The NYSE consists of thousands of publicly traded companies ranging from small startups all the way up to massive corporations in the Fortune 500. Each stock is individually affected by a whole host of interwoven factors, including its employees, its management, its ability to sell goods or produce services, its competitors, and the health of the

market as a whole, to name a few. This complexity hides the fact that stocks have connections to each other that are not readily observable to the naked eye. We will reveal these connections.

Some of our portfolios are of no surprise, for instance it is easily imaginable that two health insurance provides like Anthem and UnitedHealth Group should experience similar market dynamics. However, we will see that there are some stocks that are heavily correlated with no discernible reason why, such as telecom company AT&T and fast food restaurant McDonald's.

Methodology

Using a bi-layered feed forward neural network, we identify portfolios of stocks with similar correlation and daily changes. The first layer is Principal Component Analysis, a data reduction technique used frequently on high dimensional data to reduce high dimensional data to a few key components. The second layer is k-means clustering, a labelling technique that is used to group data of similar structure. Linear algebra is used extensively in this paper, so we will discuss our notation.

Notation

$\mathbf{X}$	Matrices, all entries in R
$\mathbf{x}$	Vectors and columns of matrices, all in R
n	Scalar values and elements of vectors, all in R
$\|\| \cdot \|\|_2$	Euclidean Norm, $\|\|\boldsymbol{x}\|\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$
$\| \cdot \|$	Cardinality of a set

We additionally define a validity metric that will be used frequently throughout the paper which we call the Mean Percent of Days or MPD. The MPD gives the average of the percent of days each pair of stocks in a portfolio both increase or both decrease from their respective prices on the previous day. If we define the percent of days on which a pair of stocks $\boldsymbol{a}$ and $\boldsymbol{b}$ with m returns without dividends observations both increase or both decrease to be

$$p_{ab} = \frac{1}{m}|\{i|a_i b_i > 0 \; for \; i = 1, \ldots, n\}|$$

then

$$MPD = \frac{2}{n(n-1)} \sum_{i \in S} \left(\sum_{j \in S \setminus \{i\}} p_{ij} \right)$$

where S is a set of stock return vectors.

Principal Component Analysis

Our PCA algorithm follows that of Cadima Jolliffe [2]. The stock price data can be considered to be a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ where m is the number of *observations* and n is the number of *variables*.

$$\boldsymbol{A} = (\boldsymbol{x}_1 \, \boldsymbol{x}_2 \ldots \boldsymbol{x}_n) \; where \; \boldsymbol{x}_i \in R^{m \times 1} \; for \; i = 1, \ldots, n$$

The original data will consist of m observations of daily returns without dividends on n different stocks. Daily returns without dividends are calculated by

$$x_{ij} = \frac{p_j - p_{j-1}}{p_{j-1}}$$

where p_j is the closing price of the stock on day $j = 2, \dots, m+1$. We can expect that $Rank(A) \leq n$ since this is a stock price dataset, and variations in stock prices are not perfectly linearly related to the prices of other stocks. The z-score normalized data $\boldsymbol{z}_i$ for each stock is used in place of the original data, which for a given stock $\boldsymbol{x}_i$ is found by

$$\boldsymbol{z}_i = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}_i}{\sigma_i} \; where \; \mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}\,, \boldsymbol{\mu}_i = \mu_i \cdot \; \boldsymbol{1}^{m\times 1} \; and \; \sigma_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(x_{ij} - \mu_i\right)^2}$$

where $\boldsymbol{1}^{m\times 1}$ is a vector of size $m \times 1$ with a one in every entry. The value of μ_i for each $\boldsymbol{x}_i$ is stored in a vector $\boldsymbol{\mu} \in \mathbb{R}^{1\times n}$, which will be used later in Section 2.4. Z-score normalized data has a mean of 0 and a standard deviation of 1, making it easier to extract patterns in the data. We now have a normalized data matrix $\widehat{\boldsymbol{A}}$ with new variables $\boldsymbol{z} \in \mathbb{R}^{m\times 1}$ as the columns and new observations $\boldsymbol{y} \in \mathbb{R}^{1\times n}$ as the rows.

$$\widehat{\boldsymbol{A}} = (\boldsymbol{z}_1\, \boldsymbol{z}_2 \dots \boldsymbol{z}_n\,) = (\boldsymbol{y}_1\, \boldsymbol{y}_2 \dots \boldsymbol{y}_m\,)^T$$

