Assignment1

Sadavath Sharma

August 7, 2015

```
library(ggplot2)
library(foreach)
library(fImport)

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

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: lattice
## 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
```

Answer1:

Read the data file

```
georgia =
read.csv("https://raw.githubusercontent.com/jgscott/STA380/master/data/georgi
a2000.csv", header=TRUE)
head(georgia)
```

```
##     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
```

```
summary(georgia)
```

```
##       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
##  Min.   :   249   Min.   :   271
```

```
##  1st Qu.:  1386   1st Qu.:  1804  
##  Median :  2326   Median :  3597  
##  Mean   :  7020   Mean   :  8929  
##  3rd Qu.:  4430   3rd Qu.:  7468  
##  Max.   :154509   Max.   :140494  
##

attach(georgia)
```

Create indicators for countries having voting undercount

```
georgia$undercount=ifelse(georgia$ballots>georgia$votes,1,0)
```

Find out the undercount counties on the basis of different equipment

```
xtabs(~equip+undercount,data=georgia)

##          undercount
## equip      0  1
##   LEVER    2 72
##   OPTICAL  0 66
##   PAPER    0  2
##   PUNCH    0 17
```

Lever has least reported instances of undercount

All other equipments have 100% undercounts

Use %age of undercount votes as the parameter in order to find out the efficiency of the equipment

Aggregate the counts of ballots on the basis of equipment and merging them

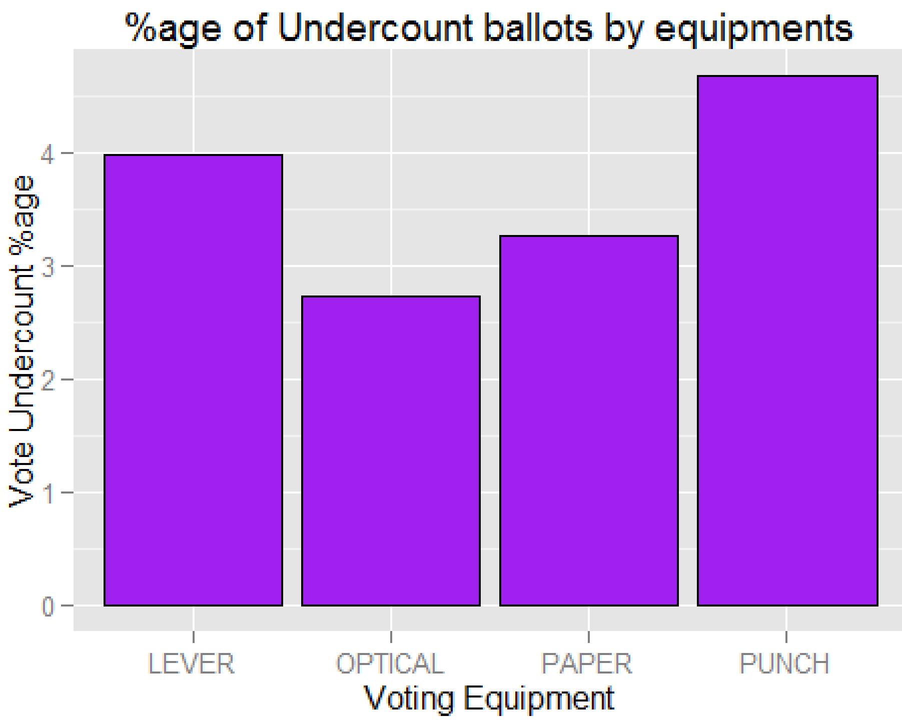
```
votes <-aggregate(votes ~ equip,data=georgia,FUN=sum, na.rm=TRUE)
ballots=aggregate(ballots~equip,data=georgia,FUN=sum,na.rm=TRUE)

ballot_undercount=merge(votes,ballots,by.x="equip",by.y="equip")
```

Find the undercount for each equipment type

```
ballot_undercount$percent_ballot_diff= ((ballot_undercount$ballots -
ballot_undercount$votes)/ballot_undercount$ballots)*100
```

Plot the undercount %age for each equipment type

```
ggplot(ballot_undercount, aes(x=ballot_undercount$equip,
y=ballot_undercount$percent_ballot_diff)) +
geom_bar(stat="identity",fill="purple", colour="black")+
  labs(x="Voting Equipment",y="Vote Undercount %age",title="%age of
Undercount ballots by equipments")
```

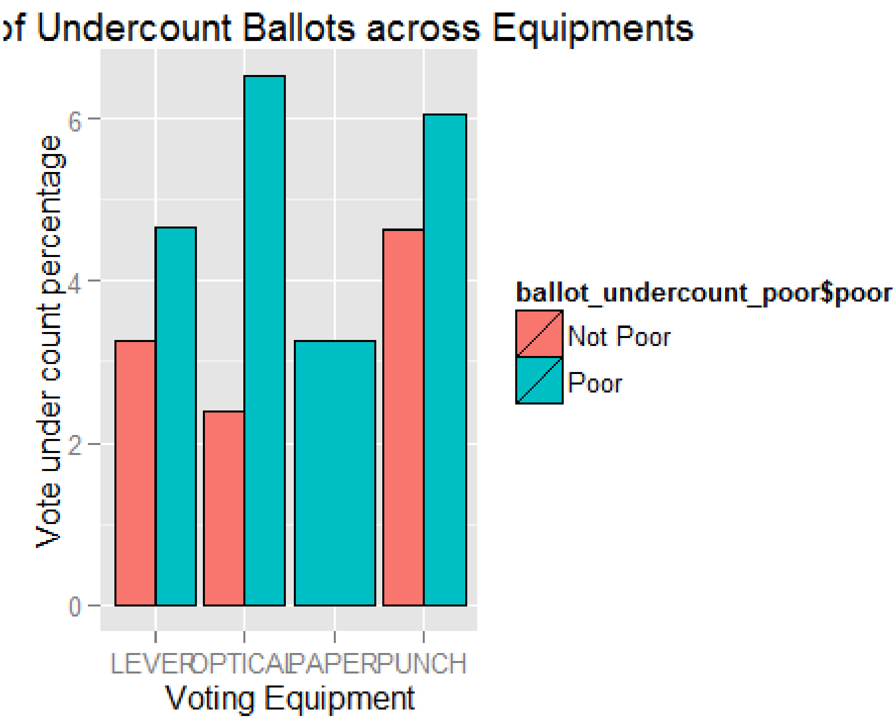


####Aggregate %age of undercount to find out the effect of ballot undercount on poor segments and minorities ####New data frame of counted votes, ballots and %age on the basis of poor and non-poor

```
votes_poor <-aggregate(votes ~ equip+poor,data=georgia,FUN=sum, na.rm=TRUE)
ballots_poor=aggregate(ballots~equip+poor,data=georgia,FUN=sum,na.rm=TRUE)
ballot_undercount_poor=merge(votes_poor,ballots_poor,by=c("equip","poor"))
ballot_undercount_poor$poor=ifelse(ballot_undercount_poor$poor==1,"Poor","Not
Poor")
ballot_undercount_poor$poor=factor(ballot_undercount_poor$poor)
ballot_undercount_poor$percent_ballot_diff= ((ballot_undercount_poor$ballots
- ballot_undercount_poor$votes)/ballot_undercount_poor$ballots)*100
```

From the plot below, we notice that Voting Undercount is higher for poorer areas

```
ggplot(ballot_undercount_poor, aes(x=ballot_undercount_poor$equip,
y=ballot_undercount_poor$percent_ballot_diff))+

geom_bar(stat="identity",aes(fill=ballot_undercount_poor$poor),colour="black"
,position=position_dodge())+
  labs(x="Voting Equipment",y="Vote under count percentage",title="%age of
Undercount Ballots across Equipments")
```



Answer2:

Import the stocks

```
mystocks = c("SPY", "TLT", "LQD","EEM","VNQ")
myprices = yahooSeries(mystocks, from='2011-01-01', to='2015-08-05')
```

