Assignment1

Abhishek Sanghavi

Friday, August 07, 2015

Exploratory Data Analysis

Read the csv for the voting across counties in Georgia:

```
georgiaData = read.csv('../data/georgia2000.csv')
```

Calculate the undercounts and the fraction of undercounts

```
georgiaData$underCount<-georgiaData$ballots-georgiaData$votes
georgiaData$underCountPerCent<-round(100*(georgiaData$underCount/georgiaDa
ta$ballots),2)
```

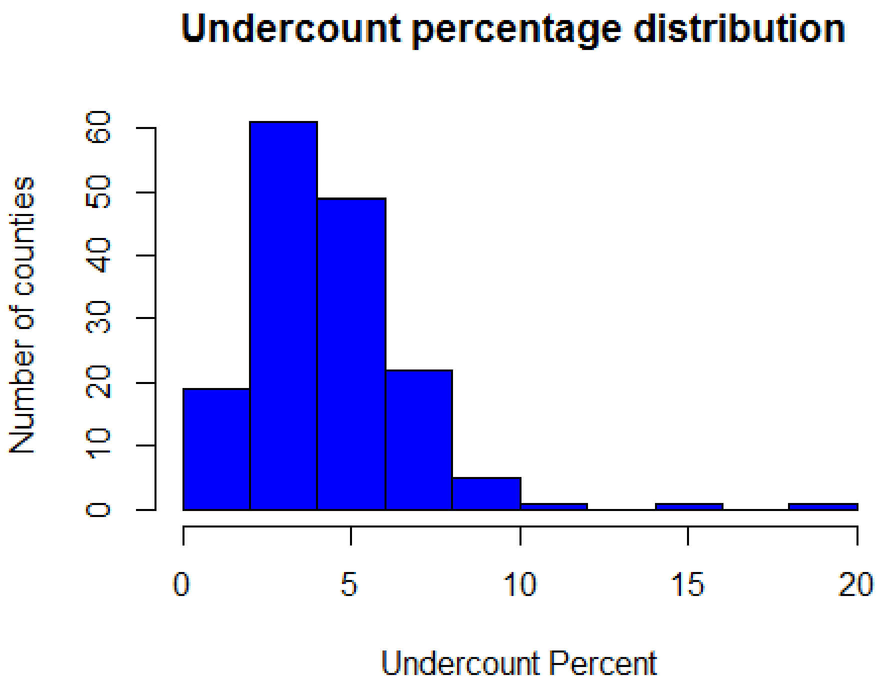
Summary of Georgia dataset

- There are a total of 159 counties and each county has a different equipment for voting (4 different equiments - LEVER, OPTICAL, PUNCH, PAPER)
- 2596633 out of 2691314 ballots, were counted leading to an undercount of 3.52% in Georgia

```
summary(georgiaData)

##       county       ballots            votes              equip
##  APPLING :  1   Min.   :   881   Min.   :   832   LEVER   :74
##  ATKINSON:  1   1st Qu.:  3694   1st Qu.:  3506   OPTICAL:66
##  BACON   :  1   Median :  6712   Median :  6299   PAPER  : 2
##  BAKER   :  1   Mean   : 16927   Mean   : 16331   PUNCH  :17
##  BALDWIN :  1   3rd Qu.: 12251   3rd Qu.: 11846
##  BANKS   :  1   Max.   :280975   Max.   :263211
##  (Other) :153
##       poor             urban              atlanta              perAA
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.1115
##  Median :0.0000   Median :0.0000   Median :0.00000   Median :0.2330
```

```
##  Mean    :0.4528   Mean    :0.2642   Mean    :0.09434   Mean    :0.2430
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.3480
##  Max.    :1.0000   Max.    :1.0000   Max.    :1.00000   Max.    :0.7650
##
##       gore              bush           underCount       underCountPerCent
##  Min.    :   249   Min.    :   271   Min.    :    0.0   Min.    : 0.000
##  1st Qu.:  1386   1st Qu.:  1804   1st Qu.:  152.5   1st Qu.: 2.780
##  Median :  2326   Median :  3597   Median :  296.0   Median : 3.980
##  Mean    :  7020   Mean    :  8929   Mean    :  595.5   Mean    : 4.379
##  3rd Qu.:  4430   3rd Qu.:  7468   3rd Qu.:  523.5   3rd Qu.: 5.650
##  Max.    :154509   Max.    :140494   Max.    :17764.0   Max.    :18.810
##
```

```
hist(georgiaData$underCountPerCent, main = "Undercount percentage distribu
tion ", ylab="Number of counties",xlab = "Undercount Percent",col = "blue")
```

