Analysis of price fluctuations of financial stocks and construction of stock portfolios

Tvisha Gupta & Abhishek Behera
Chennai Mathematical Institute

Contents

1 Introduction

The random character of stock market prices was first modelled by Jules Regnault, a French broker, in 1863 and then by Louis Bachelier, a French mathematician, in his 1900 Ph.D. thesis, "The Theory of Speculation".

The efficient-market hypothesis was developed by Professor Eugene Fama at the University of Chicago Booth School of Business as an academic concept of study through his published Ph.D. thesis in the early 1960s. The efficient-market hypothesis (EMH), or the joint hypothesis problem, asserts that financial markets are "informationally efficient". In other words EMH, claims that prices on traded assets (e.g., stocks, bonds, or property) already reflect all past publicly available information. Stonger versions of EMH even additionally claim that prices instantly reflect even hidden or "insider" information. Eversince there has been much interest in discering patterns in the prices of stock markets. And in today's world where automated algorithmic trading dominates the scenario, the subject matter is even more serious and pertinent.

In our project we wish to apply popular prevalent techniques such as Principal Component Analysis, Random Matrix Theory and Markowitz Theory of Portfolio Optimization to discover information from the trading history of the stocks listed in BSE.

2 Correlation Matrix

Much of the fundamentals of many trading strategies rely on exploiting the correlation between different stocks. And such an exercise not only uncovers the underlying "interactions" for the stock market, but also furthers our understanding of the economy as a complex dynamical system.

In order to quantify correlations, we first calculate the price change ("return") of stock i = 1, ..., N:

$$g_i(t) = \ln S_i(t+1) - lnS_i(t)$$

where $S_i(t)$ denotes the price of stock i on the t^{th} day. Since different stocks have varying levels of volatality (standard deviation), we define a normalized return:

$$G_i(T) = \frac{g_i(t) - \langle g_i \rangle}{\sigma_i}$$

where σ_i is the standard deviation of g_i, and $\langle ... \rangle$ denotes a time average over a period studied. We then compute the equal-time cross-correlation matrix C with elements:

$$C_{ij} = \langle G_i(t) G_j(t) \rangle$$

In other words:

$$C_{ij} = \frac{1}{T}(G^T G)_{ij}$$

So, C thus formed is $N \times N$ matrix.

By construction, the elements are restrited to the domain $-1 \leq C_{ij} \leq 1$, where $C_{ij} = 1$ corresponds to perfect correlations, $C_{ij} = -1$ corresponds to perfect anticorrelations, and $C_{ij} = 0$ corresponds to uncorrelated pairs of stocks.

2.1 Collection of Data

- We collected data for the 200 companies that constitute the S&P BSE index from www.finance.yahoo.com for the period 1^{st} January 2007 to $31^{st} December 2012$.
- We wrote a Python code to automatically download data from the website. The Python code can be found in the Appendix.
- As a part of pre-processing we deleted the companies for which we couldn't find enough data.
- We also removed dates on which most of the market didn't trade and less than 0.25 fraction of the companies traded.
- For trading dates where more than 0.25 fraction of the companies traded, we used the prices of the last available trading date for the companies that didn't trade.
- In a situation where the data for a company was not available for a few days (either because the company was not listed in the exchange at that time or during the 2008 crash when a lot of companies did not list their prices), we interpolated to get data between dates whose data was available.
- We were left with 170 companies and 1513 trading dates to work with, i.e. $N = 170$ and $T = 1513$.
- We constructed of correlation matrix C by the above mentioned procedure.

3 Principal Component Analysis

The correlation matrix defined above is a symmetric matrix, that can be diagonalized.

This is the basis of the well known Principal Component Analysis (PCA), aiming at decomposing the fluctuations of $G_i(t)$.

Notice that $C \sim G^T G$ (differ by a scalar factor of I/T). Construct $G' = \frac{G}{\sqrt{T}}$ so that $C = G'^T G'$. So finding the eigenvalue-decompositon of C is equivalent to finding the singular value decomposition of G'.

Suppose the SVD of G' is $U \Sigma V^T$, then we may write:

$$G'_{ij} = \sum_{\alpha=1}^{N} \sqrt{\lambda_\alpha} u_i^\alpha v_j^\alpha$$

Now if we consider the k-largest singular values and truncate it there after, then:

$$G'_{ij} \approx \sum_{\alpha=1}^{k} \sqrt{\lambda_\alpha} u_i^\alpha v_j^\alpha$$

is a reasonably good approximation to the original data.