A Helper Function for calculating %age returns from a Yahoo Series

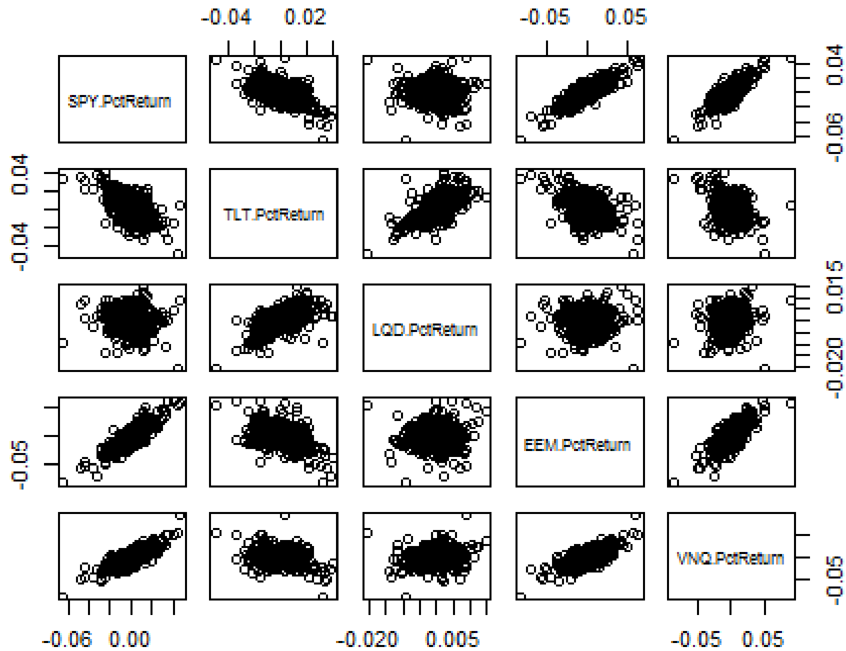
```
YahooPricesToReturns = function(series) {
  mycols = grep('Adj.Close', colnames(series))
  closingprice = series[,mycols]
  N = nrow(closingprice)
  percentreturn = as.data.frame(closingprice[2:N,]) /
as.data.frame(closingprice[1:(N-1),]) - 1
  mynames = strsplit(colnames(percentreturn), '.', fixed=TRUE)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

Compute the returns from the closing prices

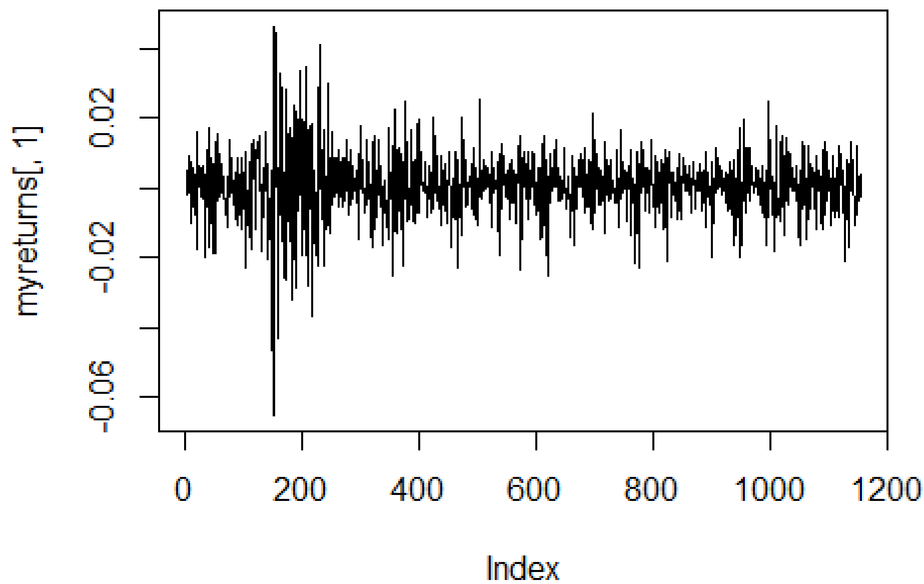
```
myreturns = YahooPricesToReturns(myprices)
```

These returns can be viewed as draws from the joint distribution

```
pairs(myreturns)
```




```
plot(myreturns[,1], type='l')
```



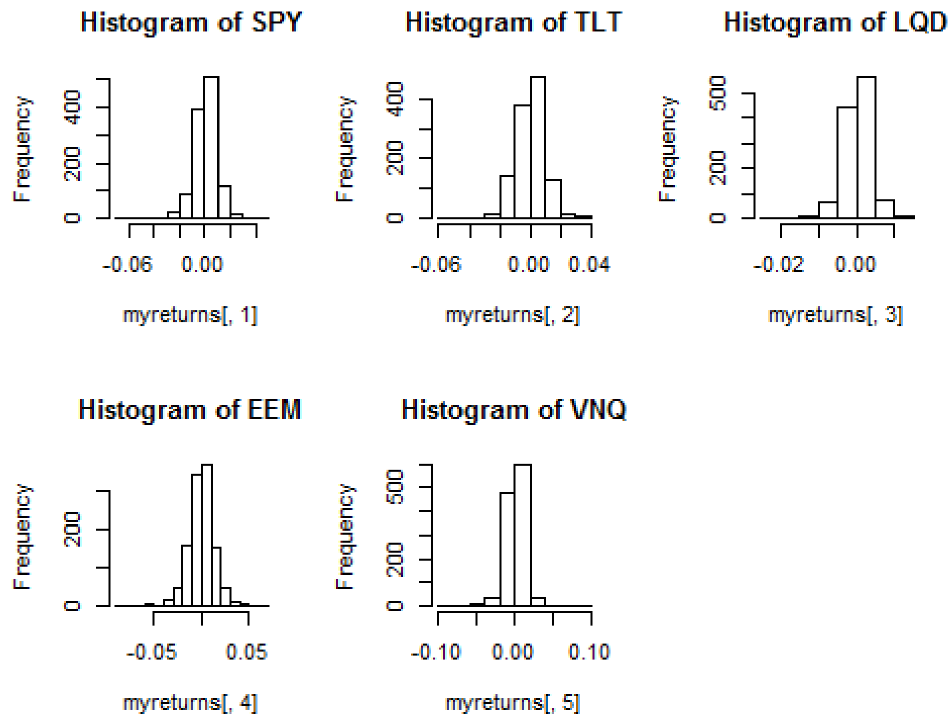

```
mu_SPY = mean(myreturns[,4])
sigma_SPY = sd(myreturns[,4])

mynames = sapply(data.frame(myreturns), function(x) sd(x))
mynames

## SPY.PctReturn TLT.PctReturn LQD.PctReturn EEM.PctReturn VNQ.PctReturn
##   0.009405747   0.009593108   0.003490087   0.013843982   0.011444111
```

Compute the moments of a one-day change in your portfolio

```
totalwealth = 100000
weights = c(0.20,0.20,0.20,0.20,0.20)     # What percentage of your wealth
will you put in each stock?
```

How much money do we have in each stock?

```
holdings = weights * totalwealth
par(mfrow=c(2,3))
hist(myreturns[,1],main = paste("Histogram of SPY" ))
hist(myreturns[,2],main = paste("Histogram of TLT"))
hist(myreturns[,3],main = paste("Histogram of LQD" ))
hist(myreturns[,4],main = paste("Histogram of EEM" ))
hist(myreturns[,5],main = paste("Histogram of VNQ" ))
```



####The standard deviation values helps in characterizing the risk/return properties for these stocks ####LQD and and SPY safe stocks to purchase since they have smaller standard deviations ####EEM and VNQ are riskier stocks to purchase since they have higher standard deviations ####Portfolio with equal split amongst stocks

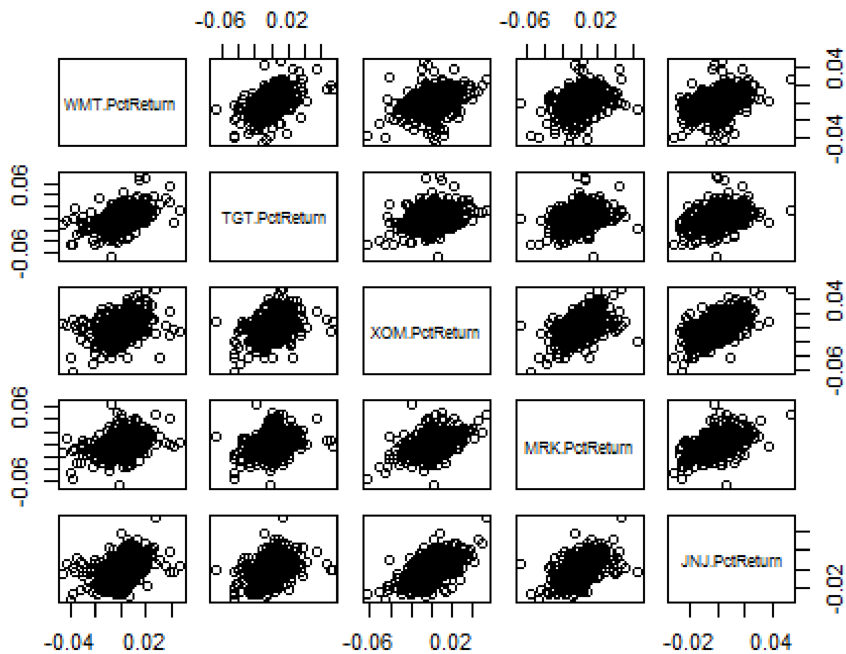
```
totalwealth = 100000
weights = c(0.20,0.20,0.20,0.20,0.20)
holdings = weights * totalwealth
```

Now use a bootstrap approach with more stocks

```
mystocks = c("WMT", "TGT", "XOM", "MRK", "JNJ")
myprices = yahooSeries(mystocks, from='2011-01-01', to='2015-07-30')
```

Compute the returns from the closing prices

```
myreturns = YahooPricesToReturns(myprices)
pairs(myreturns)
```



####Sample a random return day

```
return.today = resample(myreturns, 1, orig.ids=FALSE)
```

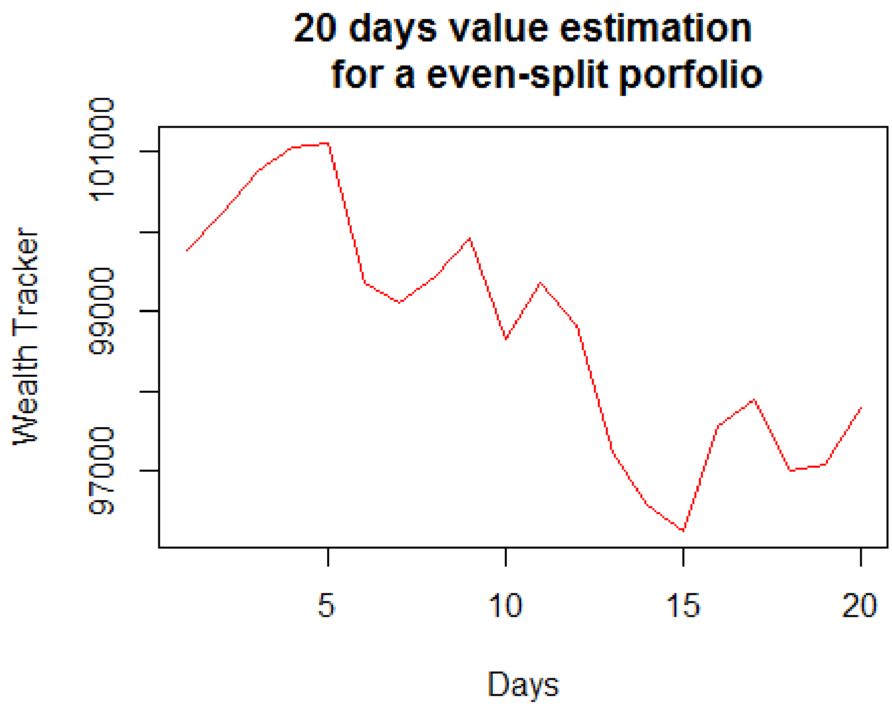
Update the value of the holdings and compute new wealth

