Assignment 1

Carlton Washburn

August 5, 2015

Exploratory Analysis

Loading of the dataset and libraries.

```
vote.data = read.csv("georgia2000.csv")
library(mosaic)
```

```
## 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
```

```
library(ggplot2)
```

For the exploratory analysis, I looked to see if there was a connection between the number of votes that were not counted, and the voting conditions and demographics of the voting county. The first thing done was to add several columns to the dataset to help with the exploratory analysis. I added a column which shows the number of ballots that went uncounted as disparity.

```
vote.data['disparity'] = vote.data$ballots - vote.data$votes
```

I also added columns which recorded poor and atlanta as string values.

```
vote.data$ispoor = 'p'
for (i in c(1:159)) {
  if (vote.data$poor[i] == 1) {
    vote.data$ispoor[i] = 'poor'
  }
  else {
    vote.data$ispoor[i] = 'not poor'
  }
}
vote.data$isatlanta = 'n'
for (i in c(1:159)){
  if (vote.data$atlanta[i] == 1) {
    vote.data$isatlanta[i] = 'atlanta'
  }
  else {
    vote.data$isatlanta[i] = 'not atlanta'
  }
}
```

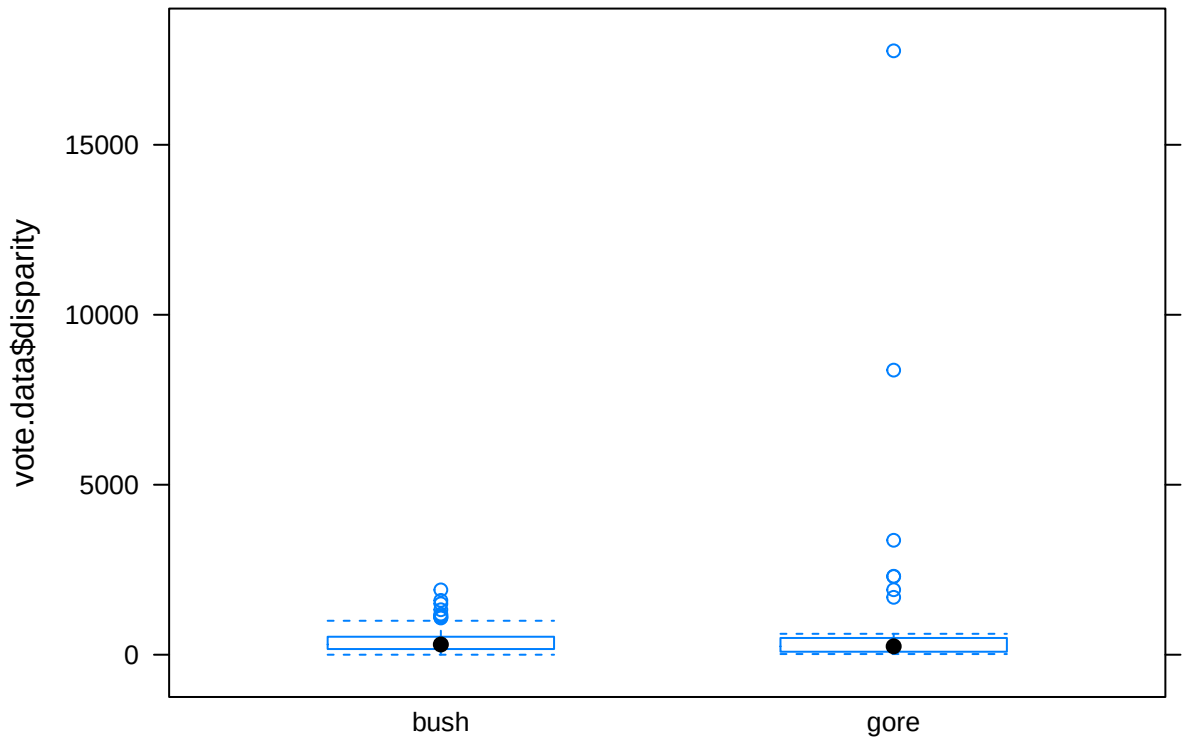
Finally, I added a column which recorded whether the majority of voters in the county voted for Gore or Bush in a column labeled candidate.

```
vote.data$candidate = 'p'
for (i in c(1:nrow(vote.data))){
  if (vote.data$gore[i] > vote.data$bush[i]) {
    vote.data$candidate[i] = 'gore'
  }
  else if (vote.data$bush[i] > vote.data$gore[i]) {
    vote.data$candidate[i] = 'bush'
  }
}
```

After adding the columns to the dataset, I plotted several boxplots versus the number of uncounted ballots to see if any of the categorical variables displayed a higher number of uncounted ballots for any category.

First is whether the majority of voters voted for Bush or Gore and the number of uncounted votes.

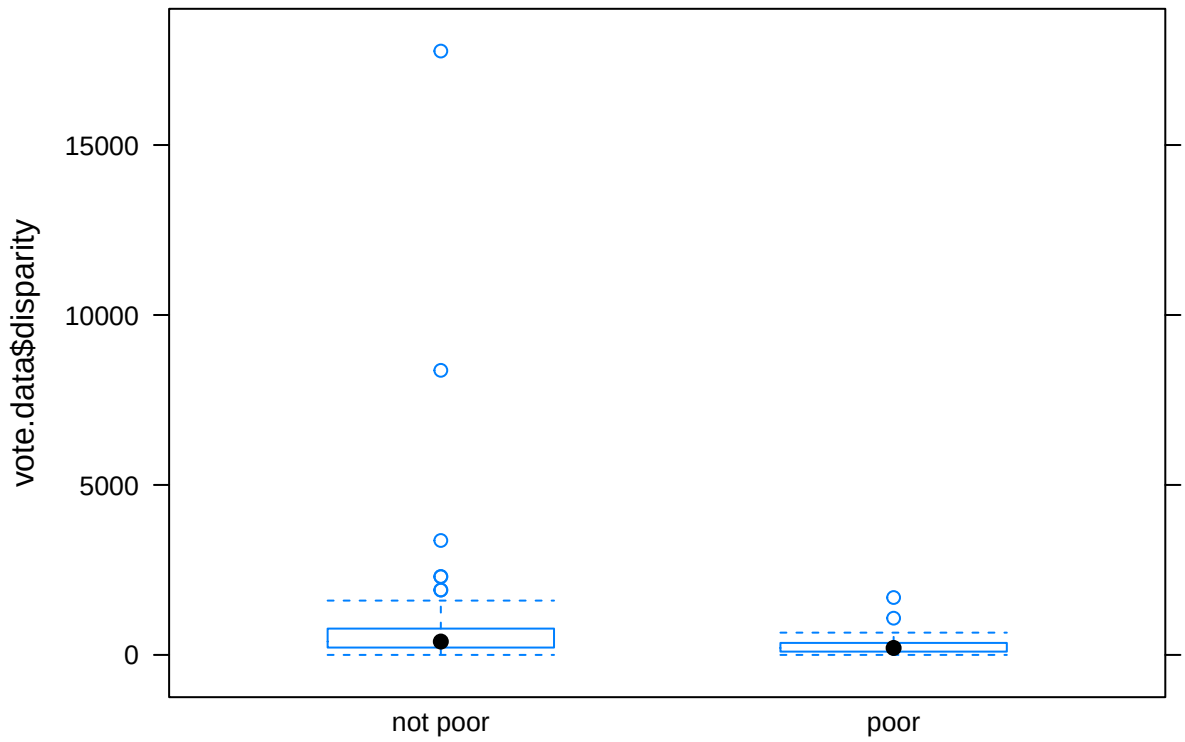
```
bwplot(vote.data$disparity ~ vote.data$candidate)
```



In this we see that while both counties that voted Bush and Gore had counties with 1000 plus uncounted votes but the largest disparity occured in counties which voted for Gore.

