Habits

1) Exploratory Analysis- Georgia Vote Undercount

Reading the data into a vector georgia2000

```
georgia2000=read.csv("C:/Users/Ramyasai/Desktop/Predictive modelling/James/STA380-maste
r(1)/STA380-master/data/georgia2000.csv")
vote_data = georgia2000
head(vote_data)
```

```
##     county ballots votes   equip poor urban atlanta perAA gore bush
## 1  APPLING    6617  6099   LEVER    1     0       0 0.182 2093 3940
## 2 ATKINSON    2149  2071   LEVER    1     0       0 0.230  821 1228
## 3    BACON    3347  2995   LEVER    1     0       0 0.131  956 2010
## 4    BAKER    1607  1519 OPTICAL    1     0       0 0.476  893  615
## 5  BALDWIN   12785 12126   LEVER    0     0       0 0.359 5893 6041
## 6    BANKS    4773  4533   LEVER    0     0       0 0.024 1220 3202
```

Creating a variable undercount_values which is the difference between ballots and votes

```
# Making the variable with undercounts magnitude
vote_data['Undercount_Values'] = vote_data['ballots'] - vote_data['votes']
head(vote_data)
```

```
##     county ballots votes   equip poor urban atlanta perAA gore bush
## 1  APPLING    6617  6099   LEVER    1     0       0 0.182 2093 3940
## 2 ATKINSON    2149  2071   LEVER    1     0       0 0.230  821 1228
## 3    BACON    3347  2995   LEVER    1     0       0 0.131  956 2010
## 4    BAKER    1607  1519 OPTICAL    1     0       0 0.476  893  615
## 5  BALDWIN   12785 12126   LEVER    0     0       0 0.359 5893 6041
## 6    BANKS    4773  4533   LEVER    0     0       0 0.024 1220 3202
##   Undercount_Values
## 1               518
## 2                78
## 3               352
## 4                88
## 5               659
## 6               240
```

For comparsion, of undercounts in different counties, we should calculate % undercounted votes (i.e standardize) so that a smaller county doesnt get undue advantage of having smaller magnitude of undercounts

```
vote_data['Undercount_Rate%'] = round((vote_data['Undercount_Values']/vote_data['ballot
s'])*100,2)
head(vote_data)
```

```
##     county ballots votes   equip poor urban atlanta perAA gore bush
## 1  APPLING    6617  6099   LEVER    1     0       0 0.182 2093 3940
## 2 ATKINSON    2149  2071   LEVER    1     0       0 0.230  821 1228
## 3    BACON    3347  2995   LEVER    1     0       0 0.131  956 2010
## 4    BAKER    1607  1519 OPTICAL    1     0       0 0.476  893  615
## 5  BALDWIN   12785 12126   LEVER    0     0       0 0.359 5893 6041
## 6    BANKS    4773  4533   LEVER    0     0       0 0.024 1220 3202
##   Undercount_Values Undercount_Rate%
## 1               518             7.83
## 2                78             3.63
## 3               352            10.52
## 4                88             5.48
## 5               659             5.15
## 6               240             5.03
```

Checking if any counties had no undercounts

```
perfect_counties = subset(vote_data, select = c('Undercount_Values'))
cat("Number of counties which had no undercount problems in election:", length(perfec
t_counties[perfect_counties$Undercount_Values== 0,]))
```

```
## Number of counties which had no undercount problems in election: 2
```

Almost all counties in Georgia had this undercount issue.

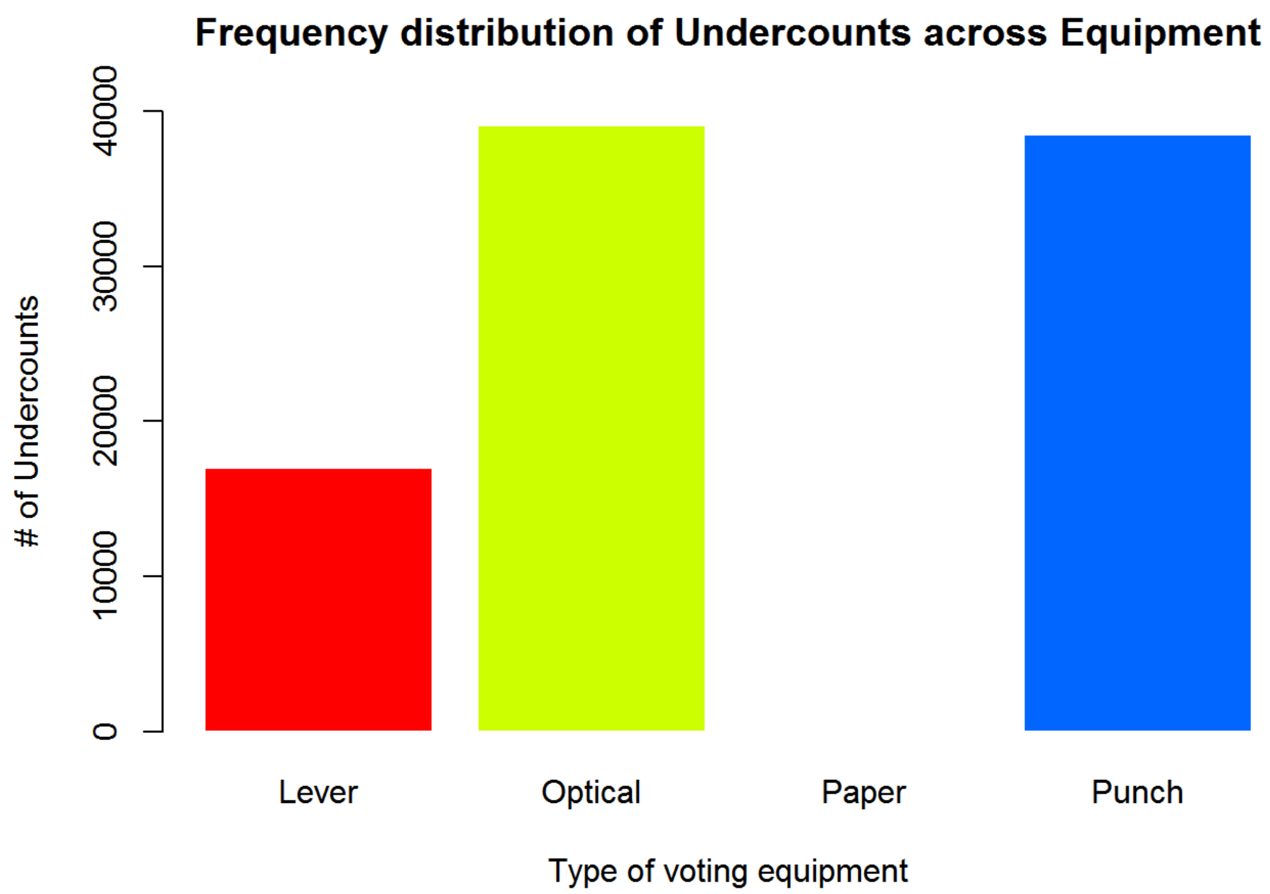
Getting the frequency distribution of voteunder rates by voting equipment used. First we'll visualize using a table since very few categories of Equipment are there.For easy visualization, a bar plot is drawn

```
Undercounts_Equip = data.frame(aggregate(Undercount_Values~equip, sum, data=vote_data))
Undercounts_Equip
```

```
##     equip Undercount_Values
## 1   LEVER             17016
## 2 OPTICAL             39090
## 3   PAPER               113
## 4   PUNCH             38462
```

Now it looks like Optical and Punches have too many undercountings. Now we can plot equipment vs the number of undercounts they cause

```
freq = Undercounts_Equip$Undercount_Values
barplot(freq,names.arg = c("Lever", "Optical", "Paper", "Punch"), col=rainbow(5), xlab=
"Type of voting equipment", ylab="# of Undercounts", main="Frequency distribution of Un
dercounts across Equipment", ylim=c(0,40000), border=FALSE)
```



We need to standardize the number of undercounts. It is possible that overall ballots for Optical was e.g. 80000, in that case the 38000 is just close to 50% times. Getting the number of ballots assigned to each equipment

```
Ballots_Equip= data.frame(aggregate(ballots~equip, sum, data=vote_data))
```

Integtaring the undercounts information in the table with equipment and ballots

```
Ballots_Equip_Undercounts = cbind(Ballots_Equip, freq)
```

Creating a percentage undercounts column

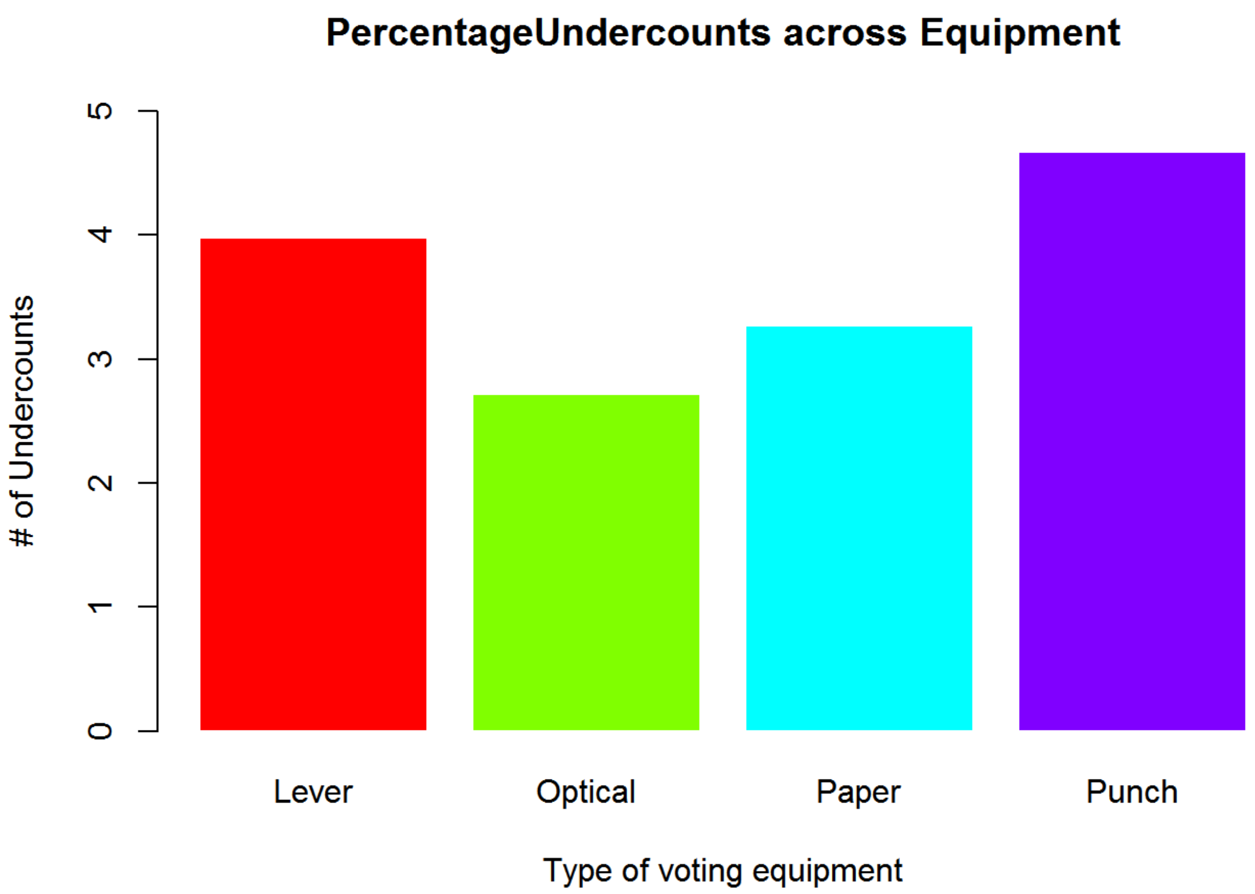
```
Ballots_Equip_Undercounts['per_Undercounts'] = round((Ballots_Equip_Undercounts['fre
q']/Ballots_Equip_Undercounts['ballots'])*100,2)
Ballots_Equip_Undercounts
```

```
##     equip ballots  freq per_Undercounts
## 1   LEVER  427780 17016            3.98
## 2 OPTICAL 1436159 39090            2.72
## 3   PAPER    3454   113            3.27
## 4   PUNCH  823921 38462            4.67
```

