Homework 1 STA380:P2

Zahref Beyabani

August 6, 2015

Analyzing Vote Undercount within the GA2000 Dataset

Read dataset

```
ga2000 =
read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/
georgia2000.csv', row.names=1)
```

Create a calculated attribute to hold the undercount numbers, and inspect the first few rows including undercount numbers of our dataset

```
ga2000$vote_undercount=(ga2000$ballots-ga2000$votes)
head(ga2000)

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

Factorize the factor variables so they are visible to R

```
ga2000$poor = as.factor(ga2000$poor)
ga2000$urban = as.factor(ga2000$urban)
ga2000$atlanta = as.factor(ga2000$atlanta)
```

Load the ggplot2 library to facilitate making plots

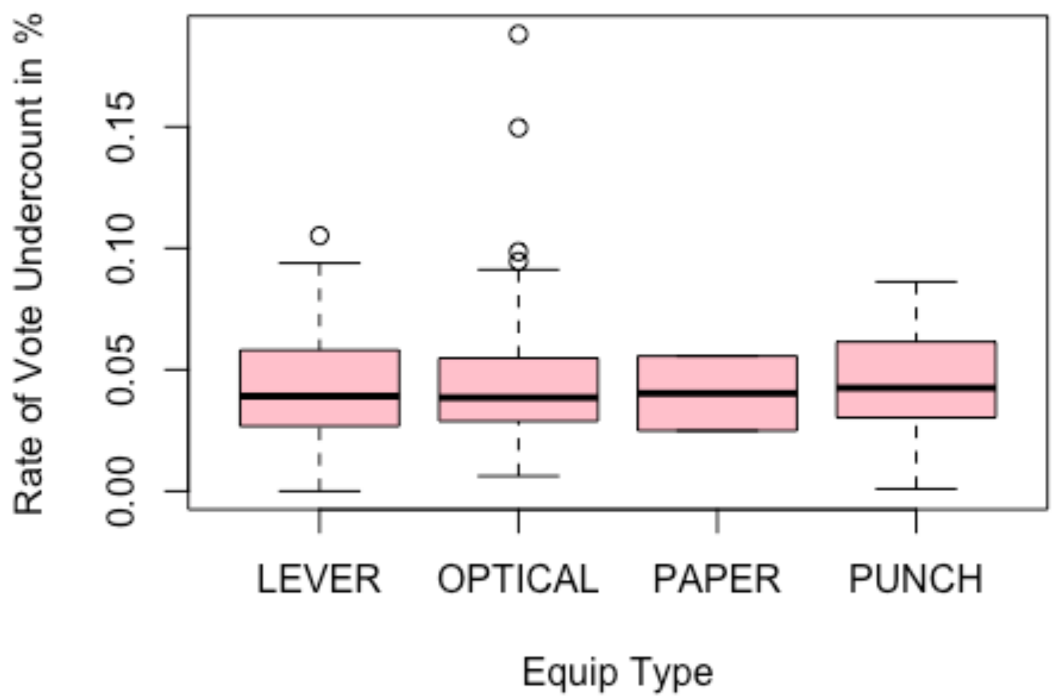
```
library(ggplot2)
```

Create a percentage undercount Variable

```
ga2000$pctUC = ((ga2000$vote_undercount/ga2000$ballots))
```

Plot the rate of undercount versus voting medium

```
plot(ga2000$equip, ga2000$pctUC, col="pink", ylab = "Rate of Vote
Undercount in %", xlab = "Equip Type")
```



- This tells us the undercount percentage for each machine
- We see that paper is the odd one out with not much variability
  - **Lets see how many times paper was used as a voting medium:**

```
summary(ga2000$equip)

##   LEVER OPTICAL   PAPER   PUNCH
##      74      66       2      17
```

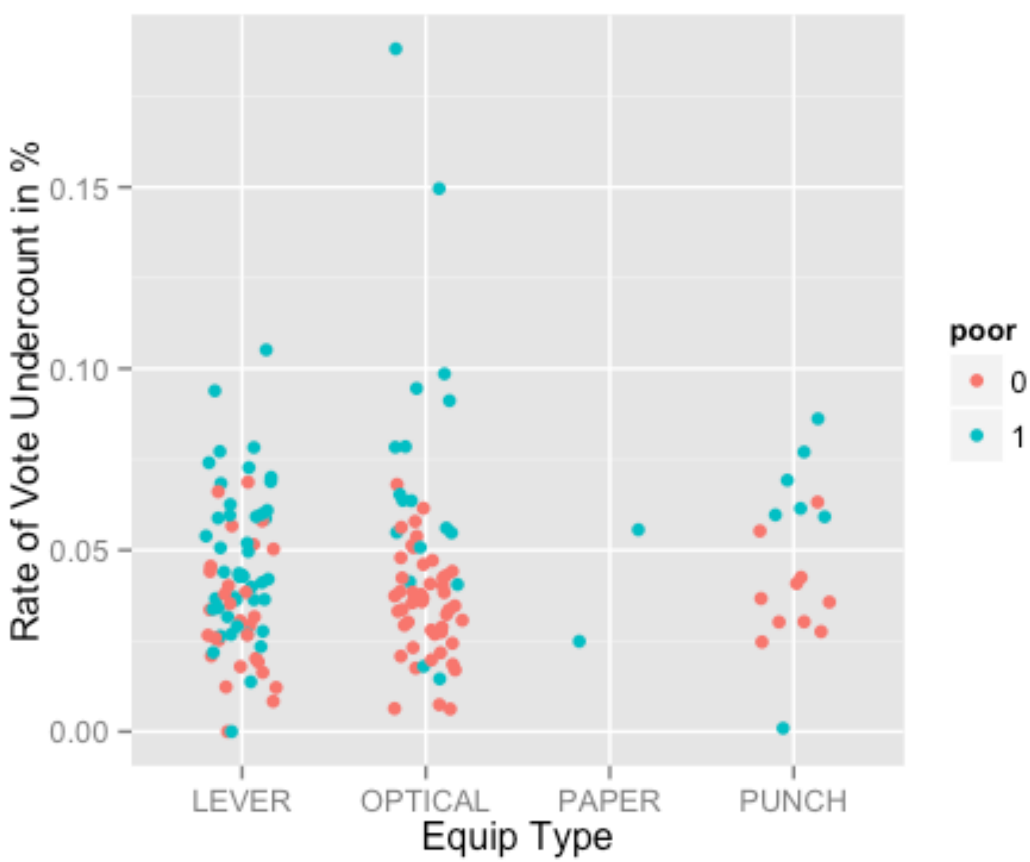
- Paper has only been used twice so we can safely say there are not enough data points to make inferences on paper's contribution to rate of vote undercount

- For the other three machines, however, we can say they are similar in how they contribute to the rate of undercount

Lets see if there's any connection with the county being classified as a poor one:

Plot the equipment vs undercount percentage with whether or not the county is identified as poor

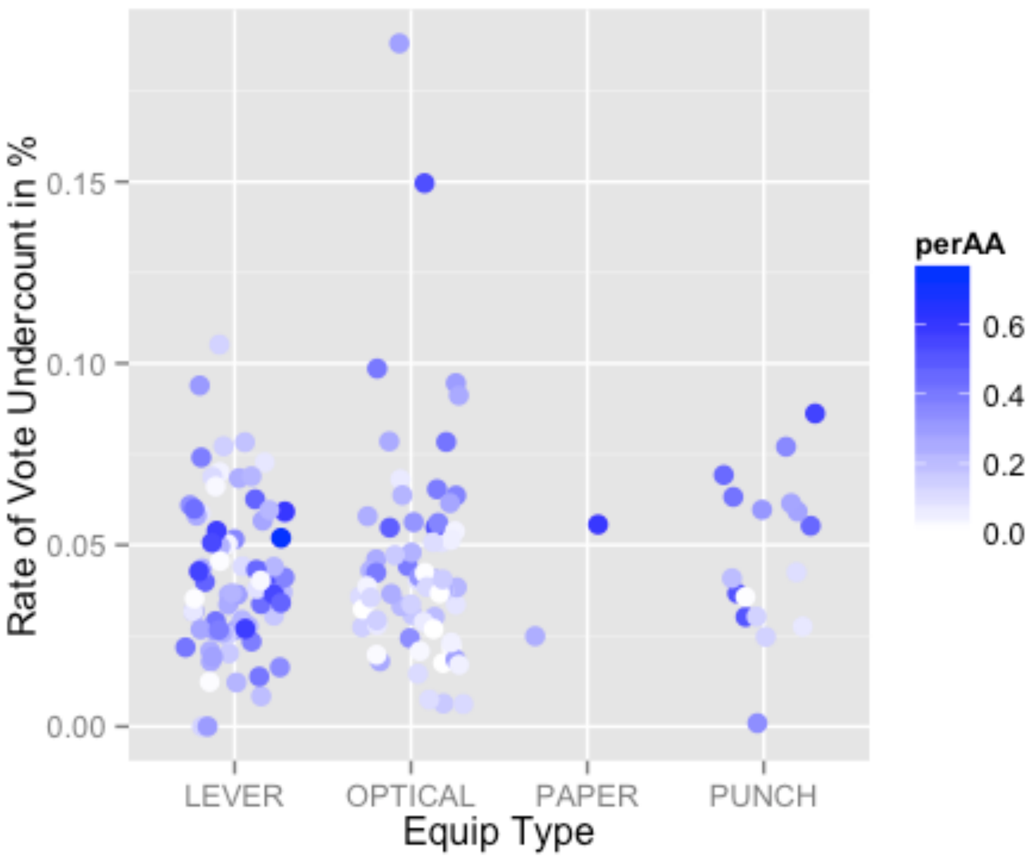
```
ggplot(ga2000, aes(y=ga2000$pctUC , col=poor, x=equip)
)+geom_point(size=2, position=position_jitter(width=0.20))+ylab("Rate
of Vote Undercount in %")+xlab("Equip Type")
```



- A reasonable conclusion is that counties that are identified as poor will have a higher rate of vote undercount regardless of the machine used

Now let's see if theres any connection with the percentage minority population living in that particular county and the vote undercount percentage

```
ggplot(ga2000, aes(y=ga2000$pctUC , col=perAA, x=equip)
)+geom_point(size=3, position=position_jitter(width=0.30))+ylab("Rate
of Vote Undercount in %")+xlab("Equip
Type")+scale_colour_gradient2(low="red", high="blue")
```



- Conclusion from the above plot : Minority status does not contribute to the vote undercount

---

Returns analysis on a balanced, risky, and safe portfolio

Using Monte-Carlo simulations, I will atempmt to analyze the returns on a perfectly balanced, a safe, and a risky portfolio (risk will be based on the stocks beta, and the variance of its returns) This will be done across 20 trading days. The dollar amount I will be simulating investing is $100,000.

Importing required libraries

Setting the random generator's seed to ensure reproducibility. Initialize the number of days variable to 20.