In this boxplot we looked at number of uncounted votes and whether the county was considered poor or not.

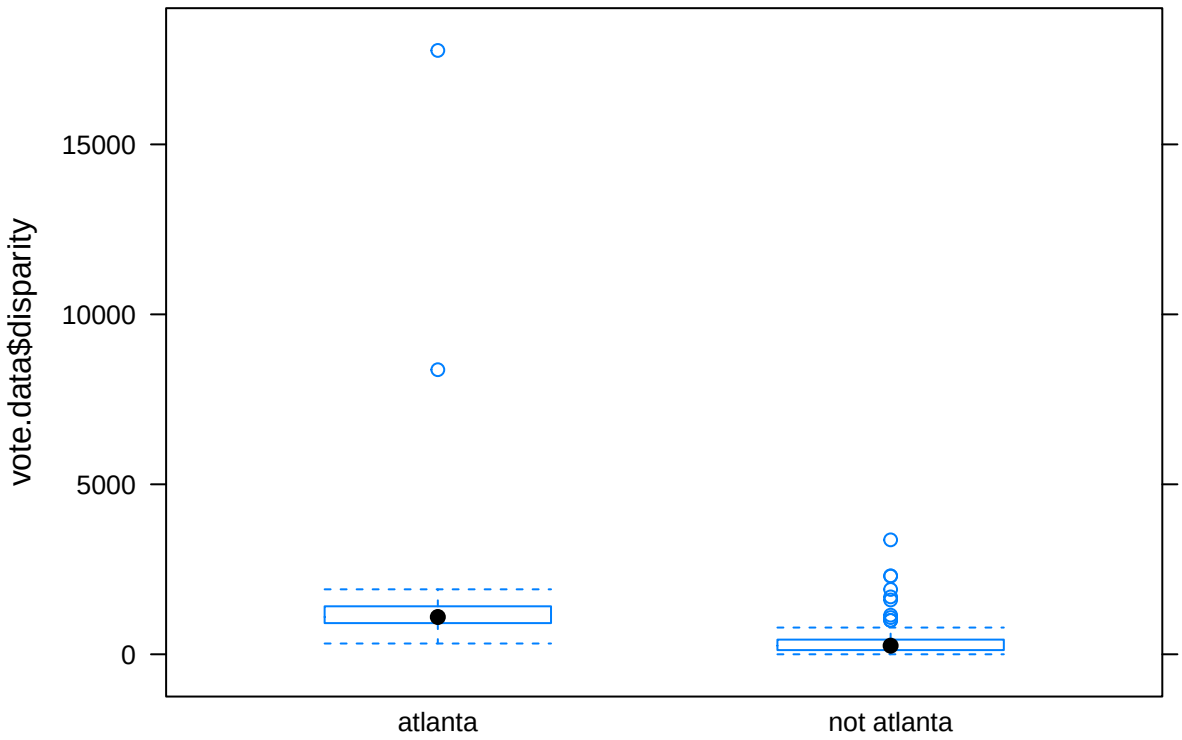
```
bwplot(vote.data$disparity ~ vote.data$ispoor)
```



We see that the counties with highest number of uncounted votes appear to bve from counties which are marked as not poor.

In this next one we examine whether the county is in Atlanta or not and number of uncounted votes.

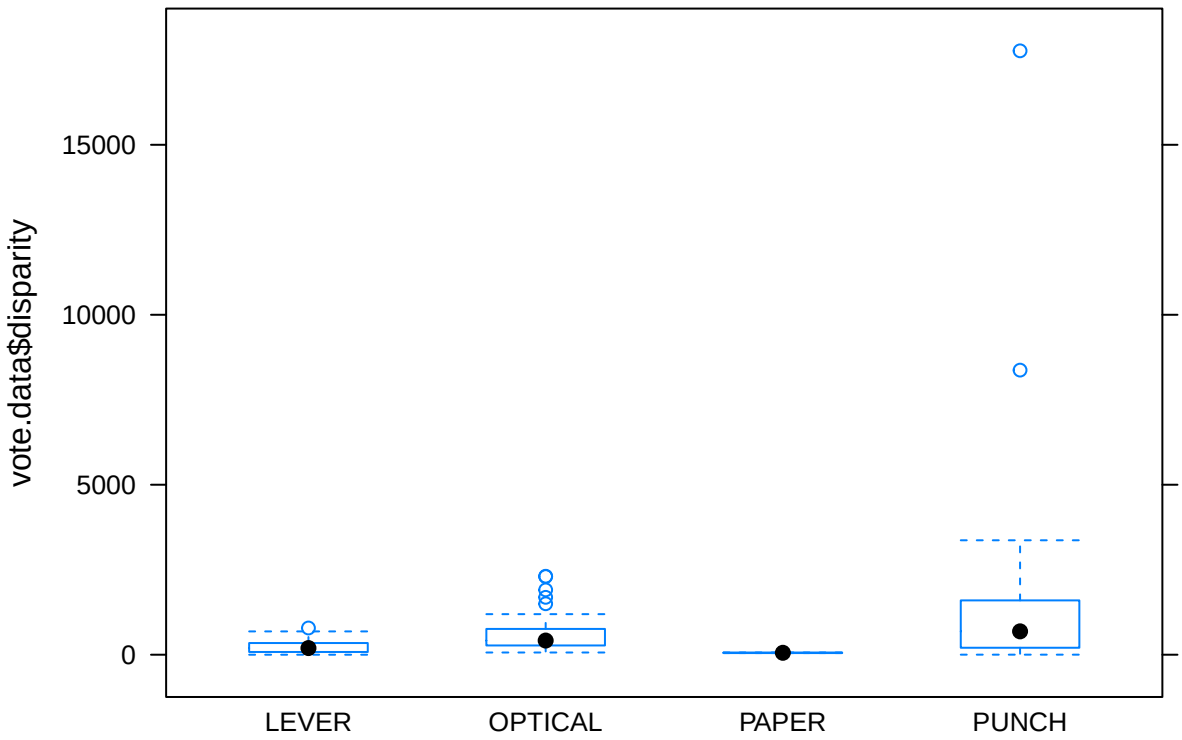
```
bwplot(vote.data$disparity ~ vote.data$isatlanta)
```



We see the counties with the highest number of uncounted votes were in Atlanta, and that counties not in Atlanta generally have a lower number of uncounted votes, but also has several counties with a thousand plus uncounted votes.