Punch has the highest %age of Undercounts, and not Optical. If one were to check devices for tampering(and if people not been able to follow instructions was ruled out), I'd defintely check the Punches

Optical and Punches have too many undercountings. Now we can plot equipment vs the number of undercounts they cause

```
percent_undercount = Ballots_Equip_Undercounts$per_Undercounts
barplot(percent_undercount,names.arg = c("Lever", "Optical", "Paper", "Punch"), col=rai
nbow(4), xlab= "Type of voting equipment",ylab="# of Undercounts", main="PercentageUnde
rcounts across Equipment", ylim=c(0,5), border=FALSE)
```



Checking if the undercounts for these machines, affects poverty and minorities

However, with the available data it is useful to see, % of undercounts by poverty and equipment

Replacing Poor values as ‘Poor’ and ‘Rich’ (instead of 1 and 0) to make it a categorical variable

```
vote_data = transform(vote_data, isPoor = ifelse(poor == 0, ifelse(poor == 1,0, "Ric
h"), "Poor"))
```

Calculating sum of overall ballots and sum of overall votes across, each machine and poverty flag

```
library(sqldf)
```

```
## Loading required package: gsubfn
## Loading required package: proto
## Loading required package: RSQLite
## Loading required package: DBI
```

```
equip_poverty_sums = data.frame(sqldf('SELECT equip, isPoor, SUM(votes) AS sum_votes, S
UM(ballots) AS sum_ballots FROM vote_data GROUP BY equip, isPoor'))
```

```
## Loading required package: tcltk
```

```
equip_poverty_sums['Undercounts'] = equip_poverty_sums['sum_ballots'] - equip_poverty_s
ums['sum_votes']

equip_poverty_sums['Per_Undercounts'] = round((equip_poverty_sums['Undercounts'])/(equi
p_poverty_sums['sum_ballots'])*100,2)
```

Having a look at the table of percentage undercounts by equipment and poverty level
equip_poverty_sums

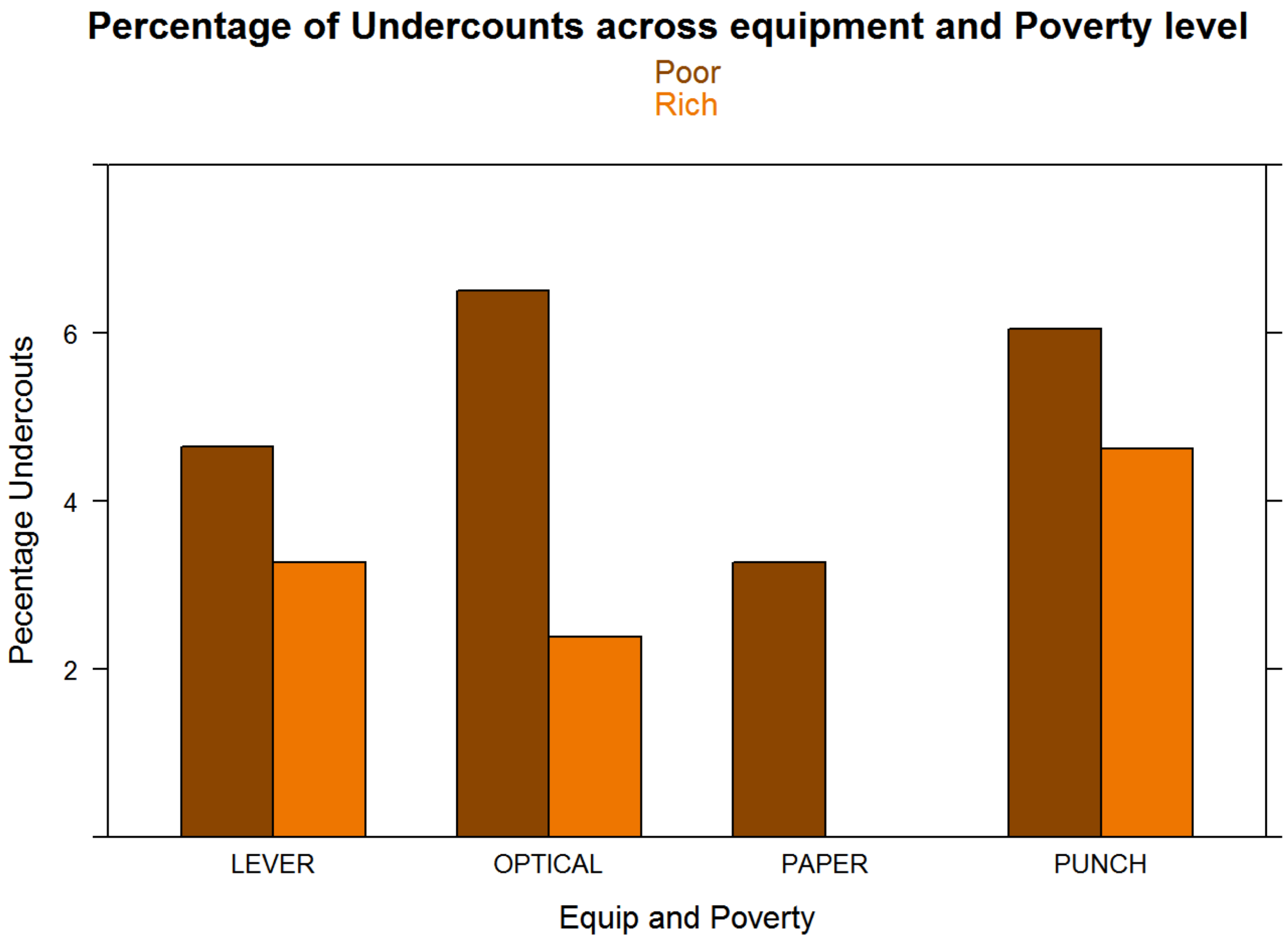
From the numerical table, it looks like percentage undercounts are much more in counties which have more people people We can plot it to have more easy readability

Creating a temporary dataframe to hold only the final columns we need for plotting

```
temp_poverty_equip_data = equip_poverty_sums[,c(1,2,6)]
```

Plotting the undercounts across equipment and poverty level

```
library(lattice)
barchart(Per_Undercounts~equip,data=temp_poverty_equip_data,groups=isPoor, xlab= "Equip
and Poverty", ylab = "Pecentage Undercouts", main= "Percentage of Undercounts across eq
uipment and Poverty level", ylim=c(0,8), col=c("darkorange4","darkorange2"), key= lis
t(space="top", text=list(c("Poor", "Rich"),col=c("darkorange4", "darkorange2"))))
```



Calculating number of African Americans in each county assuming total population is ballots

Question2

2) Bootstrapping - Exchange Traded Fund

After obtaining the data for 5 years from Yahoo, I will first simulate a scenarios where I equal money into all stocks.

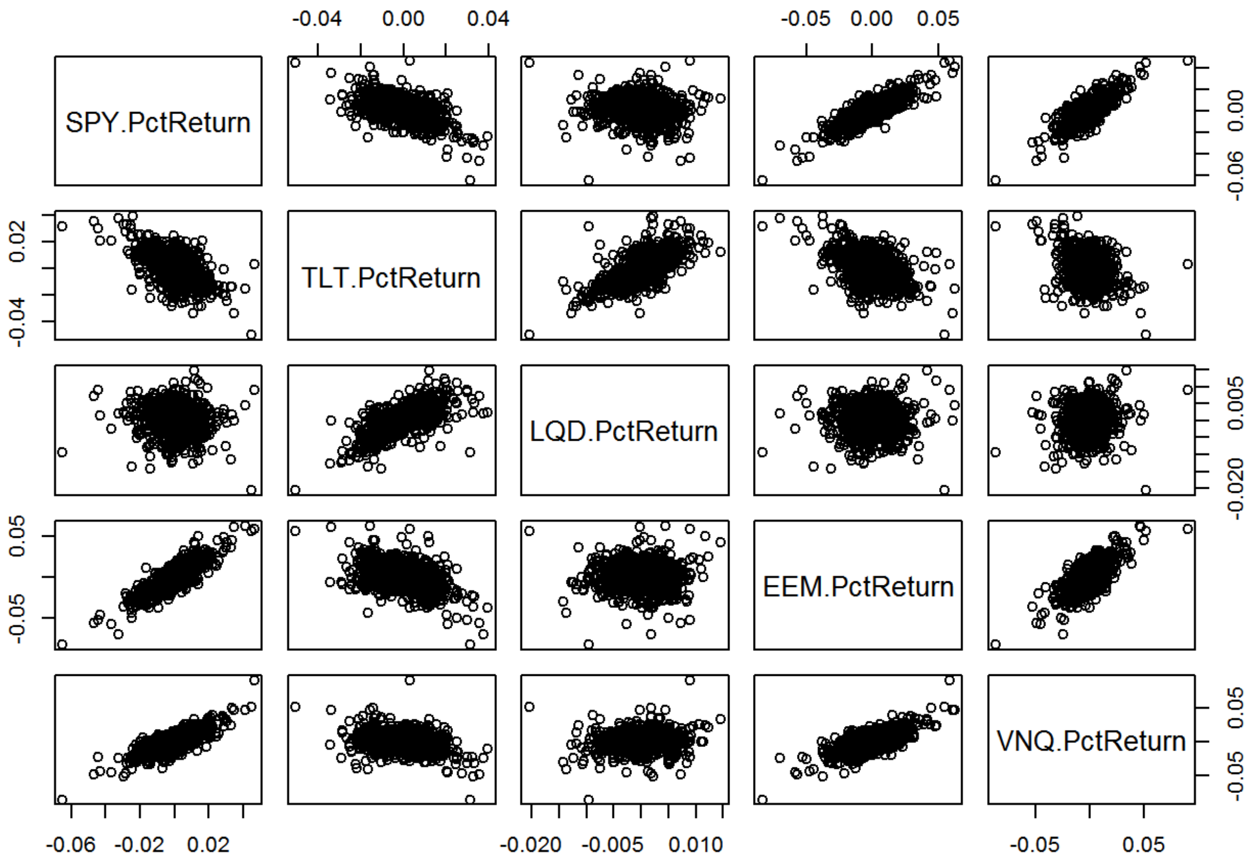
```
# Downloading 5 year data for SPY, TLT, LQD, EEM and VNQ
library(fImport)
```

```
## Loading required package: timeDate
## Loading required package: timeSeries
```

```
tickers = c("SPY", "TLT", "LQD", "EEM", "VNQ")
history_data_stocks = yahooSeries(tickers, from='2010-08-01', to='2015-08-01')
# The first few rows
head(history_data_stocks)
```

```
## GMT
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adj.Close
## 2010-08-02   111.99   112.94  111.54    112.76  188263200      101.8326
## 2010-08-03   112.48   112.77  111.85    112.22  146657300      101.3450
## 2010-08-04   112.53   113.11  112.16    112.97  158171700      102.0223
## 2010-08-05   112.25   112.91  112.08    112.85  140473800      101.9139
## 2010-08-06   111.74   112.57  110.92    112.39  239728300      101.4985
## 2010-08-09   112.92   113.18  112.32    112.99  120800400      102.0403
##            TLT.Open TLT.High TLT.Low TLT.Close TLT.Volume TLT.Adj.Close
## 2010-08-02    99.24    99.33   98.75     98.75    5769200      84.60973
## 2010-08-03    99.20    99.66   98.93     99.32    4363500      85.09811
## 2010-08-04    99.50    99.51   98.56     98.56    3820400      84.44693
## 2010-08-05    99.34    99.49   98.84     99.02    3704200      84.84106
## 2010-08-06    99.79   100.21   99.49    100.10    6042400      85.76641
## 2010-08-09    99.74    99.98   99.61     99.73    2578500      85.44940
##            LQD.Open LQD.High LQD.Low LQD.Close LQD.Volume LQD.Adj.Close
## 2010-08-02   109.91   109.95  109.56    109.63     764100      90.53107
## 2010-08-03   109.90   110.04  109.70    109.90    1060700      90.75404
## 2010-08-04   109.83   109.95  109.55    109.56     859900      90.47327
## 2010-08-05   109.69   109.89  109.59    109.76    1093400      90.63843
## 2010-08-06   110.19   110.48  110.06    110.39     685700      91.15867
## 2010-08-09   110.46   110.78  110.30    110.75     844400      91.45596
##            EEM.Open EEM.High EEM.Low EEM.Close EEM.Volume EEM.Adj.Close
## 2010-08-02    42.18    42.59   42.07     42.47   69623700      38.51627
## 2010-08-03    42.14    42.43   41.93     42.27   60207900      38.33489
## 2010-08-04    42.28    42.43   42.00     42.33   55875600      38.38930
## 2010-08-05    42.02    42.20   41.87     42.14   43650600      38.21699
## 2010-08-06    41.86    42.19   41.60     42.08   65731600      38.16258
## 2010-08-09    42.36    42.39   42.16     42.30   27051000      38.36209
##            VNQ.Open VNQ.High VNQ.Low VNQ.Close VNQ.Volume VNQ.Adj.Close
## 2010-08-02    51.78    52.81   51.62     52.66    3018300      43.59576
## 2010-08-03    52.53    52.57   51.78     52.15    1955500      43.17355
## 2010-08-04    52.39    52.54   51.90     52.52    2041300      43.47986
## 2010-08-05    52.24    52.50   51.75     51.86    1847300      42.93346
## 2010-08-06    51.31    51.78   50.76     51.62    1836100      42.73477
## 2010-08-09    51.88    52.38   51.56     52.21    2914500      43.22322
```

```
# Using Prof. Scott's helper function, which helps us in calculating returns of each ti
cker
# or stock

Returns_YahooStocks = function(series) {
  mycols = grep('Adj.Close', colnames(series))
    closingprice = series[,mycols]
    N = nrow(closingprice)
    percentreturn = as.data.frame(closingprice[2:N,]) / as.data.frame(closingprice[1:
(N-1),]) - 1
    mynames = strsplit(colnames(percentreturn), '.', fixed=TRUE)
    mynames = lapply(mynames, function(x) return(paste0(x[1], ".PctReturn")))
    colnames(percentreturn) = mynames
    as.matrix(na.omit(percentreturn))
}

# Calculating returns for the whole data which was downloaded from Yahoo
myreturns = Returns_YahooStocks(history_data_stocks)

# The pair plots help in giving us a preliminary idea about the stocks. Looks like EEM
and LQD # are highly correlated. SPY and TLT are highly correlated
pairs(myreturns)
```




```
library(mosaic)
```

```
## Loading required package: car
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:timeSeries':
##
##     filter, lag
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
##
## Loading required package: ggplot2
## Loading required package: mosaicData
##
## Attaching package: 'mosaic'
##
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
##
## The following object is masked from 'package:car':
##
##     logit
##
## The following objects are masked from 'package:timeSeries':
##
##     quantile, sample
##
## The following object is masked from 'package:timeDate':
##
##     sample
##
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cov, D, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
##
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
library(fImport)
library(foreach)
```