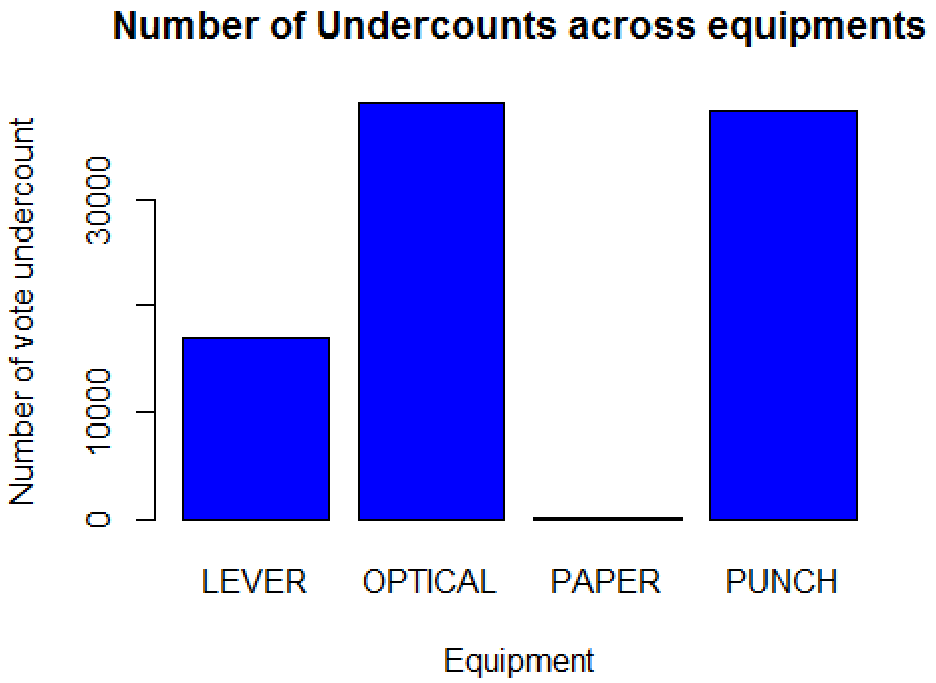


Deciphering the reasons of vote undercount

We can analyze the equipments responsible for most invalid votes

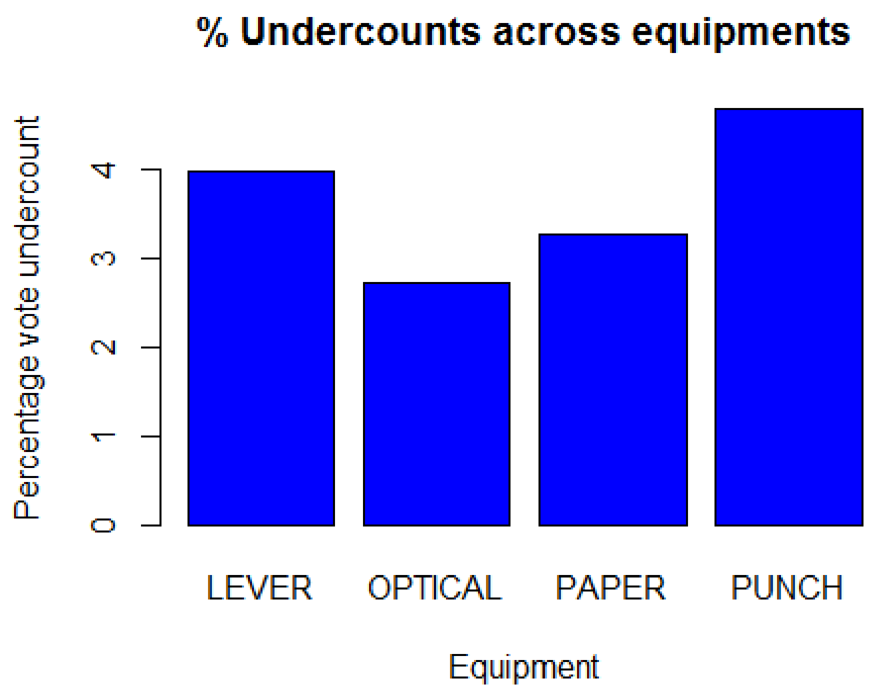
```
votes_by_equip= aggregate(cbind(ballots,votes)~equip,data=georgiaData,sum)
```

```
votes_by_equip$under_percent<-100*(votes_by_equip$ballots-votes_by_equip$votes)/(votes_by_equip$ballots)


barplot((votes_by_equip$ballots-votes_by_equip$votes),col="blue",main="Number of Undercounts across equipments",names.arg = votes_by_equip$equip,xlab = "Equipment",ylab = "Number of vote undercount")
```



Optical has the highest and Paper based equipment has the least number of vote undercounts

```
barplot(votes_by_equip$under_percent,col="blue",main="% Undercounts across equipments",names.arg = votes_by_equip$equip,xlab = "Equipment",ylab = "Percentage vote undercount")
```



Normalizing the number of ballots in each equipment, we realize that punch has the highest % of undercounts as compared to optical (which has the least)

It can be concluded that people have issues with interpreting the PUNCH and LEVER ballot system as compared to others

Impact on the poor and minority communities

Poor voters

```
Georgiapoor<-georgiaData[georgiaData$poor==1,]

poor=aggregate(cbind(ballots,votes)~equip,data=Georgiapoor,sum)
poor$undercountPercent<-100*(poor$ballots-poor$votes)/(poor$ballots)

poor

##     equip ballots  votes undercountPercent
## 1   LEVER  219254 209054          4.652139
## 2 OPTICAL  114465 107008          6.514655
## 3   PAPER    3454   3341          3.271569
## 4   PUNCH   23612  22183          6.052007
```

Rich voters

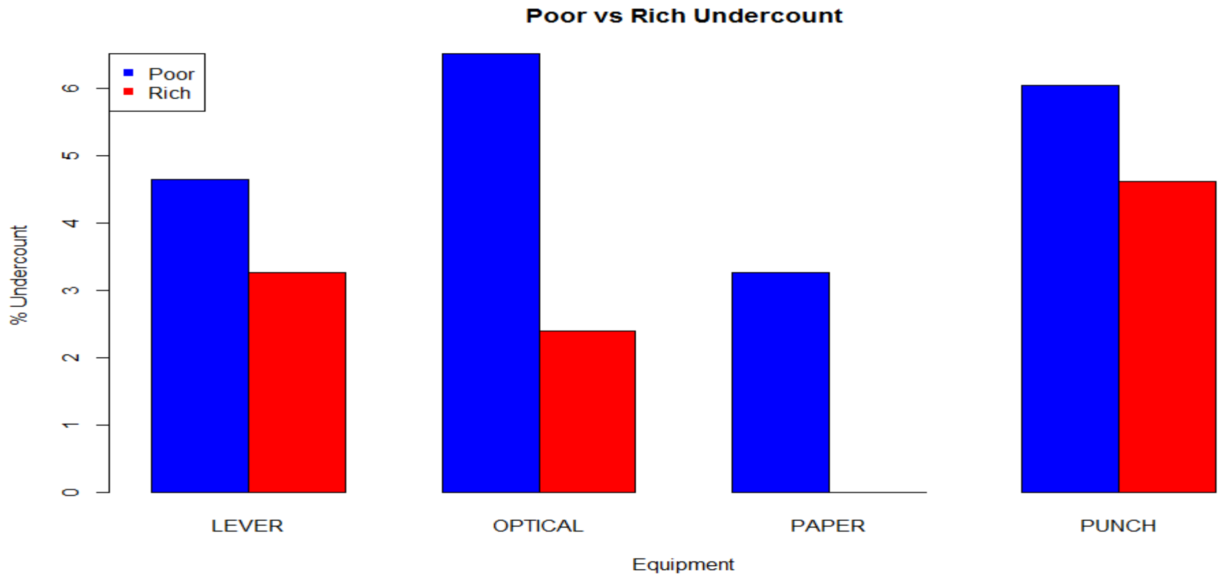
```
Georgiarich<-georgiaData[georgiaData$poor==0,]

rich=aggregate(cbind(ballots,votes)~equip,data=Georgiarich,sum)
rich$undercountPercent<-100*(rich$ballots-rich$votes)/(rich$ballots)

rich=rbind(rich,c("PAPER",0,0,0))
rich=rbind(rich[1:2,],rich[4,],rich[3,])

rich

##     equip ballots   votes undercountPercent
## 1   LEVER  208526  201710   3.2686571458715
## 2 OPTICAL 1321694 1290061  2.39336790512781
## 4   PAPER       0       0                 0
## 3   PUNCH  800309  763276  4.62733769081692
```

Observations

- Counties with higher percentage of poor people have higher undercounts irresective of equipment they use. Thus poverty more than the equipment used is a deciding factor.

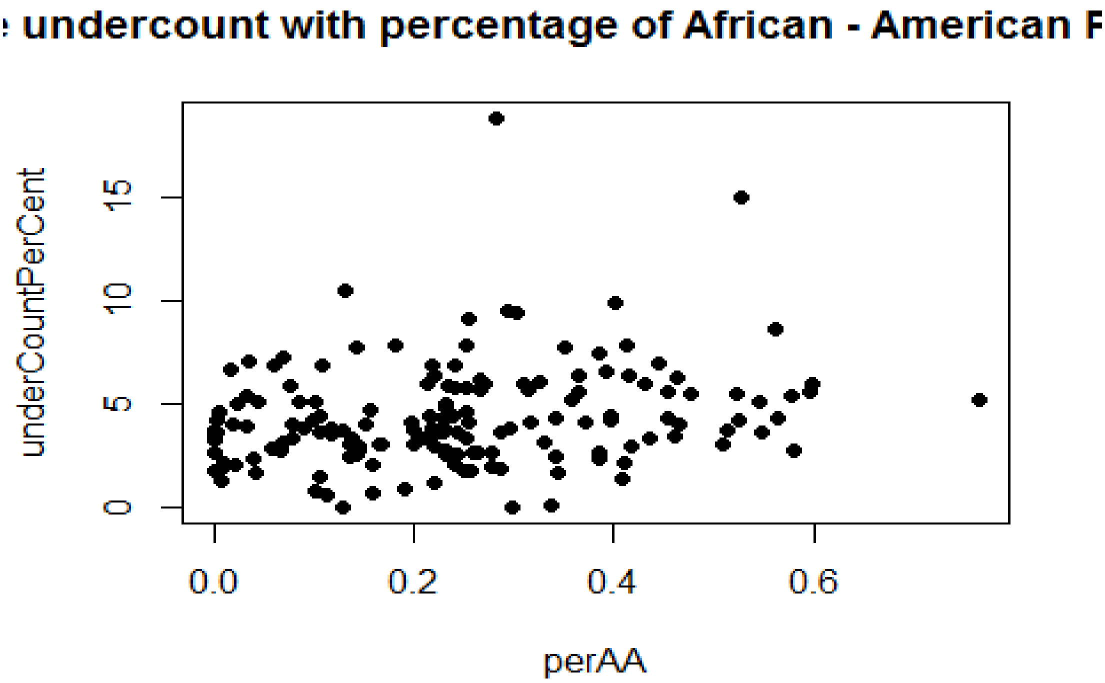
- Optical devices have the highest difference for the richer counties as compared to poor counties. This points to problems in the device.

```
barplot(matrix(c(as.numeric(poor$undercountPercent),as.numeric(rich$underc
ountPercent)),nr=2,byrow = TRUE), beside=T, col=c("blue","red"),names.arg=
poor$equip,xlab="Equipment",ylab="% Undercount",main="Poor vs Rich Underco
unt")

legend("topleft", c("Poor","Rich"), pch=15,
       col=c("blue","red"))
```