Further more if the largest singular value $\lambda_1 \gg$ other singular values then:

$$G'_{ij} \approx \sqrt{\lambda_1} u_i^1 v_j^1$$

The eigen-vectors corresponding to these first few largest eigenvalues are called as the *Principal Components*.

3.1 Calculating the Eigenvalues and Eigenvectors

With the idea of how eigenvalues and eigenvectors might be useful in studying C, we move towards calculating its eigenvalues and the corresponding eigenvectors. Physically, eigenvalue analysis can give insight into the behaviour of the matrix and thus tend to be the most powerful tool available in matrix theories. The eigenvalues and the corresponding eigenvectors are calculated in MATLAB using the QR algorithm. The program for the same is given the appendix.

4 Markowitz Portfolio Theory

In this section we give a very short introduction to Markowitz Portfolio Theory (MPT).

MPT is the mathematical formulation of the concept of diversification in investing, with the aim of selecting a collection of investment assets that collectively has lower risk than any individual asset. Harry Markowitz won a Nobel memorial prize for the theory.

4.1 Formulation

In MPT a portfolio Π consists of weights w_i, where w_i is the fraction of the wealth is invested in stock S_i and $\sum w_i = 1$. Then the return on the stock is given by:

$$\Phi = \sum_{i=1}^{N} w_i g_i$$

And the corresponding risk in holding the portfolio Π is given by the variance of returns:

$$\Omega^2 = \sum_{j=1}^{N} \sum_{i=1}^{N} w_i w_j C_{ij} \sigma_i \sigma_j$$

MPT assumes that investors are risk averse, meaning that given two portfolios that offer the same expected return, investors will prefer the less risky one. Thus, an investor will take on increased risk only if compensated by higher expected returns. Conversely, an investor who wants higher expected returns must accept more risk.

Mathematically the above can be formulated as the following constrained optimization problem:

Minimize:

$$\Omega^2 = \sum_{j=1}^{N}\sum_{k=1}^{N} w_k w_j C_{ij}\sigma_i\sigma_j$$

Subject to:

$$\Phi = \sum_{k=1}^{N} w_k G_k$$

$$\sum_{k=1}^{N} w_k = 1$$

4.2 Eigenportfolios and Eigenvalues

The list of numbers $\{u_k^i\}$ can be seen as the weights of the different stock $k = 1, ..., N$ in a certain portfolio Π_i, where some stocks are 'long' ($u_k^i > 0$) while others are 'short' ($u_k^i < 0$).

The realized risk is measured by the variance of the returns, given by:

$$\Omega_i^2 = \frac{1}{T}\sum_t \left(\sum_k u_k^i G_{kt}\right)^2 = \sum_{jk} u_j^i u_k^i C_{jk} = \lambda_i$$

.

It is customary to call the portfolios corresponding to the eigenvectors as eigenportfolios. And the essential synopsis is that the risk of the investment in an eigenportfolio is its corresponding eigenvalue.

5 Random Matrix Theory

Much of the success of the trading strategies depends on whether the correlations are true correlations or not. We use random matrix theory to help us decide that.

5.1 Random matrices and the correlation matrix

The project started with an expectation that the price changes are random. We claim that it is indeed so and verify this claim. Since our measure of the price changes is determined by the correlation matrix C, we intend to show that C behaves like a random matrix and exhibits most of its properties. Since the properties of random matrices are known, we can thereby use them to study C.

A random matrix R is defined as a matrix whose elements are all uncorrelated, have mean 0 and variance 1. To construct such a matrix, we take A to be an $T \times N$ random matrix (constructed using MATLAB) with elements that belong to a standard normal distribution having mean 0 and variance 1. Define R as

$$R = \frac{1}{T} A^T A$$

So R is a random matrix and henceforth will be used throughout the project for the comparison of properties with C.

The tests used to prove the claim are as follows:-

- Probability distribution of the eigenvalues of C is almost the same as the probability distribution of the eigenvalues of R.
- Probability distribution of the components of "almost" all eigenvectors converges to standard normal distribution, i.e., $N(0, 1)$ which is the probability distribution of the components of every eigenvector of a random matrix.
- Probability distribution of the 'nearest neighbour unfolded eigenvalues' is the same as that of R.

Note that all 3 tests are necessary but not sufficient to prove our claim. However, there does not exist any sufficient test for this. So we verify whether the necessary tests hold or not. In the event of non-compliance with any of the tests, our whole claim will fail.

For this project, we perform the first 2 tests. The third test, although the most powerful test which if done, can show that the 'nearest neighbour unfolded eigenvalues' of C and R are same for more than 99.5% eigenvalues is being skipped because of the depth of the probability and measure theoretic topics involved in that study. We hope to study it in the near future if we can.