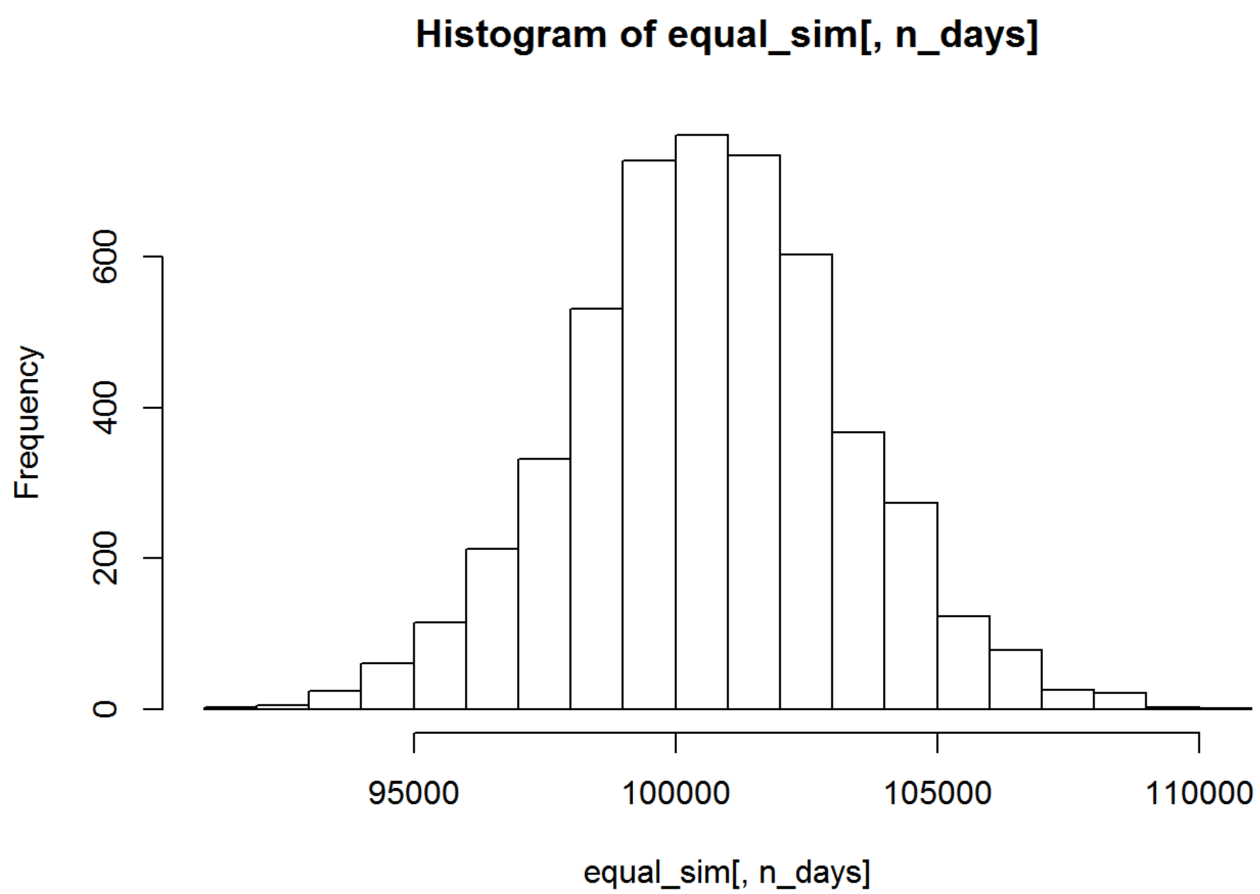
Now simulate many different possible trading years assuming that my portfolio is rebalanced each day at zero transaction cost.

```
set.seed(10)
equal_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * mywealth
  n_days=20 # 4 business weeks
    equal_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        equal_wealthtracker[today] = mywealth
    }
    equal_wealthtracker

}
mywealth
```

```
## [1] 100356.9
```

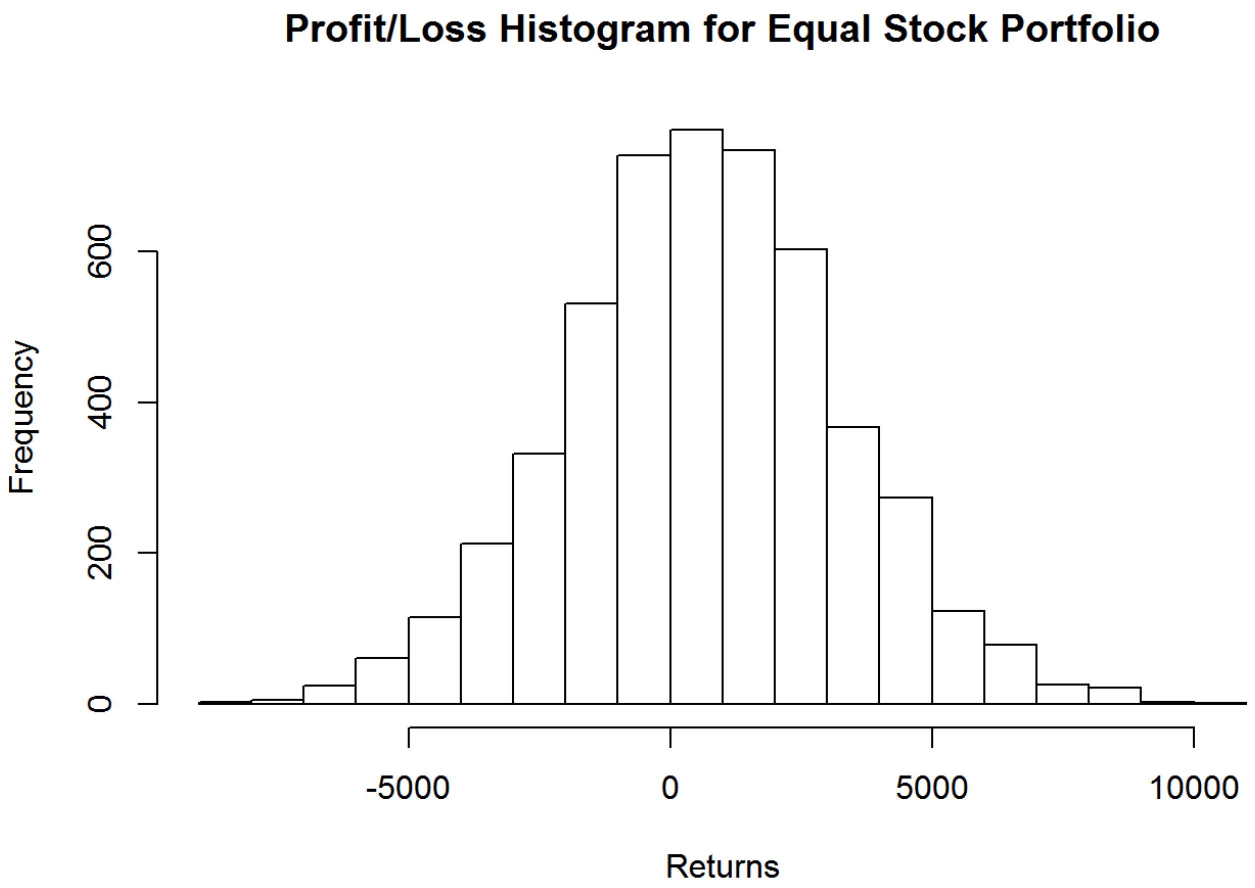
Plotting a histogram based on the simulation

```
hist(equal_sim[,n_days])
```



Checking for Profit/loss

```
hist(equal_sim[,n_days]- 100000,main="Profit/Loss Histogram for Equal Stock Portfoli
o",xlab="Returns")
```