```
n_days = 20
set.seed(512)
```

Defining what stocks I want to analyze and, pulling the relevant data from Yahoo finance's API, for those stocks for the last 5 years.

```
mystocks = c("SPY", "TLT", "LQD", "EEM", "VNQ")
myprices = yahooSeries(mystocks, from='2010-01-01', to='2015-07-30')
```

A helper function for calculating percent returns from a Yahoo Series

Once sourced to the console, it will be availble for use in our portfolios

```
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)
```

Now lets look at the riskiness of our individual Stocks:

First fit the market model to each stock:

```
lm_TLT = lm(myreturns[,2] ~ myreturns[,1])
lm_LQD = lm(myreturns[,3] ~ myreturns[,1])
lm_EEM = lm(myreturns[,4] ~ myreturns[,1])
lm_VNQ = lm(myreturns[,5] ~ myreturns[,1])
```

Lets inspect the Betas of each stock:

```
cat(paste("Beta of TLT",lm_TLT$coefficients[2]))

## Beta of TLT -0.547628662133637
```

```r
cat(paste("Beta of LQD",lm_LQD$coefficients[2]))

## Beta of LQD -0.0381982702157758

cat(paste("Beta of EEM",lm_EEM$coefficients[2]))

## Beta of EEM 1.24343172386549

cat(paste("Beta of VNQ",lm_VNQ$coefficients[2]))

## Beta of VNQ 1.02949741644464
```

- Interpretability of the Beta of a stock is as follows:

"If a stock's price movements, or swings, are less than those of the market, then the beta value will be less than 1. Since increased volatility of stock price means more risk to the investor, it's reasonable to expect greater returns from stocks with betas over 1."

TL;DR: Higher-beta stocks are more volatile and therefore more riskier.

- From the above output we can determine that VNQ and EEM are the most riskiest, and that TLT and LQD are the least riskiest. SPY is the fund that tracks the market, and it will have a beta = 1.

Along the same lines, lets inspect the variance of the historical returns for the above mentioned assets, to substantiate our insight from the betas:

```r
#Computations:

##Varience of the returns
varSPY = var(myreturns[,1])
varTLT = var(myreturns[,2])
varLQD = var(myreturns[,3])
varEEM = var(myreturns[,4])
varVNQ = var(myreturns[,5])

#Outputs:
cat(paste("Varience on the return percentage of SPY : "),varSPY)

## Varience on the return percentage of SPY :  9.551231e-05

cat(paste("Varience on the return percentage of TLT : "),varTLT)

## Varience on the return percentage of TLT :  9.39768e-05

cat(paste("Varience on the return percentage of LQD : "),varLQD)

## Varience on the return percentage of LQD :  1.258955e-05

cat(paste("Varience on the return percentage of EEM : "),varEEM)

## Varience on the return percentage of EEM :  0.0002037578
```

```
cat(paste("Varience on the return percentage of VNQ : "),varVNQ)

## Varience on the return percentage of VNQ :  0.000159674
```

- With this output its clear, our safest asset classes are those with the least varience and which also have lower betas, and our riskiest asset classes are those with the higher variences and higher betas.
- Financially this makes sesnse, as we expect returns on a risky asset to vary much more.

Based on these insights from the data lets build our portfolios.

Safe Portfolio:

- LQD - Lowest beta, and lowest variance which means lowest risk; **91** % of my wealth goes here
- TLT - 6% of my wealth here because because it is the next lower beta stock
- SPY - 3% of my wealth here beacuase because it is the next lower beta stock after TLT.

Selecting the returns from these stocks for the above mentioned safe portfolio:

```
safe_assets_returns=myreturns[,c(1:3)]
```

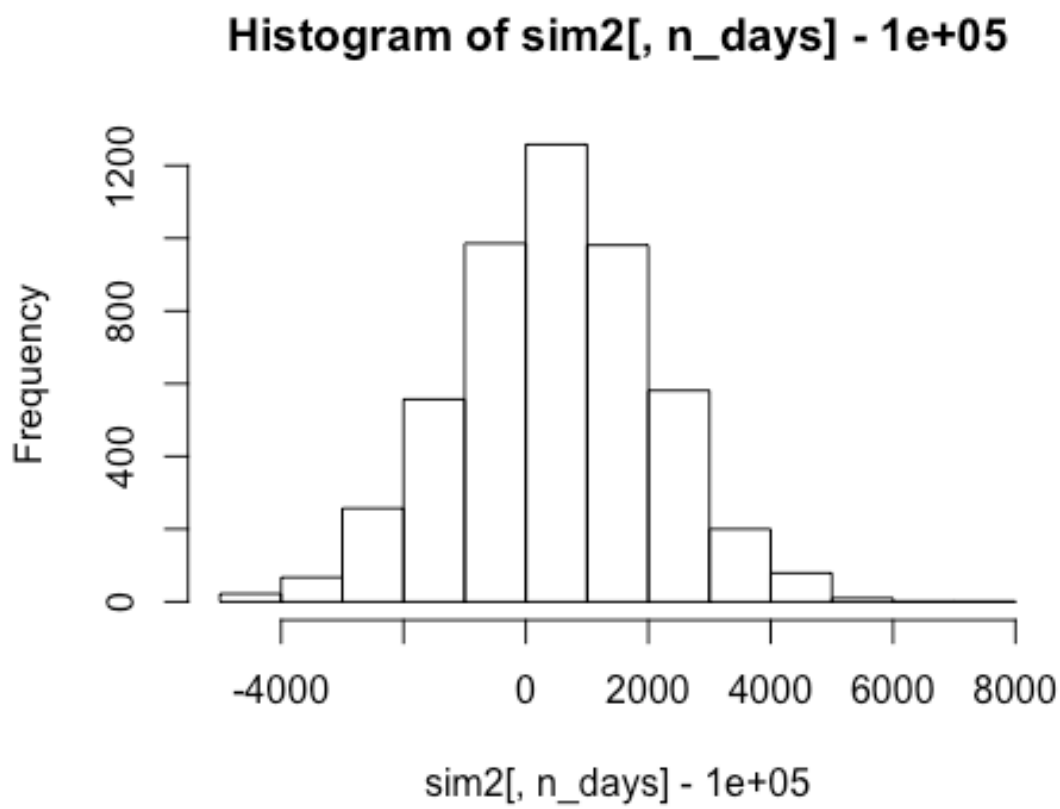
Running a Monte-Carlo simulation to simulate 20 trading days 5000 times for the "safe asset classes portfolio", rebalancing the portfolio at the end of each day:

```
set.seed(512)

sim2 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.03, 0.06, 0.91)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(safe_assets_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
    ##rebalancing the portfolio at the end of each trading day
    holdings = weights * totalwealth
  }
  wealthtracker
}
```

Now that the simulation has been run, lets see how well our asset classes did!

Let us visualize the distribution of our profits

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



Let's now inspect the 5% value at risk:

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

##       5%
## -2249.12
```

- Not bad! 5% of the time we will make a loss of $2249.12 or more

==

100% Balanced Portfolio across all asset classes:

Running a Monte-Carlo simulation to simulate 20 trading days 5000 times for the balanced portfolio across all assets, rebalancing the portfolio at the end of each day:

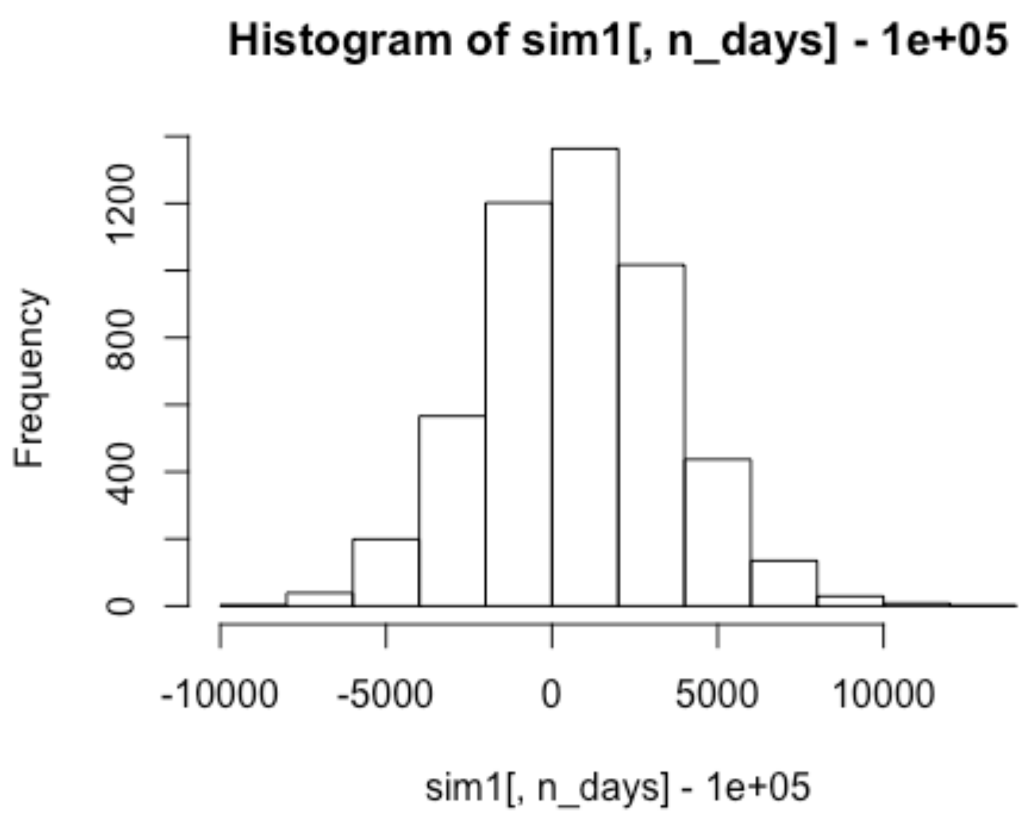
```
set.seed(512)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
```

```
  weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total 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
    ##rebalancing portfolio at the end of each trading day
    holdings = weights * totalwealth
  }
  wealthtracker
}
```

Now that the simulation has been run, lets see how well our asset classes did within the balanced portfolio!