```
holdings = holdings + holdings*return.today
totalwealth = sum(holdings)
par(mfrow=c(3,1))
```

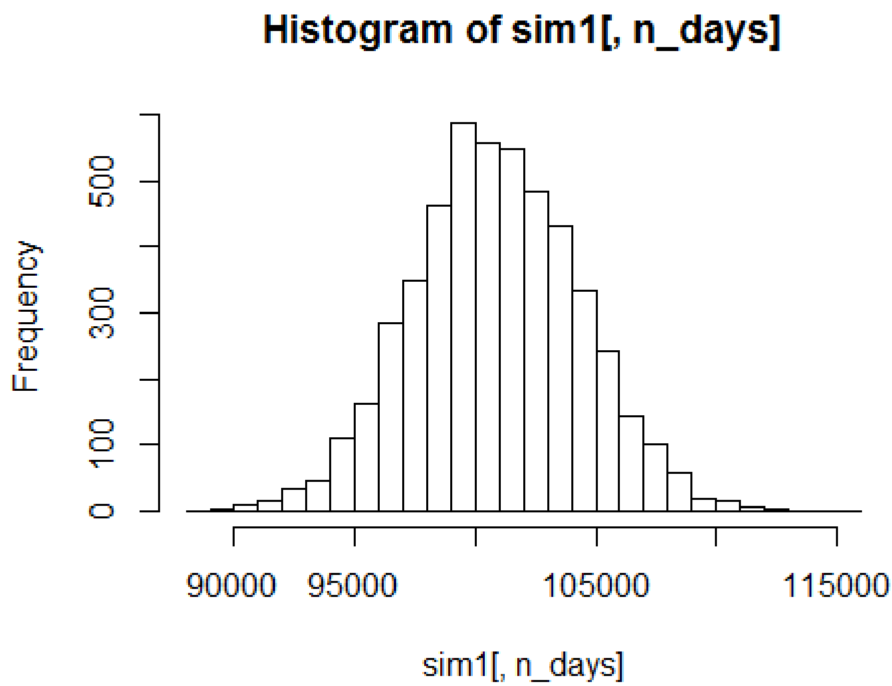
Bootstrapping for even split portfolio for a 20 day trading window

```
n_days=20
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
  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
  }
  wealthtracker
}
plot(wealthtracker, type='l',xlab="Days",ylab="Wealth Tracker",main="20 days
```

```
value estimation
     for a even-split porfolio",col="red")
```

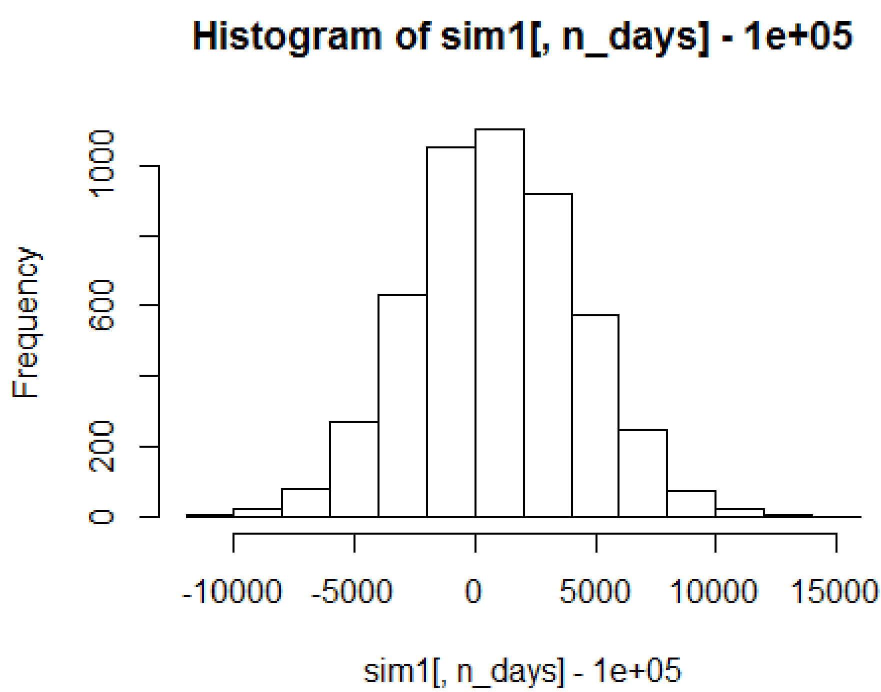



```
hist(sim1[,n_days], 25)
```



####Find profit/loss and Calculate 5% value at risk

```
hist(sim1[,n_days]- 100000)
```



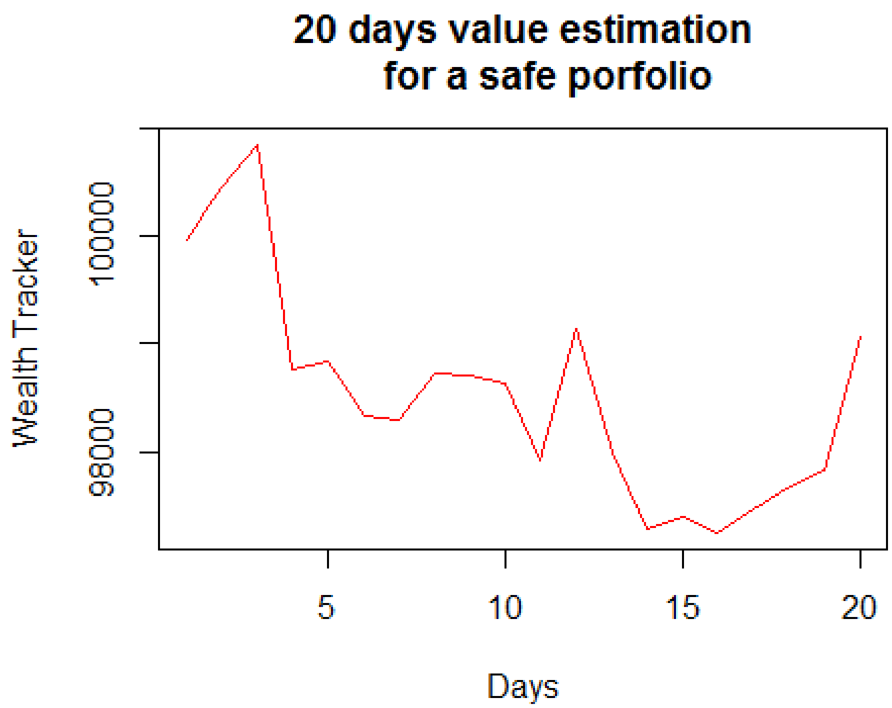
```r
quantile(sim1[,n_days], 0.05) - 100000
```

```
##        5%
## -4731.794
```

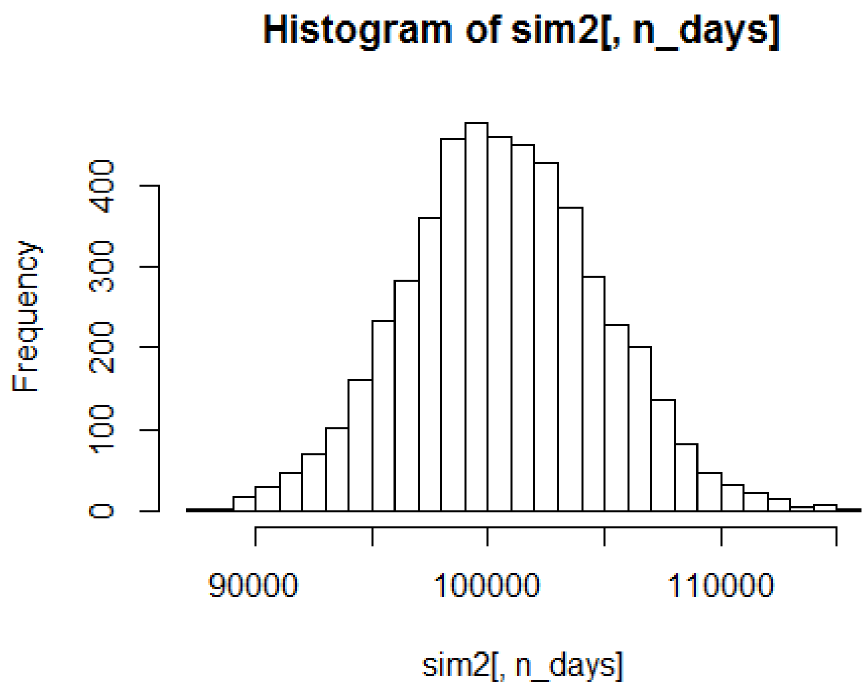
Bootstrapping for safer portfolio over two trading weeks

Considering the portfolio of SPY,TLT and LQD as a safe portfolio