Finally we look at equipment versus the number of uncounted votes.

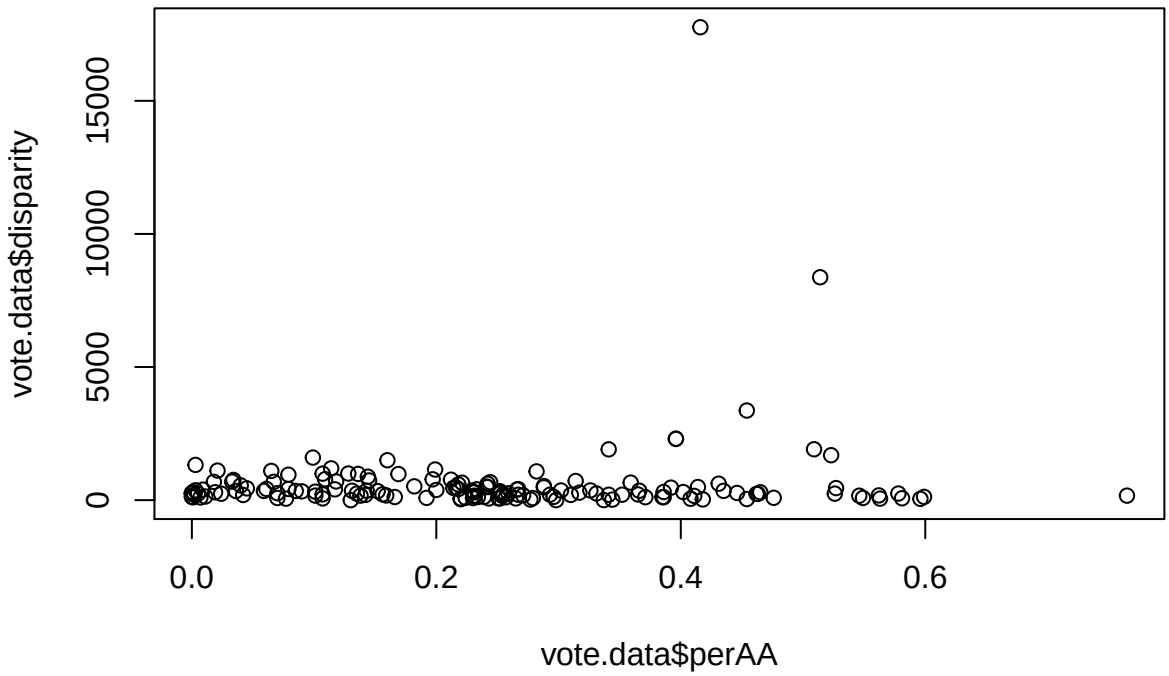
```
bwplot(vote.data$disparity ~ vote.data$equip)
```



From this we see that counties that used Punch for equipment saw the some of the highest number of uncounted votes and that optical equipment also had several counties that had large numbers of uncounted votes.

So, know we look at percent African American versus the number of uncounted votes.

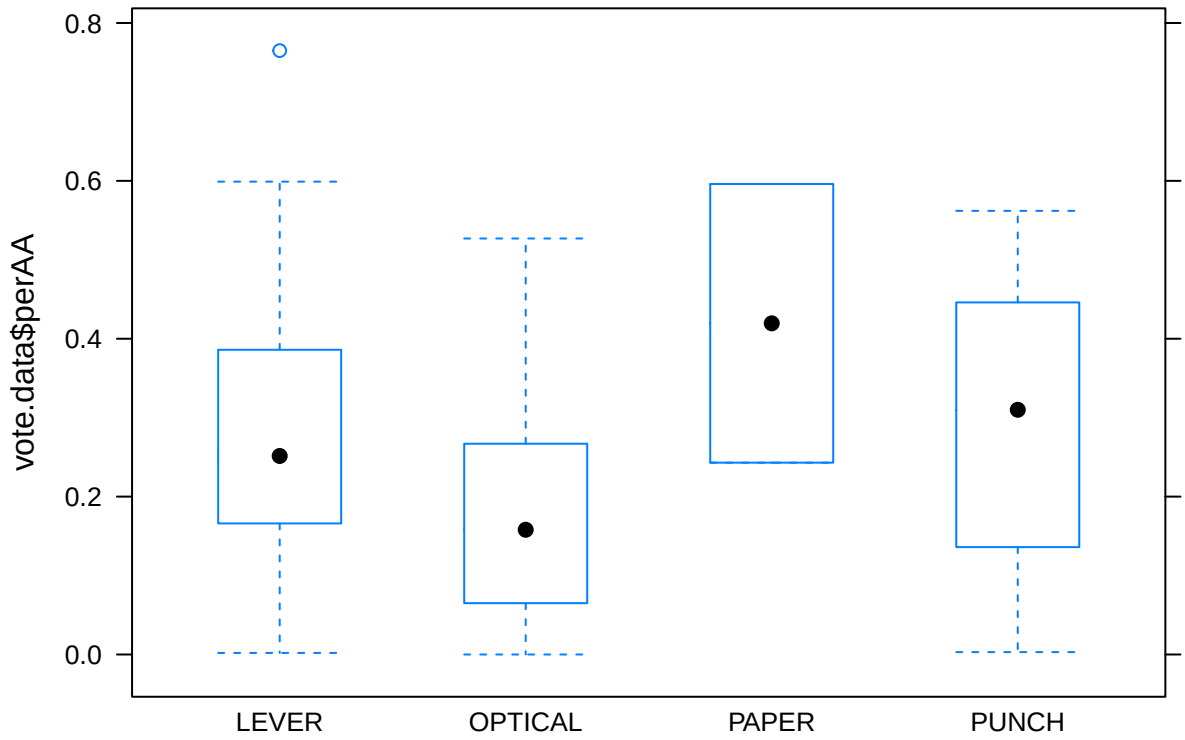
```
plot(vote.data$disparity ~ vote.data$perAA)
```



Here we see that the counties with the highest number of uncounted votes, came from counties with over 35 percent African American Populations.

So to check to see whether counties with high percentage of African American voters are using equipment that results in a high number of votes not being counted we examine percent African American versus the Equipment.

```
bwplot(vote.data$perAA ~ vote.data$equip)
```



Here the box plot suggests that punch equipment is found in counties with high percent of African Americans, which according to our earlier box plot results in a higher number of uncounted votes.

Bootstrapping

Given a $100000 to invest in some combination of S&P500, US Treasury Bonds, Investment Grade Corporate Bonds, Emerging Market Equities, and Real Estate what are the proportions that would result in a safer and riskier portfolio then just an even split between the five choices. To assess which of the options were risky and which of the options were safer we looked at the mean and staandard deviation of the percentage change in value of the Exchange Traded Funds.