5.2 Test 1: Probability distribution of eigenvalues

Eigenvalue decomposition does not get changed for similar matrices. Since C and R are expected to show a lot of similar behaviour, therefore mathematically, they are expected to be 'somewhat' similar. This 'somewhat' similar mathematical nature is determined by whether their eigenvalues are almost equal or not.

The probability distribution of eigenvalues denotes the probability of an eigenvalue lying in a small interval. If in a very small interval, the probability of an eigenvalue of C and of R is very close to each other, we can safely conclude that their eigenvalues are almost equal. For this we follow the follwing steps:

- Find the largest and smallest eigenvalues of C.
- Divide the interval between these eigenvalues in to a lot of smaller subintervals (around 500 in number).
- Find the number of eigenvalues of C that lie in each subinterval.
- Divide these numbers with the total number of eigenvalues of C.

We compare this distribution with the probability density of the eigenvalues of the random matrix R which is given by the formula

$$P_{rm}(\lambda) = \frac{Q}{2\pi} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda}$$

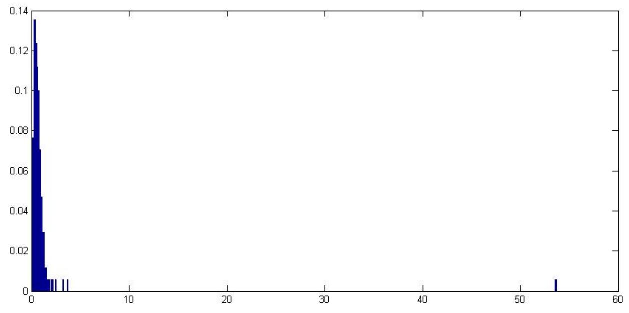
where $Q = \frac{T}{N}$ and λ_+ and λ_- are the largest and the smallest eigenvalues respectively.
T is the number of dates for which data was collected and N is the number of stocks being studied.

According to random matrix theory, the largest and the smallest eigenvalues of a random matrix are given by

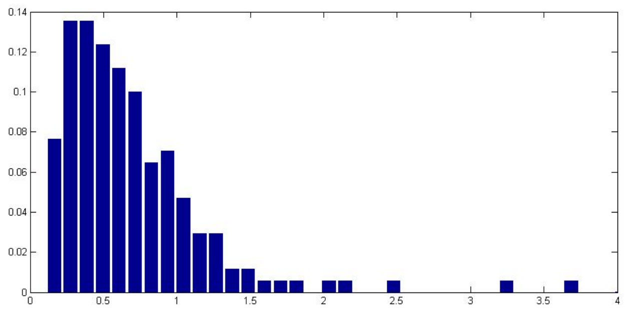
$$\lambda_{\pm} = 1 + \frac{1}{Q} \pm 2\sqrt{\frac{1}{Q}}$$

The fact that different random matrices will have different eigenvalues lying between the same range explains the possibility of infinitely many values that can be an eigenvalue of a randomly chosen R. This is the reason why we consider probability density here.

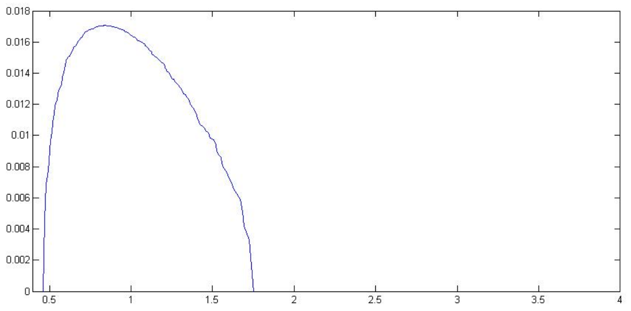
Plotting the two distributions, we note that the distributions have a lot of similarity.



Probability distribution of eigenvalues of C.



Probability distribution of eigenvalues of C (with modified axes).