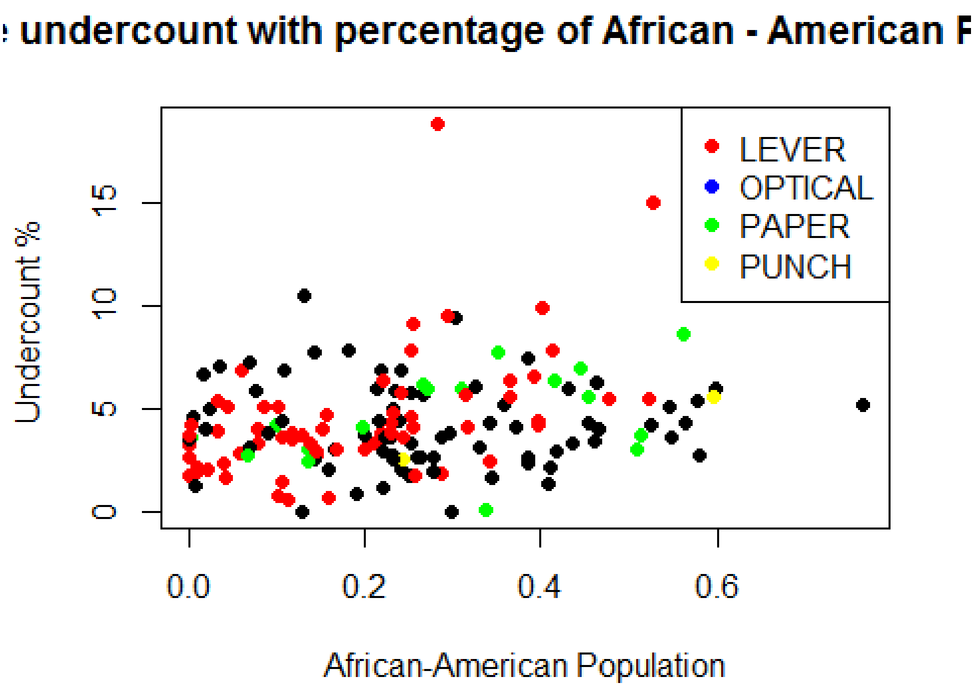


Minority community analysis

```
attach(georgiaData)

## The following object is masked _by_ .GlobalEnv:
##
##     poor

plot(x=perAA,y=underCountPerCent,main="%vote undercount with percentage of
 African - American Population",col="black",pch=19)
```




```
plot(x=perAA,y=underCountPerCent,main="%vote undercount with percentage of
 African - American Population",pch=19,col=c("black","red","yellow","green
")[equip],xlab="African-American Population",ylab="Undercount % ")

legend(x="topright", legend = levels(georgiaData$equip), col=c("red","blue
","green","yellow"), pch=19)
```




```
detach(georgiaData)
```

Conclusion

- Percentage of minorities(African Americans) in a county does not impact % vote undercount in a large way
- Majority of the counties with higher minorities(African American) Population have Lever and Optical equipments for ballots.
- Counties with comparitavely high vote undercount generally used optical or lever based equiments

Bootstrapping

Downloading the data and return over each stock

- Download data for stock price at a daily level using tickers

```
## Loading required package: car
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## 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: lattice
## 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: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
##
## Loading required package: timeDate
## Loading required package: timeSeries
```

- Create a helper function to calculate the return at a daily level

Analyzing the profitability and risk of each exchange traded fund

- Returns of each stock/ticker can be gauged by looking at the distribution of each of their return distribution
- Let us look at return distribution of each ticker and take a call on the risk/return profiles for each

```
# Identity matrix (used for weights) for each iteration

ETF=diag(5)



for (j in 1:5)
{
    n_days=20
    set.seed(15)

    # Now simulate many different possible trading years!
    sim1 = foreach(i=1:500, .combine='rbind') %do% {
        totalwealth = 100000

        #Simulate return of each stock
        weights =ETF[j,]

        holdings = weights * totalwealth
        wealthtracker = rep(0, n_days) # Set up a placeholder to track tot
al wealth

            for(today in 1:n_days)
      {
            return.today = resample(myreturns, 1, orig.ids=FALSE)
            holdings = holdings + holdings*return.today
```

```
            totalwealth = sum(holdings)
            wealthtracker[today] = totalwealth
        }

        wealthtracker
    }

    head(sim1)




    cat(mystocks[j],"\n")

    # Calculate 5% value at risk
    cat("5% : ",quantile(sim1[,n_days], 0.05) - 100000)

    # Mean
    cat("\nMean : ",mean(sim1[,n_days]- 100000))


    # SD
    cat("\nStandard Deviation : ",sd(sim1[,n_days]- 100000))

    # Calculate 5% value at risk
    cat("\n95 percentile : ",quantile(sim1[,n_days], 0.95) - 100000)

    cat("\n\n")
}
```

```
## SPY
## 5% :  -5872.313
## Mean :  1297.183
## Standard Deviation :  4149.845
## 95 percentile :  8656.182
##
## TLT
## 5% :  -6445.759
## Mean :  514.744
```

```
## Standard Deviation :  4453.335
## 95 percentile :  7897.752
##
## LQD
## 5% :  -2250.228
## Mean :  353.2371
## Standard Deviation :  1564.517
## 95 percentile :  2964.769
##
## EEM
## 5% :  -8982.338
## Mean :  297.7715
## Standard Deviation :  5996.186
## 95 percentile :  11285.39
##
## VNQ
## 5% :  -7136.143
## Mean :  1242.889
## Standard Deviation :  4994.409
## 95 percentile :  9753.325
```

- Lower the 5th quantile (left tail of distribution) higher the risk related to the stock/portfolio

Risk Return profiles of each of the stocks

- The risk / return of a stock can be gauged by the median return within a period of time (20 days in this case)

ETFs in order of increasing volatility

- LQD is the safest stock option. It has minimal risk of losses (5 percentile loss of about 2.2k on 100,000$ investment) and has a standard deviation of 1.5k

- SPY is the second most safe option among the five,in a 20 day period on a 100,000$ investment and a loss profile of 5.8$ at the lowest 5% times. It has a standard deviation of 4k