Five years of daily data was downloaded with the following code:

```
library(mosaic)
library(fImport)
```

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

```
library(foreach)

# Import a few stocks
```

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

The following function provided by Dr. Scott was used to calculate the daily returns for the different exchange traded funds:

```
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))
}
```

The daily reutrns, mean of the daily return, and the standard deviation of the daily returns were calculated with the following code.

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

```
##  SPY.PctReturn        TLT.PctReturn        LQD.PctReturn
##  Min.   :-0.0651232   Min.   :-0.0504495   Min.   :-0.0205232
##  1st Qu.:-0.0036944   1st Qu.:-0.0057748   1st Qu.:-0.0018335
##  Median : 0.0007576   Median : 0.0007709   Median : 0.0004937
##  Mean   : 0.0006403   Mean   : 0.0003233   Mean   : 0.0001978
##  3rd Qu.: 0.0053486   3rd Qu.: 0.0065048   3rd Qu.: 0.0022556
##  Max.   : 0.0464992   Max.   : 0.0396555   Max.   : 0.0146677
##  EEM.PctReturn        VNQ.PctReturn
##  Min.   :-8.337e-02   Min.   :-0.0868671
##  1st Qu.:-7.740e-03   1st Qu.:-0.0050551
##  Median : 4.649e-04   Median : 0.0009249
##  Mean   : 7.693e-05   Mean   : 0.0005595
##  3rd Qu.: 7.747e-03   3rd Qu.: 0.0066850
##  Max.   : 6.240e-02   Max.   : 0.0910393
```

```
apply(myreturns,2,sd)
```

```
## SPY.PctReturn TLT.PctReturn LQD.PctReturn EEM.PctReturn VNQ.PctReturn
##   0.009371236   0.009774380   0.003579140   0.013743189   0.011556416
```

As we can see the mean daily percent return for the ETFs range from .007 percent to .06 percent and the standard deviations range .35 percent to 1.37 percent. From this, it was decided the safest ETFs were Investment Grade Corporate Bonds, S&P500, and Treasury Bonds in that order. The riskiest assets were Emerging Market Equities and Real Estate with Emerging Market Equities being the riskiest. So for our safe portfolio we would reccommend 60 percent in Investment Grade Corporate Bonds, 30 percent in S&P500, and 10 percent in US Treasury Bonds. For the risky portfolio we recommend 60 percent in Emerging Market Equities and 40 percent in Real Estate.

In the following code we simulate a many different possible 20 day trading periods to assess what 20 day value at risk at the 5 percent level is:

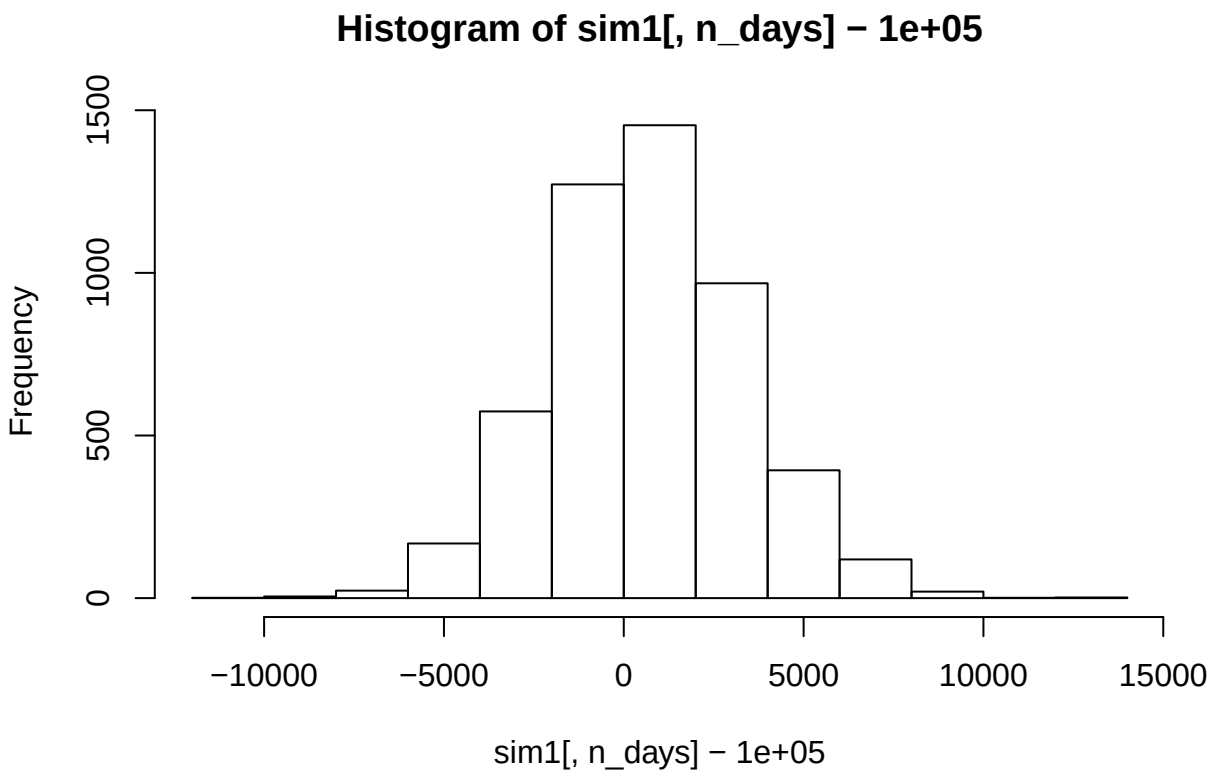
```
n_days = 20
set.seed(13)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
  }
  wealthtracker
}
quantile(sim1[,n_days], 0.05) - 100000
```

```
##        5%
## -3669.575
```

We see for an evenly split portfolio, there is a five percent chance that the portfolio will lose $3669.58 during the twenty day trading period.