```r
n_days=20
sim2 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.15, 0.15, 0.70, 0, 0)
  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
  }
  wealthtracker
}
plot(wealthtracker, type='l',xlab="Days",ylab="Wealth Tracker",main="20 days value estimation
     for a safe porfolio",col="red")
```

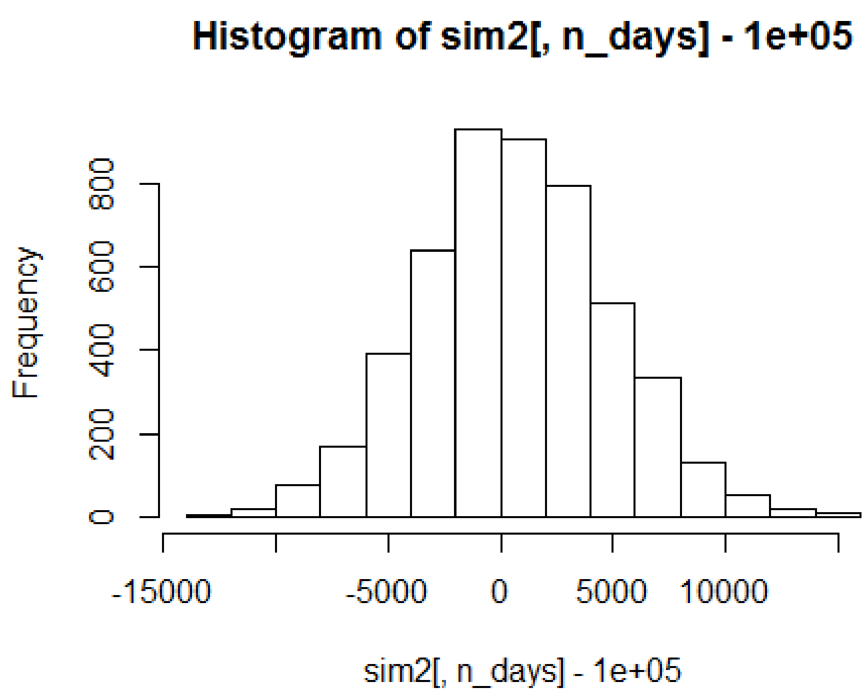



```r
hist(sim2[,n_days], 25)
```



####Find profit/loss and Calculate 5% value at risk

```
hist(sim2[,n_days]- 100000)
```

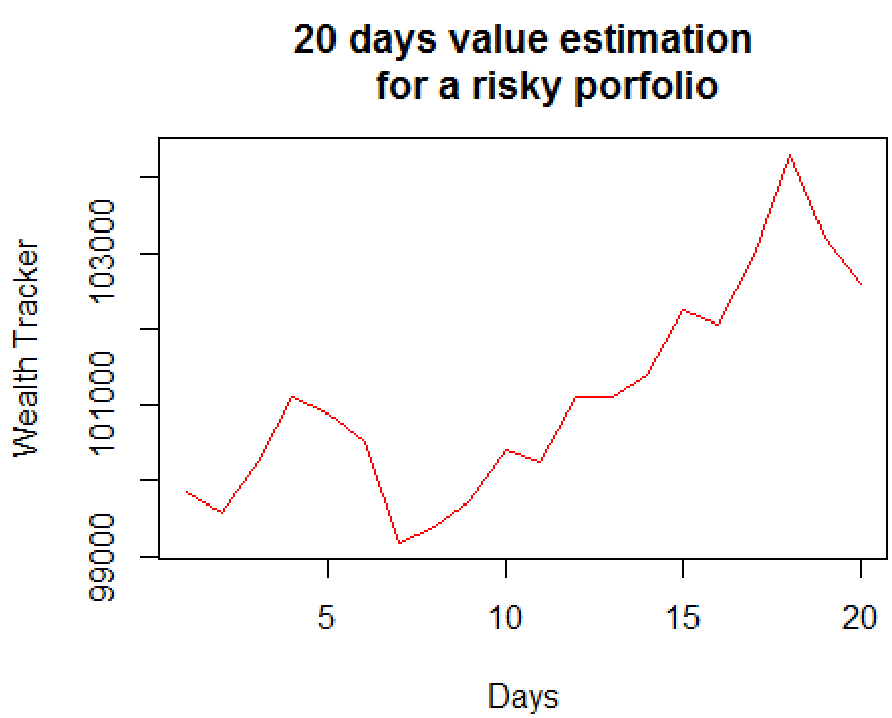


```
quantile(sim2[,n_days], 0.05) - 100000

##        5%
## -6124.526
```

Bootstrapping for riskier portfolio over two trading weeks

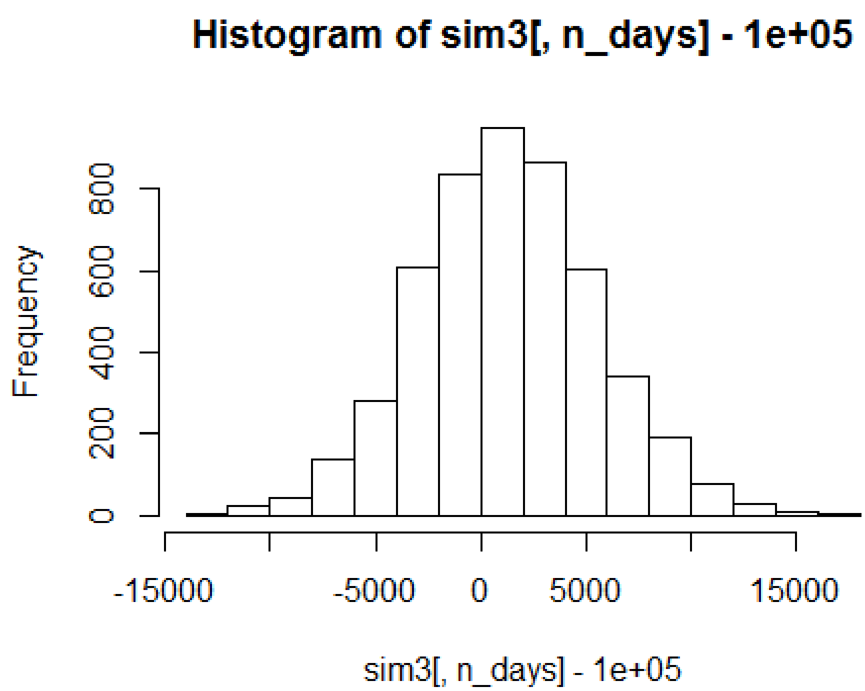
Considering the portfolio of EEM and VNQ as a risky portfolio

```
n_days=20
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0,0,0,0.55, 0.45)
  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
  }
  wealthtracker
}
plot(wealthtracker, type='l',xlab="Days",ylab="Wealth Tracker",main="20 days value estimation
     for a risky porfolio",col="red")
```



####Find profit/loss and Calculate 5% value at risk

```r
hist(sim3[,n_days]- 100000)
```




```r
quantile(sim3[,n_days], 0.05) - 100000

##        5%
## -5668.889
```

Answer3:

```r
winedata = 
read.csv("https://raw.githubusercontent.com/jgscott/STA380/master/data/wine.c
sv", header=TRUE)
head(winedata)

##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
```

```
## 6                   13                    40  0.9978 3.51      0.56     9.4
##   quality color
## 1       5   red
## 2       5   red
## 3       5   red
## 4       6   red
## 5       5   red
## 6       5   red
```

```
names(winedata)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
## [13] "color"
```

Removing Quality and Color variables from the original dataset

```
winedata_num = winedata[,1:11]
head(winedata_num)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
```

Scaling the data

```
winedata_scaled <- scale(winedata_num, center=TRUE, scale=TRUE)
head(winedata_scaled)
```

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## [1,]     0.1424623        2.1886645   -2.192664     -0.7447208 0.5699140
## [2,]     0.4510010        3.2819823   -2.192664     -0.5975941 1.1978825
## [3,]     0.4510010        2.5531038   -1.917405     -0.6606484 1.0266184
## [4,]     3.0735801       -0.3624106    1.660957     -0.7447208 0.5413699
## [5,]     0.1424623        2.1886645   -2.192664     -0.7447208 0.5699140
## [6,]     0.1424623        1.9457049   -2.192664     -0.7657389 0.5413699
##      free.sulfur.dioxide total.sulfur.dioxide   density         pH
## [1,]          -1.1000552           -1.4462472 1.0349132  1.8129500
## [2,]          -0.3112961           -0.8624022 0.7014323 -0.1150642
```

```
## [3,]          -0.8746955          -1.0924018 0.7681285  0.2580999
## [4,]          -0.7620156          -0.9862481 1.1016093 -0.3638402
## [5,]          -1.1000552          -1.4462472 1.0349132  1.8129500
## [6,]          -0.9873753          -1.3400936 1.0349132  1.8129500
##      sulphates    alcohol
## [1,] 0.1930819 -0.9153937
## [2,] 0.9995017 -0.5800235
## [3,] 0.7978967 -0.5800235
## [4,] 0.3274852 -0.5800235
## [5,] 0.1930819 -0.9153937
## [6,] 0.1930819 -0.9153937
```

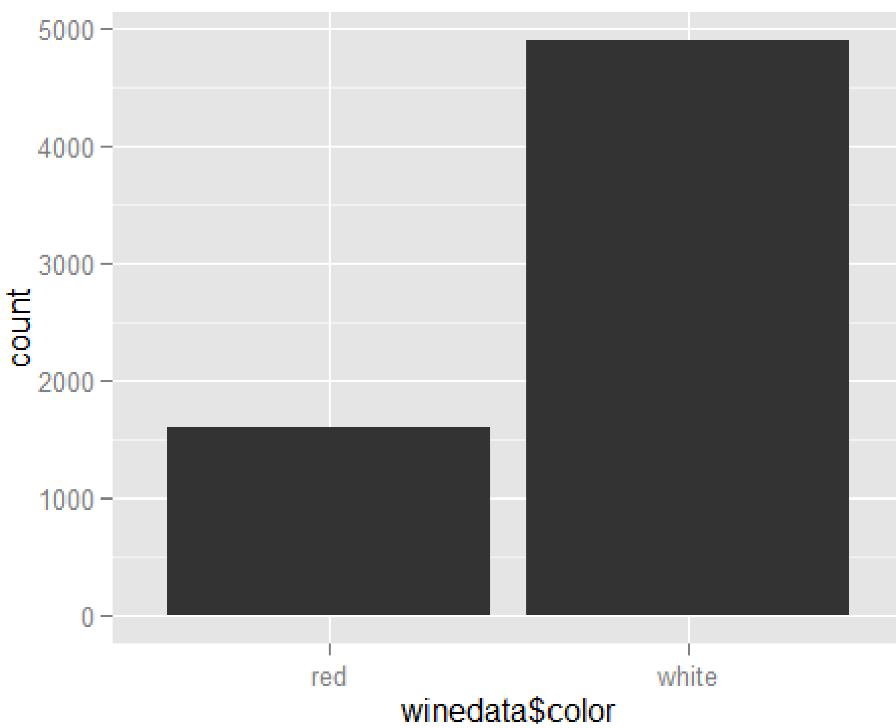
Clustering based on all remaining variables using k-means

```
winedata_clustered <- kmeans(winedata_scaled, centers=2, nstart=50)
```