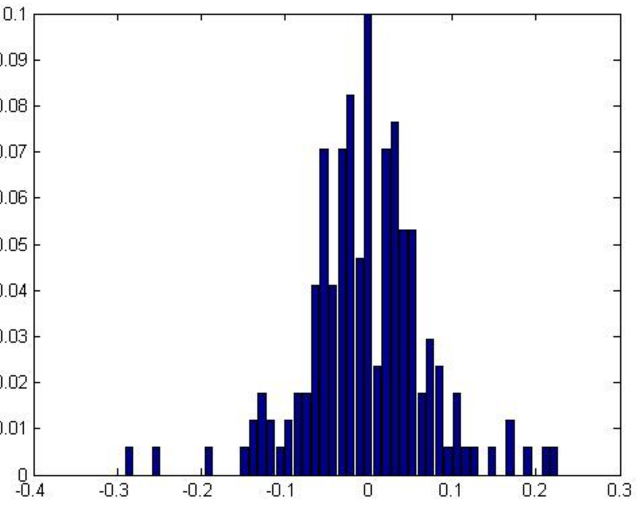


Probability distribution of eigenvalues of the random matrix R.

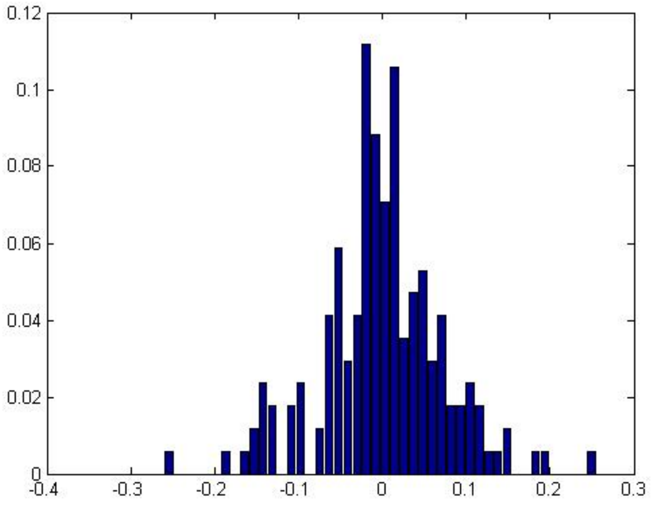
Since the two graphs are very similar to each other, our first test is successfully verified.

5.3 Test 2: Probability distribution of eigenvector components

According to random matrix theory, the components of any eigenvector follows standard normal distribution. We see if the same is true for the components of eigenvectors of C or not. For this we find the probability distribution of the componets of some eigenvectors of C.



Probability distribution of components of u^{20}



Probability distribution of components of u^{35}

For a majority of the eigenvectors, we see that the test holds.

5.4 Areas where the tests fail

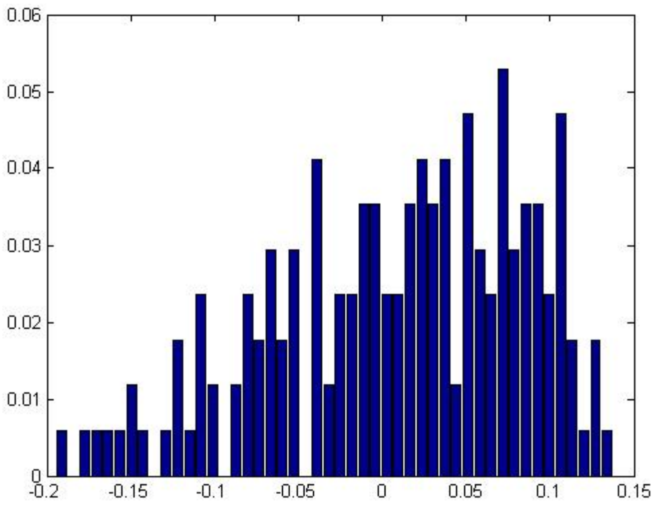
As we saw, RMT gives bounds for the largest and the smallest eigenvalues of a random matrix (given by an equation written in section 5.2). These bounds are dependent purely on the dimensions of the data collected. Thus if C was a perfect random matrix, these bounds predicted by RMT would have been the actual bounds for the eigenvalues of C.

The upper bound according to RMT predictions is 1.7496 and the lower bound is 0.4589.

But C has some eigenvalues outside these bounds. This shows that C is not a complete random matrix.

This can be seen in both the tests as well. In test 1,there are some eigenvalues much outside the predicted plot of eigenvalues of a random matrix.

In test 2, the components of eigenvectors corresponding to eigenvalues outside the RMT predicted bounds do not have a standard normal distribution as displayed below:



Probability distribution of components of u^3

So the RMT predicted bounds give a range within which most of the eigenvalues lie, those that correspond to uncorrelated and random data. The existence of eigenvalues outside the bounds means that there are some elements of C that have some correlation and the genuine information about these correlations is obtained from the eigenvalues that are outside the bounds.