We can see in the following histogram the potential changes in value for the portfolio over a twenty day period:

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



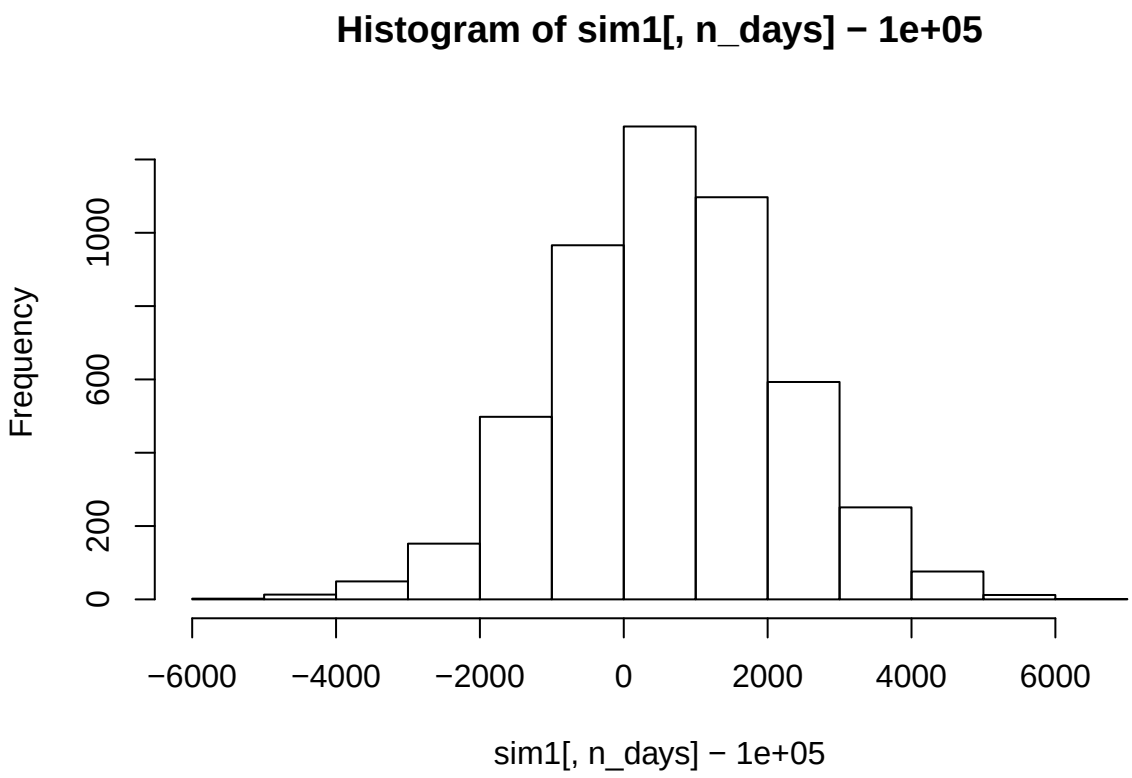
We now simulate the trading of the safe portfolio and estimate the value at risk:

```
set.seed(13)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.3, 0.1, 0.60, 0.0, 0.0)
  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
  }
  wealthtracker
}
quantile(sim1[,n_days], 0.05) - 10000
```

```
##       5%
## 88096.97
```

We see for the safe portfolio the value at risk is around $1903.03. In the following histogram we see the spread of potential gains or losses.

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



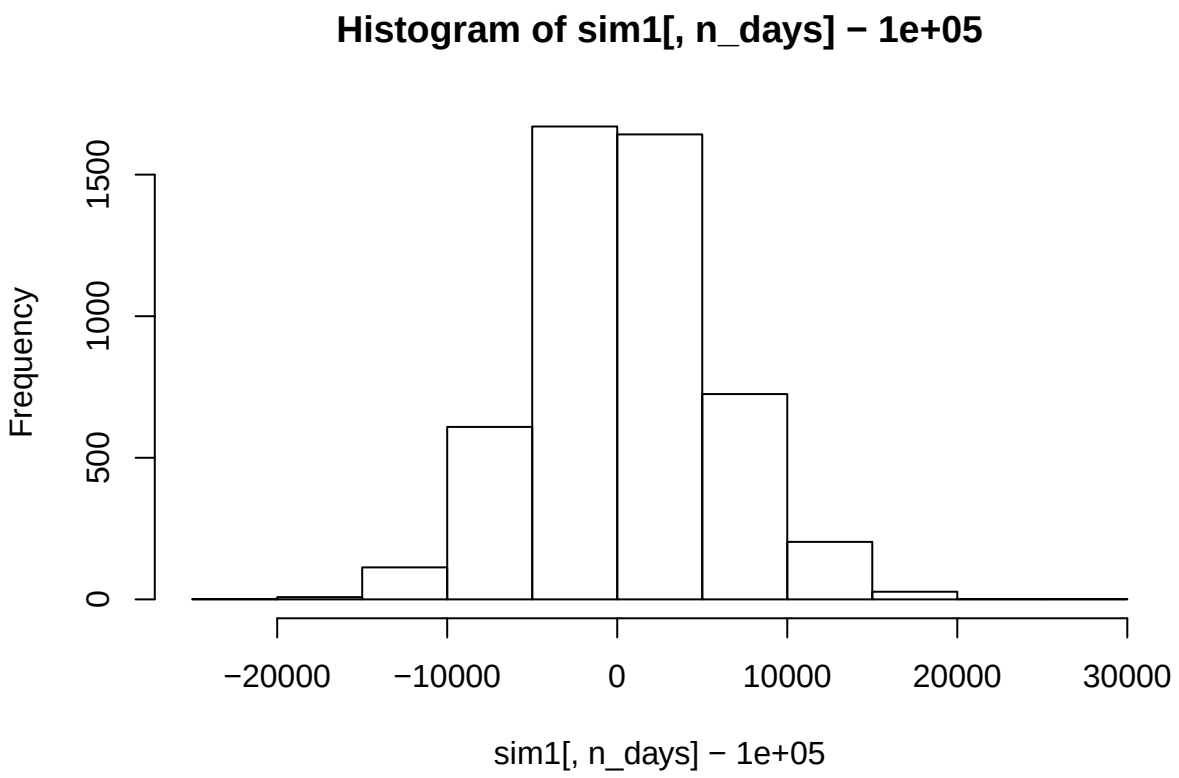
Finally we look at the risky portfolio and estimate its value at risk:

```
set.seed(13)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.0, 0.0, 0.0, 0.6, 0.4)
  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
  }
  wealthtracker
}
quantile(sim1[,n_days], 0.05) - 100000
```

```
##        5%
## -8124.578
```

We see the value at risk for the risky portfolio is $8124.58. We now look at the spread of possible gains and losses:

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



From the spreads we see that while the value at risk is lower for a balanced portfolio and a safe portfolio, the potential gains are also less while the risky portfolio has a greater value at risk, but also has a larger potential for gains.

Clustering and PCA

In this section we examine a dataset of wine and see if clustering or principle component analysis can predict whether the wine is red or white.

PCA

First we do a principle component analysis to see if it can predict the color of the wine.