Checking if k-means can help us distinguish Red from White wine

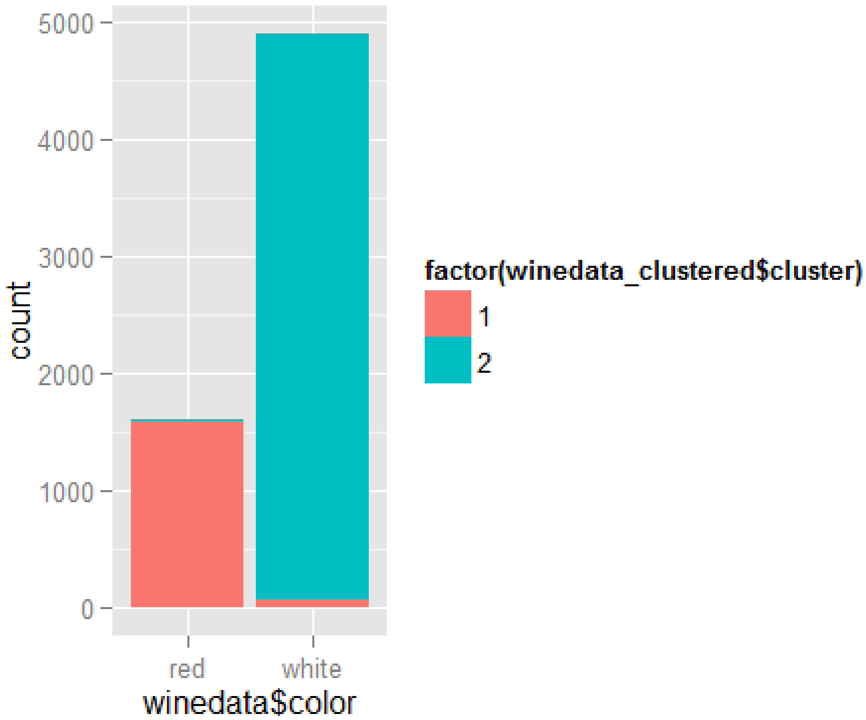
Plotting points in the dataset as Red or White wine and then superimposing predictions from the k-means clustering technique

```
qplot(winedata$color)
```



####After superimposing it can be seen that k-means was able to cluster effectively.

```
qplot(winedata$color, fill = factor(winedata_clustered$cluster) )
```



####Calculating accuracy% of clustering using a contingency table and proportions

```
color_accuracy = table(winedata$color,winedata_clustered$cluster)
color_accuracy2 = prop.table(color_accuracy, margin =1)
head(color_accuracy2*100)
```

```
##
##                 1         2
##   red   98.499062  1.500938
##   white  1.388322 98.611678
```

Conclusion : k-means clustering technique does a very good job at distinguishing red wine from white wine

Checking if k-means can help us distinguish the quality of wine

```
winedata_clustered_qual <- kmeans(winedata_scaled, centers= 7,iter.max= 50,
nstart=50)
```

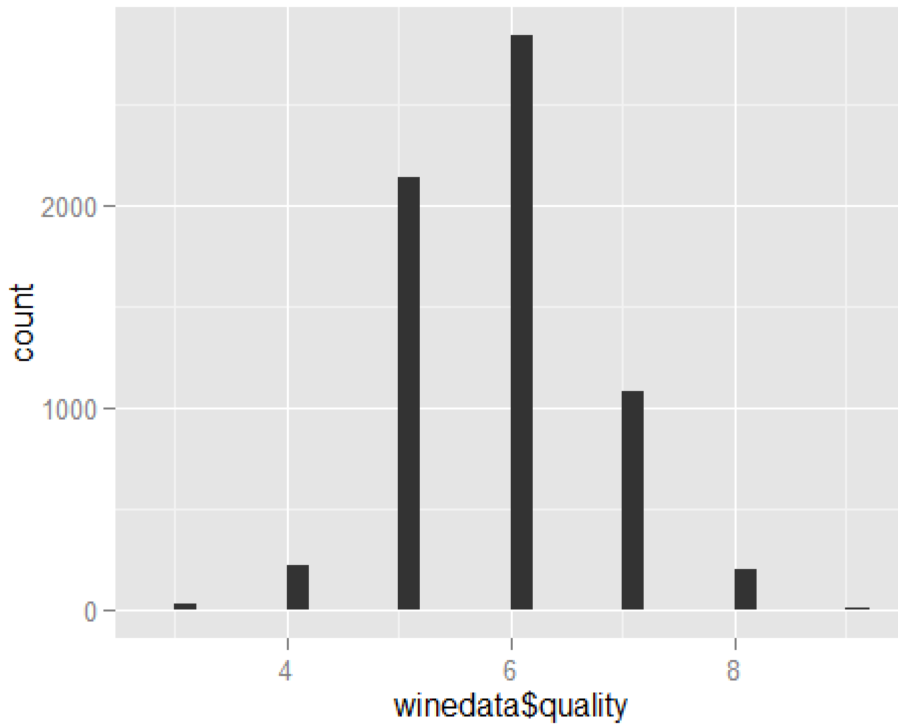
The column below gives the cluster type

```
## [1] 1 1 1 7 1 1
```

Plotting to show distribution of wines by quality and then superimposing predictions from the k-means clustering technique

```
qplot(winedata$quality)
```

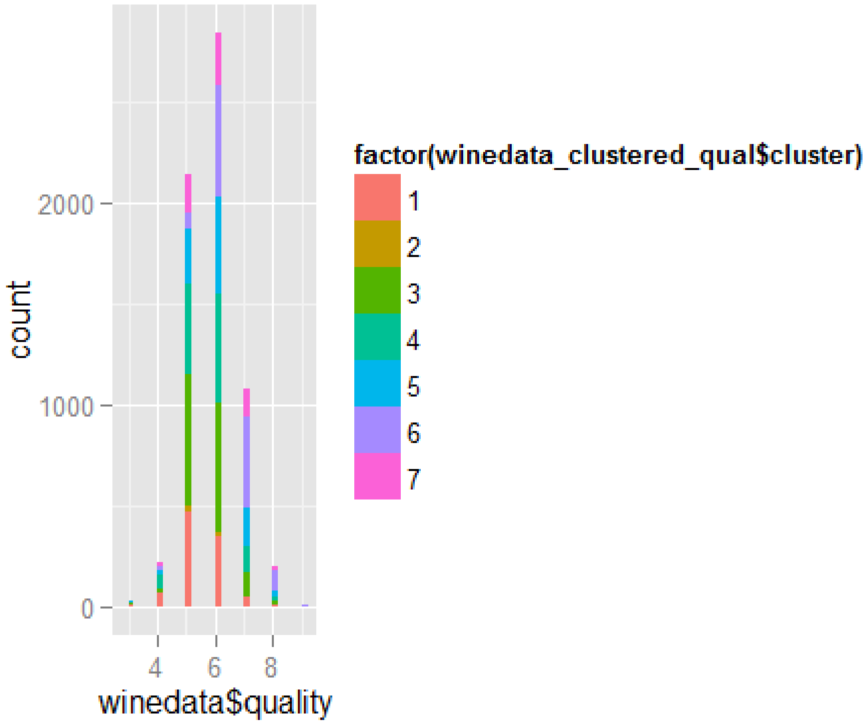
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



####After superimposing it can be seen that k-means was able to cluster effectively.

```
qplot(winedata$quality, fill = factor(winedata_clustered_qual$cluster) )
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



####Calculating accuracy% of clustering using a contingency table and proportions

```
quality_accuracy = table(winedata$quality,winedata_clustered_qual$cluster)
quality_accuracy2 = prop.table(quality_accuracy, margin =1)
head(quality_accuracy2*100)
```

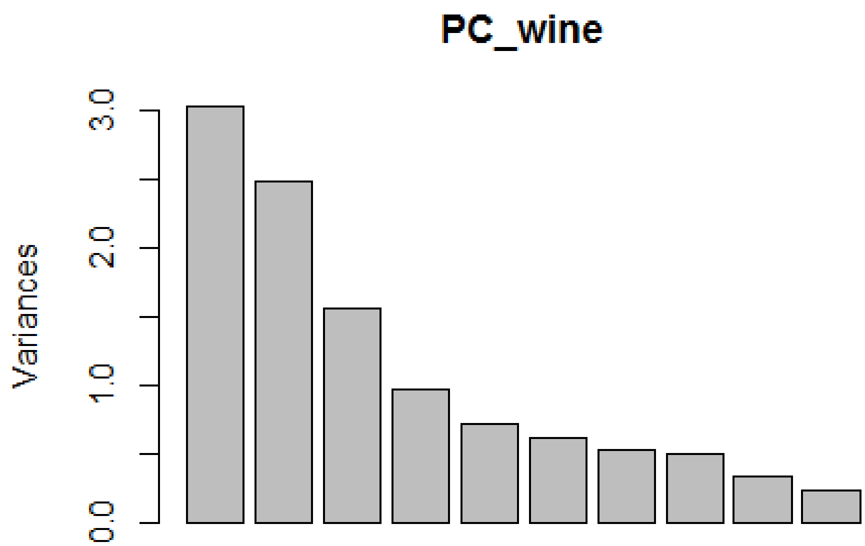
```
##
##             1         2        3        4         5         6         7
##   3 20.000000 6.6666667 23.33333 16.66667  6.666667 13.333333 13.333333
##   4 29.166667 0.9259259 11.11111 29.16667 12.037037 10.648148  6.944444
##   5 21.983162 1.2628625 30.30870 21.18803 12.441534  3.601497  9.214219
##   6 12.200282 0.5641749 22.56700 19.21721 16.748942 19.464034  9.238364
##   7  3.985171 0.1853568 11.30677 12.32623 17.608897 41.612604 12.974977
##   8  1.036269 0.0000000 11.39896 12.95337 16.062176 51.295337  7.253886
```

Conclusion : k-means clustering technique does not do a good job at distinguishing high-quality wine from a low-quality wine

Checking if PCA can help us distinguish the quality of wine

PCA