Let us visualize the distribution of our profits

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



Let's now inspect the 5% value at risk:

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

##        5%
## -3900.527
```

- Okay, 5% of the time we will make a loss of $3900.53 or more, obviously we expected something like this, as we do better in the safe portfolio by about $1000.

==

Risky Portfolio:

- EEM - HIGHEST beta amongst all 5 which means highest risk, and it has the highest variance, so theres a potential for great returns, so **98** % of my wealth goes here.
- VNQ - 2% of my wealth here because it's the next riskiest asset

Selecting the returns from these stocks for the above mentioned RISKY portfolio:

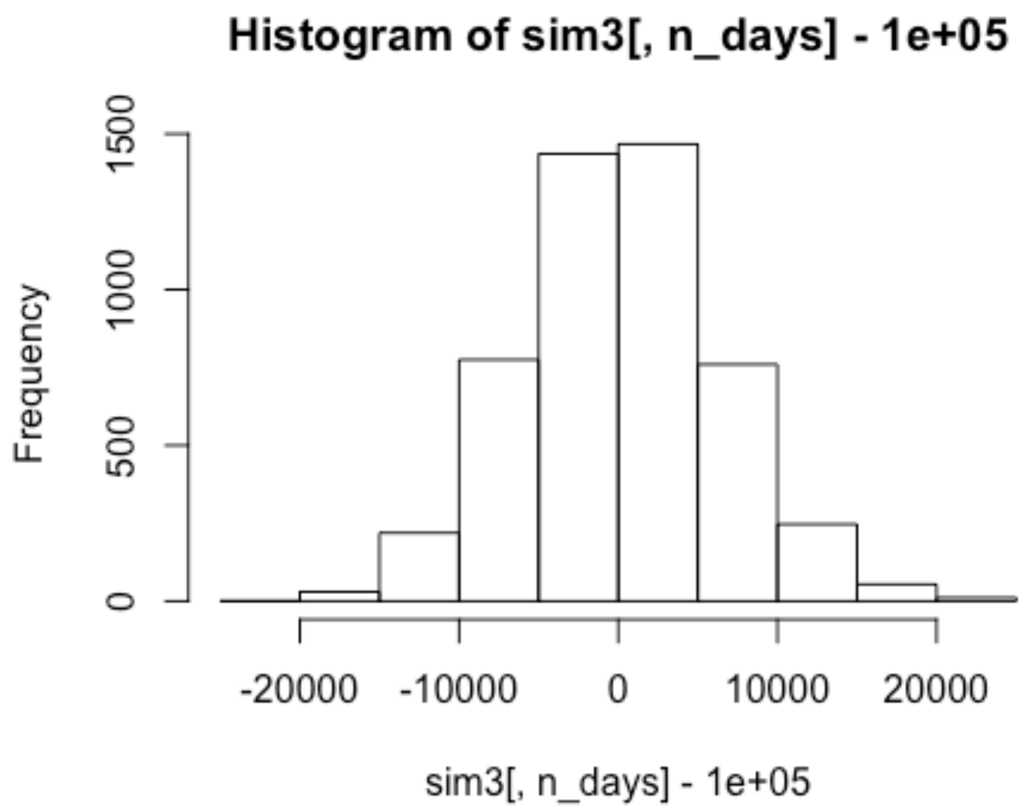
```
risky_assets_returns=myreturns[,c(4:5)]
```

Running a Monte-Carlo simulation to simulate 20 trading days 5000 times for the risky portfolio, rebalancing the portfolio at the end of each day:

```
set.seed(512)
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.98,0.02)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
    return.today = resample(risky_assets_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
    ##rebalancing portfolio at the end of each trading day
    holdings = weights * totalwealth
  }
  wealthtracker
}
```

Let us visualize the distribution of our profits

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



Let's now inspect the 5% value at risk:

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

##        5%
## -9990.475
```

- Ouch! 5% of the time we will make a loss of $9990.48 or more; obviously we expected something like this, as we are choosing very volatile stocks compared to our balanced and safe portfolio.

Clustering and PCA on the Wine Data Set

Importing required libraries

Reading in the data from the class' github page, and setting a global seed

```
set.seed(512)
wine =
```

```
read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/wine.csv', header=TRUE)
```

Removing the last two columns as per the assginment statement, and using that dataset so we can perform unsupervised learning on it, also factoring the relevant factor variables (like quality)

```
wine$quality = as.factor(wine$quality)
wine_dim=wine[,c(1:11)]
```

Scaling the numeric data to mean = 0 and s.d = 1

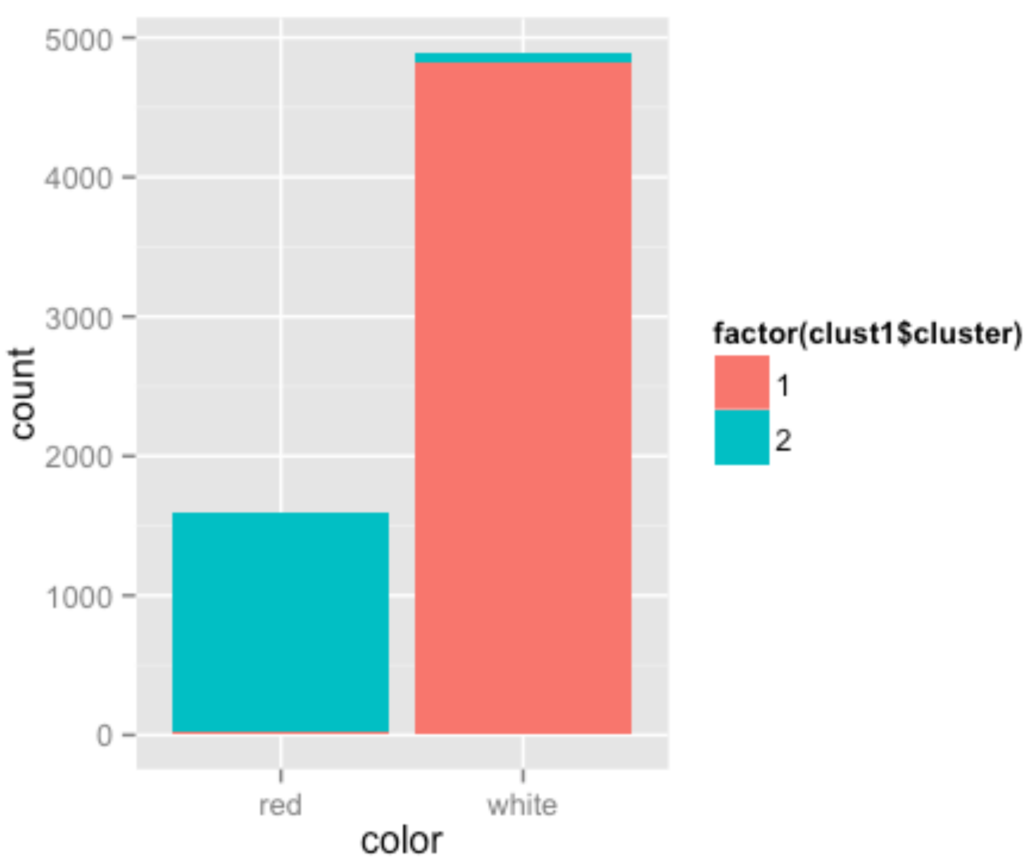
```
wine_dim = scale(wine_dim, center=TRUE, scale=TRUE)
```

Running the K-Means clustering algorithm, with 2 centers, 50 random initial centroids

```
clust1 = kmeans(wine_dim, 2, nstart=500)
```

Generating a plot of the result of the clustering algorithm

```
qplot(color, fill=factor(clust1$cluster), data = wine)
```



*This plot shows us the what the k-means clustering algorithm clustered to be in cluster 1 or 2, and whether or not the actual wine was a red wine or a white wine, so it looks like the attributes in cluster 1 most likely corresponds to properties of a red wine, and attributes of cluster 2 most likely correspond to the properties of a white wine.