The following loads in the wine date and libraries need to plot the results:

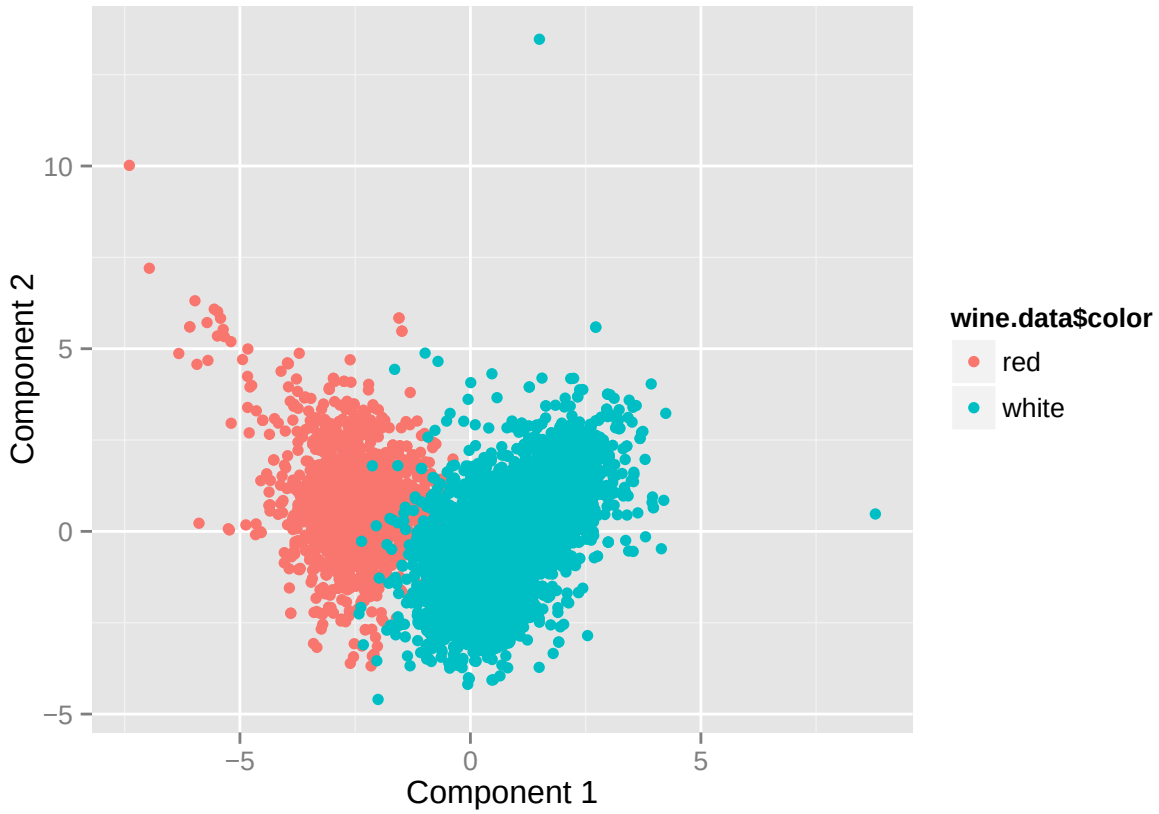
```
library(ggplot2)
wine.data = read.csv('wine.csv')
```

We then compute the priciple components using the 11 variables of the wine data.

```
wine.data.numeric = wine.data[c(1:11)]
pr.out = prcomp(wine.data.numeric, scale= TRUE,center = TRUE)
scores = pr.out$x
```

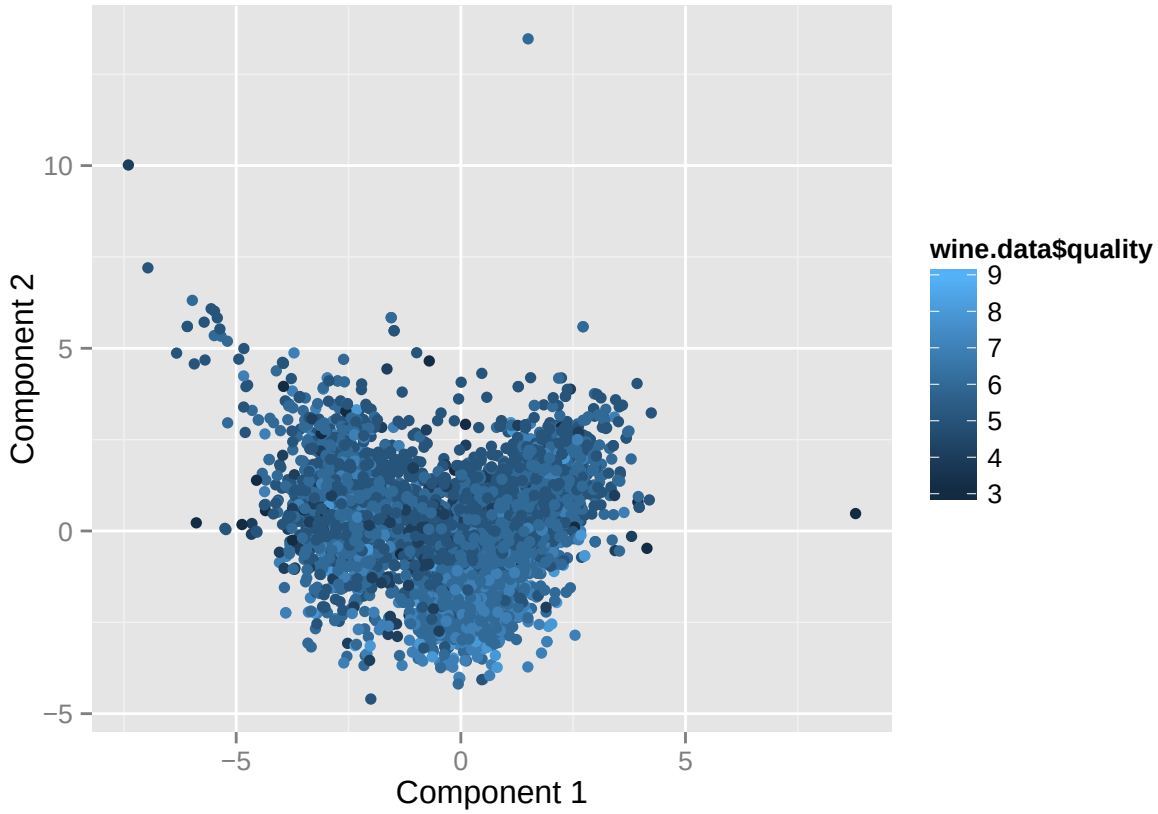
We then plot the data with principle component 1 as the x axis and principle component 2 as y axis, and color coded the points as either red wine or white wine.

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



From the plot we see that principle component 1 is capable of predicitng whether the wine is red or white. Now we check to see if it can show quality with the following plot.

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



This plot indicates that the priciple components do not appear to predict the quality of the wine.

Clustering

Here we use kmeans to see if clustering can predict the color of the wine. First we scale the data to prepare it for use in the kmeans alggorithm.