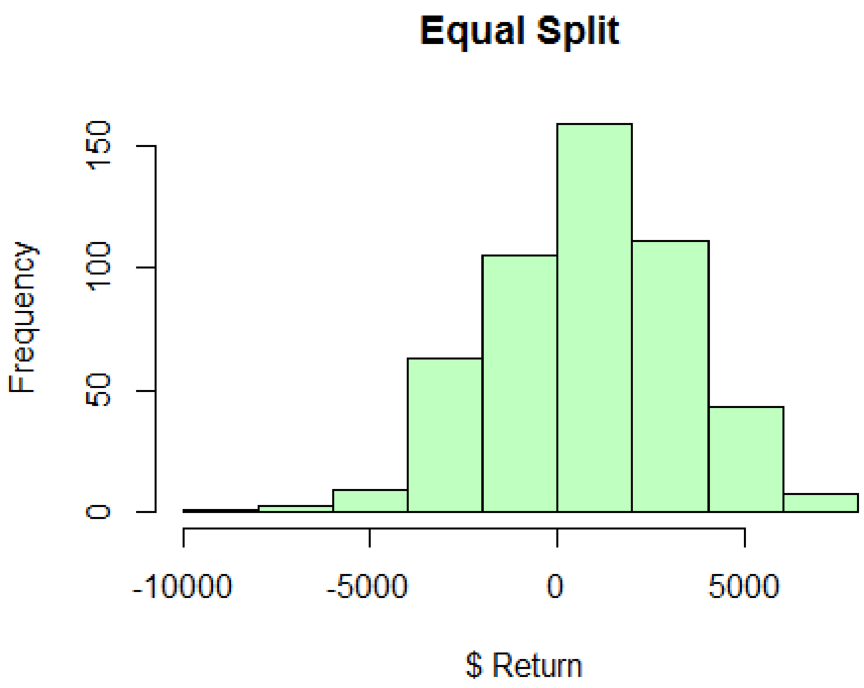
- TLT is the third most safe stock among the five with a mean return of 559$ over a 20day period on investment of 100,000$ .The 5% return is a loss of close to 6.5$ and a standard deviation of 4.5$

- VNQ is the second most volatile stock among the options (5 presented in the portfolio).The 5% return is a loss of close to 7k$ and a standard deviation of 5k$
- EEM is the most volatile stock among the others in the portfolio with a 5% returns greater than 9k$ in losses. The standard deviation of this stock is very varied 6k, thus havig a high standard deviation

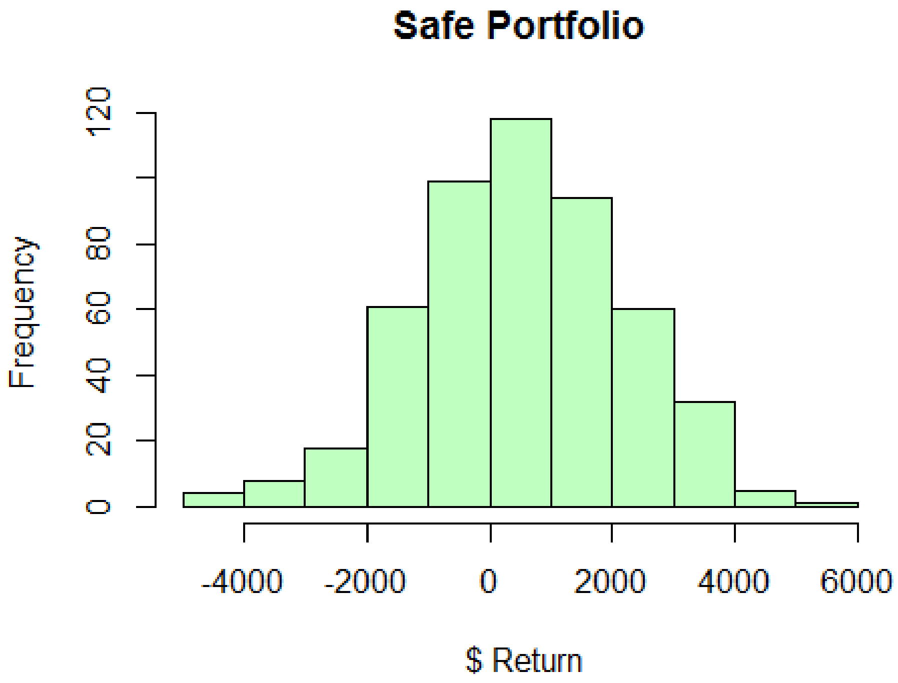
Creating portfolios

```
x=matrix(c(0.2, 0.2, 0.2, 0.2, 0.2,0.2,0.2,0.6,0,0,0,0,0,0.9,0.1),nrow=3,b
yrow = T)

portfolio=c("Equal Split","Safe Portfolio","Aggresive Portfolio")

for (z in 1:3)
{
    n_days=20

  sim1 = foreach(i=1:500, .combine='rbind') %do% {
    totalwealth = 100000
    weights = x[z,]
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days)

        for(today in 1:n_days)
  {
        return.today = resample(myreturns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        wealthtracker[today] = totalwealth

x=matrix(c(0.2, 0.2, 0.2, 0.2, 0.2,0.2,0.2,0.6,0,0,0,0,0,0.9,0.1),nrow=3,b
yrow = T)
  holdings = weights * totalwealth
```

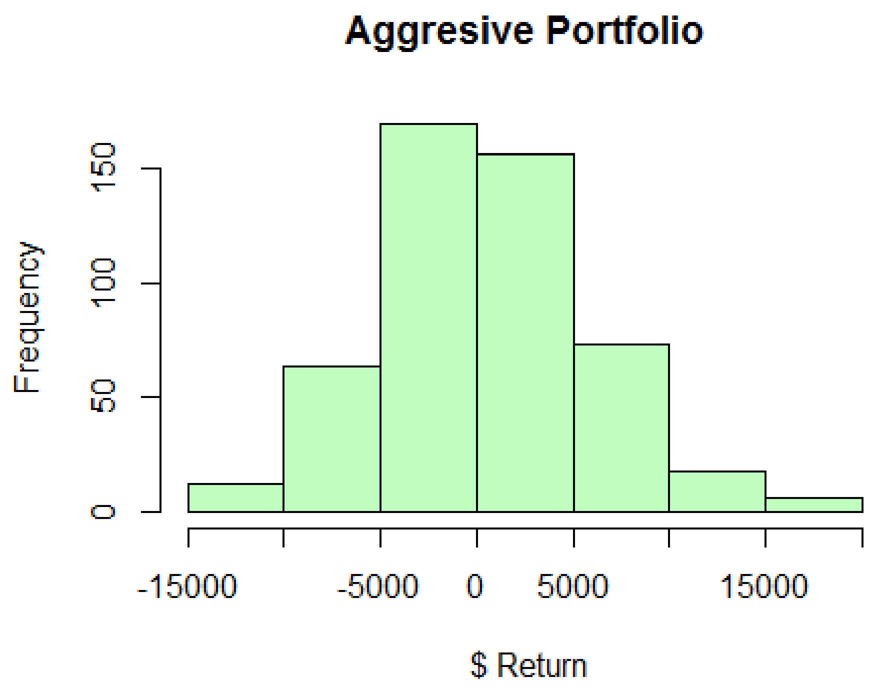
```
    }

    wealthtracker
  }



            # Profit/loss
        hist(sim1[,n_days]- 100000,col=rgb(0,1,0,1/4),main = portfolio[z],
xlab=" $ Return")

        cat(portfolio[z],"\n")



        # Calculate 5% value at risk
        cat("5% : ",quantile(sim1[,n_days], 0.05) - 100000)



        # Mean
        cat("\nMean : ",mean(sim1[,n_days]- 100000))



        # SD
        cat("\nStandard Deviation : ",sd(sim1[,n_days]- 100000))

        # Calculate 5% value at risk
        cat("\n95 percentile : ",quantile(sim1[,n_days], 0.95) - 100000)

        cat("\n\n")
}
```




```
## Equal Split
## 5% :  -3305.527
## Mean :  760.8078
## Standard Deviation :  2518.489
## 95 percentile :  4755.2
```




```
## Safe Portfolio
## 5% :  -2087.095
## Mean :  527.926
## Standard Deviation :  1673.089
## 95 percentile :  3443.76
```




```
## Aggresive Portfolio
## 5% :  -8909.414
## Mean :  430.5341
## Standard Deviation :  5625.627
## 95 percentile :  9913.429
```