Tabular representation of the above plot

```
t1=table(wine$color, clust1$cluster)
t1

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

This is telling us the numerical accuracy of the clusters, i.e: Cluster 1 ensconsed 4854 wines, and Cluster 2 ensconsed 1643 wines. However, in cluster 1, it assigned 24 wines inaccurately, and 68 in cluster 2

Probablity table of the above result

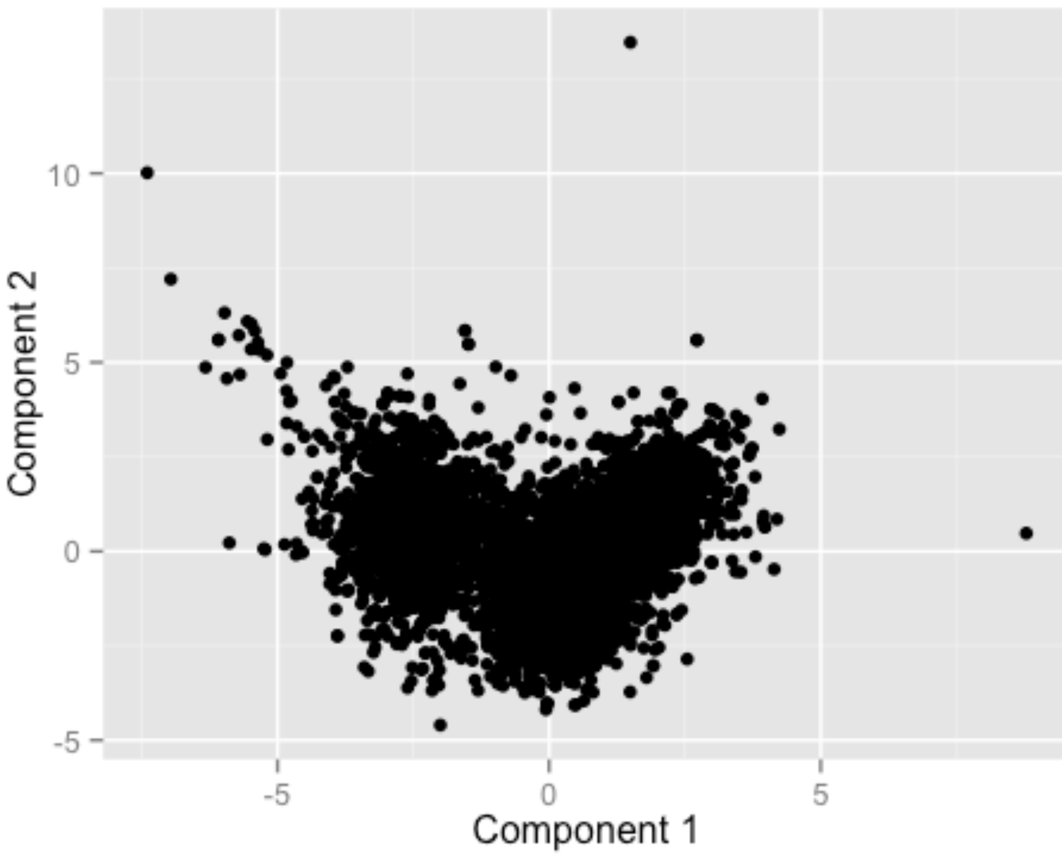
```
p1 = prop.table(t1, margin = 1)
p1

##
##                   1          2
##   red   0.01500938 0.98499062
##   white 0.98611678 0.01388322
```

- This is telling us the percentage accuracy with which each cluster classifies the wine.

Now lets try PCA on the dataset:

Running PCA on the wines data set, and plotting the first two Principal Computers along with the datapoints

```
pcomps = prcomp(wine_dim)
loadings = pcomps$rotation
scores = pcomps$x
qplot(scores[,1], scores[,2], xlab='Component 1', ylab='Component 2')
```

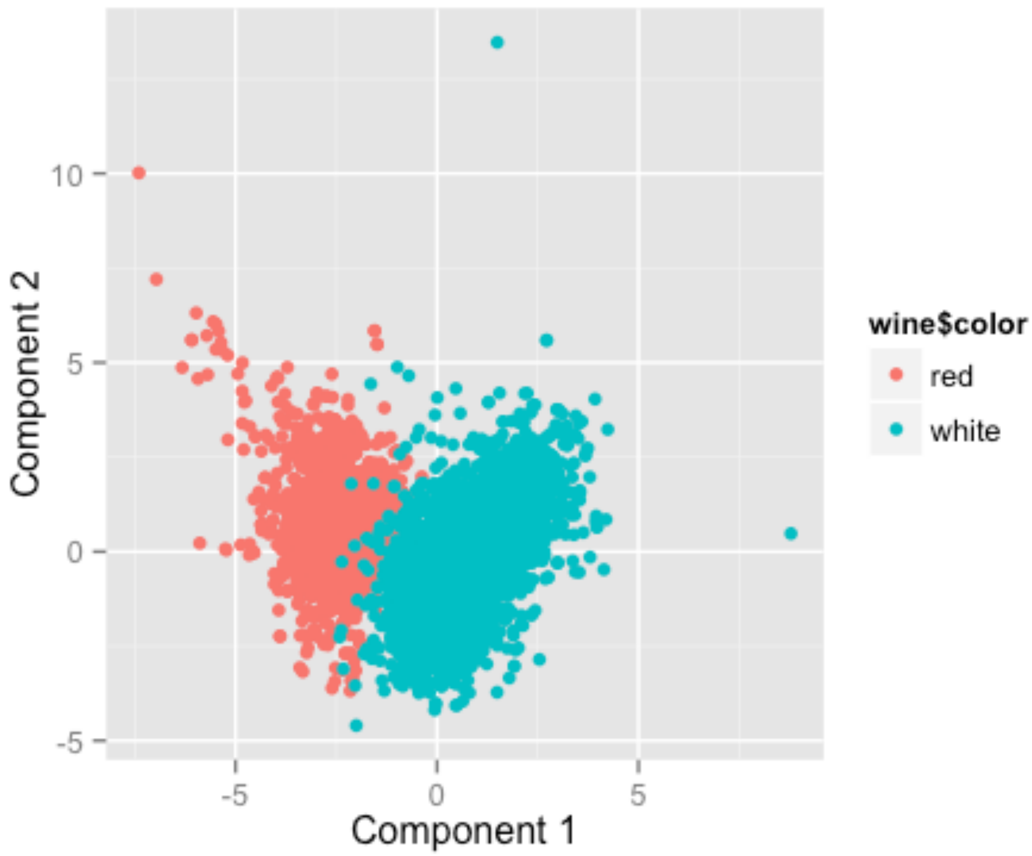


The 2 principal components plotted against each other show the projections of the points on the first two principal component vectors.

It appears as though there are two groupings with some overlap, between the two principal components

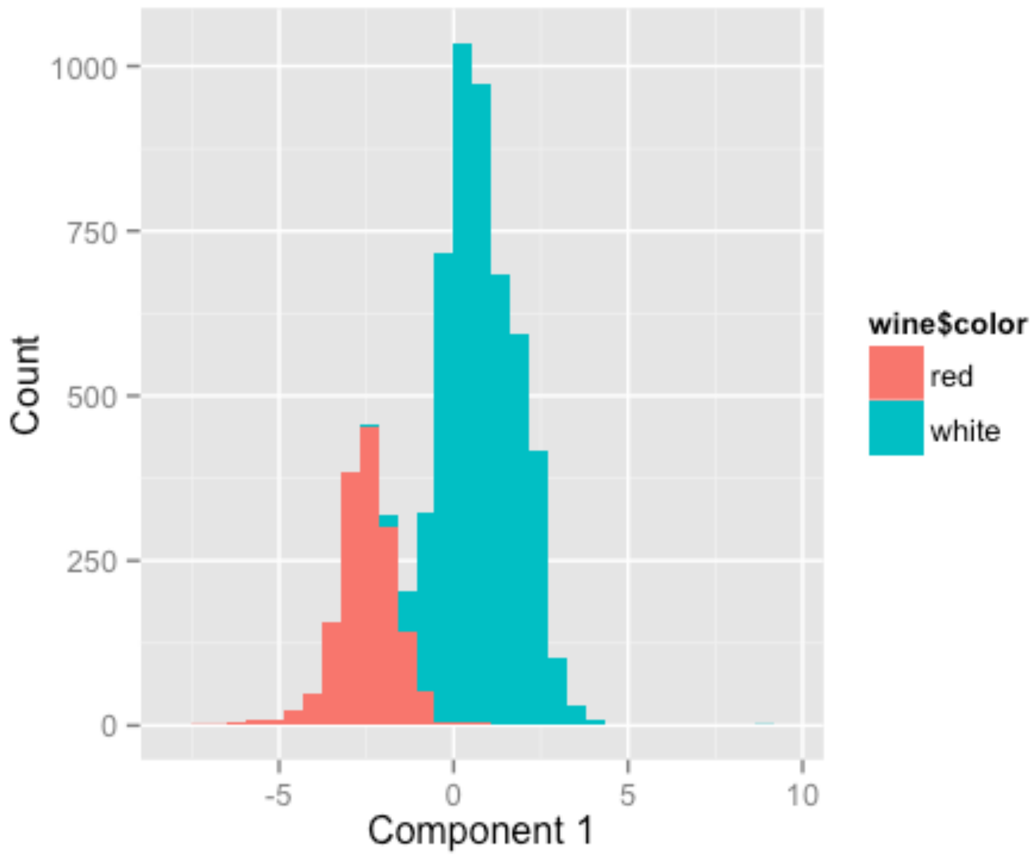
Lets superimpose our original colors of the wine onto our Principal Components to see if the groupings were relevant to the color of the wine

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



The above plot shows us that the first two principal components do a stellar job at identifying the color of the wine, with some minimal misclassification

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

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



This plot show us that just the first principle component does a decent job at predicting the wine color as it looks like it has identified two groupings, and the superimposition of the actual colors of the wines shows us that it has done a decent job.

However, two principal components capture the groups well, as seen above.

A conclusion drawn from this is that both K-means clustering and PCA does a good job at classifying the color of the wine.

Now lets tackle the wine quality classification!

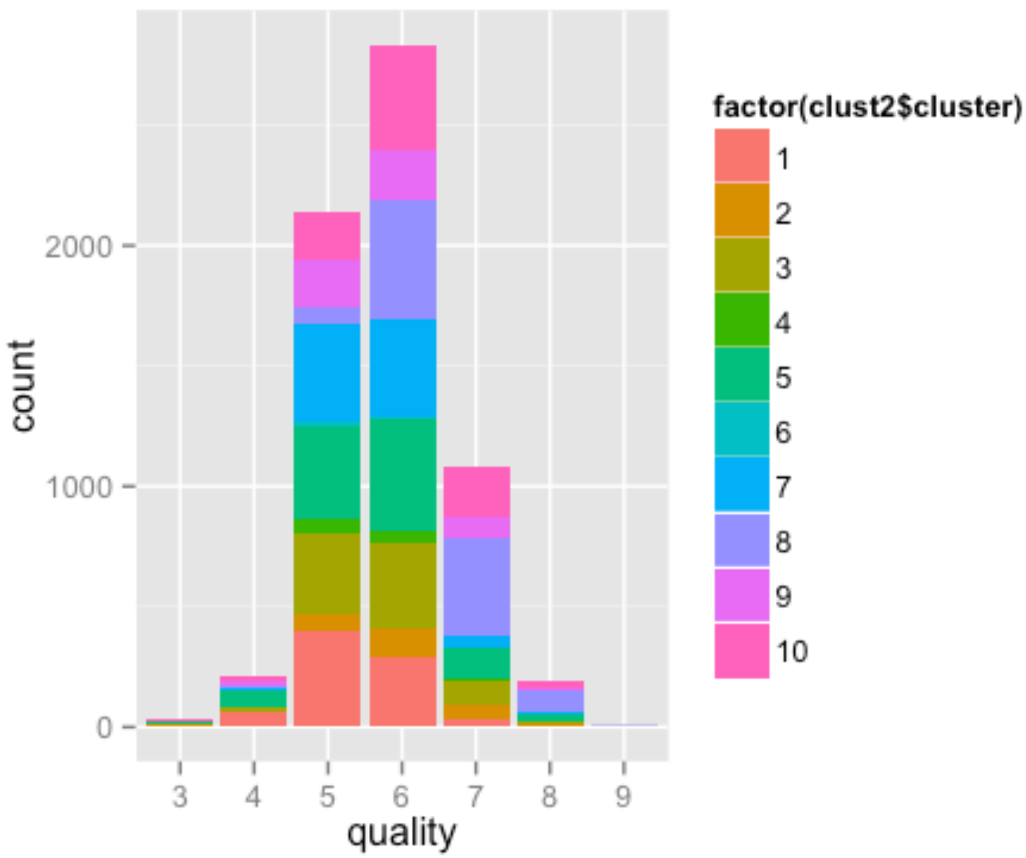
Running the K-Means clustering algorithm, with 10 centers because , 50 random initial centroids