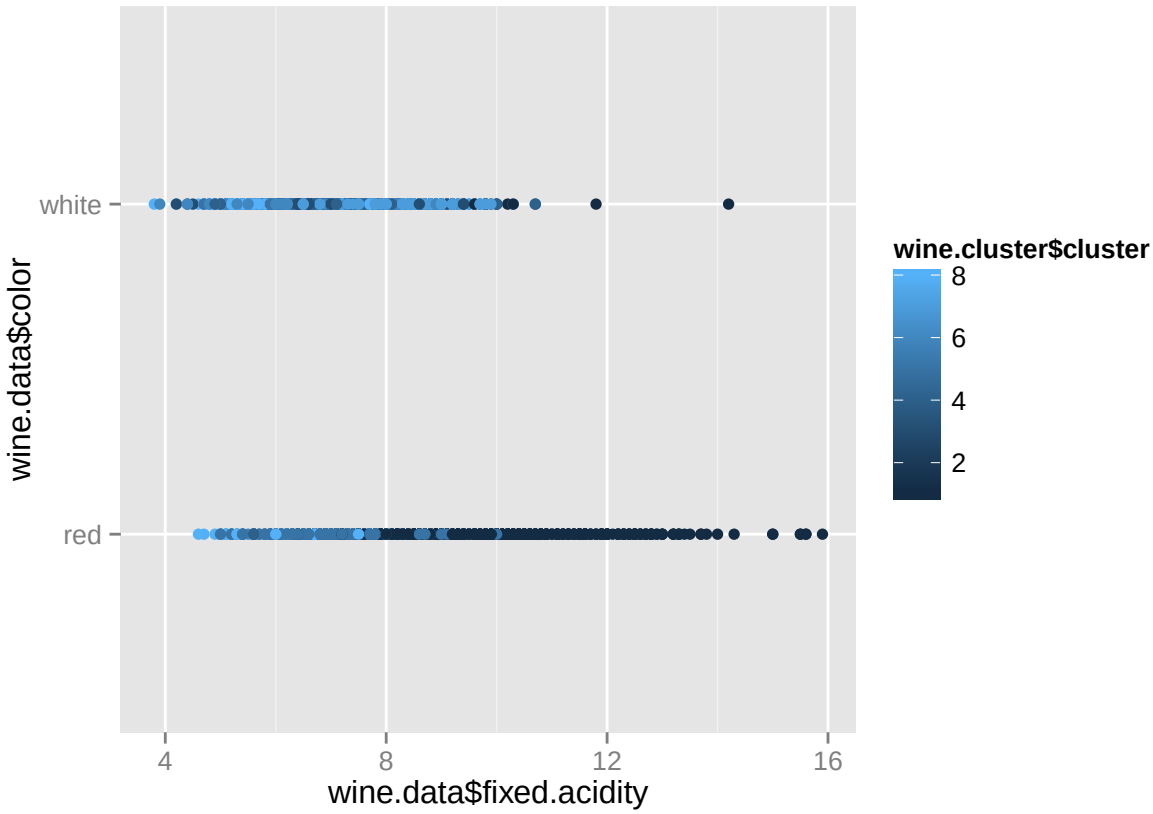
```
wine.data.numeric.scaled = scale(wine.data.numeric,center = TRUE, scale = TRUE)
```

We then run the data through the kmeans algorithm generating two clusters, hopefully one for red wine and the other for white wine.

```
wine.cluster = kmeans(wine.data.numeric.scaled,centers = 8, nstart = 50)
```

We then plot the data points based on the acidity and wine color, with the points color coded based on which cluster they were assigned.

```
qplot(wine.data$fixed.acidity,wine.data$color,col=wine.cluster$cluster)
```



As we can see the clusters for the most part group white wines together and red wines together indicating that the clustering does predict the color of the wine.

Now we attempt to see if clustering can predict the quality of the wine with 9 clusters, one for each of the quality rating represented in the data.

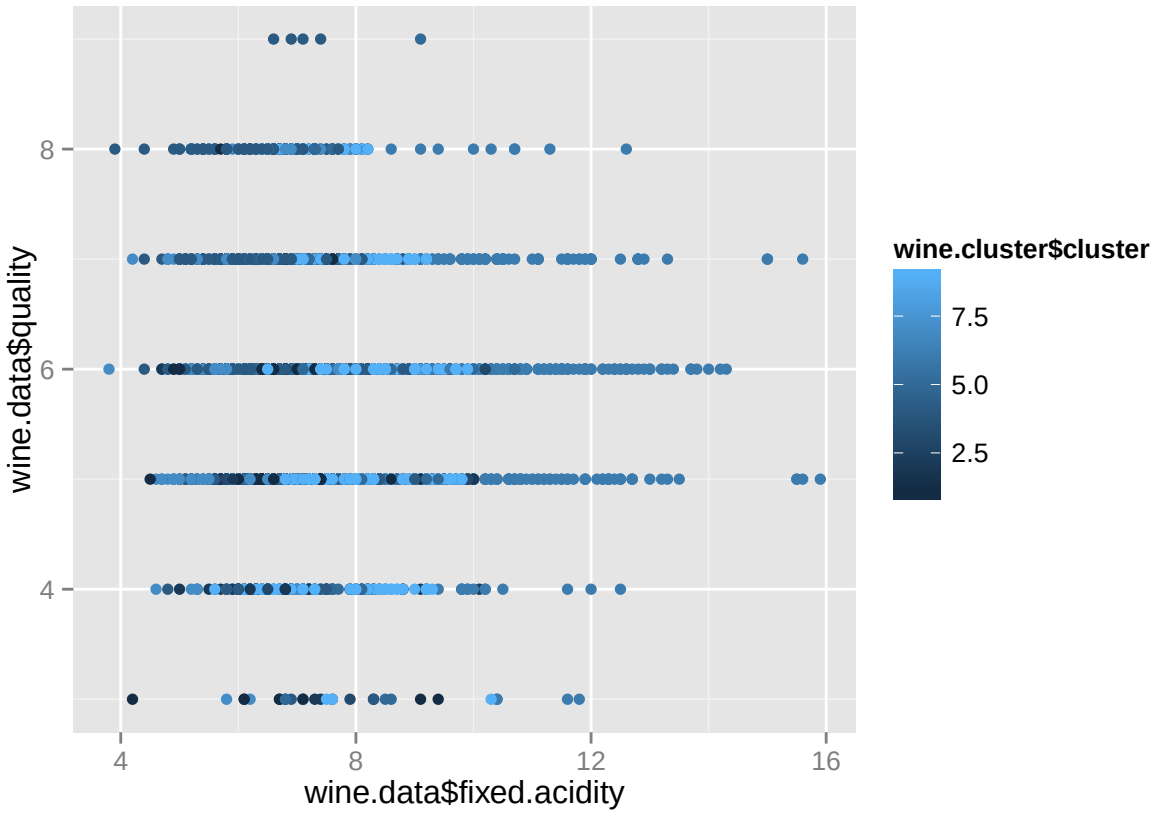
```
wine.cluster = kmeans(wine.data.numeric.scaled,centers = 9, nstart = 50)
```

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

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

Next we plot the data to see if the clusters predict the quality.

```
qplot(wine.data$fixed.acidity,wine.data$quality,col=wine.cluster$cluster)
```



From this plot we see that that the clusters do not seem to predict the quality of wine, with each cluster containing wines of many different qualities.

Market Segmentation

To examine the data and see if there are any associations between the different topics that people tweet about which indicates that people interested in certain topics are also interested in other topics we used clustering with the kmeans algorithm.