```
PC_wine = prcomp(winedata_num, scale=TRUE)
plot(PC_wine)
```



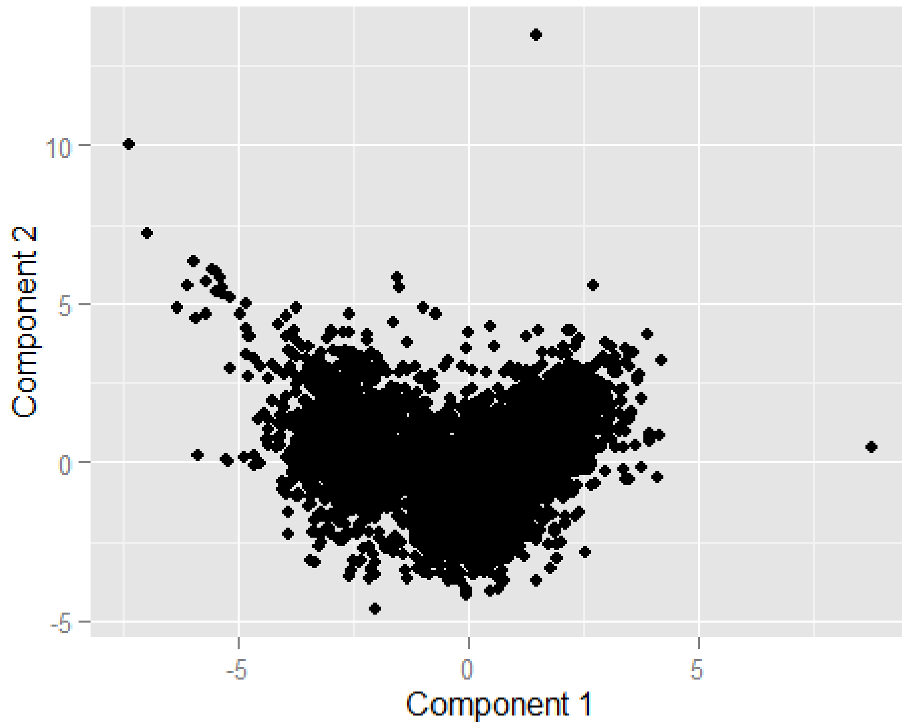
####Assigning vectors and alpha values from the principal component analysis output

```
loadings_wine = PC_wine$rotation
scores = PC_wine$x
head(loadings_wine)
```

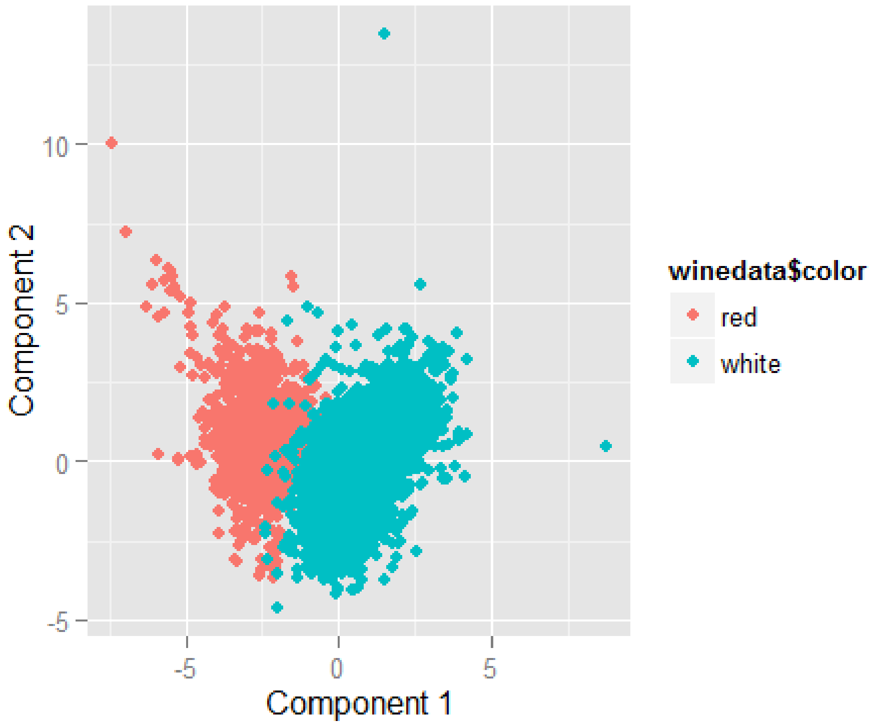
```
##                           PC1       PC2        PC3        PC4        PC5
## fixed.acidity      -0.2387989 0.3363545 -0.4343013  0.1643462 -0.1474804
## volatile.acidity   -0.3807575 0.1175497  0.3072594  0.2127849  0.1514560
## citric.acid         0.1523884 0.1832994 -0.5905697 -0.2643003 -0.1553487
## residual.sugar      0.3459199 0.3299142  0.1646884  0.1674430 -0.3533619
## chlorides          -0.2901126 0.3152580  0.0166791 -0.2447439  0.6143911
## free.sulfur.dioxide  0.4309140 0.0719326  0.1342239 -0.3572789  0.2235323
##                           PC6         PC7         PC8        PC9
## fixed.acidity      -0.2045537 -0.28307944  0.40123564  0.3440567
## volatile.acidity   -0.4921431 -0.38915976 -0.08743509 -0.4969327
## citric.acid         0.2276338 -0.38128504 -0.29341234 -0.4026887
## residual.sugar     -0.2334778  0.21797554 -0.52487294  0.1080032
## chlorides           0.1609764 -0.04606816 -0.47151685  0.2964437
## free.sulfur.dioxide -0.3400514 -0.29936325  0.20780759  0.3666563
##                           PC10          PC11
## fixed.acidity      -0.281267685 -0.3346792663
## volatile.acidity    0.152176731 -0.0847718098
## citric.acid         0.234463340  0.0011089514
## residual.sugar     -0.001372773 -0.4497650778
## chlorides          -0.196630217 -0.0434375867
## free.sulfur.dioxide  0.480243340  0.0002125351
```

Plotting projections of points on the first 2 principal components

```
qplot(scores[,1], scores[,2], xlab='Component 1', ylab='Component 2')
```

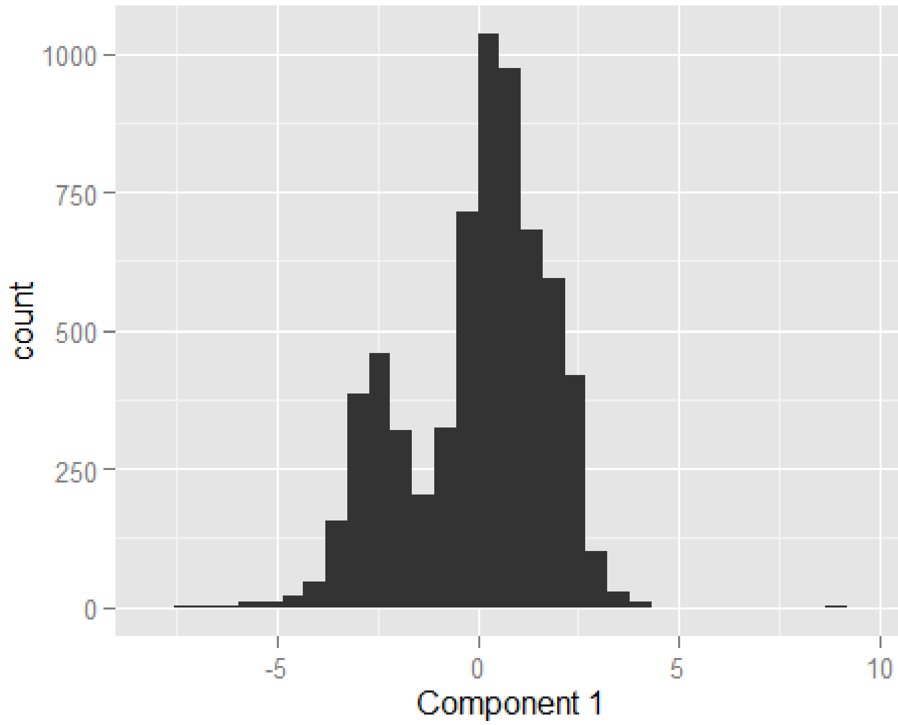