Calculate 5% value at risk

```
quantile(equal_sim[,n_days], 0.05) - 100000
```

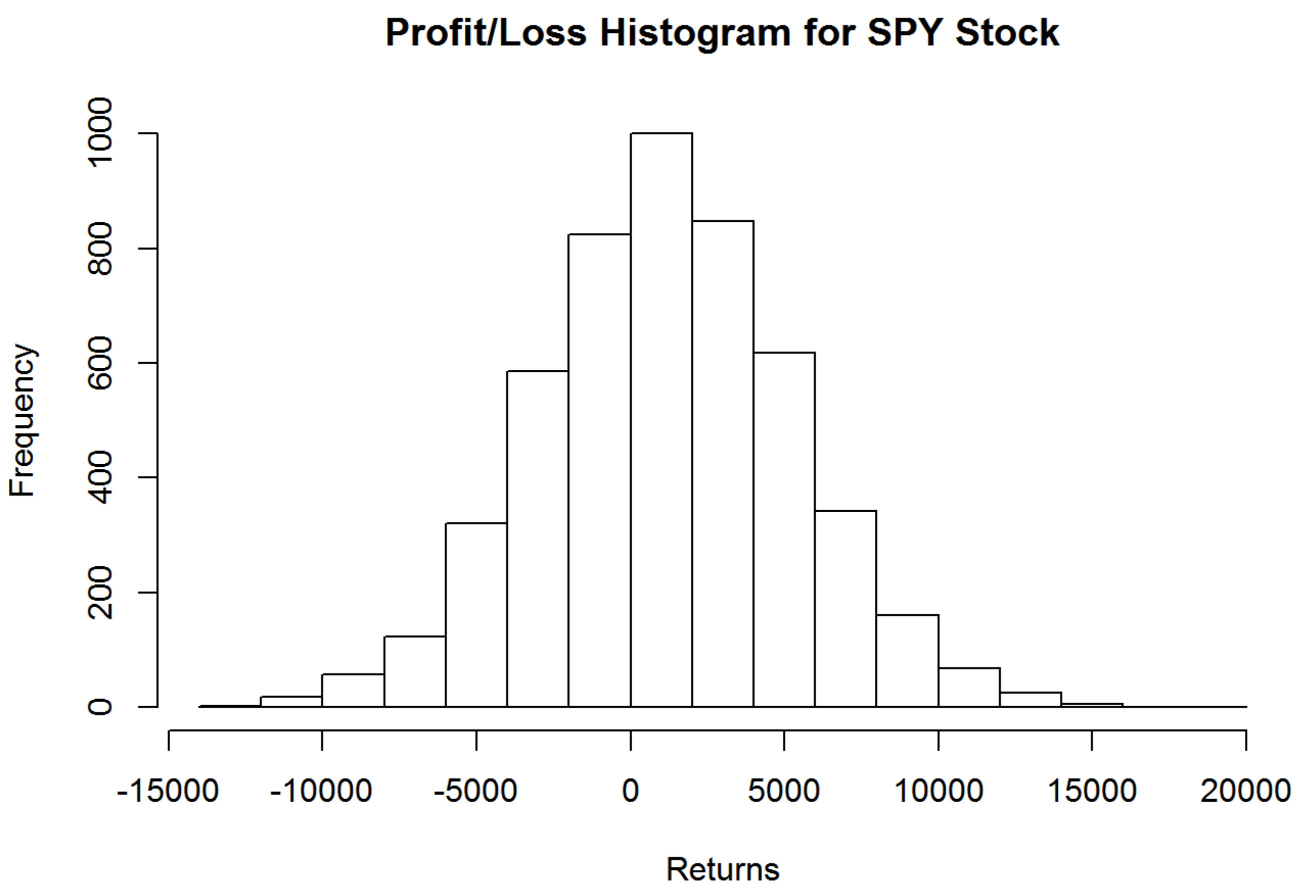
```
##        5%
## -3739.084
```

Now finding risks for each ticker **SPY**

```
set.seed(10)
SPY_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(1.0, 0.0, 0.0, 0.0, 0.0) # Putting all my money in SPY
    holdings = weights * mywealth
  n_days=20 # 4 business weeks
    SPY_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        SPY_wealthtracker[today] = mywealth
    }
    SPY_wealthtracker

}
mywealth
```

```
## [1] 99975.64
```

```
# Profit/loss
hist(SPY_sim[,n_days]- 100000,main="Profit/Loss Histogram for SPY Stock",xlab="Return
s")
```



```
# Calculate 5% value at risk
quantile(SPY_sim[,n_days], 0.05) - 100000
```

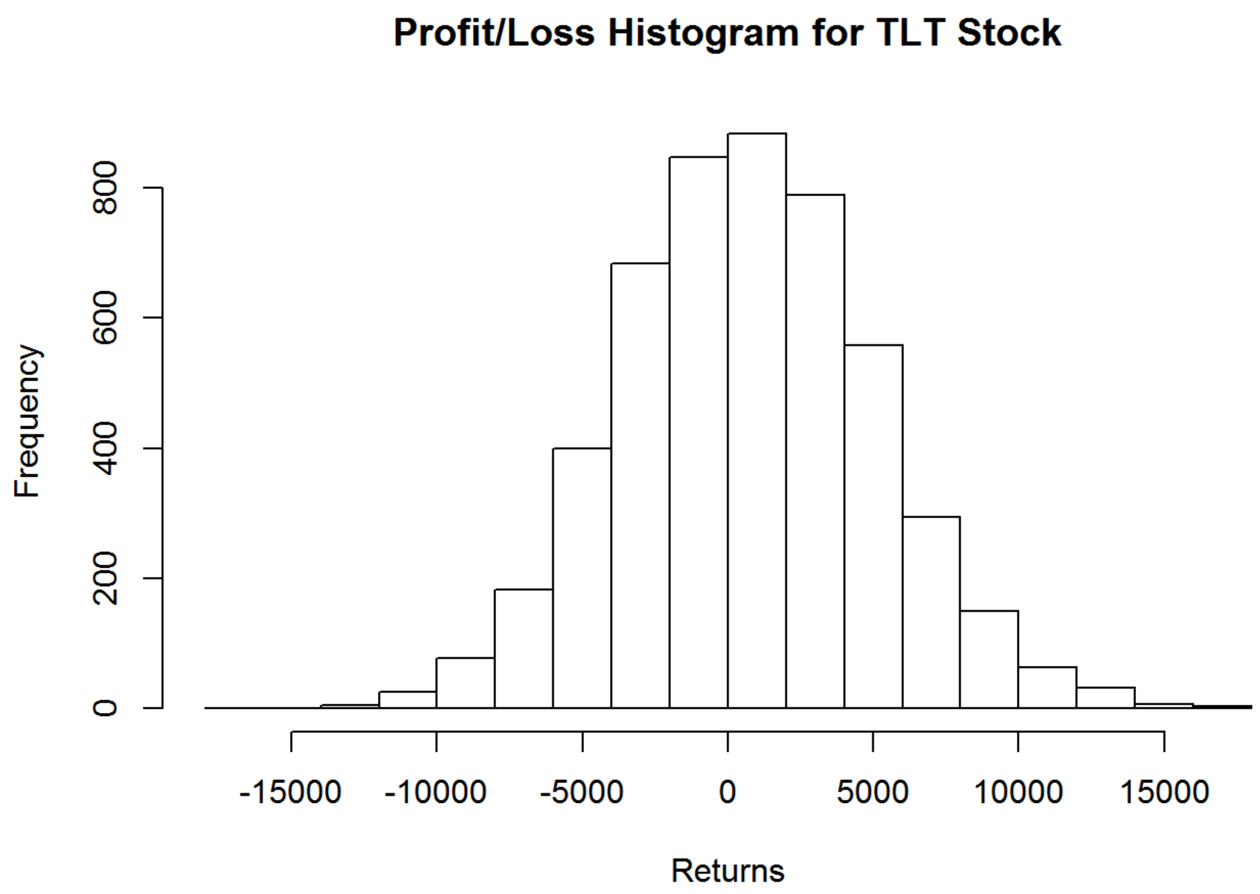
```
##        5%
## -5550.019
```

Now finding risks for each ticker **TLT**

```
set.seed(10)
TLT_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(0.0, 1.0, 0.0, 0.0, 0.0) # Putting all my money in TLT
  holdings = weights * mywealth
  n_days=20 # 4 business weeks
    TLT_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        TLT_wealthtracker[today] = mywealth
    }
    TLT_wealthtracker

}
mywealth
```

```
## [1] 99683.94
```

```
# Profit/loss
hist(TLT_sim[,n_days]- 100000,main="Profit/Loss Histogram for TLT Stock",xlab="Return
s")
```




```
# Calculate 5% value at risk
quantile(TLT_sim[,n_days], 0.95) - 100000
```

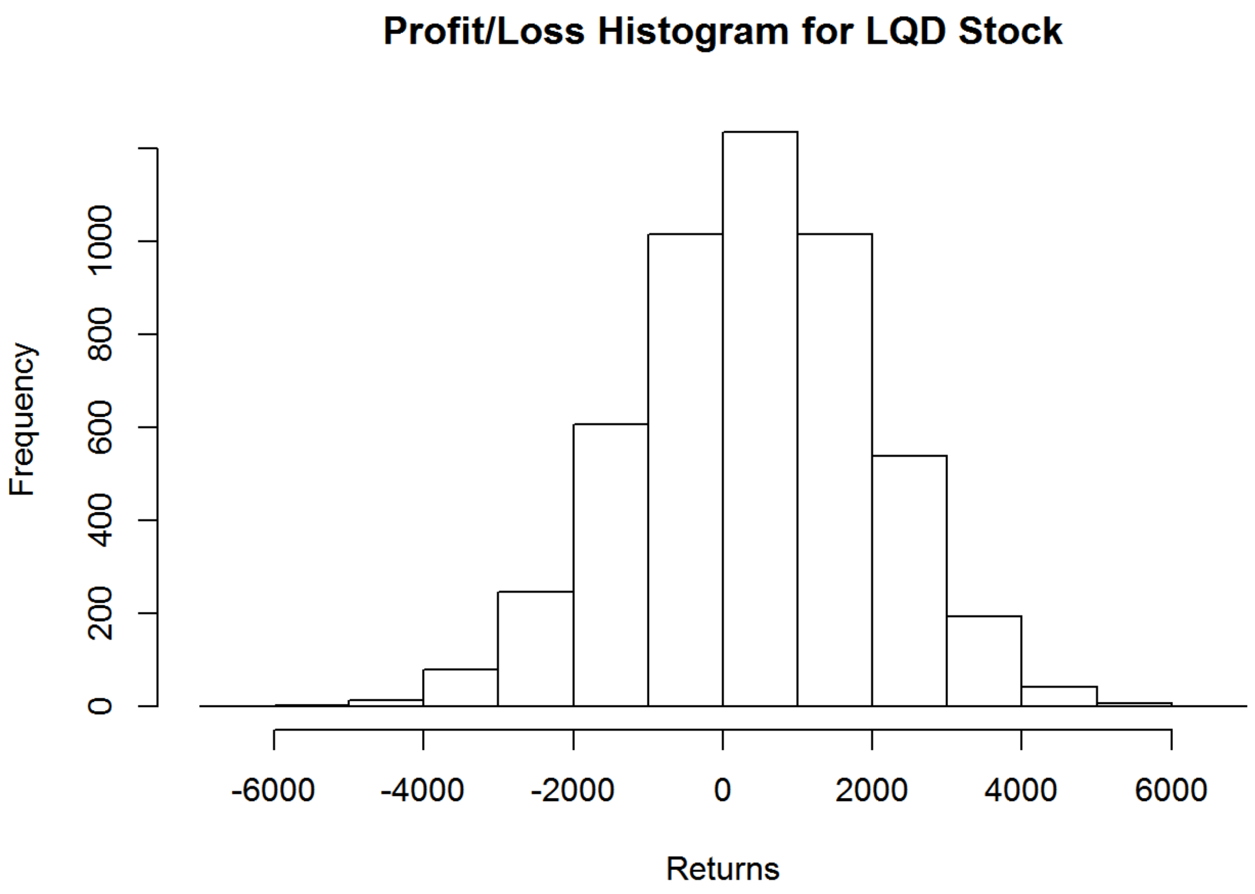
```
##      95% 
## 8020.941
```

Now finding risks for each ticker **LQD**

```
set.seed(10)
LQD_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(0.0, 0.0, 1.0, 0.0, 0.0) # Putting all my money in LQD
  holdings = weights * mywealth
  n_days=20 # 4 business weeks
  LQD_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        LQD_wealthtracker[today] = mywealth
    }
    LQD_wealthtracker

}
mywealth
```

```
## [1] 101461.9
```

```
# Profit/loss
hist(LQD_sim[,n_days]- 100000,main="Profit/Loss Histogram for LQD Stock",xlab="Return
s")
```




```
# Calculate 5% value at risk
quantile(LQD_sim[,n_days], 0.05) - 100000
```