5.5 Deviating eigenvalues and eigenvectors

The eigenvalues that lie outside the RMT predicted bounds are called deviating eigenvalues. The eigenvectors corresponding to deviating eigenvalues are called deviating eigenvectors. For portfolio construction, the aim is to find some stocks that have some positive correlation so that an increase in price of one stock increases the price of others as well and thus total profit increases. It also needs to consider a fall in the price of a stock. The stocks should not be very highly correlated because then the total loss would increase. To sum it up, correlations play an important role in portfolio construction. So the meaningful information in this regard is contained in the deviating eigenvalues and deviating eigenvectors because they corespond to existence of correlations in the market.

Deviations are of two types:

- Eigenvalues lying below the RMT lower bound
- Eigenvalues lying above the RMT upper bound

6 Principal Components of C corresponding to the Deviating Eigenvalues

From the previous section we already know that the deviating eigenvalues contain the "genuine" correlation. In this section we interpret the physical or financial meaning of the deviating eigenvectors. Before that we shall define a few statistics.

6.1 Inverse Participation Ratio & Significant Participants

The IPR of an eigenvector u^k is defined as:

$$IPR(u^k) \equiv I^k = \sum_{i=1}^{N} [u_i^k]^4$$

where N is the number of components in the eigenvector u^k and u_i^k are its components.

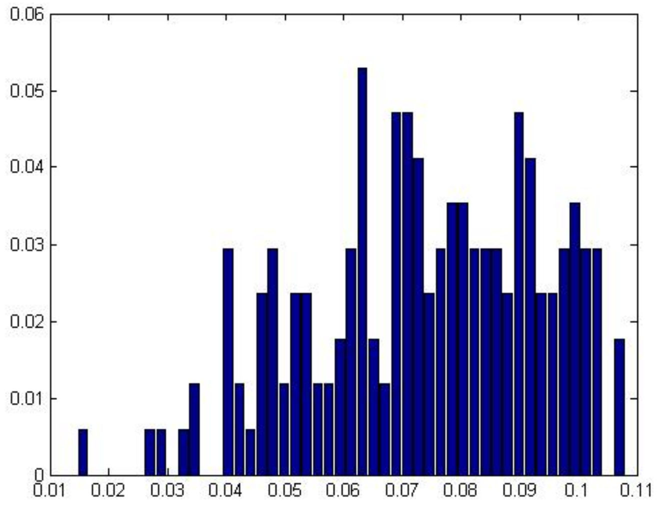
The meaning of IPR can be illustrated by two limiting cases: (i) a vector with identical components $u_i^k = \frac{1}{\sqrt{N}}$ has IPR$= \frac{1}{N}$, whereas (ii) a vector with one component $u_1^k = 1$ and the remainder zero has IPR$= 1$. Thus, the IPR quantifies the reciprocal of the number of eigenvector components that contribute significantly.

The first few of the $\frac{1}{I^k}$ components of higest magnitude in u_k are called the significant participants of u^k. The motivation is that these significant participants would almost completely describe u^k.

6.2 Interpretation of the largest Deviating Eigenvalue

For the eigenvector u^1 corresponding to the largest eigenvalue λ_1, IPR$(u^1) = 0.0072$ and hence $\frac{1}{I^1} = 138.48$ stocks out of 170 participate significantly in u^1.

The following graph shows the probability mass of each component of the eigenvector u^1:



Probability distribution of components of u^1

It is an almost uniform distribution, and it is as if one were to invest equally in the market. Thus the eigenvector u^1 depicts the entire market and it's dynamics are influenced by macroeconomic factors such as interest rate changes, GDP of the economy, inflation etc. that influence the entire market on the whole.

6.3 Interpretaion of Larger Deviating Eigenvalues

For the eigenvector u^2 corresponding to the largest eigenvalue λ_2, $\frac{1}{I^2}$ turns out to be 44.3913.

Listing the stocks corresponding to the top 10 significant participants, we get

- Wipro
- Infosys
- TCS
- HCL Tech
- Tech Mahindra
- Oracle
- Reliance Industries Ltd.
- Mphasis
- Cairn
- Tata Steel

i.e. 8 out of the top 10 significant participants of u^2 belong to the IT sector.

Repeating a similar exercise for the next few deviating eigenvectors we find:

- Most of the top significant participants of u^3 belong to the power, oil and petroleum sector.
- Most of the top significant participants of u^4 belong to the consumer products and pharmaceutical sector.
- And most of the top significant participants of u^5 belong to the banking sector.

Thus, the deviating eigenvalues above the RMT reveal the microstructure of the economy, with each correponding deviating eigenvector respresenting a cluster corresponding to the major market segments.