Analyzing the three portfolios

Even Split

- For an equal split portfolio, the returns is a combinaton of the risk profiles of all the stocks
- The average return over 20 days on an investment of $100,000 is about $760
- 5% of the times a person holding this portfolio may incur losses of 3.3k

Safe portfolio

- For a safe portfolio, we choose the safest option as the highest amount in terms of investment. LQD (60%) and the other safe (comparitavely safe) stocks 20% in SPY and 20% in TLT)
- It is safe in the sense that there is only 5% chances of losing more than $2087
- Average returns on the investment 527$

Aggressive portfolio

- For an aggressive portfolio, we choose the two most volatile stocks - EEM and VNQ and have a split of 90-10%
- The mean return is about 430$ with 5% of people gaining close to 8909$

Clustering and PCA

Reading the file and removing the columns having variables quality and color of wine as this is an unsupervised problem

```
wine<- read.csv("../data/wine.csv")
Z = wine[,1:11]
```

PCA

Running pca on the data

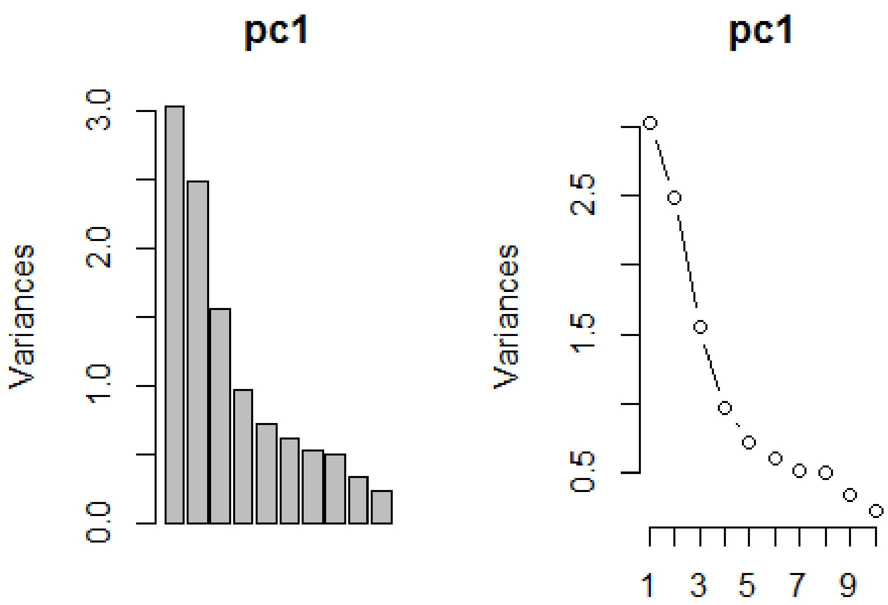
Running the summary of the pca model

```
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
```

Important principal components

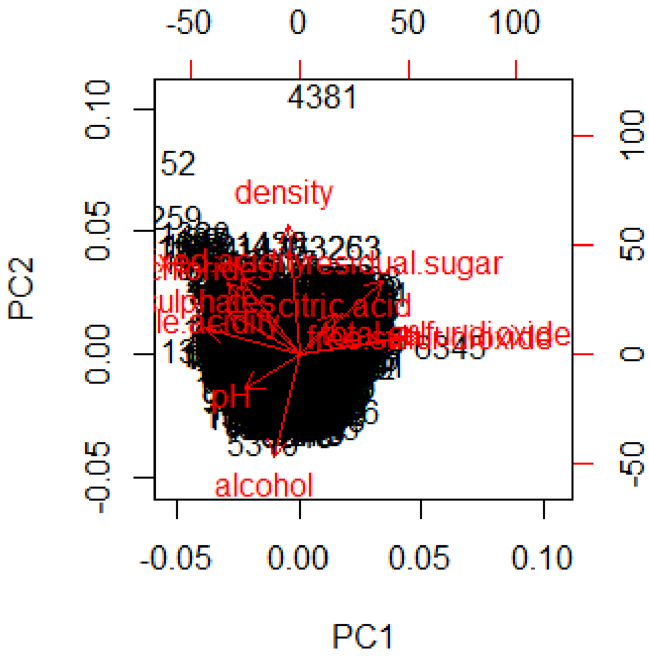
It can be seen that PC 1 through 4 combined account for about 0.75 of the Variance

```
library(RColorBrewer)
library(scales)

##
## Attaching package: 'scales'
##
## The following object is masked from 'package:mosaic':
##
##     rescale

par( mfrow = c( 1,2  ) )
plot(pc1,type="barplot")
plot(pc1,type="line")
```



The plots give a visual representation of the summary, showing the most important component vectors i.e 1,2,3,4

```
biplot(pc1)
```

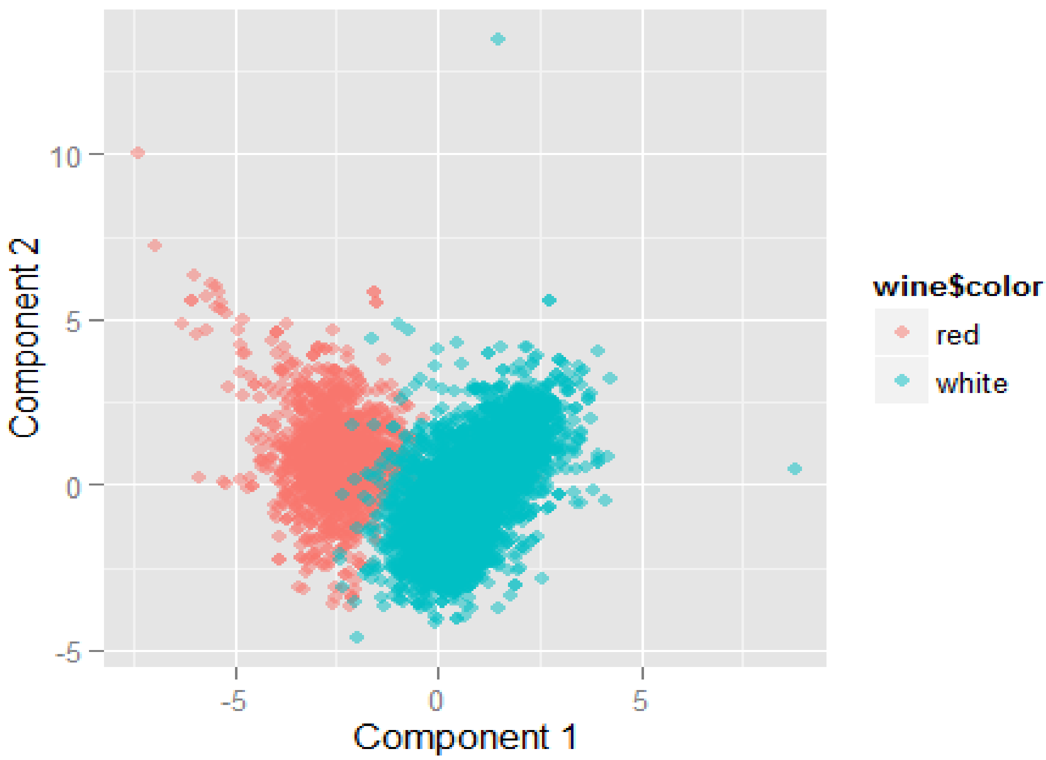


$rotation shows how each principal vector was made with contributions from the original 11 chemical properties of wine

```
##                              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  0.3357018784
```

Red and White wine



Plotting the data on pc1 as x axis and pc2 as y axis it can be seen from the plot here that although both the regions i.e red and white have about the same range on the yaxis, the x variable,pc1 can clearly separate the red and white clusters.