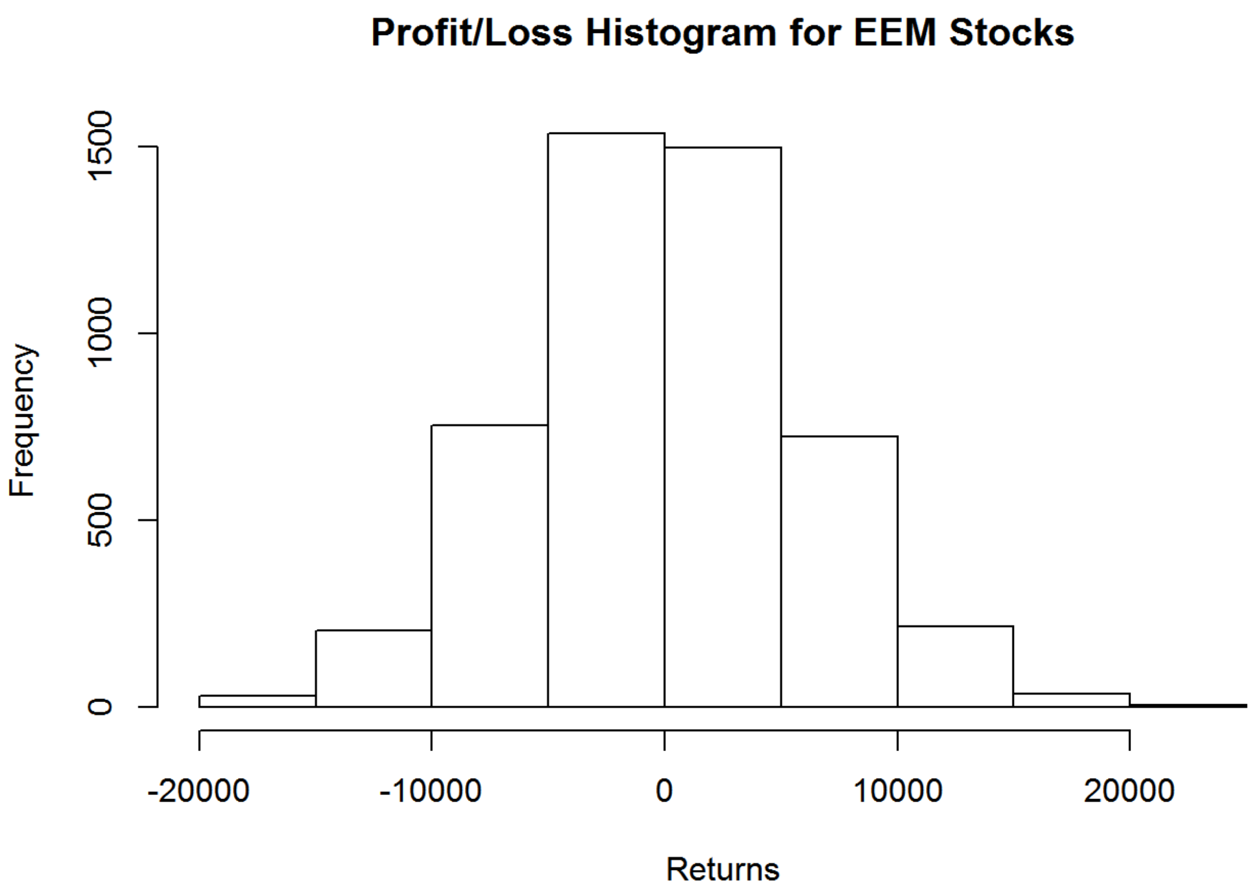
```
##        5%
## -2249.408
```

Now finding risks for each ticker **EEM**

```
set.seed(10)
EEM_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(0.0, 0.0, 0.0, 1.0, 0.0) # Putting all my money in EEM
  holdings = weights * mywealth
  n_days=20 # 4 business weeks
  EEM_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        EEM_wealthtracker[today] = mywealth
    }
    EEM_wealthtracker

}
mywealth
```

```
## [1] 92898.6
```

```
# Profit/loss
hist(EEM_sim[,n_days]- 100000,main="Profit/Loss Histogram for EEM Stocks",xlab="Return
s")
```



```
# Calculate 5% value at risk
quantile(EEM_sim[,n_days], 0.05) - 100000
```

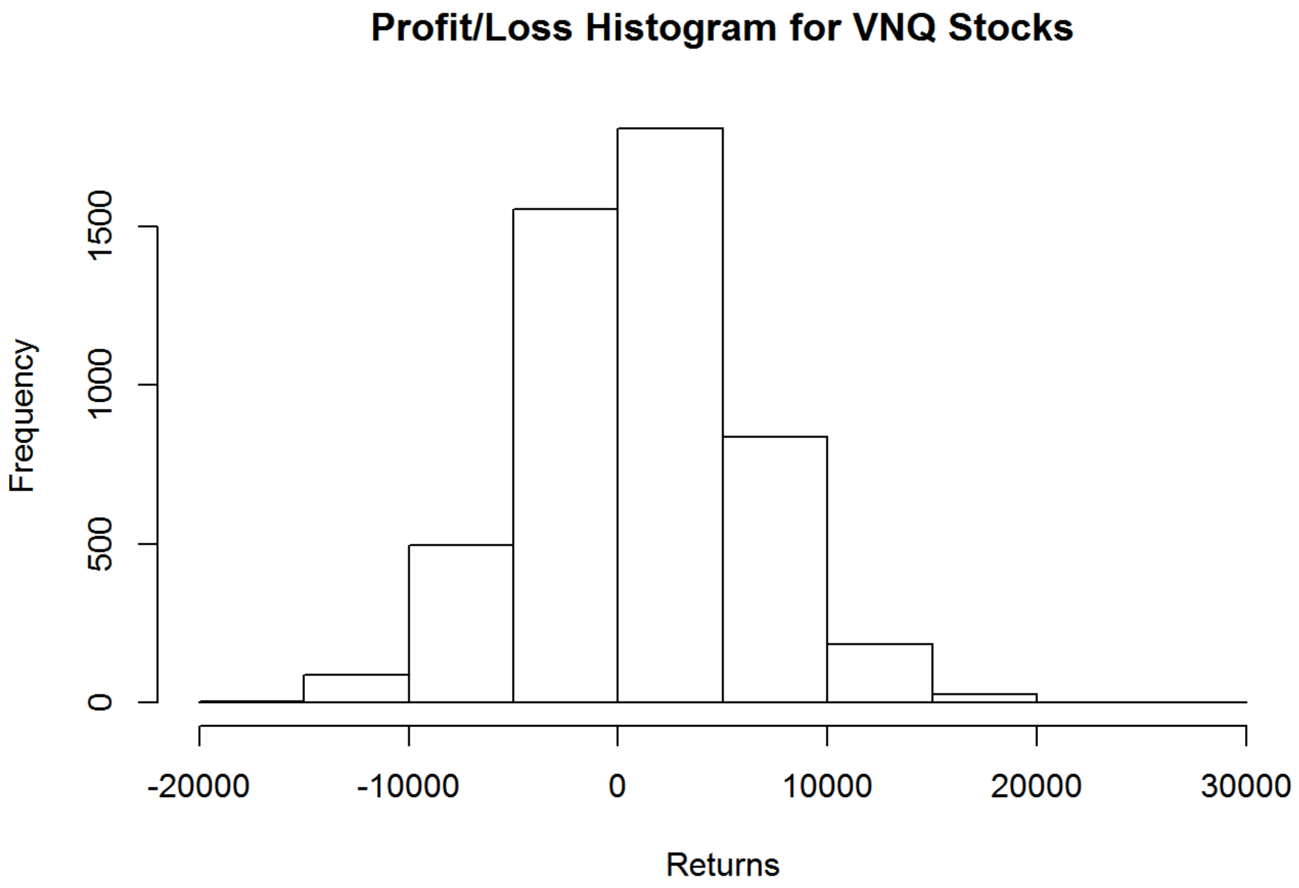
```
##        5%
## -9870.442
```

Now finding risks for each ticker **VNQ**

```
set.seed(10)
VNQ_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(0.0, 0.0, 0.0, 0.0, 1.0) # Putting all my money in EEM
  holdings = weights * mywealth
  n_days=20 # 4 business weeks
  VNQ_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
    holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        VNQ_wealthtracker[today] = mywealth
    }
    VNQ_wealthtracker

}
mywealth
```

```
## [1] 107794.7
```

```
# Profit/loss
hist(VNQ_sim[,n_days]- 100000,main="Profit/Loss Histogram for VNQ Stocks",xlab="Return
s")
```




```
# Calculate 5% value at risk
quantile(VNQ_sim[,n_days], 0.95) - 100000
```

```
##      95%
## 9654.908
```

Trying to see the risk and evaluating which stocks are risky and safe

```
quantile(SPY_sim[,n_days], 0.05) - 100000
```

```
##        5%
## -5550.019
```

```
quantile(TLT_sim[,n_days], 0.05) - 100000
```

```
##       5%
## -6369.41
```

```
quantile(LQD_sim[,n_days], 0.05) - 100000
```

```
##        5%
## -2249.408
```

```
quantile(EEM_sim[,n_days], 0.05) - 100000
```

```
##        5%
## -9870.442
```

```
quantile(VNQ_sim[,n_days], 0.05) - 100000
```

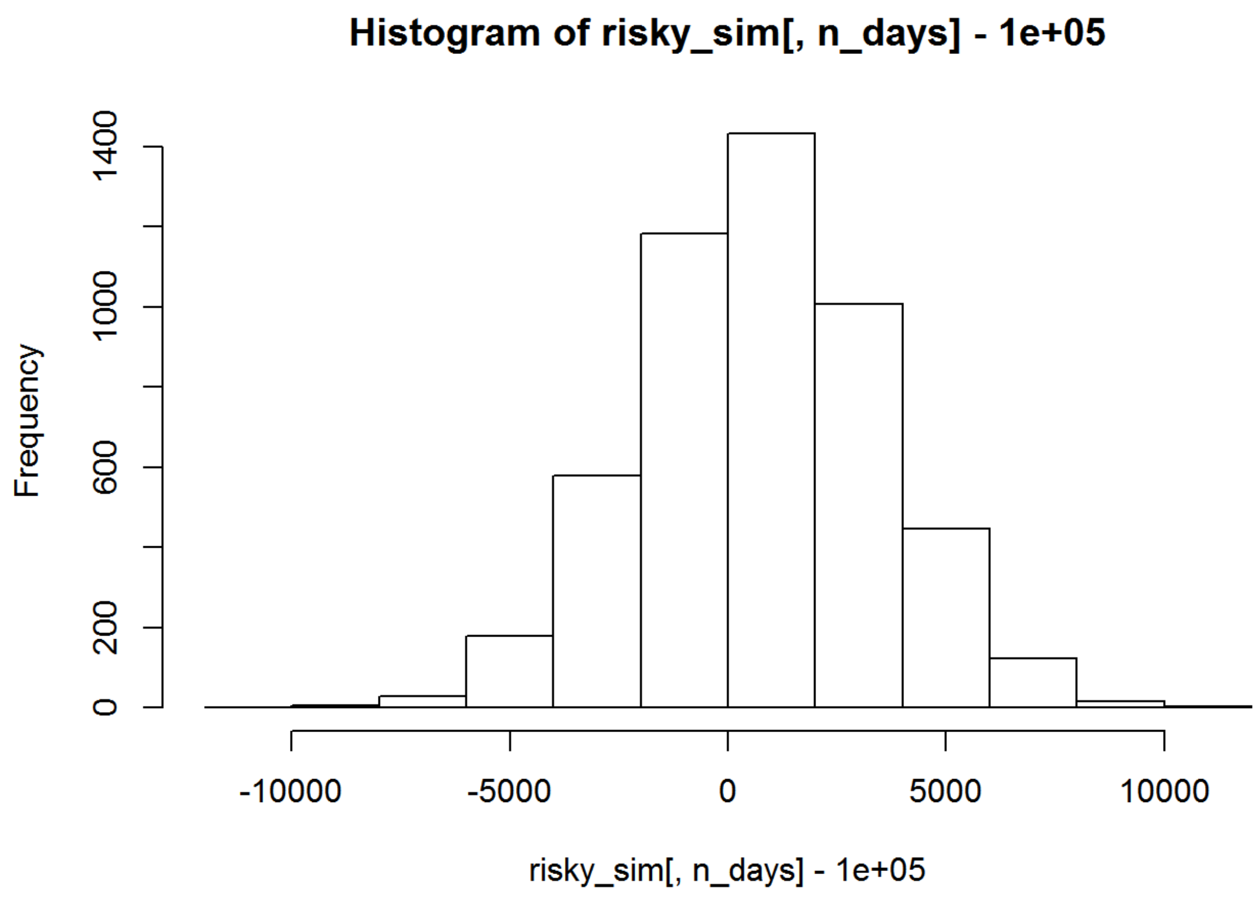
```
##        5%
## -7504.867
```

Simulating for a risky portfolio

```
set.seed(10)
risky_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(0.0, 0.5, 0.0, 0.1, 0.4)
    holdings = weights * mywealth
  n_days=20 # 4 business weeks
    risky_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        risky_wealthtracker[today] = mywealth
    }
    risky_wealthtracker

}
mywealth
```

```
## [1] 102232.3
```

```
# Profit/loss
hist(risky_sim[,n_days]- 100000)
```




```
# Calculate 5% value at risk
quantile(risky_sim[,n_days], 0.05) - 100000
```