Next, we create a covariance matrix from the normalized data.

$$\boldsymbol{C} = \frac{1}{n-1}\widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T = \begin{pmatrix} Var(\boldsymbol{y}_1) & Cov(\boldsymbol{y}_1, \boldsymbol{y}_2) & \cdots & Cov(\boldsymbol{y}_1, \boldsymbol{y}_m) \\ Cov(\boldsymbol{y}_2, \boldsymbol{y}_1) & Var(\boldsymbol{y}_2) & \cdots & Cov(\boldsymbol{y}_2, \boldsymbol{y}_m) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(\boldsymbol{y}_m, \boldsymbol{y}_1) & Cov(\boldsymbol{y}_m, \boldsymbol{y}_2) & \cdots & Var(\boldsymbol{y}_m) \end{pmatrix}$$

We decompose this matrix into its eigenvector and eigenvalue pairs.

$$\boldsymbol{C}\boldsymbol{v} = \lambda \boldsymbol{v}$$

We find n or less orthonormal eigenvector-eigenvalue pairs that describe a new space in $\mathbb{R}^n$. These eigenvectors are referred to as *principal components*. The magnitude of the eigenvalue determines how much variance in the data is accounted for by that eigenvector. The p eigenvectors associated with the p largest eigenvalues that account for 99% of the variance in the data are retained. The selection of 99% variance retention is explained in the results section. Lastly, we project the columns of $\widehat{\boldsymbol{A}}$ onto these retained eigenvectors to form a new "weight space", via

$$\boldsymbol{W} = \boldsymbol{V}^T\widehat{\boldsymbol{A}}$$

where the columns of $\boldsymbol{V}$ are the p eigenvectors found in the previous step. This means $\boldsymbol{V} \in \mathbb{R}^{m\times p}$

and $\boldsymbol{W} \in \mathbb{R}^{p \times n}$. Each column of $\widehat{\boldsymbol{A}}$ is transformed into this new space that amplifies the distinction between stocks based on their pair-wise correlations. This $\boldsymbol{W}$ matrix is fed forward into the second layer of the neural network, *k-means clustering*.

K-means Clustering

The goal of k-means clustering is to identify $k \in \mathbb{N}$ disjoint partitions in the vector space that group the data points such that each one belongs to only one cluster. In our case, the data points $\boldsymbol{w}$ are the columns of $\boldsymbol{W}$ and the k-means algorithm will partition n data points in the $\mathbb{R}^p$ vector space. The process begins by selecting k data points to be the starting "centroids". Then the following two steps are repeated until the stopping criteria is met:

1. The distance of each data point w to each centroid $\boldsymbol{c}_i$ for $i = 1, \ldots, k$ is computed via $||\boldsymbol{w} - \boldsymbol{c}_i||_2^2$, and each w is assigned to the cluster set S_i of its nearest centroid.
2. A new centroid is selected for each cluster by taking the mean of all vectors in that cluster, $\boldsymbol{c}_i = \frac{1}{|S_i|} \sum_{\boldsymbol{w}_j \in S_i} \boldsymbol{w}_j$

The stopping criteria for these iterations is met when

$$\sum_{i=1}^{k} \sum_{\boldsymbol{w}_j \in S_i} ||\boldsymbol{w}_j - \boldsymbol{c}_i||_2^2 \tag{1}$$

is minimized. This is commonly called the *intracluster distance*. Computationally, iterations are stopped when the intracluster distance changes by less than some threshold between iterations. Challenges that arise from this include determining k that forms meaningful clusters that do not overfit the data. Choosing $k = n$ would give each data point its own cluster, which while reducing the intracluster distance to zero would not provide much information about the similarity of the data points. When $k \ll n$ there is a risk that dissimilar data points will be grouped together. Finding the ideal k value between these two extremes requires careful consideration of both the intracluster distance and the problem being solved.

Forming Portfolios

Each $\boldsymbol{w}$ in the k clusters found in the previous section can be tied back to the stocks in the original data set by the transformation $\boldsymbol{x}_i = \sigma_i \boldsymbol{V} \boldsymbol{w}_i + \mu_i$. Stocks belonging to the same cluster are said to be a part of the same portfolio. Each stock in these portfolios has a very high average linear correlation with all other stocks in the same portfolio. These portfolios also have significantly high MPD. High correlation between two stocks over long periods of time implies that future prices will remain correlated as well. Take Bank of America and Wells Fargo (NYSE: BAC and WFC respectively) as an example.