6.4 Interpretation of Smaller Deviating Eigenvalues

There are also a few eigenvalues that lie below the predicted RMT bounds. As expected the IPR of these eigenvectors are large. In other words, the number of significant participants are few. So these eigenvectors represent small clusters of highly correlated stocks.

For example the significant companies in the u^{170} are (where the number to the right of it represents its component in the vector):

- Hindustan Petroleum Corporation Limited: -0.44
- Reliance Capital: -0.27
- Syndicate Bank: -0.18
- Vijay Bank: 0.51
- Bharat Petroleum Corporation Limited: 0.33
- Canara Bank: 0.14

The corresponding correlation coefficients are:

	HPCL	BPCL
HPCL	1	0.70
BPCL	0.70	1

	RELC	SYND	VIJAY	CANARA
RELC	1	0.63	0.58	0.51
SYND	0.63	1	0.74	0.59
VIJAY	0.58	0.74	1	0.60
CANBARA	0.51	0.59	0.60	1

	RELC	SYND	VIJAY	CANARA
HPCL	0.3	0.32	0.34	0.29
BPCL	0.31	0.31	0.28	0.27

So we notice that the eigenvector has two highly correlated subclusters comprising of the petroleum and the banking sector. And the subclusters themselves are weakly correlated amongst themselves. These feature of the eigenportfolio corresponding to the smaller eigenvalues, makes them ammenable to a pairs trading strategy.

7 An Illustrative Strategy: Pairs Trading Strategy

The pairs trade or pair trading is a market neutral trading strategy enabling traders to profit from virtually any market conditions: uptrend, downtrend, or sideways movement.This strategy is categorized as a statistical arbitrage and convergence trading strategy. The pair trading was pioneered by Gerry Bamberger and later led by Nunzio Tartaglia's quantitative group at Morgan Stanley in the 1980s.

The strategy monitors performance of two historically correlated securities. When the correlation between the two securities temporarily weakens, i.e. one stock moves up while the other moves down, the pairs trade would be to short the outperforming stock and to long the underperforming one, betting that the "spread" between the two would eventually converge.

Now consider the eigenportfolio u^{170} considered in the previous section. So one could make the following offsetting pair trades with:

1. HPCL: 0.44 & BPCL: -0.33
2. Vijay Bank: 0.51, Canara Bank: 0.14 & Reliance Capital: -0.27, Syndicate Bank: -0.18

However MPT tells us that by making both the above pairs trade, instead of just one of them, we can further reduce the risk in the portfolio.

8 Conclusion

Thus we categorised the stocks into 3 categories:

- Those corresponding to the small deviating eigenvalues (i.e. highly correlated small clusters)
- Those corresponding to eigenvalues lying within RMT predicted bounds (i.e uncorrelated stocks)
- Those corresponding to the large deviating eigenvalues (i.e. formation of the major diversified clusters of the market)

Depending on the investors preferences and risk appetite, he invests in either the first or the third category. The first category provides him assured profits with very low risk. But since this calls for an arbitrage opportunity, this strategy is applicable only in the short run. For higher returns, the investor makes a portfolio comprising of the top significant participants of each cluster of the eigenvectors corresponding to eigenvalues above the RMT bounds. These stocks determine how the market moves as a whole. A change in the significant participants of each cluster determines how all other stocks in the same cluster will behave and with any change in the market, the top significant participants are likely to get affected the most. This makes them more risky. But since the portfolio consists of only a limited number of stocks (that too the significant ones) rather than any stock chosen at random, the risk can be predicted well with the reduction in the data to be considered. This makes it an optimal portfolio where the returns are high due to the high risk. But at the same time, the risk management is done in a better way.

9 Appendix: Codes

1. **Python Code to Download Data**
   Code to Fetch Data:

```
from urllib.request import urlopen

base_url1 =
"http://ichart.finance.yahoo.com/table.csv?s="
base_url2 =
"&a=0&b=1&c=2007&d=11&e=30&f=2012&g=d&ignore=.csv"
def make_url(ticker_symbol):
    return base_url1 + ticker_symbol + base_url2

output_path="New folder"

def make_filename(ticker_symbol):
    return output_path + "\\" + ticker_symbol + ".csv"

def get_data(ticker_symbol):
    r=urlopen(make_url(ticker_symbol))
    outfile = open(make_filename(ticker_symbol), "wb")
    outfile.write(r.read())
    outfile.close()

def fetch(filename = 'BSEticker.txt'):
    f = open(filename)
    l = f.readlines()
    for t in l:
        get_data(t[:-1])
        print(t[:-1])
```