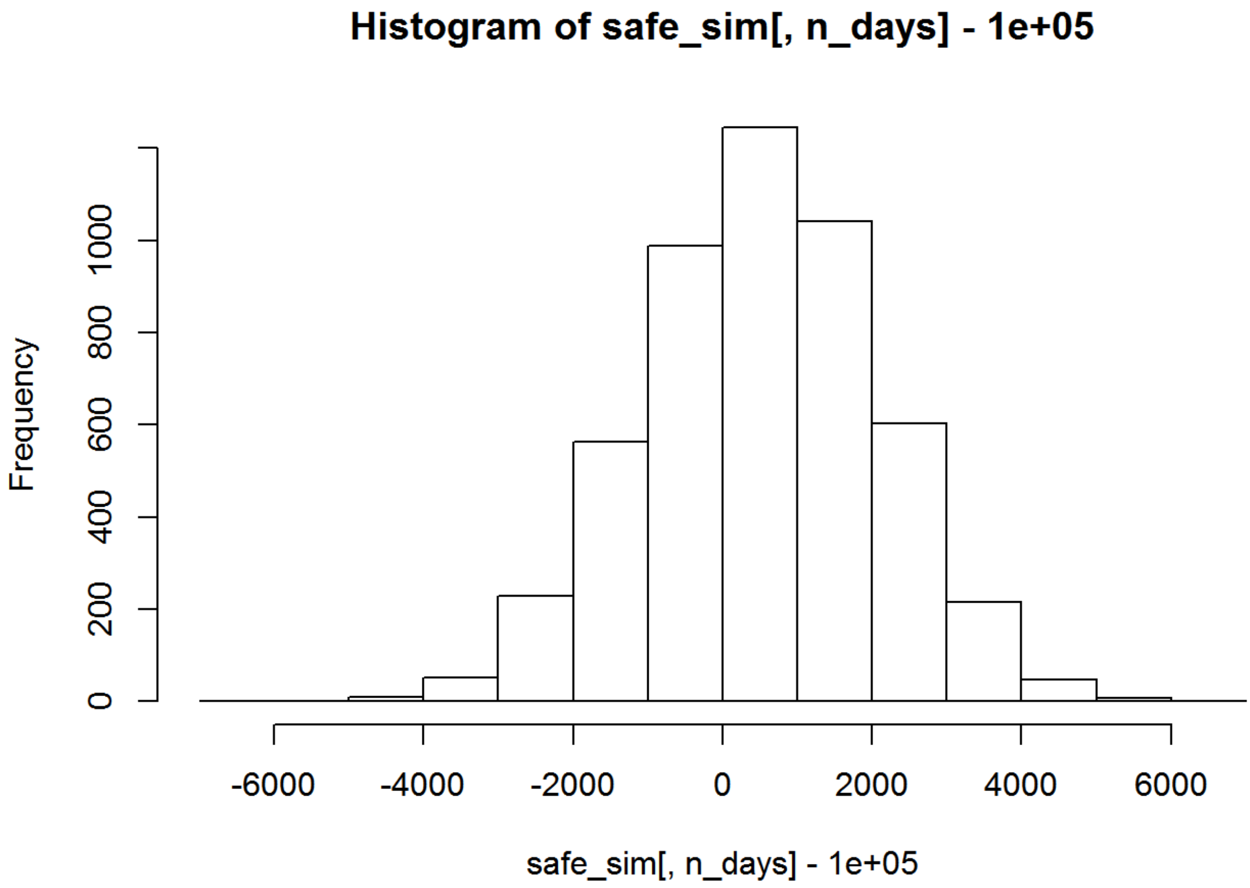
```
##        5%
## -3789.516
```

Simulating for a safe portfolio

```
set.seed(10)
safe_sim = foreach(i=1:5000, .combine='rbind') %do% {
  mywealth = 100000
  weights = c(0.1, 0.1, 0.8, 0.0, 0.0)
  holdings = weights * mywealth
  n_days=20 # 4 business weeks
    safe_wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
   return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * mywealth
        holdings = holdings + holdings*return.today
        mywealth = sum(holdings)
        safe_wealthtracker[today] = mywealth
    }
    safe_wealthtracker

}
mywealth
```

```
## [1] 101160.8
```

```
# Profit/loss
hist(safe_sim[,n_days]- 100000)
```




```
# Calculate 5% value at risk
quantile(safe_sim[,n_days], 0.05) - 100000
```

```
##        5%
## -2109.352
```

Question 3

Importing the wine data

```
wine<-read.csv("C:/Users/Ramyasai/Desktop/wine.csv")
attach(wine)
```

Encoding column quality as a factor To enable to run K means

```
wine$quality<-as.factor(wine$quality)
```

Selecting only the 11 chemical properties columns for the K means clustering and scaling the data after excluding quality and color columns

```
set.seed(25)
dfcolor<-wine$color
winescaled <- scale(wine[,-c(12,13)], center=TRUE, scale=TRUE)
```

Clustering the data using K means with K=2 Comparing the color column with the color of the datapoints in each of the two clusters

```
clusterall <- kmeans(winescaled, centers=2, nstart=50)
table(dfcolor,clusterall$cluster)
```

```
##
## dfcolor    1    2
##   red   1575   24
##   white   68 4830
```

1st cluster has 1585 red and 12 white wines only 14 red are being clustered erroneously 2nd cluster has 14 red and 4886 white wines i.e only 12 wines are being clustered erroneously

Identifying the centers and cluster after running K Means

```
clusterall$centers
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1     0.8286464        1.1678795  -0.3378091     -0.5903919  0.9216848
## 2    -0.2804833       -0.3953082   0.1143429      0.1998380 -0.3119753
##   free.sulfur.dioxide total.sulfur.dioxide    density         pH
## 1          -0.8316090           -1.1872380  0.6815493  0.5673286
## 2           0.2814861            0.4018607 -0.2306934 -0.1920315
##    sulphates     alcohol
## 1  0.8430523 -0.07569241
## 2 -0.2853595  0.02562065
```

```
#clusterall$cluster
```

Clustering with K=7 and comparing the results to check the quality of the wines of the clusters

```
clusterall <- kmeans(winescaled, centers=7, nstart=50)
dfquality=as.factor(wine$quality)
table(dfquality,clusterall$cluster)
```

```
##
## dfquality   1   2   3   4   5   6   7
##         3   7   7   4   5   1   2   4
##         4  24  61  21  64   2  29  15
##         5 656 470  80 414  20 298 200
##         6 645 347 528 538   9 503 266
##         7 122  42 451 144   1 179 140
##         8  21   2  96  30   0  30  14
##         9   1   0   4   0   0   0   0
```

When we compare the quality within these 7 clusters there is no significant pattern hence diving into 7 clusters is not an effective idea

PCA

Importing the wine data from local and scaling the first 11 chemical properties columns to run PCA

```
dim(wine)
```

```
## [1] 6497   13
```

```
wine<-read.csv("C:/Users/Ramyasai/Desktop/wine.csv")
winescaled <- scale(wine[,-c(12,13)], center=TRUE, scale=TRUE)
```

Running PCA on the 11 chemical properties

```
pc1 <-prcomp(winescaled, scale.=TRUE)
names(pc1)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```
pc1$scale
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    1                    1                    1
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    1                    1                    1
## total.sulfur.dioxide              density                   pH
##                    1                    1                    1
##            sulphates              alcohol
##                    1                    1
```

Look at the basic plotting and summary methods

```
pc1
```

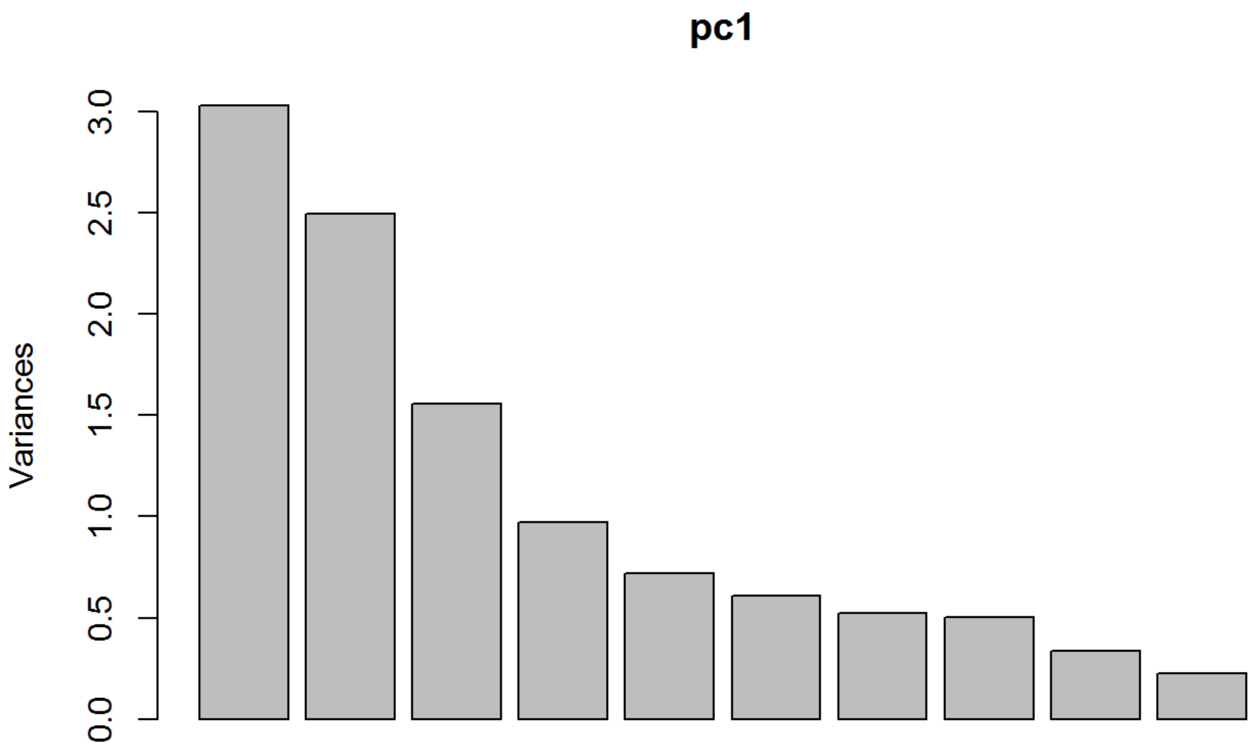
```
## Standard deviations:
##  [1] 1.7406518 1.5791852 1.2475364 0.9851660 0.8484544 0.7793021 0.7232971
##  [8] 0.7081739 0.5805377 0.4771748 0.1811927
##
## Rotation:
##                              PC1         PC2         PC3         PC4
## fixed.acidity        -0.23879890  0.33635454 -0.43430130  0.16434621
## volatile.acidity     -0.38075750  0.11754972  0.30725942  0.21278489
## citric.acid           0.15238844  0.18329940 -0.59056967 -0.26430031
## residual.sugar        0.34591993  0.32991418  0.16468843  0.16744301
## chlorides            -0.29011259  0.31525799  0.01667910 -0.24474386
## free.sulfur.dioxide   0.43091401  0.07193260  0.13422395 -0.35727894
## total.sulfur.dioxide  0.48741806  0.08726628  0.10746230 -0.20842014
## density              -0.04493664  0.58403734  0.17560555  0.07272496
## pH                   -0.21868644 -0.15586900  0.45532412 -0.41455110
## sulphates            -0.29413517  0.19171577 -0.07004248 -0.64053571
## alcohol              -0.10643712 -0.46505769 -0.26110053 -0.10680270
##                             PC5         PC6         PC7          PC8
## fixed.acidity        -0.1474804 -0.20455371 -0.28307944  0.401235645
## volatile.acidity      0.1514560 -0.49214307 -0.38915976 -0.087435088
## citric.acid          -0.1553487  0.22763380 -0.38128504 -0.293412336
## residual.sugar       -0.3533619 -0.23347775  0.21797554 -0.524872935
## chlorides             0.6143911  0.16097639 -0.04606816 -0.471516850
## free.sulfur.dioxide   0.2235323 -0.34005140 -0.29936325  0.207807585
## total.sulfur.dioxide  0.1581336 -0.15127722 -0.13891032  0.128621319
## density              -0.3065613  0.01874307 -0.04675897  0.004831136
## pH                   -0.4533764  0.29657890 -0.41890702 -0.028643277
## sulphates            -0.1365769 -0.29692579  0.52534311  0.165818022
## alcohol              -0.1888920 -0.51837780 -0.10410343 -0.399233887
##                             PC9         PC10          PC11
## fixed.acidity         0.3440567 -0.281267685 -0.3346792663
## volatile.acidity     -0.4969327  0.152176731 -0.0847718098
## citric.acid          -0.4026887  0.234463340  0.0011089514
## residual.sugar        0.1080032 -0.001372773 -0.4497650778
## chlorides             0.2964437 -0.196630217 -0.0434375867
## free.sulfur.dioxide   0.3666563  0.480243340  0.0002125351
## total.sulfur.dioxide -0.3206955 -0.713663486  0.0626848131
## density               0.1128800 -0.003908289  0.7151620723
## pH                    0.1278367 -0.141310977 -0.2063605036
## sulphates            -0.2077642  0.045959499 -0.0772024671
## alcohol               0.2518903 -0.205053085  0.3357018783
```