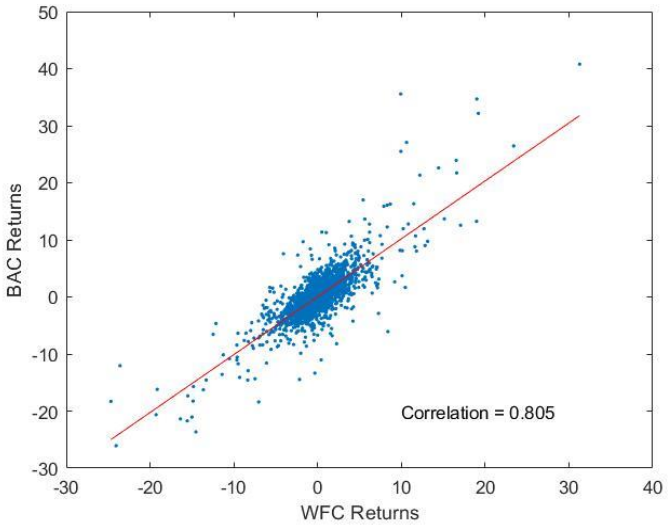

Figure 1: BAC vs WFC

From 2001 to 2017, these two multi-billion-dollar banking companies had stock returns with a linear correlation of 0.805. This is very nearly the perfect correlation of 1. A plot of the two stocks along with the least squares line of best fit is shown in Figure 1. It can be seen that the stock returns from both companies on the same days nearly match the linear fit. Therefore, the fluctuations of BAC's stock price serve as an excellent indicator for similar fluctuations in WFC's stock price, and vice-versa. This logic can be extended to the other stocks in the same portfolio as these two, as each had similarly high correlation to others in the portfolio. Therefore, each stock in the same portfolio has a very high probability to experience similar fluctuations in stock price as others in their portfolio and can therefore be used to predict the prices of those other stocks.

Results

We developed a feed-forward neural network that can identify stocks with very high correlations and similar daily movement. To do this we gathered 100 lucrative companies' stocks and put them through the methodology explained above. In our test sample of 100 stocks, the neural network formed stock groups that have a mean pair-wise correlation of 0.4299 and a MPD of approximately 65.3374% when the number of gathered clusters, k, was 50 and the percent of variance accounted for was 99%.

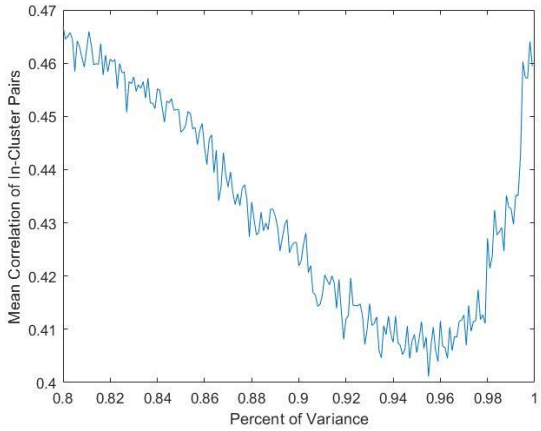

Figure 2a: Mean Correlation of Clusters vs Percent of Variance

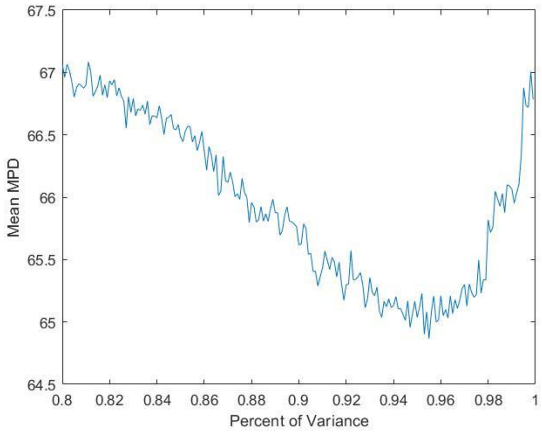

Figure 2b: MPD vs Percent of Variance

Figure 2: Observations of Mean Correlation and MPD Averaged Over 100 Trials for Varied Percent of Variance Explained. $k = 50$

It can be seen in Figure 2 that the mean correlation of in-cluster pairs and MPD both have local maxima at a variance retention of 99%. While retaining 100% of the variance would mean that correlation and MPD are maximized, it would mean that none of the dimensionality reduction from PCA was utilized. One of the goals of this project was to reduce the dimensionality of the stock market for use in k-means clustering, so we selected 99% variance retention for our analysis.