First we loaded the data into R

```
twitter.data = read.csv('social_marketing.csv')
```

We then scaled the data and excluded the chatter variable as it was felt that its inclusion in the data was not very informative.

```
twitter.scaled = scale(twitter.data[,c(3:37)], center = TRUE, scale = TRUE)
```

We then used the kmeans algorithm to generate are different groups, in this case we felt 8 clusters was a good amount.

```
social.cluster = kmeans(twitter.scaled,center = 8, nstart = 50)
```

We then examined the centers of the clusters to get an idea of what the different generated groups appear to be tweeting about.

```
social.cluster$centers
```

```
##   current_events      travel photo_sharing uncategorized     tv_film
## 1     0.19304688 -0.04849382    1.25320747    0.48174855 -0.13691559
## 2     0.09810774 -0.11628281   -0.01984714   -0.08298029 -0.10816798
## 3    -0.09205812 -0.24063720   -0.15058972   -0.14362467 -0.23556257
## 4     0.10783519  1.75482673   -0.05662221   -0.10290013 -0.04514207
## 5    -0.06864969 -0.03487717    0.02965404   -0.04893487  0.09797392
## 6     0.27684711  0.28872703   -0.09071828    0.11259854 -0.11619101
## 7     0.01053256 -0.15906974   -0.01614375    0.14378408 -0.15690019
## 8     0.28035331  0.20870271    0.01435929    0.71059034  2.53662621
##   sports_fandom    politics         food      family home_and_garden
## 1    -0.1997531 -0.11044053 -0.186458235  0.04852022      0.14223221
## 2     2.0136901 -0.20436863  1.797672670  1.45125533      0.16359705
## 3    -0.2968670 -0.26366100 -0.378683362 -0.25551695     -0.13633305
## 4     0.2148075  2.39124043  0.002967272  0.04988734      0.10185008
## 5    -0.1377727 -0.16068188 -0.082320742  0.19494657      0.06888712
## 6     0.1406567  0.15052740  0.040494220 -0.05999555      0.23510191
## 7    -0.2009970 -0.17520181  0.405634708 -0.08150131      0.14437472
## 8    -0.1055161 -0.09786249  0.082516093 -0.11643253      0.28685011
##          music          news online_gaming    shopping health_nutrition
## 1  0.519170150 -0.0670969513   -0.03514059  0.32534092      -0.07035347
## 2  0.027941242 -0.0887441734   -0.07669002  0.06304708      -0.16409036
## 3 -0.192093179 -0.2581436196   -0.22658015 -0.06992049      -0.32568785
## 4 -0.078148720  1.9928335233   -0.14380538 -0.01545698      -0.21335584
## 5 -0.043056263 -0.1897643972    3.49069377 -0.07631685      -0.16558422
## 6  0.014182641 -0.0006901999    0.08935906 -0.23782264       0.05086059
## 7  0.003606222 -0.0495687225   -0.13061432  0.04183153       2.11585257
## 8  1.124965214 -0.0188175762   -0.19603242  0.16730658      -0.19955807
##   college_uni sports_playing     cooking         eco   computers
## 1 -0.02408399     0.18074549  2.63684192  0.04393640  0.07540234
## 2 -0.13692163     0.09717116 -0.11562494  0.20113018  0.08238941
## 3 -0.25746982    -0.23131269 -0.33047452 -0.18192863 -0.23265326
## 4 -0.10138814    -0.02728855 -0.20977335  0.10835765  1.55697386
## 5  3.21817021     2.08650946 -0.12828873 -0.04004683 -0.08601849
## 6  0.12732753    -0.11129036 -0.05898219  0.44799948  0.29753290
## 7 -0.22707503    -0.03872744  0.37430019  0.52760202 -0.07401928
## 8  0.45027637     0.12590466 -0.19890250  0.11392834 -0.13317513
##      business    outdoors      crafts  automotive         art
## 1  0.27074519  0.03934651  0.10707275  0.05430844  0.01715262
## 2  0.10593408 -0.08087144  0.67787211  0.16547878 -0.01949817
## 3 -0.16403769 -0.31928637 -0.23598021 -0.17745804 -0.24136449
## 4  0.33093268  0.09738876  0.08088491  1.15581412 -0.15440550
## 5 -0.08018436 -0.14184430  0.06155206  0.06644171  0.26789775
## 6 -0.34600901  0.29780310  0.21793373  0.12453564  0.33167537
## 7  0.07214350  1.62032454  0.04193883 -0.12116513 -0.07931203
## 8  0.45749103 -0.09150139  0.61228205 -0.18582538  2.12520611
##        religion      beauty   parenting       dating      school
## 1 -0.1249817268  2.47568494 -0.07106346  0.125993521  0.19549729
## 2  2.1958765469  0.29909438  2.09195736  0.061187839  1.64326847
## 3 -0.3106321455 -0.27434377 -0.30503465 -0.097512407 -0.25950212
## 4 -0.0458266857 -0.17813393  0.02967587  0.202606632 -0.02617127
## 5 -0.1921288718 -0.23617435 -0.15933231  0.004890480 -0.19634354
```

```
## 6  0.1207017943 -0.10070195  0.18658414 -0.009528244  0.09244824
## 7 -0.1752116828 -0.21856461 -0.10268560  0.188380753 -0.15368446
## 8  0.0003095539 -0.03852012 -0.20677599 -0.003848963 -0.01994393
##   personal_fitness     fashion small_business        spam        adult
## 1       -0.0594024  2.56998775     0.21999429 -0.07768727 -0.001629525
## 2       -0.1178042  0.01455792     0.07827327 -0.07768727 -0.022295789
## 3       -0.3340178 -0.27178457    -0.15916802 -0.07768727 -0.016726832
## 4       -0.1914708 -0.18490098     0.19168251 -0.07768727 -0.107561211
## 5       -0.1734094 -0.06613010     0.11925477 -0.07768727 -0.022548548
## 6        0.1218324 -0.02044987     0.31428826 12.41886450  3.750222155
## 7        2.0751800 -0.11222806    -0.10290222 -0.07768727  0.010892572
## 8       -0.1696566 -0.05500645     0.77920654 -0.07768727 -0.054771424
```

From this we see the following:

1. Group One appears to be interested in Film and Movies, Music, and Art.
2. Group Two appears to be primarly be spammers and posters of adult orientated subject material
3. Group Three likes to share photos and is interested in cooking, fashion, and bueaty.
4. Group Four is a group of people that show a general disinterest in most topics and may not be active users of Twitter.
5. Group Five is interested in Health and Nutrtion, Personal Fitness, and the Outdoors.
6. Group Six displays interests in travel, news, politics, computers, and cars.
7. Group Seven displays interests in online gaming, college and university, and playing sports.
8. Group Eight displays interests in sports fandom, food, family, religion, parenting, and school.

From the interests, we can make educated guesses about the members of these groups. For example, we could guess that group seven is made up of college age tweeters.