```
qplot(scores[,1], scores[,2], color = winedata$color, xlab='Component 1',
ylab='Component 2')
```

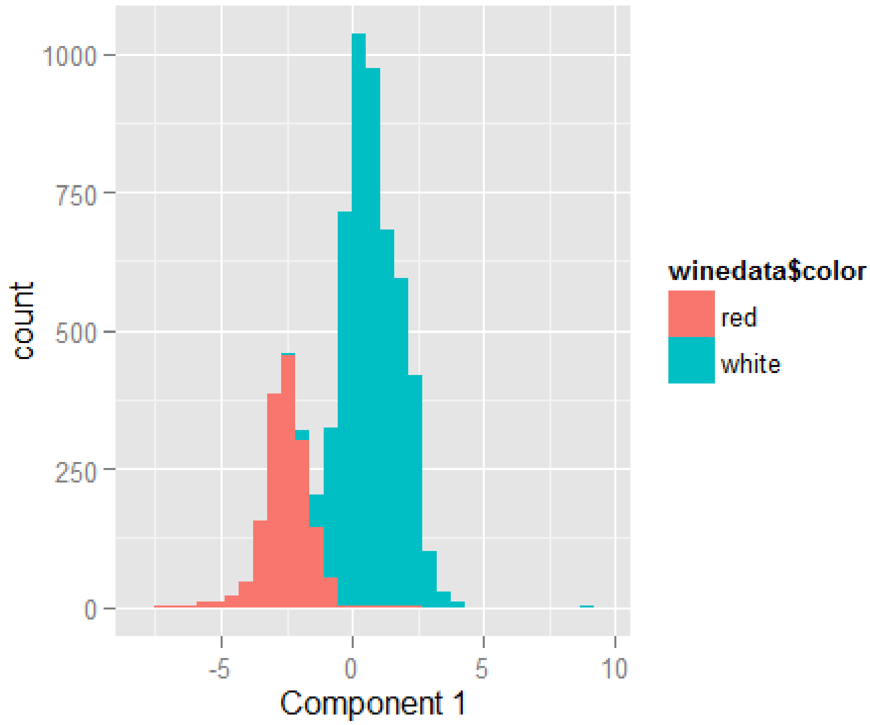


####It can be seen from the above plot that using PCA helps distinguish Red wine from White wine ####Checking to see if the first Principal Component alone helps distinguish the wines

```
qplot(scores[,1], xlab='Component 1')

## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust
this.
```

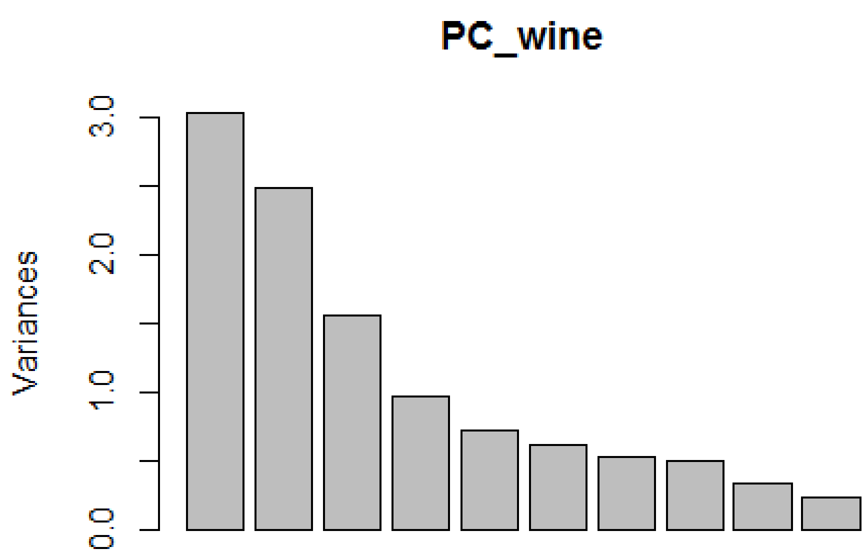



```
qplot(scores[,1], fill = winedata$color, xlab='Component 1')

## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust
this.
```

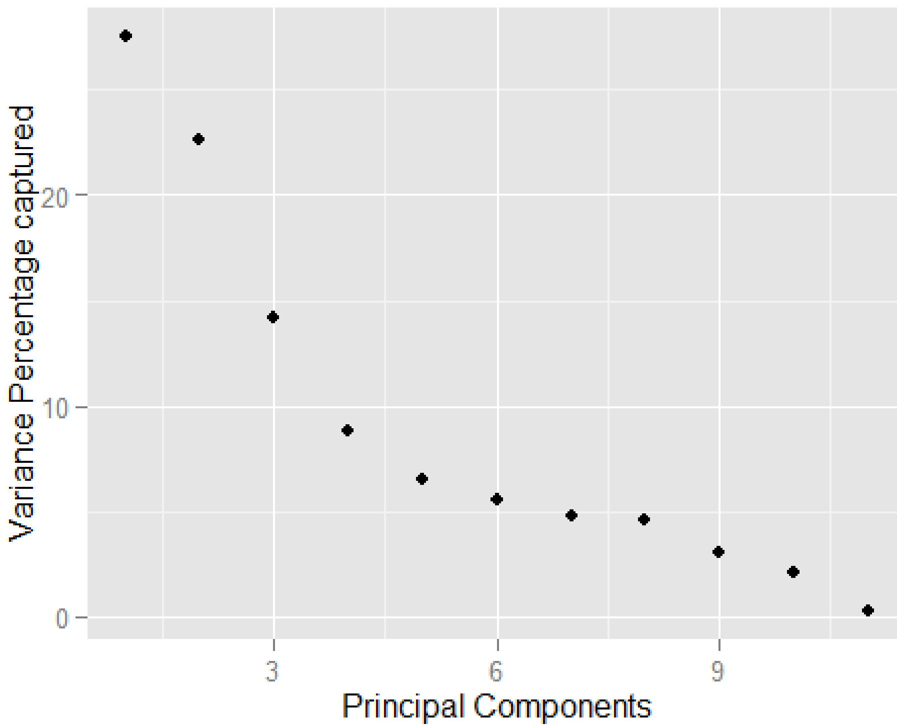


####Plot to see how the various multiple Principal components capture the variance

```
plot(PC_wine)
```



```r
summary(PC_wine)

## 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

Std_Dev_PCA = PC_wine$sdev
Variance_PCA = (Std_Dev_PCA)^2
Variance_perc = (Variance_PCA/sum(Variance_PCA)) * 100
qplot(,Variance_perc, xlab='Principal Components', ylab = 'Variance
Percentage captured',)
```



####The top features associated with each component

```r
o1_wine = order(loadings_wine[,1])
colnames(winedata)[head(o1_wine,2)]

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

colnames(winedata)[tail(o1_wine,2)]

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

Answer4:

Import the dataset and scaling the numeric dataset

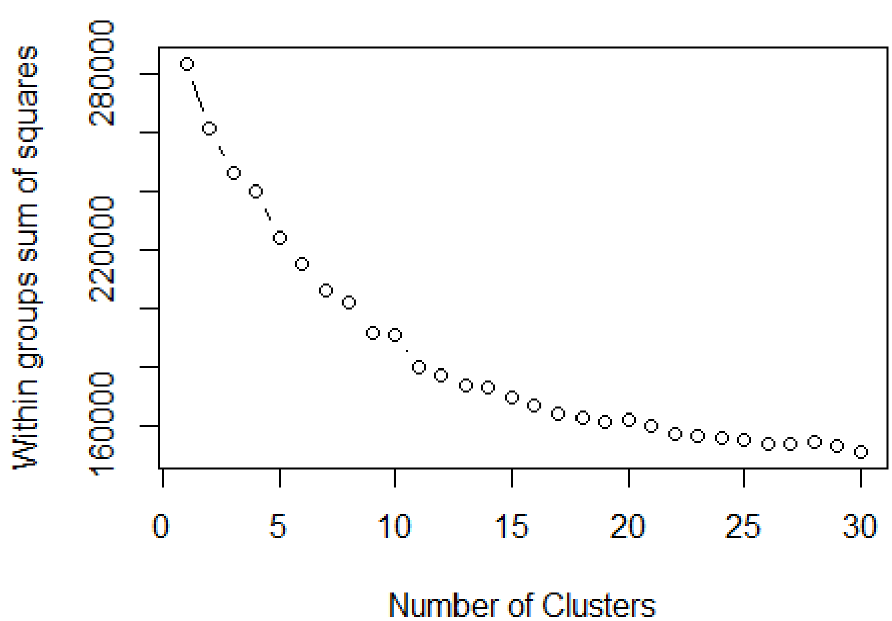
```
set.seed(5)
segmentation =
read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/social
_marketing.csv', header=TRUE)
segmentation = segmentation[,-1]
segmentation_scaled <- scale(segmentation, center=TRUE, scale=TRUE)
```

Decide the number of clusters

The denser the clusters and the more distant the clusters from each other the better

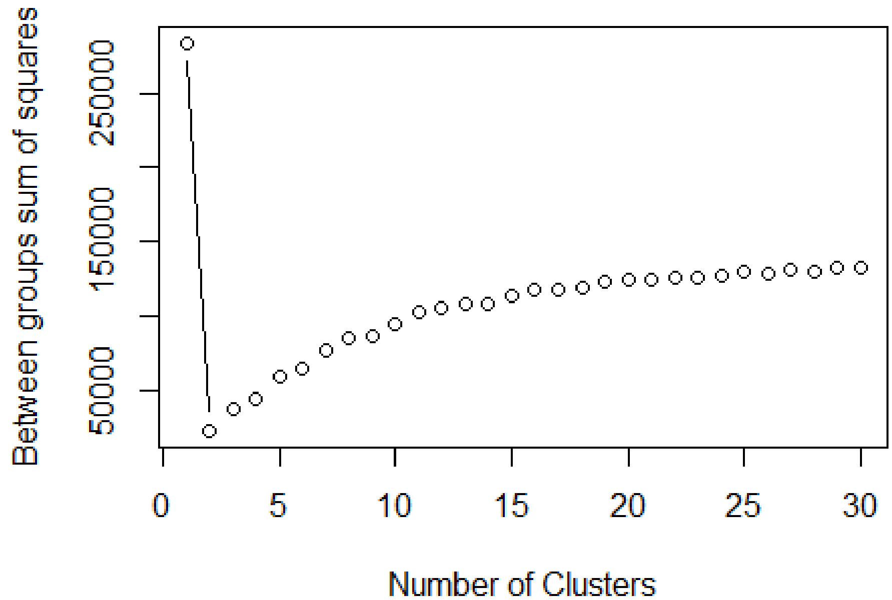
'Within groups sum of Squares' value drops sharply with increasing no. of clusters. But it starts levelling around 10 clusters.Also, the 'Between groups sum of Squares' does not increase appreciably beyond '10' clusters

```
wss <- (nrow(segmentation_scaled)-1)*sum(apply(segmentation_scaled,2,var))
for (i in 2:30) wss[i] <- sum(kmeans(segmentation_scaled,centers=i, iter.max
= 20)$withinss)
plot(1:30, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum
of squares")
```




```
bss <- (nrow(segmentation_scaled)-1)*sum(apply(segmentation_scaled,2,var))
for (i in 2:30) bss[i] <- sum(kmeans(segmentation_scaled,centers=i, iter.max
```

```
= 20)$betweenss)
plot(1:30, bss, type="b", xlab="Number of Clusters", ylab="Between groups sum
of squares")
```