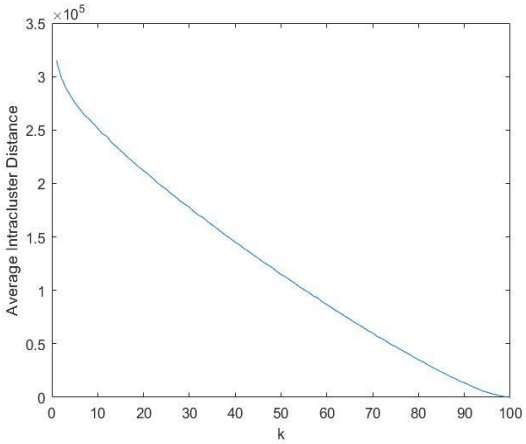

Figure 3: Average Intracluster Distance for Varied k in the 100 Stock Sample, Averaged Over 100 Trials

The value of k was determined using two analyses. Generally, k is selected to be at the point in Figure 3 where the intracluster distance begins to decrease at a decreasing rate. Here, the intracluster distance seems to decrease at a constant rate between $k = 10$ and $k = 90$, so it is difficult to infer the ideal k value from this plot alone. So we analyzed the average correlation and MPD for varied k to see if we could draw any insight from that. We found that larger numbers of clusters produced portfolios with very high correlation and high MPD but contained very few stocks, which does not yield much information to the observer. A graph of correlation vs k can be seen in Figure 4a and a graph of MPD vs k can be seen in Figure 4b. We noticed that correlation and MPD both seem to increase at a constant rate between $k = 10$ and $k = 90$, indicating that no

value of k would produce significantly better portfolios. We noticed that there are several local maxima in both plots, including one at $k = 50$. Taken in the context of the project, $k = 50$ would produce an average of 2 stocks per portfolio, which ensures that on average we can derive some insight about co-movement from every stock in the sample.

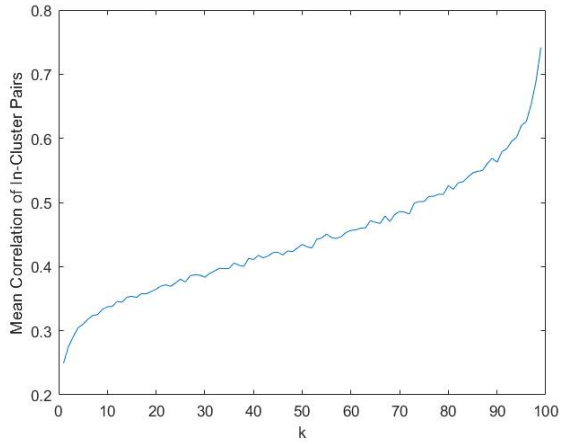

Figure 4a: Mean Correlation of Clusters vs k

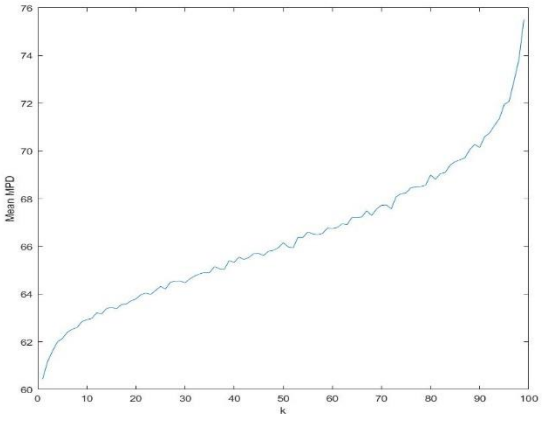

Figure 4b: MPD vs k

Figure 4: Observations of Mean Correlation and MPD Averaged Over 100 Trials for Varied k. Percent of Variance Explained = 99%

Figures 5 and 6 show a pair of stocks that were clustered into the same portfolio using our algorithm with the described values. Anthem Inc. and UnitedHealth Group Inc. (NYSE: ANTM and UNH, respectively) are a pair of lucrative health insurance corporations, so it is reasonable to assume that they would experience similar growth over time. Our algorithm paired these two together, and we do in fact see that the prices appear to be correlated over time, and their daily returns also appear to be heavily correlated.