   Code to Club Data:

```
folder = 'New folder'
filenames = 'BSEticker.txt'
outfile_path = 'Data'
outfile = 'Consolidate.csv'
output = outfile_path + '\\' + outfile


def club():

    w = open(output, 'w')
    w.write('Date\n\n')
    w.close()

    l = open(filenames).readlines()

    col = 0
    for t in l:
        now =
        open(folder+'\\'+t[:-1]+'.csv').readlines()
        print(t[:-1])
        x = now[:]
        for i in range(0,len(now)):
            x[i] = now[i].split(',')
            x[i] = [x[i][0],x[i][4]]
        insert(output,x,t[:-1],col)
        col  = col + 1

def insert(out,x,filename,col):

    y = open(out).readlines()
    for i in range(0,len(y)):
        y[i] = y[i].split(',')
        l=len(y[i])-1
        y[i][l] = y[i][l][:-1]

    y[0].append(filename)

    m = 1
    for d in x[1:]:
        append = 0
        for i in range(1,len(y)):
            if (d[0]==y[i][0]):
                y[i].append(d[1])
                append = 1
                for j in range(m,i):
                    y[j].append('')
                m = i+1

        if (append == 0):
            i = 1
            while(compare(y[i][0],d[0])):
                if (i+1 < len(y)):
                    i = i+1
            for k in range(0,col):
                d.insert(1,'')
            for j in range(m,i):
                    y[j].append('')
            y.insert(i,d)
            m = i+1

    for j in range(m,len(y)):
            y[j].append('')


    pprint(y,output)

def compare(y,x):
    y=y.split('-')
    x=x.split('-')
    if (y[0]>x[0]):
        return 1
    elif (y[0]==x[0]):
        if (y[1]>x[1]):
            return 1
        elif (y[1]==x[1]):
            if (y[2]>x[2]):
                return 1
            else:
                return 0
        else:
            return 0
    else:
        return 0

def pprint(t,file):
    y = t[:]
    for i in range(0,len(y)):
        y[i] = ''.join(str(e)+',' for e in y[i])
        y[i] = y[i][:-1]+'\n'
    y = ''.join(str(e) for e in y)
    w = open(file, 'w')
    w.write(y)
    w.close()
```

Code to Clean Data:

```
path = r'C:\Users\Abhishek\Desktop\CMI Text books\Sem - I\2 Linear Algebra\Project\Ra
data = 'Consolidate.csv'
cleaned = 'NewClean.csv'
infile = path + '\\' + data
```

```
outfile = path + '\\' + cleaned

def clean(purity=0.75):
    y = open(infile).readlines()
    for i in range(0,len(y)):
        y[i] = y[i].split(',')
        l=len(y[i])-1
        y[i][l] = y[i][l][:-1]

    l = len(y[0])
    p = purity*l

    for e in y:

        c = count(e)

        if (c < p):
            y.remove(e)
        #elif (c < l):
            #e = fill(y,e)

    pprint(y, outfile)

def count(l):
    c = len(l)
    for e in l:
        if (e == ''):
            c = c - 1

    return c

def fill(y,e):
    i = y.index(e)
    if (i == 1):
        for j in range(0,len(y[i])):
            if (y[i][j] == '' and y[i+1][j] != ''):
                y[i][j] = y[i+1][j]
    else:
        for j in range(0,len(y[i])):
            if (y[i][j] == '' and y[i-1][j] != ''):
                y[i][j] = y[i-1][j]


def pprint(t,file):
    y = t[:]
    for i in range(0,len(y)):
        y[i] = ''.join(str(e)+',' for e in y[i])
        y[i] = y[i][:-1]+'\n'
    y = ''.join(str(e) for e in y)
    w = open(file, 'w')
    w.write(y)
    w.close()
```

2. **MATLAB Code to form the Correlation Matrix**

```
num=xlsread('Project results.xlsx','Data');
l=size(num);
num1=num;
num2=zeros(l(1)-1,l(2));
for i=1:(l(1)-1)
    num2(i,:)=log(num1(i,:))-log(num1(i+1,:));
end
m=mean(num2); %gives column wise mean
for i=1:l(2)
    sqsum=0;
    for j=1:l(1)-1
        sqsum=sqsum+(num2(j,i)-m(i))^2;
    end
    v(i)=sqsum/(l(1)-1);
%gives column wise variance
end
num3=zeros(l(1)-1,l(2));
for i=1:(l(1)-1)
    for j=1:l(2)
        num3(i,j)=(num2(i,j)-m(j))/sqrt(v(j));
%standard normalize
    end
end

for i = 1:l(2)
    for j = 1:l(2)
        c(i,j) = num3(:,i)'*num3(:,j)/(l(1)-1);
%correlation matrix elements
    end
end

xlswrite('Project results.xlsx',c,'Correlation matrix');
```