Thus it can be seen that principal component analysis can be used to distinguish red and white wine in the above given case

The contributors of principal component 1 are

```
o1 = order(loadings[,1])
colnames(Z)[head(o1,3)]

## [1] "volatile.acidity" "sulphates"        "chlorides"

colnames(Z)[tail(o1,3)]

## [1] "residual.sugar"       "free.sulfur.dioxide"  "total.sulfur.dioxide
"
```

Investigating PC1

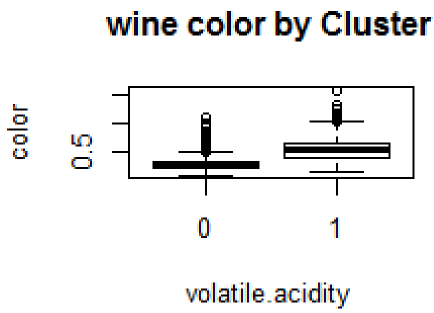
Lets investigate if vectors making PC1 can distinguish red and white wine to any extent.

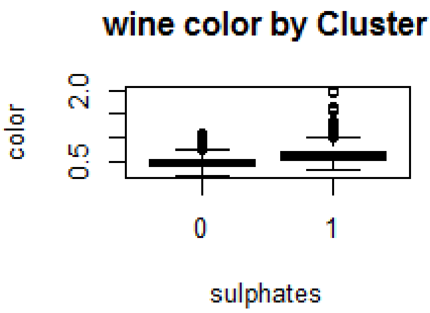
This is an excercise similar to the one done in class with regards to republicans and democrats, the difference being that in that case we had a broad knowledge about the respective idealogies. In the case concerning wines we donot know their characteristics.

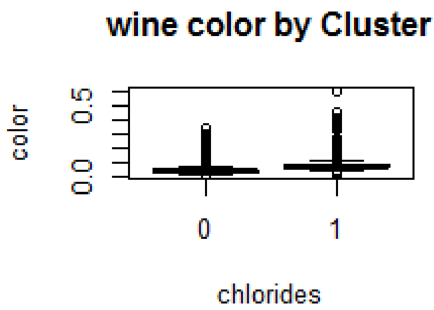
We can do this using boxplots.

Converting the factor column color to a numeric value, 1 for red and 0 for white

Boxplot of volatile.acidity, sulphates and chlorides with color







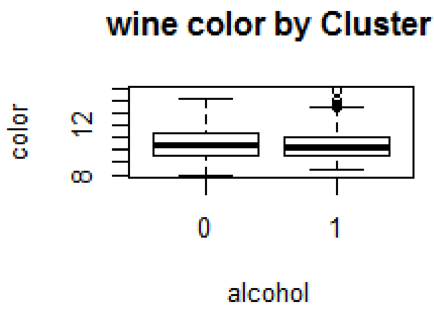
From the above graph we can conclude that the main components making pc1 can actually differentiate between red and white wine.

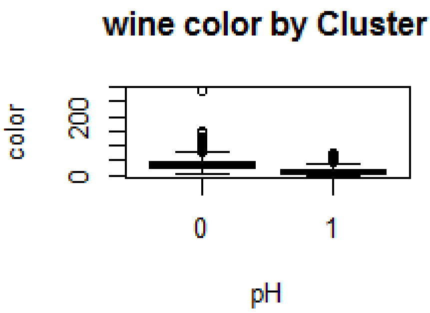
Investigating PC2

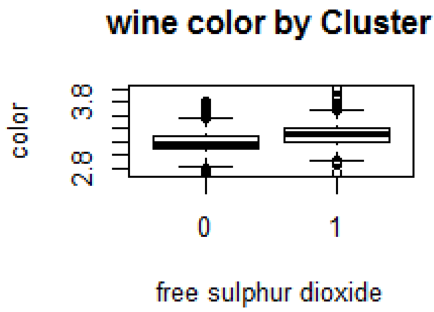
The contributors of principal component 2 are

```
o2 = order(loadings[,2])
colnames(Z)[head(o2,3)]

## [1] "alcohol"             "pH"                    "free.sulfur.dioxide"

colnames(Z)[tail(o2,3)]

## [1] "residual.sugar" "fixed.acidity"  "density"
```

Boxplot of alcohol, pH and free.sulfur.dioxide with color



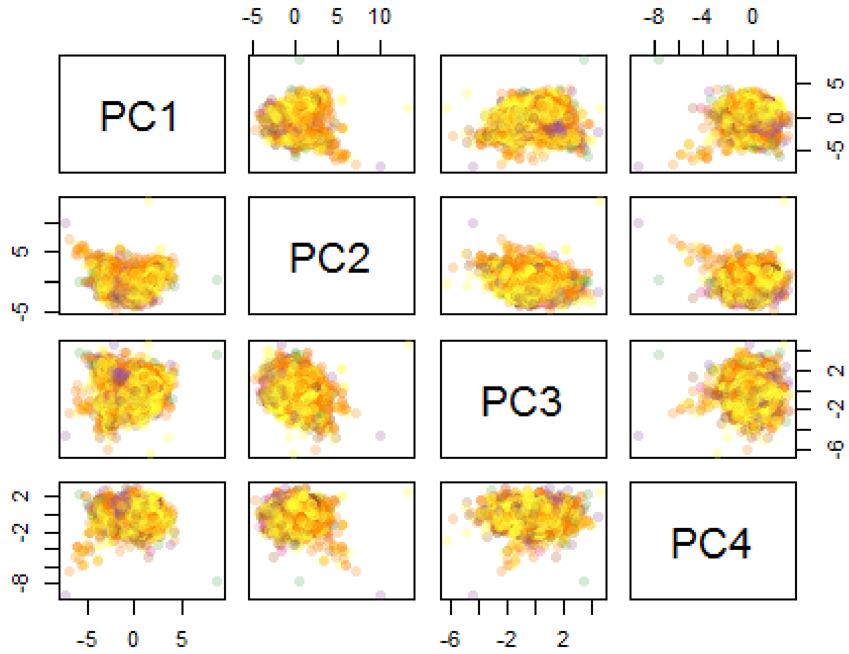




From the above graph we can conclude that the main components making pc2 cannot differentiate between red and white wine.