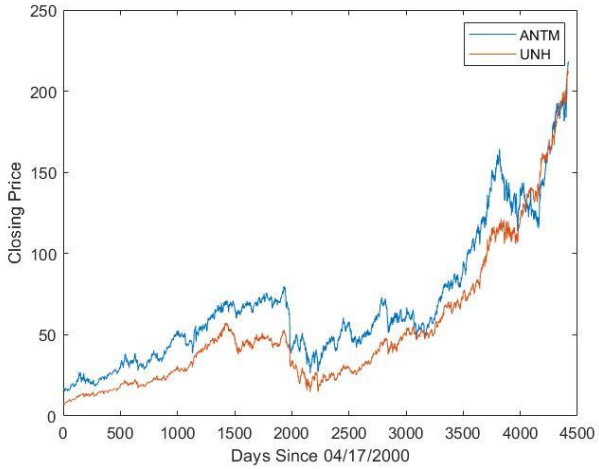

Figure 5: ANTM and UNH Closing Prices

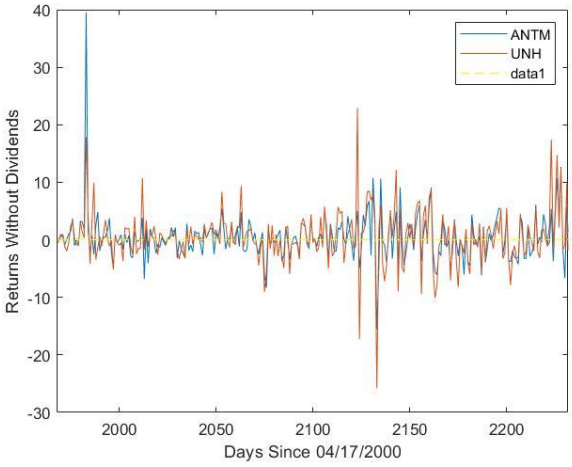

Figure 6: ANTM and UNH Returns

Similarly, Figures 7 and 8 show a pair of stocks that were clustered into the same portfolio. However these stocks, AT&T and McDonald's (NYSE: T and MCD, respectively), do not belong to the same business sector of the market. Just as with ANTH and UNH, we see that these two stocks have correlated prices and returns, providing a unique insight into the relationship of these stocks in the stock market.