```
set.seed(512)
clust2 = kmeans(wine_dim, 10, nstart=50)

## Warning: did not converge in 10 iterations
```

```
## Warning: did not converge in 10 iterations
```

Plotting the result from the k-means clustering on 10 clusters

```
qplot(quality, fill=factor(clust2$cluster), data = wine)
```



- For quality, the K-means technique does not seem capable of sorting the wines accurately, as it bins each of the distinct quality wines into different clusters which are not representative of the clusters

Tabular representation of the above plot

```
t2=table(wine$quality, clust2$cluster)
t2

## 
##       1   2   3   4   5   6   7   8   9  10
##   3   7   4   3   1   4   0   7   2   0   2
##   4  59   5  18   3  63   1  12  13  16  26
##   5 397  72 336  58 388  16 408  69 201 193
##   6 288 125 357  44 467   6 410 487 213 439
##   7  37  60  99   3 132   1  44 412  89 202
```

```
##    8    2    7   17    0   23    0   13   94    7   30
##    9    0    0    1    0    0    0    0    4    0    0
```

- This is telling us the numerical accuracy of the clusters, there seems to be a very distributed grouping of wine quality amongst the indentified 10 clusters.

Probablity table of the above result

```
p2 = prop.table(t2, margin = 1)
p2

##
##                 1            2            3            4            5
##    3 0.2333333333 0.1333333333 0.1000000000 0.0333333333 0.1333333333
##    4 0.2731481481 0.0231481481 0.0833333333 0.0138888889 0.2916666667
##    5 0.1856875585 0.0336763330 0.1571562208 0.0271281572 0.1814780168
##    6 0.1015514810 0.0440761636 0.1258815233 0.0155148096 0.1646685472
##    7 0.0342910102 0.0556070436 0.0917516219 0.0027803522 0.1223354958
##    8 0.0103626943 0.0362694301 0.0880829016 0.0000000000 0.1191709845
##    9 0.0000000000 0.0000000000 0.2000000000 0.0000000000 0.0000000000
##
##                 6            7            8            9           10
##    3 0.0000000000 0.2333333333 0.0666666667 0.0000000000 0.0666666667
##    4 0.0046296296 0.0555555556 0.0601851852 0.0740740741 0.1203703704
##    5 0.0074836296 0.1908325538 0.0322731525 0.0940130964 0.0902712816
##    6 0.0021156559 0.1445698166 0.1717207334 0.0751057828 0.1547954866
##    7 0.0009267841 0.0407784986 0.3818350324 0.0824837813 0.1872103800
##    8 0.0000000000 0.0673575130 0.4870466321 0.0362694301 0.1554404145
##    9 0.0000000000 0.0000000000 0.8000000000 0.0000000000 0.0000000000
```

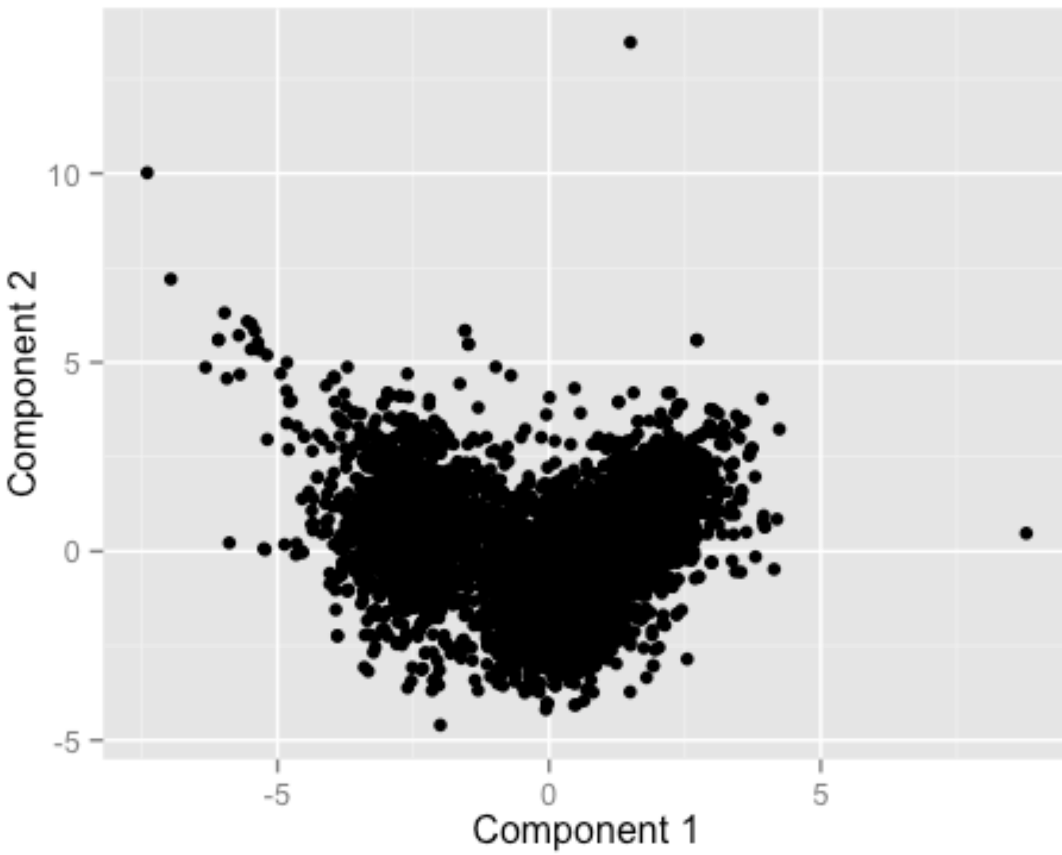
- This is telling us the probabalistic accuracy of the clusters, we dont seem to acheive any good accuracy scores for any quality within any cluster.

K means clustering does not do a good job for grouping the wines into clusters that represent different wine qualities.

Alright, Let's see how PCA does for quality

Running PCA, Again, on the wines data set, and plotting the first two Principal Computers along with the datapoints