3. **MATLAB Code to Compute Eigenvalues and Eigenvectors**

```
eigen1=xlsread('Project results.xlsx', 'Correlation matrix');
l=size(eigen1);
l1=l(1);
eigen2=zeros(l1,l1);
s=1;
for m=1:8000
    for i=1:l1 %implementing QR algorithm
        v=eigen1(:,i);
        for k=1:i-1
            r(k,i)=(q(:,k)')*eigen1(:,i);
            v=v-r(k,i)*q(:,k);
        end
        r(i,i)=norm(v);
        q(:,i)=v/r(i,i);
    end
    s=s*q; %matrix containing eigenvectors
    for i=2:l1
        for j=1:i-1
            r(i,j)=0; %putting lower triangular elements to 0
            %so that r is now upper triangular
        end
    end
    eigen1=r*q;
    disp(m); %to know the progress of the program working
end
for t=1:(l1^2)
    eigen2(t)=round(eigen1(t)*10^8)/10^8;
end

xlswrite('Project results.xlsx',s,'Eigenvectors');
xlswrite('Correlation.xlsx',eigen2,'Eigenvalues');
```

4. **MATLAB Code to find the p.m.f. of distribution of eigenvalues of Correlation Matrix**

```
a=xlsread('Project Results.xlsx', 'Eigenvalues');
l=size(a);
l1=l(1);
eig=zeros(l1,1);
for k=1:l1
    eig(k)=a(k,k);
end
l2=55/500; %dividing range of eigenvalues in 500
%smaller parts
x(1)=0;
t=zeros(1,500);
for i=2:500
    x(i)=(i-1)*l2;
    for k=1:l1
        if x(i-1)<=eig(k) && x(i)>=eig(k) %seeing
%if eigenvalue lies in the small interval
                t(i-1)=t(i-1)+1; %counting the no.
%of such eigenvalues
        end
    end
    prob(i-1)=t(i-1)/l1; %probability of eigenvalue
    y(i-1)=(x(i-1)+x(i))/2; %mid-point of the
%interval to make a probability histogram
end
bar(y,prob)
```

5. **MATLAB Code to find the p.m.f. of distribution of the components of Eigenvectors**

```
a=xlsread('Project Results.xlsx', 'Eigenvectors');
l=size(a);
k=input('kth eigenvector');
e=a(:,k);
e1=min(e);
e2=max(e);
dif=e2-e1;
l2=dif/50;
x(1)=e1;
t=zeros(1,50);
for i=2:50
    x(i)=x(i-1)+l2;
    for k=1:l(1)
        if x(i-1)<=e(k) && x(i)>=e(k) %seeing if the
%component lies in the small interval
                t(i-1)=t(i-1)+1;
%counting the no. of such components
        end
    end
    prob(i-1)=t(i-1)/l(1); %probability of component
    y(i-1)=(x(i-1)+x(i))/2; %mid-point of the interval
%to make a probability histogram
end
bar(y,prob)
```

6. **MATLAB Code to find the IPR and the Significant Participants of Eigenvectors**

```
a=xlsread('Project Results.xlsx', 'Eigenvectors');

%finding IPR
ipr=zeros(1,6);
%6 because these many eigenvalues were
%outside the upper bound
for i=1:6
    for j=1:170
        ipr(i)=ipr(i)+(a(j,i))^4;
    end
    ripr(i)=1/ipr(i); %reciprocal of ipr
end

% finding the top participants of each eigenvector
k=input('Eigenvector');
A=a(:,k); %To study the kth eigenvector.

A=sort(A,'descend');
for i=1:10
```

```
    for j=1:170
        if A(i)==a(j,k)
            disp(j);
%gives the indices of the largest participants
        end
    end
end
```

10 References

1. “Random matrix approach to cross correlations in financial data” by Vasiliki Plerou, Parameswaran Gopikrishnan, Bernd Rosenow, Lus A. Nunes Amaral, Thomas Guhr and H. Eugene Stanley PHYSICAL REVIEW E, VOLUME 65, 066126

2. “Distributions of singular values for some random matrices” by A. M. Sengupta and P. P. Mitra PHYSICAL REVIEW E, VOLUME 60, 03389

3. “Financial Applications of Random Matrix Theory: A short review” by Jean-Philippe Bouchaud, Marc Potters