####Cluster using k=10

```
set.seed(1000)
clustered <- kmeans(segmentation_scaled, centers=10,iter.max = 30, nstart=50)
```

Extracting attributes that would help us characterize the clusters from the output

```
head(clustered$center)

##       chatter current_events      travel photo_sharing uncategorized
## 1 -0.13106784    0.098568748 -0.10210840   -0.09702572   -0.10932175
## 2 -0.12055555    0.327439777  0.22299270   -0.08181427    0.69000791
## 3 -0.07726621    0.113662088  3.26563571   -0.11032799   -0.08797596
## 4 -0.12959312   -0.009409365 -0.15569133   -0.10874494    0.17199992
## 5 -0.04319508    0.177569765 -0.05423302    1.24167351    0.49901874
## 6 -0.06873643    0.072073400 -0.18660693   -0.22095375   -0.09408515
##       tv_film sports_fandom    politics       food      family
## 1 -0.09782764     2.0931845 -0.22395732  1.8526328  1.51930134
## 2  2.74947376    -0.1153915 -0.09202017  0.1493241 -0.11125482
## 3 -0.07173772    -0.2085897  3.11928957  0.1569816 -0.09231701
## 4 -0.14834245    -0.1983635 -0.20003892  0.4552042 -0.08904256
## 5 -0.13629038    -0.2057172 -0.12751983 -0.2037098  0.02911547
## 6 -0.01145700     0.6679035  1.22557740 -0.1542867  0.23545654
##   home_and_garden         music         news online_gaming     shopping
```

```
## 1      0.15922839  0.024736105 -0.110548369   -0.07770529 -0.02250247
## 2      0.33434666  1.004182705  0.004992348   -0.16802032  0.01956446
## 3      0.05166238 -0.041908204  1.140617963   -0.17046322 -0.07586007
## 4      0.15751336 -0.004650472 -0.074283081   -0.11065146 -0.05833223
## 5      0.14196331  0.552566673 -0.075788892   -0.02286982  0.20257187
## 6      0.16019548 -0.089179919  2.663930892   -0.12194071 -0.18819576
##   health_nutrition college_uni sports_playing     cooking           eco
## 1      -0.14332129 -0.13128067     0.10219662 -0.09767488  0.1844765002
## 2      -0.16017157  0.36662549     0.14097260 -0.14242668  0.0975311087
## 3      -0.16949729 -0.04922176     0.04384399 -0.18660894  0.1608323131
## 4       2.21843983 -0.20898762    -0.01853799  0.41620469  0.5642380561
## 5      -0.06622745 -0.01816877     0.20154607  2.82395159 -0.0009452388
## 6      -0.24281192 -0.19448941    -0.08412803 -0.23462522 -0.0962396906
##     computers    business     outdoors      crafts  automotive
## 1  0.09123101  0.10014569 -0.066878958  0.69985909  0.11801947
## 2 -0.15108700  0.34573339 -0.089221674  0.73532196 -0.22724285
## 3  2.91153602  0.55987463 -0.038264028  0.20332987 -0.13134399
## 4 -0.08444139  0.05256166  1.731014683  0.06666309 -0.17473888
## 5  0.05656488  0.22792402  0.007366432  0.08238866  0.01204133
## 6 -0.18667073 -0.12312259  0.310743356 -0.16067078  2.59007457
##             art    religion      beauty   parenting      dating
## 1 -0.0241511326  2.29792873  0.32148174  2.17066963  0.01821377
## 2  2.6369004837  0.01482072  0.01184033 -0.19635839 -0.05974777
## 3 -0.1616973087  0.11627370 -0.17714921  0.02354578  0.30530203
## 4 -0.0756353608 -0.16542539 -0.20155916 -0.08900958  0.19875142
## 5  0.0009203335 -0.12128984  2.63819768 -0.05784476  0.04883143
## 6 -0.1615620527 -0.17886371 -0.17643498  0.04114091 -0.03394992
##        school personal_fitness     fashion small_business        spam
## 1  1.68634487      -0.08971009  0.01242245     0.09195084 -0.07768727
## 2 -0.04757675      -0.15376088 -0.02202118     0.79092336 -0.07768727
## 3 -0.10592364      -0.14802999 -0.17050897     0.40150860 -0.07768727
## 4 -0.16501784       2.15735943 -0.09426523    -0.11649828 -0.07768727
## 5  0.17246492      -0.04418512  2.72842640     0.16429557 -0.07768727
## 6  0.01502133      -0.22990371 -0.21485572    -0.15569556 -0.07768727
##           adult
## 1 -0.0047783954
## 2 -0.0403803982
## 3 -0.1434066109
## 4  0.0181280362
## 5  0.0004888515
## 6 -0.1092934662
```

```r
mean = attr(segmentation_scaled,"scaled:center")
std_dev =attr(segmentation_scaled,"scaled:scale")
clustered$centers[1,]
```

```
##          chatter   current_events           travel    photo_sharing
##     -0.131067843      0.098568748     -0.102108399     -0.097025721
##    uncategorized          tv_film    sports_fandom         politics
##     -0.109321755     -0.097827641      2.093184497     -0.223957316
```

```
##             food           family  home_and_garden            music
##      1.852632778      1.519301344      0.159228392      0.024736105
##             news    online_gaming         shopping health_nutrition
##     -0.110548369     -0.077705294     -0.022502465     -0.143321290
##      college_uni   sports_playing          cooking              eco
##     -0.131280666      0.102196617     -0.097674880      0.184476500
##        computers         business         outdoors           crafts
##      0.091231014      0.100145691     -0.066878958      0.699859089
##       automotive              art         religion           beauty
##      0.118019466     -0.024151133      2.297928733      0.321481740
##        parenting           dating           school personal_fitness
##      2.170669629      0.018213768      1.686344874     -0.089710087
##          fashion   small_business             spam            adult
##      0.012422449      0.091950839     -0.077687267     -0.004778395
```

```
clustered$centers[1,]*std_dev + mean
```

```
##          chatter   current_events           travel    photo_sharing
##     3.936202e+00     1.651335e+00     1.351632e+00     2.431751e+00
##    uncategorized          tv_film   sports_fandom         politics
##     7.106825e-01     9.080119e-01     6.117211e+00     1.109792e+00
##             food           family  home_and_garden            music
##     4.686944e+00     2.584570e+00     6.379822e-01     7.047478e-01
##             news    online_gaming         shopping health_nutrition
##     9.732938e-01     1.000000e+00     1.348665e+00     1.922849e+00
##      college_uni   sports_playing          cooking              eco
##     1.169139e+00     7.388724e-01     1.663205e+00     6.543027e-01
##        computers         business         outdoors           crafts
##     7.566766e-01     4.925816e-01     7.017804e-01     1.087537e+00
##       automotive              art         religion           beauty
##     9.910979e-01     6.854599e-01     5.495549e+00     1.132047e+00
##        parenting           dating           school personal_fitness
##     4.210682e+00     7.433234e-01     2.771513e+00     1.246291e+00
##          fashion   small_business             spam            adult
##     1.019288e+00     3.931751e-01    -2.341877e-17     3.946588e-01
```

To characterize each cluster, it helps to look at the scaled and unscaled center value of each cluster with repect to all of the twitter interests. If the standard deviation is greater than 2 then that interest can be labeled significant for that particular cluster.

Cluster1

```
a1 <- rbind(clustered$center[1,],(clustered$center[1,]*std_dev + mean))
```

Cluster2

```
a2 <- rbind(clustered$center[2,],(clustered$center[2,]*std_dev + mean))
```

Cluster3

```
a3 <-rbind(clustered$center[3,],(clustered$center[3,]*std_dev + mean))
```

Cluster4

```
a4 <-rbind(clustered$center[4,],(clustered$center[4,]*std_dev + mean))
```

Cluster5

```
a5 <-rbind(clustered$center[5,],(clustered$center[5,]*std_dev + mean))
```

Cluster6

```
a6 <-rbind(clustered$center[6,],(clustered$center[6,]*std_dev + mean))
```

Cluster7

```
a7 <-rbind(clustered$center[7,],(clustered$center[7,]*std_dev + mean))
```

Cluster8

```
a8 <-rbind(clustered$center[8,],(clustered$center[8,]*std_dev + mean))
```

Cluster9

```
a9 <-rbind(clustered$center[9,],(clustered$center[9,]*std_dev + mean))
```

Cluster10

```
a10 <-rbind(clustered$center[10,],(clustered$center[10,]*std_dev + mean))
```

Cluster1 has teenagers who talk about computers, food, photo-sharing

Cluster4 has parents who talk about religion, parenting and food

Cluster6 has females who talk about cooking, fashion and beauty