Verifying if PCA can distinguish quality of the wine

```
comp <- data.frame(pc1$x[,1:4])
palette(alpha(brewer.pal(9,'Set1'), 0.25))
plot(comp, col=wine$quality, pch=16)
```



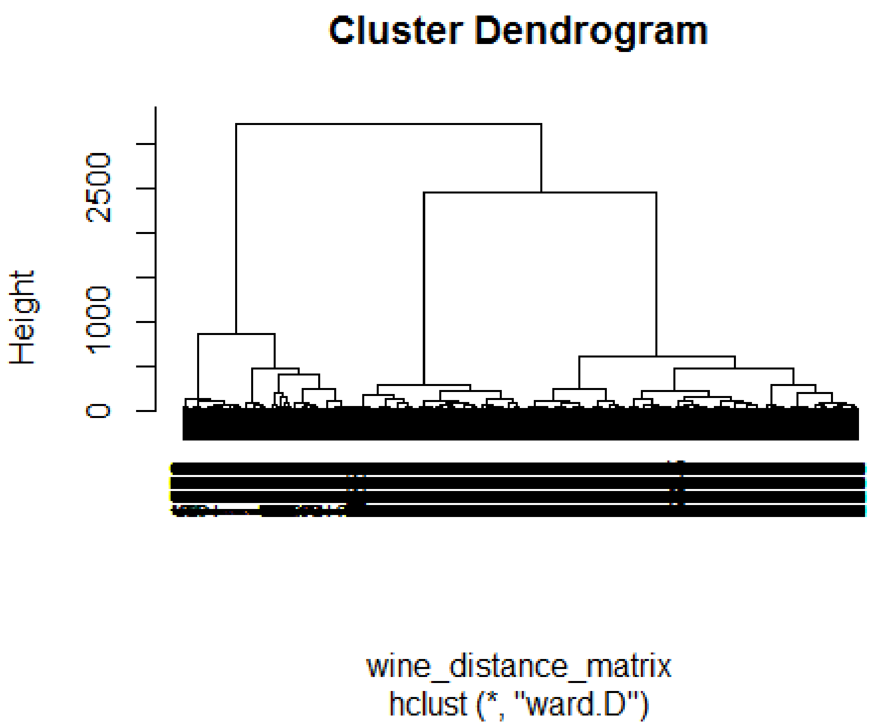
As can be seen here that there can be no clear conclusion about the quality of the wine using PCA, as this plot which is a 2d representation of 3d space has no face in which wines of a particular quality can be distinguished from others

Hierarchical clustering

Scaling and centering the data

Calculating the distance matrix using euclidean method and clustering the distance matrix with ward method.

```
wine_distance_matrix = dist(wine_scaled, method='euclidean')
set.seed(13)
hier_wine = hclust(wine_distance_matrix, method='ward.D')
plot(hier_wine, cex=0.8)
```



Using k=4 we select 4 clusters on the basis of the above plotted dendrogram

```
cluster1 = cutree(hier_wine, k=4)
summary(factor(cluster1))

##    1    2    3    4
##  580 1106 1598 3213
```

The summary function gives us the number of objects in each cluster

We can identify the number of red or white wines in each cluster using the table function

Red and White clusters

Cluster 1

```
table(wine[which(cluster1 == 1),13])

##
##   red white
##   572     8
```

The above cluster is predominantly Red.

Cluster 2

```
table(wine[which(cluster1 == 2),13])

##
##   red white
##   982   124
```

The above cluster is predominantly Red.

Cluster 3

```
table(wine[which(cluster1 == 3),13])

##
##   red white
##     8  1590
```

The above cluster is predominantly White.

Cluster 4

```
table(wine[which(cluster1 == 4),13])

##
##   red white
##    37  3176
```

The above cluster is predominantly White.

```
table(wine$quality)

##
##     3    4    5    6    7    8    9
##    30  216 2138 2836 1079  193    5
```

The table above provides a summary of the number of winess of each quality. It is seen that most data points lie in the values 5 to 7

Verifying if clustering can distinguish quality of the wine

Trying to verify the components of each cluster for quality

Cluster 1

```
table(wine[which(cluster1 == 1),12])
```

```
##
##   3   4   5   6   7   8
##   6  37 264 235  35   3
```

Cluster 2

```
table(wine[which(cluster1 == 2),12])
```

```
##
##   3   4   5   6   7   8
##   6  30 458 438 158  16
```

Cluster 3

```
table(wine[which(cluster1 == 3),12])
```

```
##
##   3   4   5   6   7   8   9
##   3  29 709 712 123  21   1
```

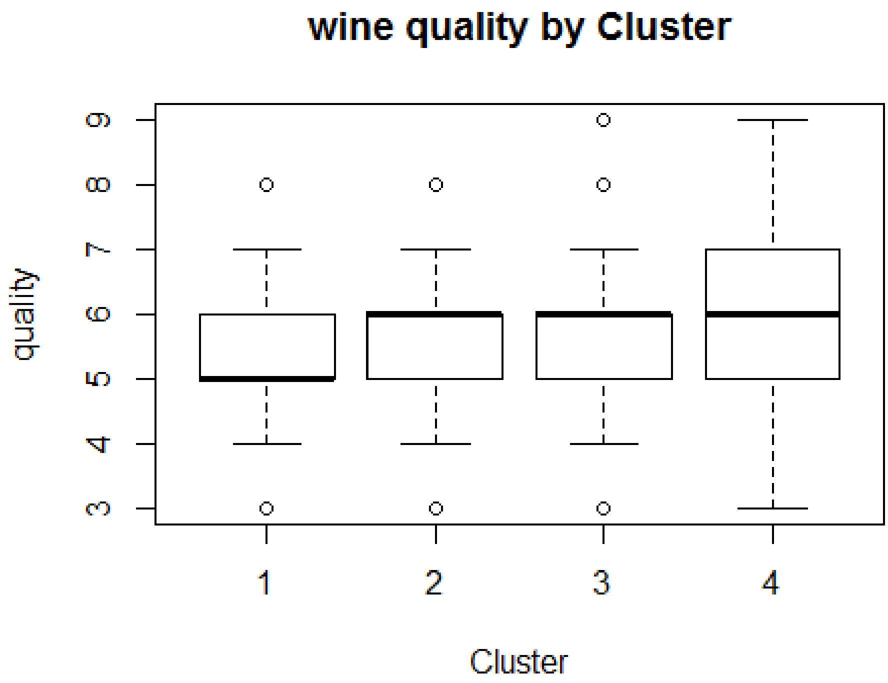
Cluster 4

```
table(wine[which(cluster1 == 4),12])
```

```
##
##    3    4    5    6    7    8    9
##   15  120  707 1451  763  153    4
```

There is not much decipherable difference in the clusters with respect to quality, a boxplot may be able to give us better clarity in this case

```
boxplot(wine$quality ~ cluster1,
        xlab='Cluster', ylab='quality',
        main='wine quality by Cluster')
```



It is seen that the quality cannot be accurately inferred from the clustering method in use, although we see differences in median qualities of wine in all the clusters.

Conclusion

Thus it can be concluded that although both PCA and clustering can differentiate red wine from white wine, clustering seems to be little bit more informative about the quality of wine. None of the methods though gave any answer in regards to the quality of wine with a degree of certainness.

Social Marketing

```
social <- read.csv("../data/social_marketing.csv", row.names=1)
```

We read in the social_marketing csv file

Chatter and Uncategorized

As has been stated in the question, chatter(column 1) and uncategorized(column 5) are the tags of tweets which could not be classified. Using these in our clustering analysis will provide no additional information.