Summary of pc1 gives the std error, proportion of variance and cumulative proportion
7 principal components cover 88% of the variance

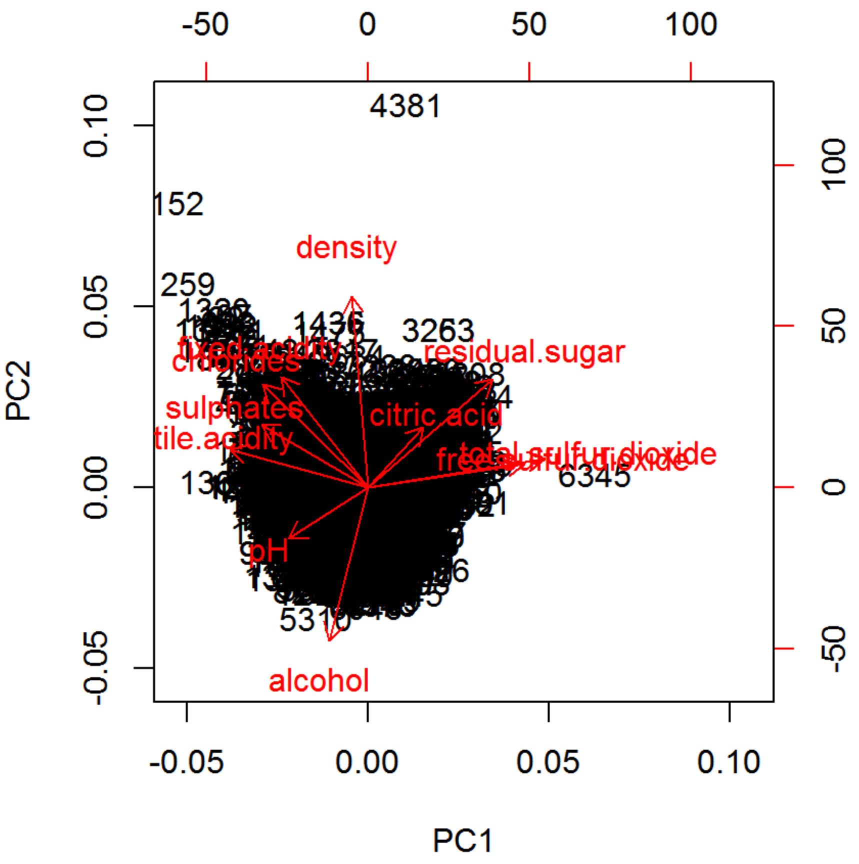
```
summary(pc1)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.7407 1.5792 1.2475 0.98517 0.84845 0.77930
## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521
## Cumulative Proportion  0.2754 0.5021 0.6436 0.73187 0.79732 0.85253
##                            PC7     PC8     PC9   PC10    PC11
## Standard deviation     0.72330 0.70817 0.58054 0.4772 0.18119
## Proportion of Variance 0.04756 0.04559 0.03064 0.0207 0.00298
## Cumulative Proportion  0.90009 0.94568 0.97632 0.9970 1.00000
```

```
plot(pc1)
```




```
biplot(pc1)
```



plot function is used to plot the variance on y axis and the principal components on the x axis biplot of the principal component A more informative biplot

```
loadings = pc1$rotation
loadings
```

```
##                             PC1         PC2         PC3         PC4
## fixed.acidity        -0.23879890  0.33635454 -0.43430130  0.16434621
## volatile.acidity     -0.38075750  0.11754972  0.30725942  0.21278489
## citric.acid           0.15238844  0.18329940 -0.59056967 -0.26430031
## residual.sugar        0.34591993  0.32991418  0.16468843  0.16744301
## chlorides            -0.29011259  0.31525799  0.01667910 -0.24474386
## free.sulfur.dioxide   0.43091401  0.07193260  0.13422395 -0.35727894
## total.sulfur.dioxide  0.48741806  0.08726628  0.10746230 -0.20842014
## density              -0.04493664  0.58403734  0.17560555  0.07272496
## pH                   -0.21868644 -0.15586900  0.45532412 -0.41455110
## sulphates            -0.29413517  0.19171577 -0.07004248 -0.64053571
## alcohol              -0.10643712 -0.46505769 -0.26110053 -0.10680270
##                            PC5         PC6         PC7          PC8
## fixed.acidity        -0.1474804 -0.20455371 -0.28307944  0.401235645
## volatile.acidity      0.1514560 -0.49214307 -0.38915976 -0.087435088
## citric.acid          -0.1553487  0.22763380 -0.38128504 -0.293412336
## residual.sugar       -0.3533619 -0.23347775  0.21797554 -0.524872935
## chlorides             0.6143911  0.16097639 -0.04606816 -0.471516850
## free.sulfur.dioxide   0.2235323 -0.34005140 -0.29936325  0.207807585
## total.sulfur.dioxide  0.1581336 -0.15127722 -0.13891032  0.128621319
## density              -0.3065613  0.01874307 -0.04675897  0.004831136
## pH                   -0.4533764  0.29657890 -0.41890702 -0.028643277
## sulphates            -0.1365769 -0.29692579  0.52534311  0.165818022
## alcohol              -0.1888920 -0.51837780 -0.10410343 -0.399233887
##                            PC9         PC10          PC11
## fixed.acidity         0.3440567 -0.281267685 -0.3346792663
## volatile.acidity     -0.4969327  0.152176731 -0.0847718098
## citric.acid          -0.4026887  0.234463340  0.0011089514
## residual.sugar        0.1080032 -0.001372773 -0.4497650778
## chlorides             0.2964437 -0.196630217 -0.0434375867
## free.sulfur.dioxide   0.3666563  0.480243340  0.0002125351
## total.sulfur.dioxide -0.3206955 -0.713663486  0.0626848131
## density               0.1128800 -0.003908289  0.7151620723
## pH                    0.1278367 -0.141310977 -0.2063605036
## sulphates            -0.2077642  0.045959499 -0.0772024671
## alcohol               0.2518903 -0.205053085  0.3357018783
```

```
scores = pc1$x
#scores
```

Tabulating the loadings

```
o1 = order(loadings[,1])
colnames(winescaled)[head(o1,5)]
```

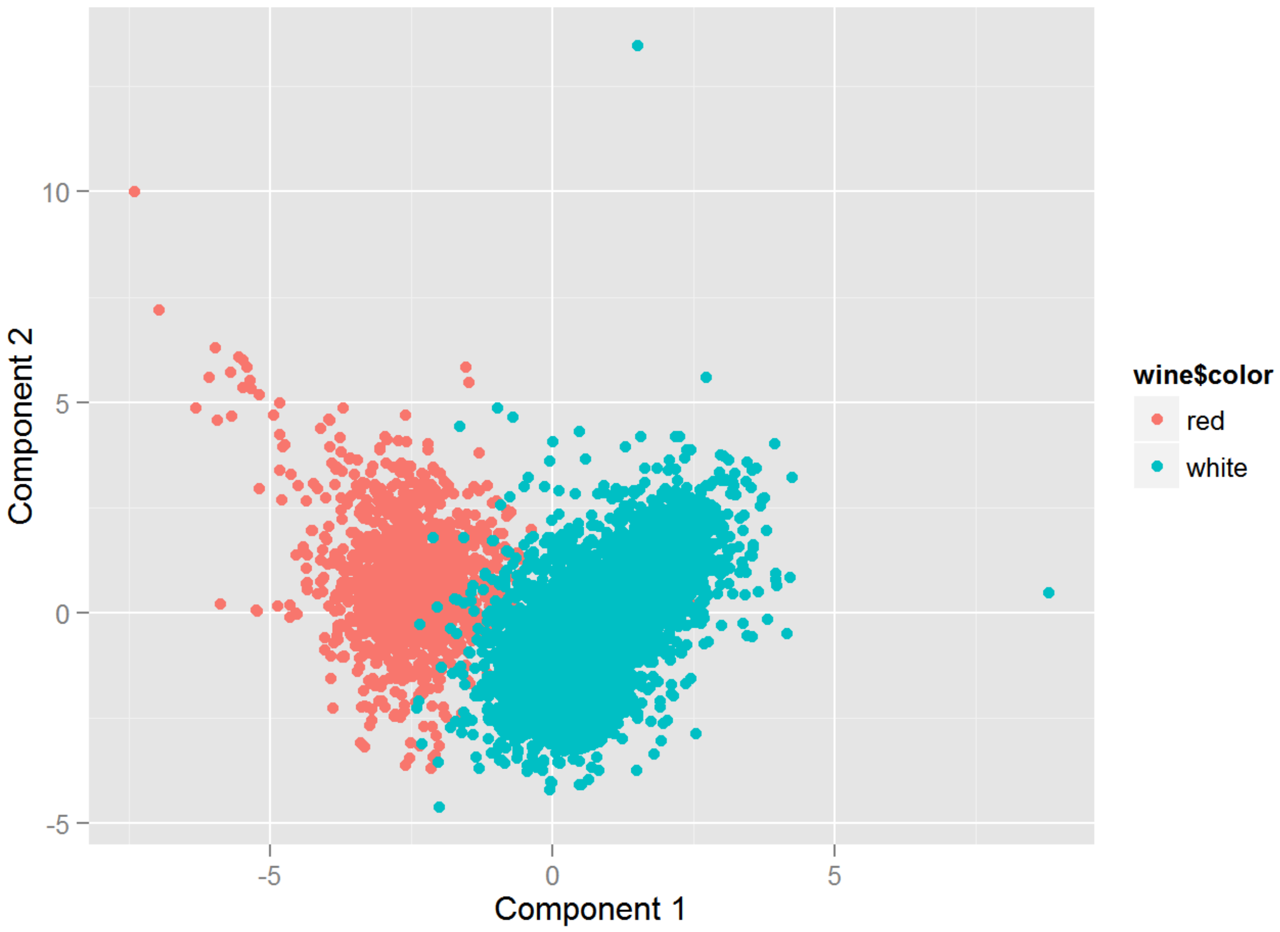
```
## [1] "volatile.acidity" "sulphates"        "chlorides"
## [4] "fixed.acidity"    "pH"
```

```
colnames(winescaled)[tail(o1,5)]
```

```
## [1] "density"              "citric.acid"          "residual.sugar"
## [4] "free.sulfur.dioxide"  "total.sulfur.dioxide"
```