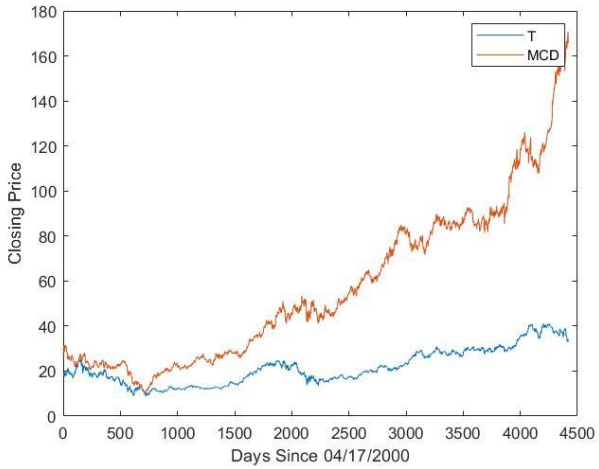

Figure 7: T and MCD Closing Prices

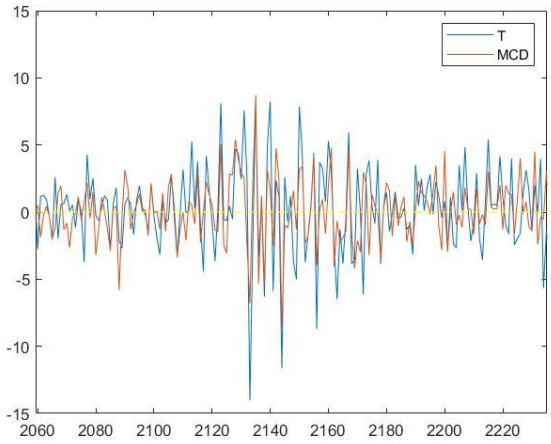

Figure 8: T and MCD Return

Limitations

For the feed-forward neural network to work, the data matrix A must be a complete matrix in $R^{m \times n}$ without any empty entries. Therefore, all stocks must have recorded prices for each day in the sample range. This means that stocks that open or close their public trading during that time period cannot be used for portfolio extraction. Furthermore, the time period on which the analysis is performed should be sufficiently large. Upwards of three years of data should provide actionable results, but longer time periods will be more accurate since linear correlation values are most significant for large numbers of data points. Additionally, a sufficient number of stocks should be used in the original data set. We used 100 stocks for the majority of our experimentation, and we do not recommend using much fewer than that.

Acknowledgements

The research team would like to thank our mentor Dr. Melinda Lanius for her time and input through every stage of this project. We cannot thank her enough for her continued support and encouragement. This project would not have been completed without her kindness and generosity of self.

We would also like to thank the University of Arizona for bringing this team together, and for providing us with the opportunity to publish our findings.

Appendix

MATLAB Code for Clustering Stock Portfolios

```
clear
rng(1)
%% User Inputs
% Place Path to Folder Here
testfiledir = 'Data 100 Stocks';
% Set minimum number of days that must be present in stock file
min_days = 0;
% Set the start date for analysis (# of days before last day in dataset)
start_date = 260*17;
% Set the end date for analysis (# of days before last day in dataset)
end_date = 260*0;
% Percent of variance to be explained
percent_explained = 0.99;
%% Read in Files
matfiles = dir(fullfile(testfiledir, '*.txt'));
nfiles = length(matfiles);
disp('Total Number of files:')
disp(nfiles)
j = 1;
for i = 1:nfiles
    fid = fullfile(testfiledir, matfiles(i).name);
    M = importdata(fid);
    if isempty(M)
        continue
    end
    if string(M.textdata(end,1)) == '2017-11-10' && size(M.textdata,1) >= min_days
        data{j} = M;
        file_names{j} = matfiles(i).name;
        j = j + 1;
    end
end
nfiles = j - 1;
disp(string(nfiles)+' Files Loaded')

%% Calculate returns and place into a matrix based on date
min_sz = size(data{i}.data, 1);
for i = 1:nfiles
    A = data{i};
    if size(A.data, 1) < min_sz
        min_sz = size(A.data, 1);
    end
end
returns_matrix = [];
if start_date < min_sz && start_date ~= 0
    min_sz = start_date;
end
for i = 1:nfiles
    A = data{i};
    ndays = size(A.data, 1);
    r = ((A.data(1:(ndays-1),4)-A.data(2:ndays,4))./A.data(2:ndays,4)).*100;
    sz = size(r, 1)+1;
    r =  r(sz-min_sz:end,:);
    returns_matrix = [returns_matrix, r];
end
date_diff = start_date - end_date;
returns_matrix = returns_matrix(1:date_diff,:);
%% Perform PCA
```

```
A = returns_matrix;
nobservations = size(A,1);
nvariables = size(A,2);
mu = mean(A,1);
stdevs = std(A);
T = (A - mu)./stdevs;
C = cov(T');
[V,D] = eigs(C,min(nvariables,nobservations));
E = diag(D);
explained = E./sum(E);
tot_explained = 0;
j = 1;
V_red = [];
while tot_explained < percent_explained
    v_temp = V(:,j);
    v = v_temp./norm(v_temp);
    V_red = [V_red, v];
    tot_explained = tot_explained + explained(j);
    j = j + 1;
end

%% Find weight vector transformation
W = V_red'*T;

%% Calculate MPD and correlation for each pair
MPD = zeros(nvariables);
for i = 1:nvariables
    for j = 1:nvariables
        updown_1 = returns_matrix(:,i) > 0;
        updown_2 = returns_matrix(:,j) > 0;
        MPD(i,j) = sum((updown_1 - updown_2) == 0)/nobservations;
    end
end
R = corr(A,A);
%% Kmeans Clustering
k = ceil(nfiles/2);
[labels,~,SUMD] = kmeans(W', k);
k_correlations = [];
stock_names = {};
j = 1;
k_percent_accurate = [];
for i = 1:k
    cluster = find(labels == i)';
    if size(cluster,2) < 2
        continue
    end
    pairs = nchoosek(cluster,2);
    for row = 1:size(pairs,1)
        val = R(pairs(row,1),pairs(row,2));
        k_correlations = [k_correlations; val];
        val = MPD(pairs(row,1),pairs(row,2));
        k_percent_accurate = [k_percent_accurate; val];
    end
end
%% Print Results
k_indexes = {};
for i = 1:k
    cluster = find(labels == i)';
    k_indexes{i} = cluster;
    disp('Stocks in Portfolio '+string(i))
    if size(cluster,2) < 2
        name = split(file_names{cluster(1)}, ".");
        fprintf('%s\n', upper(name{1}))
```

```
        continue
    end
    for index = cluster
        name = split(file_names{index}, ".");
        fprintf('%s\n', upper(name{1}))
    end
end
[~,~,R_wo] = find(R - eye(nfiles));
R_Q = quantile(R_wo, [0.5, 0.75], 'all');
disp('Median and Third Quantile Correlation of All Stock Pairs')
disp(R_Q)
[~,~,P_wo] = find(MPD - eye(nfiles));
P_Q = quantile(P_wo, [0.5, 0.75], 'all')*100;
disp('Median and Third Quantile MPD of All Stock Pairs')
disp(P_Q)
disp('Mean Correlation of In-Cluster Pairs')
disp(mean(k_correlations))
disp('Mean MPD for In-Cluster Pairs')
disp(mean(k_percent_accurate)*100)
```