Detecting bots

The few bots that might have slipped into the dataset would have values for adult and spam, using subset function we can try to separate the.

```
bots=subset(social[-c(1,5)], adult>=1 & spam>=1)
sort(sapply(bots,mean))

##         business          beauty  sports_playing  small_business
##        0.1956522       0.5217391       0.5434783       0.5652174
##            music          crafts          dating home_and_garden
##        0.5869565       0.6956522       0.7173913       0.7391304
##           family         tv_film             eco          school
##        0.8043478       0.8478261       0.8478261       0.8913043
##          fashion        shopping      automotive            spam
##        0.9565217       1.0217391       1.0217391       1.0434783
##        computers        outdoors            news       parenting
##        1.0652174       1.1086957       1.2173913       1.2608696
##              art        religion            food   online_gaming
##        1.3043478       1.3478261       1.4565217       1.5217391
## personal_fitness  current_events         cooking     college_uni
##        1.5869565       1.8478261       1.8695652       1.9130435
##    sports_fandom        politics          travel   photo_sharing
##        1.9347826       2.2391304       2.3260870       2.4782609
## health_nutrition           adult
##        2.5652174       7.6739130
```

As can be seen that members of this subset have a mean value for "adult", much greater than others, so these members qualify for classification as bots

Clustering

Why?

Clustering is used as a technique in this case as this is problem of classifying a dataset(market segments) and not a problem of dimension reduction where PCA may turn out to be an apt choice.

We scale the data to apply clustering algorithms

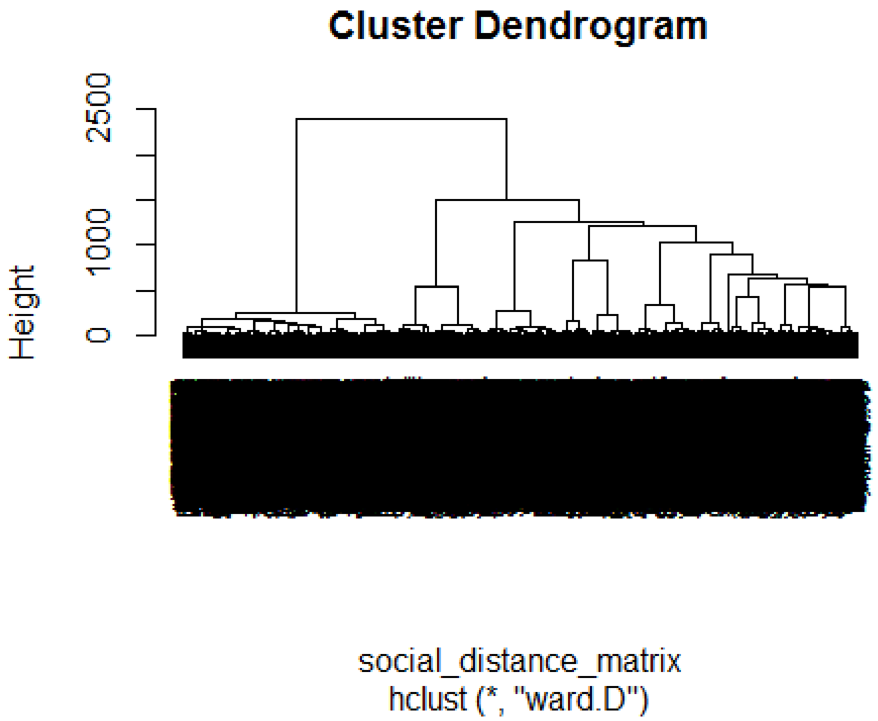
```
social_scaled <- scale(social[-c(1,5)], center=TRUE, scale=TRUE)
```

Calculating the euclidean distance and applying hierarchical clustering

```
social_distance_matrix = dist(social_scaled, method='euclidean')
set.seed(13)
hier_social = hclust(social_distance_matrix, method='ward.D')
```

Plotting the dendogram to understand the possible clusters

```
par( mfrow = c( 1,1  ) )

plot(hier_social, cex=0.8)
```



Choosing the number of clusters as 6 by choosing a line that cuts the dendogram into substantial branches

```
cluster1 = cutree(hier_social, k=6)
```

The number of members in the clusters

```
summary(factor(cluster1))

##    1    2    3    4    5    6 
##  849 1031 1887 2513  886  716
```

The cluster with the largest members can be seen as the leftmost branch of the dendogram

Adding the vector with the cluster numbering into the original data frame

```
social$clust=cluster1
```

Making subsets of the original dataset using the cluster vector which was provided by hierarchical clustering

```
clust1=subset(social[-c(1,5)], clust==1)


clust2=subset(social[-c(1,5)], clust==2)
```

```r
clust3=subset(social[-c(1,5)], clust==3)

clust4=subset(social[-c(1,5)], clust==4)

clust5=subset(social[-c(1,5)], clust==5)

clust6=subset(social[-c(1,5)], clust==6)
```

Analyzing members of each cluster and their properties. It is noted that since **photo_sharing** is a very generic activity it will occur in all clusters.

```r
tail(sort(sapply(clust1[-35],mean)),7)
```

```
##   current_events             food    photo_sharing         outdoors
##         1.442874         2.070671         2.518257         2.653710
##          cooking personal_fitness health_nutrition
##         3.095406         6.040047        11.565371
```

Fitness Enthusiast

The above table provides the average value per variable in the cluster. The highest variables being

- health_nutrition
- personal_fitness
- cooking
- outdoors

The above features describes people who can be classified as **fitness_enthusiasts**.

```r
tail(sort(sapply(clust2[-35],mean)),7)
```

```
##        family        school photo_sharing     parenting          food
##      2.104753      2.141610      2.191077      3.217265      3.755577
##      religion sports_fandom
##      4.238603      4.815713
```

Parents

The above table provides the average value per variable in the cluster. The highest variables being

- sports_fandom

- religion
- food
- parenting
- school

The above features describe people who can be classified as **Parents**.

```
tail(sort(sapply(clust3[-35],mean)),7)

##   current_events health_nutrition          tv_film         shopping
##         1.635400         1.756227         2.023847         2.054054
##    online_gaming      college_uni    photo_sharing
##         2.550609         3.409645         3.560678
```

College Students

The above table provides the average value per variable in the cluster. The highest variables being

- college_uni
- online_gaming
- shopping
- tv_film
- health_nutrition

The above features describe people who can be classified as **College_Students**.

```
tail(sort(sapply(clust4[-35],mean)),7)

##      college_uni         politics         shopping           travel
##        0.7584560        0.8786311        1.0159172        1.0429765
## health_nutrition   current_events    photo_sharing
##        1.1703144        1.4468762        1.8730601
```

Generic

The above table provides the average value per variable in the cluster.The highest variable being

- current_events
- health_nutrition
- travel
- shopping

The above features are very diverse and the mean values for each are very low, this could indicate that this group of users have a very limited activity on Twitter. The group can be described as **Generic**.

```
tail(sort(sapply(clust5[-35],mean)),7)
```

```
##      computers sports_fandom photo_sharing    automotive        travel
##       1.883747      2.153499      2.246050      2.343115      4.425508
##           news      politics
##       4.879233      7.420993
```

Working male

The above table provides the average value per variable in the cluster.The highest variable being

- politics
- news
- travel
- automotive

This group can be classified as **Working male professional**.

```
tail(sort(sapply(clust6[-35],mean)),7)
```

```
##         shopping   current_events health_nutrition           beauty
##         1.371508         1.562849         1.973464         3.311453
##          fashion    photo_sharing          cooking
##         4.750000         4.808659         9.222067
```

Women

The above table provides the average value per variable in the cluster.The highest variable being

- cooking
- fashion
- beauty
- health_nutrition

This group can be classified as **Women**.

Conclusion

Clustering helps us identify distinct clusters except the generic cluster which was hard to classify.