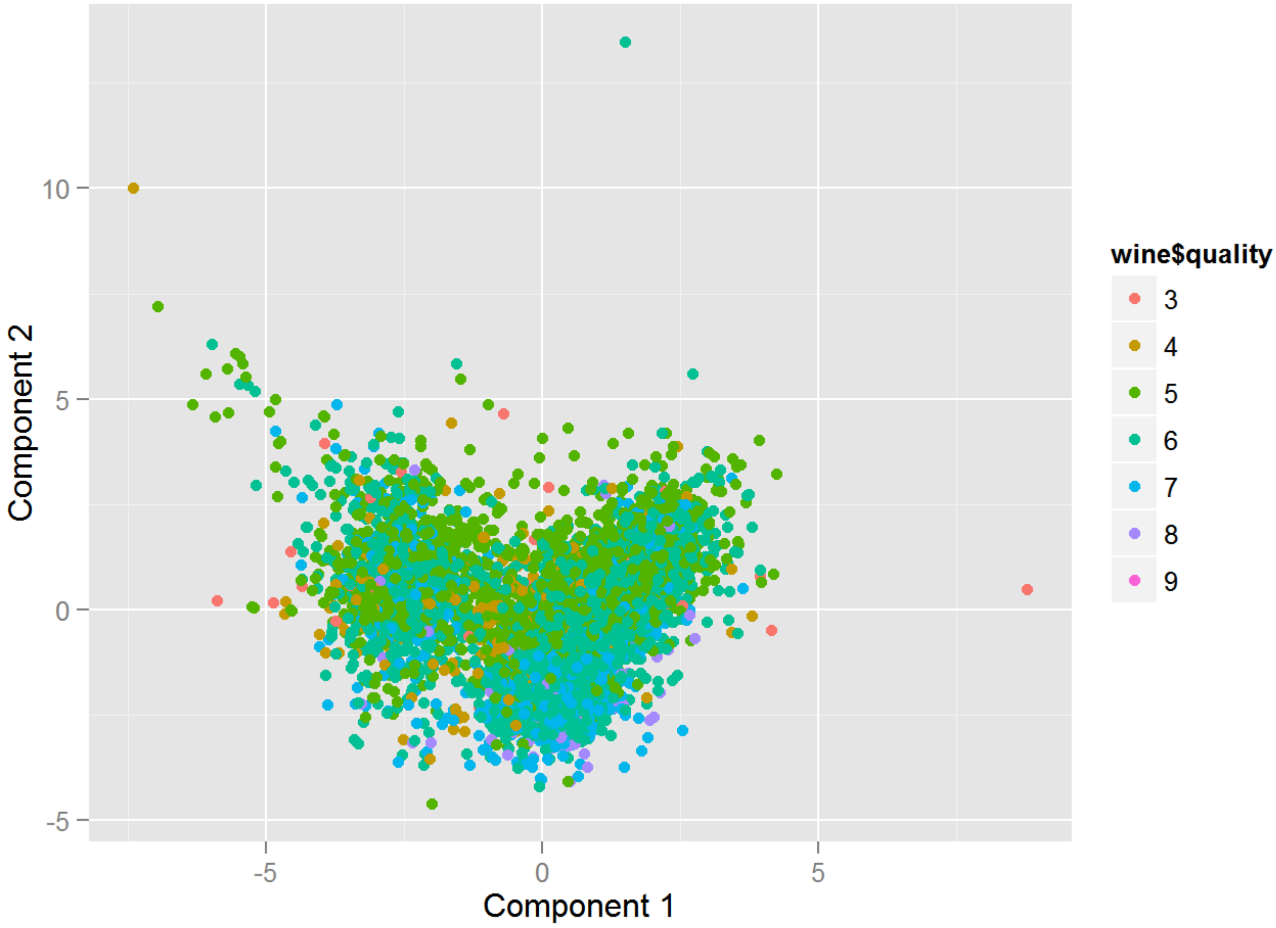
Plotting the scores and the color using qplot

```
View(scores)
library("ggplot2")
qplot(scores[,1], scores[,2], color=wine$color, xlab='Component 1', ylab='Component 2')
```



Plotting the scores and quality using qplot

```
wine$quality<-as.factor(wine$quality)
qplot(scores[,1], scores[,2], color=wine$quality, xlab='Component 1', ylab='Component
2')
```



After looking at the qplot between quality and the scores for first and second principal components we understand that the datapoints are quite cluttered and cant make the difference significantly.

Conclusion: Both K means and PCA are able to cluster color very well.But for quality both K means and PCA are not showing significant clusters. Still I will go with K means beacause the error for color is very less and for K means

Question 4
** 4) Market Segmentation **

IMporting the social marketing data

```
##Social Marketing

par(mfrow=c(1,1))
social_marketing = read.csv("C:/Users/Ramyasai/Desktop/Predictive modelling/James/STA38
0-master(1)/STA380-master/data/social_marketing.csv", header=TRUE)
```

After basic exploration of the data we are removing the rows which are spam and adult data

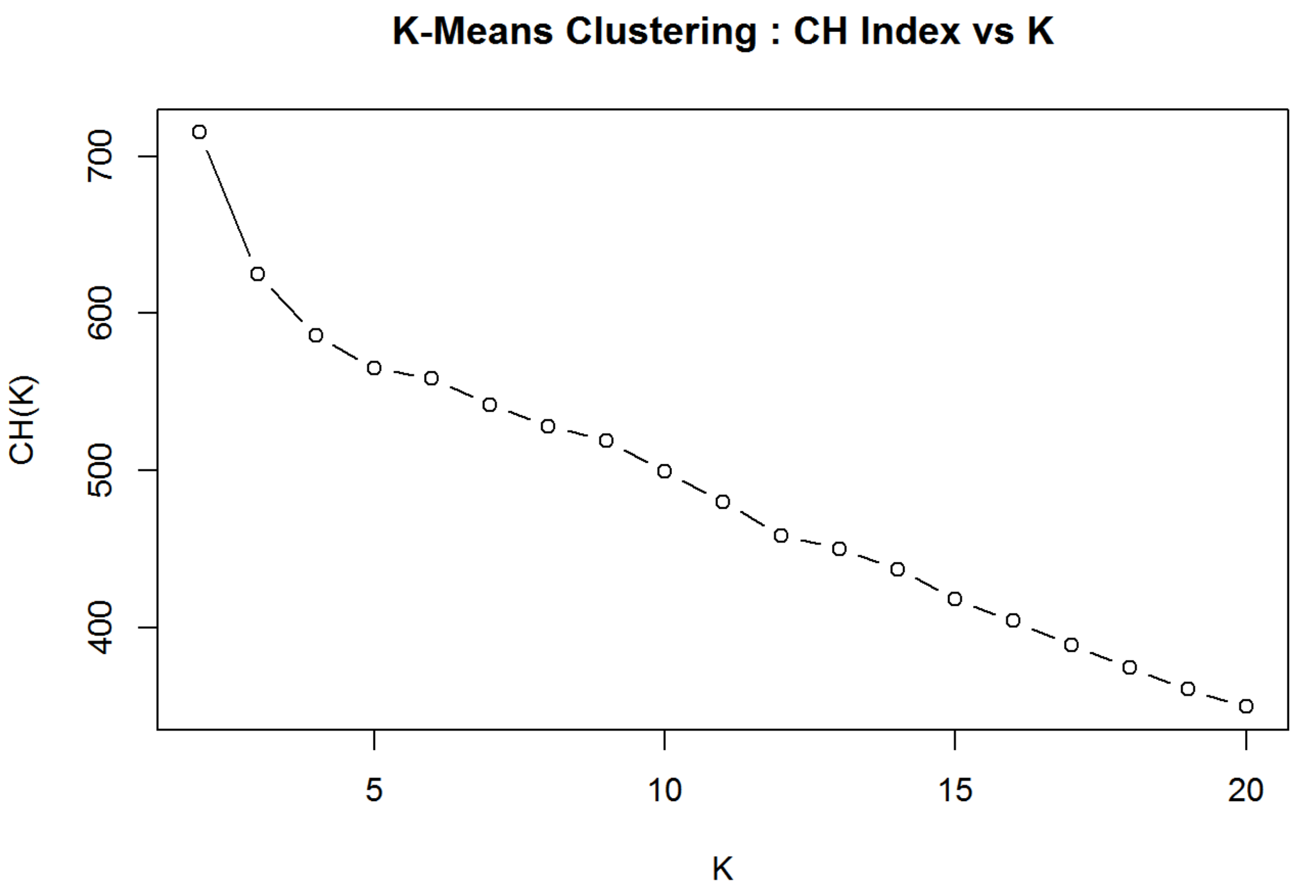
```
social_marketing=social_marketing[-(social_marketing$spam >= 0 & social_marketing$adul
t>=0),]
```

Removing extra columns like uncategorized,chatter and photo sharing as we are not going to concentrate on this

```
social_marketing = social_marketing[,c(-6,-2,-5)]
```

Calculate the CH index to calculate the number of clusters to identify the K size

```
social_marketing_scaled <- scale(social_marketing[,-1], center=TRUE, scale=TRUE)

#To compute the value of k

n= dim(social_marketing_scaled)[1]
ch = numeric(length=20)
set.seed = 12

for(i in 2:20){

  kmean = kmeans(social_marketing_scaled, centers=i, nstart=25)
  ch[i-1] = (sum(kmean$betweenss)/(i-1))/(kmean$tot.withinss/(n-i))
}

plot(2:20, ch[1:19], xlab='K', ylab='CH(K)', type='b', main='K-Means Clustering : CH In
dex vs K' )
```



We can understand that k value is should be taken close to 5 from the graph

```
set.seed = 12
cluster_all <- kmeans(social_marketing_scaled, centers=5, nstart=25)
names(cluster_all)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
cluster1 = cluster_all$cluster
social_marketing$cluster = cluster1
```

For Cluster 1

```
cluster1 = subset(social_marketing,cluster == 1)

head(sort(sapply(cluster1[,c(-35,-1)],mean),decreasing=TRUE))
```

```
## health_nutrition personal_fitness          cooking         outdoors
##        11.841321         6.334398         3.287540         2.685836
##             food   current_events
##         2.117146         1.544196
```

The group of people under cluster 4 can be identified as those interested in politics, travel and news. These can be classified as people of middle age group

Cluster 2

```
cluster2 = subset(social_marketing,cluster == 2)

head(sort(sapply(cluster2[,c(-35,-1)],mean),decreasing=TRUE))
```

```
##          cooking          fashion           beauty health_nutrition
##        10.540362         5.451400         3.825371         2.144975
##      college_uni         shopping
##         1.957166         1.906096
```

We find that the data shows features about people who are young and in college

Cluster 3

```
cluster3 = subset(social_marketing,cluster == 3)

head(sort(sapply(cluster3[,c(-35,-1)],mean),decreasing=TRUE))
```

```
##      college_uni   current_events         shopping    online_gaming
##         1.528548         1.453455         1.286719         1.176252
##           travel health_nutrition
##         1.113157         1.063715
```

The group of people under cluster 3 could be classified as those interested in health

Cluster 4

```
cluster4 = subset(social_marketing,cluster == 4)

head(sort(sapply(cluster4[,c(-35,-1)],mean),decreasing=TRUE))
```

```
## sports_fandom      religion          food     parenting        school
##      5.887612      5.272031      4.556833      4.029374      2.697318
##        family
##      2.490421
```

We find that the segment of observations here are young citizen who like to cook, fashion, beauty and shopping

cluster 5

```
cluster5 = subset(social_marketing,cluster == 5)

head(sort(sapply(cluster5[,c(-35,-1)],mean),decreasing=TRUE))
```

```
##      politics        travel          news     computers    automotive
##      8.823120      5.559889      5.210306      2.477716      2.331476
## sports_fandom
##      1.993036
```

This cluster classifies people who are parents and in their middle