Sample Output for 100 Stock Sample

Portfolio 1
GD
LMT
Portfolio 2
MU
Portfolio 3
AA
Portfolio 4
ADBE
CSCO
IBM
INTC
Portfolio 5
BIIB
Portfolio 6
ANTM
UNH
Portfolio 7
BRK-B
CCL
COF
SCHW
Portfolio 8
DLTR
Portfolio 9
F
GE
Portfolio 10
JCP
JWN

Portfolio 11
GIS
PEP
PG
Portfolio 12
MSFT
Portfolio 13
CI
Portfolio 14
TXN
Portfolio 15
ABC
CAH
MCK
Portfolio 16
MMC
Portfolio 17
GS
OMC
Portfolio 18
KR
Portfolio 19
BBY
Portfolio 20
CL
CLX
Portfolio 21
SWK
Portfolio 22
X

Portfolio 23
DDS
Portfolio 24
NOC
Portfolio 25
TJX
Portfolio 26
UNP
Portfolio 27
AIG
Portfolio 28
ADM
Portfolio 29
CAT

Portfolio 30
COST
HD
TGT
WMT
Portfolio 31
BA
Portfolio 32
HON
Portfolio 33
ADP
AFL
AXP
CMCSA

GT
MET
SYY
VZ
WBA
Portfolio 34
HSY
K
KO
Portfolio 35
NKE
Portfolio 36
DIS
Portfolio 37
JNJ
MRK
Portfolio 38
SYK
Portfolio 39
AAPL
Portfolio 40
PFE
Portfolio 41
XRX
Portfolio 42
ORCL
Portfolio 43
COP
CVS
CVX

Portfolio 44	T	HPQ	Portfolio 49
DTE	XOM	Portfolio 47	DAL
ECL	Portfolio 45	AMZN	LUV
ITW	BBBY	EXC	Portfolio 50
MAR	KSS	FDX	BAC
MCD	LOW	UPS	C
PPG	ROST	Portfolio 48	JPM
SBUX	Portfolio 46	DE	WFC

References

[1] Forgy, Edward W. (1965). ``Cluster analysis of multivariate data: efficiency versus interpretability of classifications". Biometrics. 21 (3): 768–769. JSTOR 2528559.

[2] Jolliffe IT, Cadima J. 2016 Principal component analysis: a review and recent developments. *Phil. Trans. R. Soc. A* 374: 20150202. http://dx.doi.org/10.1098/rsta.2015.0202

[3] Marjanovic, Boris 2017. Huge Stock Market Dataset. www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

[4] Karl Pearson F.R.S. (1901) LIII. On lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2:11, 559-572, DOI: 10.1080/14786440109462720