```
pcomps = prcomp(wine_dim)
loadings = pcomps$rotation
scores = pcomps$x
qplot(scores[,1], scores[,2], xlab='Component 1', ylab='Component 2')
```

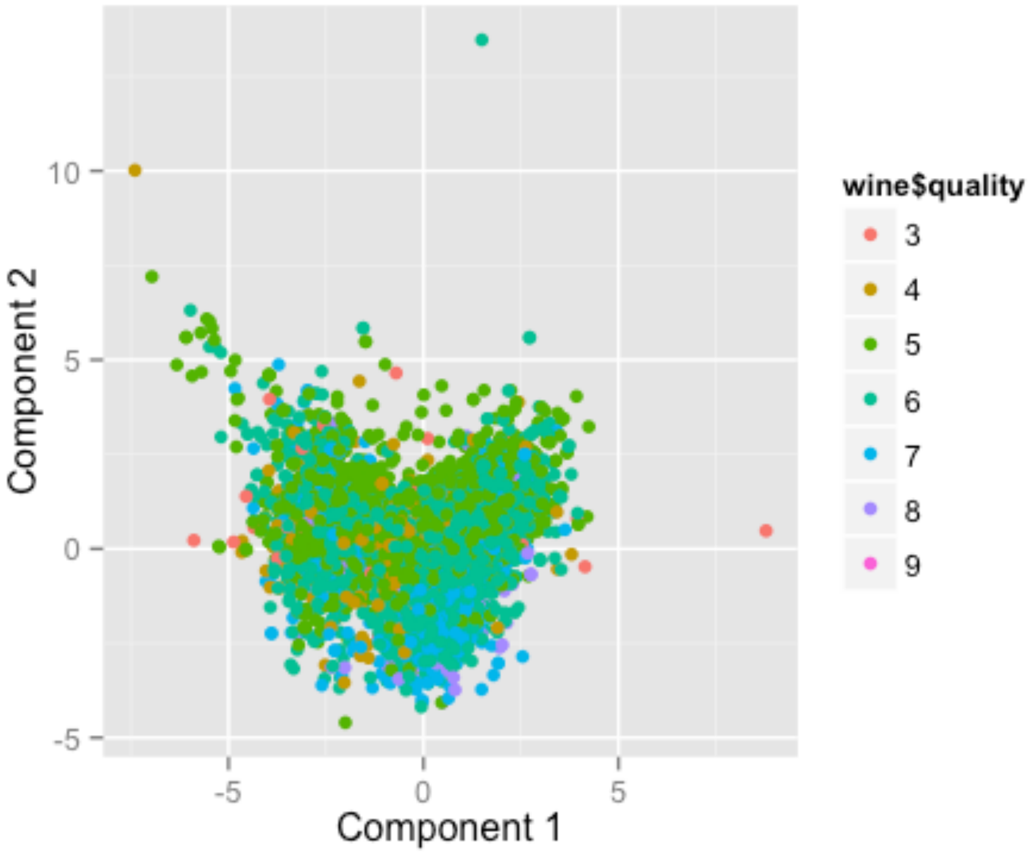


The 2 principal components plotted against each other show the projections of the points on the first two principal component vectors.

It appears as though there are two groupings with some overlap, between the two principal components

Lets superimpose our original qualities of the wine onto our Principal Components to see if the groupings were relevant to the qulaity of the wine.

```
qplot(scores[,1], scores[,2], col=wine$quality, xlab='Component 1',
ylab='Component 2')
```



We see that two principal components do not do a good job at classifying the qualities of the wine.

Conclusion :

Both algorithms are effective at discerning wine color, but not wine quality

---

Market segmentation based on collected tweets data with features identified for each tweet.

Reading in the data from the class' github page, and setting a global seed

```
tweets =
read.csv("https://raw.githubusercontent.com/jgscott/STA380/master/data/
social_marketing.csv",header = TRUE)
set.seed(512)
```

Deleting the first column from the dataset, as it does not provide a lot of value

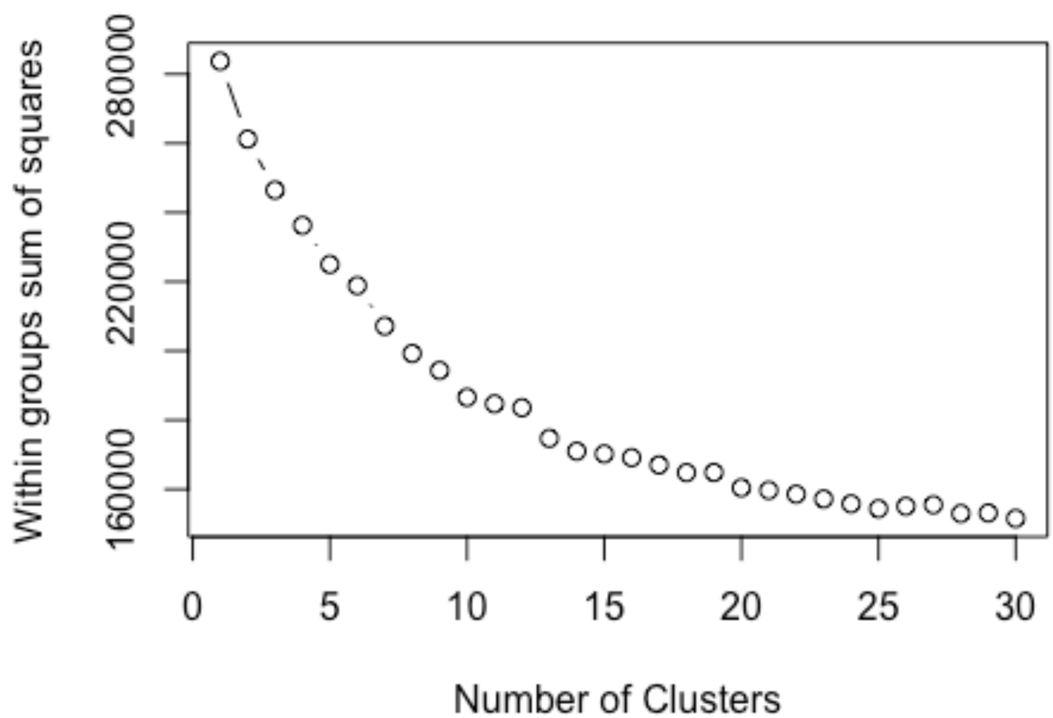
```
tweets = tweets[,-1]
```

Scaling the data so it has a mean of 1 and s.d = 0

```
tweets_scaled = scale(tweets, center=TRUE, scale=TRUE)
```

Picking a good value of K for the k-means model by analyzing the within group sum of squares for the different numbers of clusters

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



Based on this plot, we can see that k=10 seems to be providing good interpretability while not being too complex of a model.

Running a K-means clustering technique with 10 centers on the scaled tweets dataset

```
set.seed(512)
clusttwits = kmeans(tweets_scaled, centers=10, nstart=50)
```

Unscale the dataset to see the true values of the cluster centers

```
mu=attr(tweets_scaled,"scaled:center")
sigma=attr(tweets_scaled,"scaled:scale")
clusttwits$centers[1,]*sigma + mu
```

```
##          chatter   current_events           travel    photo_sharing
##     4.085714e+00     1.411429e+00     1.511429e+00     2.657143e+00
##    uncategorized          tv_film    sports_fandom         politics
##     7.800000e-01     1.234286e+00     1.302857e+00     1.257143e+00
##             food           family  home_and_garden            music
##     1.237143e+00     1.097143e+00     5.742857e-01     6.257143e-01
##             news    online_gaming         shopping health_nutrition
##     8.114286e-01     1.093714e+01     1.142857e+00     1.742857e+00
##      college_uni   sports_playing          cooking              eco
##     1.113714e+01     2.734286e+00     1.594286e+00     4.600000e-01
##        computers         business         outdoors           crafts
##     5.542857e-01     3.542857e-01     6.142857e-01     5.428571e-01
##       automotive              art         religion           beauty
##     9.228571e-01     1.171429e+00     7.257143e-01     4.085714e-01
##        parenting           dating           school personal_fitness
##     7.257143e-01     6.914286e-01     4.971429e-01     1.022857e+00
##          fashion   small_business             spam            adult
##     8.771429e-01     4.142857e-01     5.030698e-17     3.657143e-01
```

To interpret the cluster, we now bind the rows with the centers of the cluster and the average unscaled values.

This gives us the true cluster mean, and how many standard deviations away from the feture mean it is (for each feature)

Cluster 1:

```
rbind(clusttwits$center[1,],(clusttwits$center[1,]*sigma + mu))
```

Cluster 2:

```
rbind(clusttwits$center[2,],(clusttwits$center[2,]*sigma + mu))
```

Cluster 3:

```
rbind(clusttwits$center[3,],(clusttwits$center[3,]*sigma + mu))
```

Cluster 4:

```
rbind(clusttwits$center[4,],(clusttwits$center[4,]*sigma + mu))
```

Cluster 5:

```
rbind(clusttwits$center[5,],(clusttwits$center[5,]*sigma + mu))
```

Cluster 6:

```
rbind(clusttwits$center[6,],(clusttwits$center[6,]*sigma + mu))
```

Cluster 7:

```
rbind(clusttwits$center[7,],(clusttwits$center[7,]*sigma + mu))
```

Cluster 8:

```
rbind(clusttwits$center[8,],(clusttwits$center[8,]*sigma + mu))
```

Cluster 9:

```
rbind(clusttwits$center[9,],(clusttwits$center[9,]*sigma + mu))
```

Cluster 10:

```
rbind(clusttwits$center[10,],(clusttwits$center[10,]*sigma + mu))
```

How do we interpret this (cluster centers are in the appendix)?

We will segment users based on these clusters' characteristics that seem significant.

Significant is defined as a cluster center being 2 or more standard deviations away from the "global attribute mean"

The second row gives the average number of tweets for those features by each cluster that we are inspecting

Here are the market segments that have been identified based on the analysis detailed above within the tweets dataset.

Cluster 1:

```
1. Significant features: Online gaming, college uni, sports_playing.
+ I would classify this cluster as the college going males cluster.
+ This is because these topics are most likely to be relevant to young
males starting out their university education. They are probably
researching and talking about university/college related activities,
and one easy hobby to retain when in college (Gaming)
```

Cluster 2:

```
2. There are no significant features in this cluster based on the
evaluation criteria
```

Cluster 3:

```
3. There are no significant features in this cluster based on the
evaluation criteria
```

Cluster 4:

```
4. Significant features: news, automative.
+ I would classify this cluster as the individuals who are interested
in current events and automotive enthusiasts cluster. Perhaps even
interested in news related to cars. This is likely to be males who are
not fathers as they do not seem to be tweeting about parenting.
```

Cluster 5:

5. Significant features: spam and adult.
    - This is clearly the bots who spam segemnt.

Cluster 6:

6. Significant features: tvfilm and art.
    - This is the creative folks segment as the tweets in this cluster seem to relate mostly to TV & Flim, which is typical of people critiquing or talking about the arts, and they also tweet about art. One could also resonably assume that these are the individuals interested in Indie-Films.

Cluster 7:

7. Significant features: food, family, school, sports_fandom, religion, parenting.
    - This cluster clearly identifies the parents segment. The features are self explanatory. Except maybe for sports_fandom which could represent the dads who are also into sports.

Cluster 8:

8. Significant features: personal fitness, health nutrition, outdoors, photo_sharing.
    - This cluster seems to be of fitness and health conscious people as the tweets revolve around fitness & health. Typical crossfit junkies.

Cluster 9:

9. Significant features:travel, politics, computers.
    - This cluster identifies the segment of the people who are young profesionals just out of college, possibly in the tech industries (the tweets of the computers), and who are looking to get out and travel.

Cluster 10:

10. Significant features: photo sharing, beauty, cooking, fashion.
    - This cluster seems to segment the stereo-typical socialite, social media involved woman on the internet. We could infer that these women are not married or moms as there are not a significant amount of tweets regarding parenting within this cluster.

Apendix:

Cluster Centers

Cluster 1:

```
rbind(clusttwits$center[1,],(clusttwits$center[1,]*sigma + mu))

##          chatter current_events      travel photo_sharing
uncategorized
## [1,] -0.08870253    -0.09049938 -0.03219177   -0.01451015   -
0.03525299
## [2,]  4.08571429     1.41142857  1.51142857    2.65714286
0.78000000
##         tv_film sports_fandom   politics        food    family
## [1,] 0.09886703    -0.1347365 -0.1753446 -0.09030691 0.2059718
## [2,] 1.23428571     1.3028571  1.2571429  1.23714286 1.0971429
##      home_and_garden       music        news online_gaming   shopping
## [1,]      0.07276547 -0.05199434 -0.1875984      3.619885 -0.1362808
## [2,]      0.57428571  0.62571429  0.8114286     10.937143  1.1428571
##      health_nutrition college_uni sports_playing    cooking
eco
## [1,]       -0.1833537    3.309338       2.147690 -0.1177682 -
0.06795483
## [2,]        1.7428571   11.137143       2.734286  1.5942857
0.46000000
##        computers    business    outdoors     crafts automotive
art
## [1,] -0.08036615 -0.09959447 -0.1392195 0.03305173 0.06806834
0.2740668
## [2,]  0.55428571  0.35428571  0.6142857 0.54285714 0.92285714
1.1714286
##        religion     beauty  parenting      dating     school
## [1,] -0.1930684 -0.2233443 -0.1290952 -0.01090226 -0.2276908
## [2,]  0.7257143  0.4085714  0.7257143  0.69142857  0.4971429
##      personal_fitness     fashion small_business          spam
adult
## [1,]       -0.1826045 -0.06531985      0.1261023 -7.768727e-02 -
0.02073959
## [2,]        1.0228571  0.87714286      0.4142857  5.030698e-17
0.36571429
```

Cluster 2:

```r
rbind(clusttwits$center[2,],(clusttwits$center[2,]*sigma + mu))
```

```
##          chatter current_events   travel photo_sharing uncategorized
## [1,] -0.07726621      0.1136621 3.265636     -0.110328   -0.08797596
## [2,]  4.12607450      1.6704871 9.048711      2.395415    0.73065903
##          tv_film sports_fandom politics      food      family
## [1,] -0.07173772    -0.2085897  3.11929 0.1569816 -0.09231701
## [2,]  0.95128940     1.1432665 11.24355 1.6762178  0.75931232
##      home_and_garden      music     news online_gaming    shopping
## [1,]      0.05166238 -0.0419082 1.140618    -0.1704632 -0.07586007
## [2,]      0.55873926  0.6361032 3.601719     0.7507163  1.25214900
##      health_nutrition college_uni sports_playing    cooking
eco
## [1,]       -0.1694973 -0.04922176     0.04384399 -0.1866089
0.1608323
## [2,]        1.8051576  1.40687679     0.68194842  1.3581662
0.6361032
##      computers  business    outdoors    crafts automotive        art
## [1,]  2.911536 0.5598746 -0.03826403 0.2033299 -0.1313440 -0.1616973
## [2,]  4.083095 0.8108883  0.73638968 0.6819484  0.6504298  0.4613181
##       religion     beauty  parenting   dating     school
personal_fitness
## [1,] 0.1162737 -0.1771492 0.02354578 0.305302 -0.1059236        -
0.148030
## [2,] 1.3180516  0.4699140 0.95702006 1.255014  0.6418338
1.106017
##         fashion small_business          spam      adult
## [1,] -0.1705090      0.4015086 -7.768727e-02 -0.1434066
## [2,]  0.6848138      0.5845272  5.030698e-17  0.1432665
```

Cluster 3:

```r
rbind(clusttwits$center[3,],(clusttwits$center[3,]*sigma + mu))
```

```
##         chatter current_events     travel photo_sharing
uncategorized
## [1,] -0.1295931   -0.009409365 -0.1556913    -0.1087449
0.1719999
## [2,]  3.9414062    1.514322917  1.2291667     2.3997396
0.9739583
##         tv_film sports_fandom   politics      food      family
## [1,] -0.1483424    -0.1983635 -0.2000389 0.4552042 -0.08904256
## [2,]  0.8242187     1.1653646  1.1822917 2.2057292  0.76302083
##      home_and_garden        music        news online_gaming
shopping
## [1,]       0.1575134 -0.004650472 -0.07428308    -0.1106515 -
0.05833223
## [2,]       0.6367187  0.674479167  1.04947917     0.9114583
1.28385417
##      health_nutrition college_uni sports_playing   cooking       eco
## [1,]          2.21844  -0.2089876    -0.01853799 0.4162047 0.5642381
```

```
## [2,]         12.54167   0.9440104     0.62109375 3.4257812 0.9466146
##        computers   business outdoors     crafts automotive
art
## [1,] -0.08444139 0.05256166 1.731015 0.06666309 -0.1747389 -
0.07563536
## [2,]  0.54947917 0.45963542 2.876302 0.57031250  0.5911458
0.60156250
##        religion     beauty   parenting    dating     school
## [1,] -0.1654254 -0.2015592 -0.08900958 0.1987514 -0.1650178
## [2,]  0.7786458  0.4375000  0.78645833 1.0651042  0.5716146
##      personal_fitness     fashion small_business          spam
adult
## [1,]         2.157359 -0.09426523     -0.1164983 -7.768727e-02
0.01812804
## [2,]         6.651042  0.82421875      0.2643229 -4.076600e-17
0.43619792
```

Cluster 4:

```
rbind(clusttwits$center[4,],(clusttwits$center[4,]*sigma + mu))
```

```
##         chatter current_events     travel photo_sharing
uncategorized
## [1,] -0.1310678     0.09856875 -0.1021084   -0.09702572    -
0.1093218
## [2,]  3.9362018     1.65133531  1.3516320    2.43175074
0.7106825
##          tv_film sports_fandom   politics     food   family
## [1,] -0.09782764      2.093184 -0.2239573 1.852633 1.519301
## [2,]  0.90801187      6.117211  1.1097923 4.686944 2.584570
##      home_and_garden      music       news online_gaming    shopping
## [1,]       0.1592284 0.02473611 -0.1105484   -0.07770529 -0.02250247
## [2,]       0.6379822 0.70474777  0.9732938    1.00000000  1.34866469
##      health_nutrition college_uni sports_playing     cooking
eco
## [1,]       -0.1433213  -0.1312807      0.1021966 -0.09767488
0.1844765
## [2,]        1.9228487   1.1691395      0.7388724  1.66320475
0.6543027
##       computers  business    outdoors    crafts automotive
art
## [1,] 0.09123101 0.1001457 -0.06687896 0.6998591  0.1180195 -
0.02415113
## [2,] 0.75667656 0.4925816  0.70178042 1.0875371  0.9910979
0.68545994
##      religion    beauty parenting     dating   school
personal_fitness
## [1,] 2.297929 0.3214817  2.170670 0.01821377 1.686345      -
0.08971009
## [2,] 5.495549 1.1320475  4.210682 0.74332344 2.771513
1.24629080
```

```
##         fashion small_business          spam        adult
## [1,] 0.01242245     0.09195084 -7.768727e-02 -0.004778395
## [2,] 1.01928783     0.39317507 -2.341877e-17  0.394658754
```

Cluster 5:

```
rbind(clusttwits$center[5,],(clusttwits$center[5,]*sigma + mu))
```

```
##          chatter current_events     travel photo_sharing
uncategorized
## [1,] -0.06873643      0.0720734 -0.1866069    -0.2209537   -
0.09408515
## [2,]  4.15617716      1.6177156  1.1585082     2.0932401
0.72494172
##        tv_film sports_fandom politics       food    family
home_and_garden
## [1,] -0.011457     0.6679035 1.225577 -0.1542867 0.2354565
0.1601955
## [2,]  1.051282     3.0372960 5.503497  1.1235431 1.1305361
0.6386946
##            music     news online_gaming   shopping health_nutrition
## [1,] -0.08917992 2.663931    -0.1219407 -0.1881958       -0.2428119
## [2,]  0.58741259 6.801865     0.8811189  1.0489510        1.4755245
##      college_uni sports_playing    cooking         eco  computers
## [1,]  -0.1944894    -0.08412803 -0.2346252 -0.09623969 -0.1866707
## [2,]   0.9860140     0.55710956  1.1934732  0.43822844  0.4289044
##        business  outdoors     crafts automotive        art
religion
## [1,] -0.1231226 0.3107434 -0.1606708   2.590075 -0.1615621 -
0.1788637
## [2,]  0.3379953 1.1585082  0.3846154   4.368298  0.4615385
0.7529138
##          beauty  parenting      dating     school personal_fitness
## [1,] -0.1764350 0.04114091 -0.03394992 0.01502133       -0.2299037
## [2,]  0.4708625 0.98368298  0.65034965 0.78554779        0.9090909
##         fashion small_business          spam      adult
## [1,] -0.2148557     -0.1556956 -7.768727e-02 -0.1092935
## [2,]  0.6037296      0.2400932  5.377643e-17  0.2051282
```

Cluster 6:

```
rbind(clusttwits$center[6,],(clusttwits$center[6,]*sigma + mu))
```

```
##         chatter current_events     travel photo_sharing
uncategorized
## [1,] -0.3749515     -0.2007606 -0.2200161    -0.4279099    -
0.1748575
## [2,]  3.0755059      1.2715192  1.0821504     1.5279372
0.6493506
##         tv_film sports_fandom   politics       food     family
## [1,] -0.2240837    -0.3243659 -0.3018291 -0.3621585 -0.3038108
```

```
## [2,]  0.6985805     0.8930837  0.8737542  0.7544548  0.5197825
##      home_and_garden      music       news online_gaming   shopping
## [1,]      -0.2021051 -0.2313703 -0.3085987    -0.2316691 -0.4009288
## [2,]       0.3717910  0.4409544  0.5572335     0.5862277  0.6641498
##      health_nutrition college_uni sports_playing    cooking
eco
## [1,]       -0.3122336  -0.2554426     -0.2635711 -0.3259902 -
0.2751543
## [2,]        1.1633947   0.8094231      0.3820598  0.8800966
0.3005134
##       computers   business   outdoors     crafts automotive
art
## [1,] -0.2563483 -0.2466151 -0.3244816 -0.2935791 -0.3123111 -
0.2373941
## [2,]  0.3467230  0.2524917  0.3902144  0.2760495  0.4032014
0.3379644
##        religion     beauty  parenting     dating     school
## [1,] -0.3002986 -0.2728771 -0.3223924 -0.1748173 -0.3235242
## [2,]  0.5203866  0.3427967  0.4327998  0.3992751  0.3832679
##      personal_fitness    fashion small_business          spam
adult
## [1,]       -0.3313633 -0.2938050     -0.2099046 -7.768727e-02 -
0.01272528
## [2,]        0.6650559  0.4593778      0.2065841 -2.818926e-16
0.38024766
```

Cluster 7:

```
rbind(clusttwits$center[7,],(clusttwits$center[7,]*sigma + mu))
```

```
##        chatter current_events   travel photo_sharing uncategorized
## [1,] 0.0720588      0.2768471 0.288727   -0.09071828     0.1125985
## [2,] 4.6530612      1.8775510 2.244898    2.44897959     0.9183673
##        tv_film sports_fandom  politics       food      family
## [1,] -0.116191     0.1406567 0.1505274 0.04049422 -0.05999555
## [2,]  0.877551     1.8979592 2.2448980 1.46938776  0.79591837
##      home_and_garden      music          news online_gaming
shopping
## [1,]       0.2351019 0.01418264 -0.0006901999    0.08935906 -
0.2378226
## [2,]       0.6938776 0.69387755  1.2040816327    1.44897959
0.9591837
##      health_nutrition college_uni sports_playing     cooking
eco
## [1,]       0.05086059   0.1273275     -0.1112904 -0.05898219
0.4479995
## [2,]       2.79591837   1.9183673      0.5306122  1.79591837
0.8571429
##      computers   business   outdoors    crafts automotive       art
## [1,] 0.2975329 -0.3460090 0.2978031 0.2179337  0.1245356 0.3316754
## [2,] 1.0000000  0.1836735 1.1428571 0.6938776  1.0000000 1.2653061
```

```
##       religion     beauty parenting       dating     school
## [1,] 0.1207018 -0.1007020 0.1865841 -0.009528244 0.09244824
## [2,] 1.3265306  0.5714286 1.2040816  0.693877551 0.87755102
##      personal_fitness     fashion small_business      spam    adult
## [1,]        0.1218324 -0.02044987      0.3142883 12.418865 3.750222
## [2,]        1.7551020  0.95918367      0.5306122  1.040816 7.204082
```

Cluster 8:

```
rbind(clusttwits$center[8,],(clusttwits$center[8,]*sigma + mu))
```

```
##         chatter current_events    travel photo_sharing uncategorized
## [1,] -0.1205556      0.3274398 0.2229927   -0.08181427     0.6900079
## [2,]  3.9733010      1.9417476 2.0946602    2.47330097     1.4587379
##       tv_film sports_fandom    politics      food     family
## [1,] 2.749474    -0.1153915 -0.09202017 0.1493241 -0.1112548
## [2,] 5.631068     1.3446602  1.50970874 1.6626214  0.7378641
##      home_and_garden    music        news online_gaming   shopping
## [1,]       0.3343467 1.004183 0.004992348    -0.1680203 0.01956446
## [2,]       0.7669903 1.713592 1.216019417     0.7572816 1.42475728
##      health_nutrition college_uni sports_playing    cooking
eco
## [1,]       -0.1601716   0.3666255      0.1409726 -0.1424267
0.09753111
## [2,]        1.8470874   2.6116505      0.7766990  1.5097087
0.58737864
##       computers  business    outdoors   crafts automotive      art
## [1,] -0.1510870 0.3457334 -0.08922167 0.735322 -0.2272429 2.636900
## [2,]  0.4708738 0.6626214  0.67475728 1.116505  0.5194175 5.021845
##        religion     beauty  parenting      dating      school
## [1,] 0.01482072 0.01184033 -0.1963584 -0.05974777 -0.04757675
## [2,] 1.12378641 0.72087379  0.6237864  0.60436893  0.71116505
##      personal_fitness     fashion small_business          spam
adult
## [1,]       -0.1537609 -0.02202118      0.7909234 -7.768727e-02 -
0.0403804
## [2,]        1.0922330  0.95631068      0.8252427  6.245005e-17
0.3300971
```

Cluster 9:

```
rbind(clusttwits$center[9,],(clusttwits$center[9,]*sigma + mu))
```

```
##        chatter current_events     travel photo_sharing uncategorized
## [1,] 1.486842      0.3134087 -0.1988543      1.082114    0.07240402
## [2,] 9.646009      1.9239437  1.1305164      5.652582    0.88075117
##         tv_film sports_fandom   politics       food      family
## [1,] -0.1363380    -0.1997163 -0.1483498 -0.3031910 -0.05224633
## [2,]  0.8441315     1.1624413  1.3389671  0.8591549  0.80469484
##      home_and_garden     music       news online_gaming shopping
## [1,]       0.1051214 0.1381796 -0.2703628    -0.1643534 1.361241
```

```
## [2,]       0.5981221 0.8215962  0.6375587     0.7671362 3.851643
##      health_nutrition college_uni sports_playing    cooking
eco
## [1,]       -0.2355931  -0.1047478    -0.05748673 -0.2321748
0.2822709
## [2,]        1.5079812   1.2460094     0.58309859  1.2018779
0.7295775
##        computers  business   outdoors     crafts automotive
art
## [1,] -0.03285014 0.3447953 -0.2465205 0.07771806 0.07642405 -
0.1999017
## [2,]  0.61032864 0.6619718  0.4845070 0.57934272 0.93427230
0.3990610
##        religion     beauty  parenting    dating    school
personal_fitness
## [1,] -0.2611732 -0.1838373 -0.1959448 0.3076251 0.0983022       -
0.1940594
## [2,]  0.5953052  0.4610329  0.6244131 1.2591549 0.8845070
0.9953052
##          fashion small_business          spam       adult
## [1,] -0.07002243      0.1743849 -7.768727e-02 -0.02979344
## [2,]  0.86854460      0.4441315  6.071532e-17  0.34929577
```

Cluster 10:

```
rbind(clusttwits$center[10,],(clusttwits$center[10,]*sigma + mu))
```

```
##          chatter current_events      travel photo_sharing
uncategorized
## [1,] -0.04319508      0.1775698 -0.05423302      1.241674
0.4990187
## [2,]  4.24631579      1.7515789  1.46105263      6.088421
1.2800000
##         tv_film sports_fandom   politics       food     family
## [1,] -0.1362904    -0.2057172 -0.1275198 -0.2037098 0.02911547
## [2,]  0.8442105     1.1494737  1.4021053  1.0357895 0.89684211
##      home_and_garden     music        news online_gaming  shopping
## [1,]       0.1419633 0.5525667 -0.07578889   -0.02286982 0.2025719
## [2,]       0.6252632 1.2484211  1.04631579    1.14736842 1.7557895
##      health_nutrition college_uni sports_playing   cooking
eco
## [1,]      -0.06622745 -0.01816877      0.2015461  2.823952 -
0.0009452388
## [2,]       2.26947368  1.49684211      0.8357895 11.684211
0.5115789474
##       computers  business    outdoors     crafts automotive
art
## [1,] 0.05656488 0.2279240 0.007366432 0.08238866 0.01204133
0.0009203335
## [2,] 0.71578947 0.5810526 0.791578947 0.58315789 0.84631579
0.7263157895
```

```
##         religion   beauty   parenting     dating    school
personal_fitness
## [1,] -0.1212898 2.638198 -0.05784476 0.04883143 0.1724649      -
0.04418512
## [2,]  0.8631579 4.208421  0.83368421 0.79789474 0.9726316
1.35578947
##       fashion small_business          spam        adult
## [1,] 2.728426      0.1642956 -7.768727e-02 0.0004888515
## [2,] 5.985263      0.4378947  3.295975e-17 0.4042105263
```