Homework 1

Michael Zhang

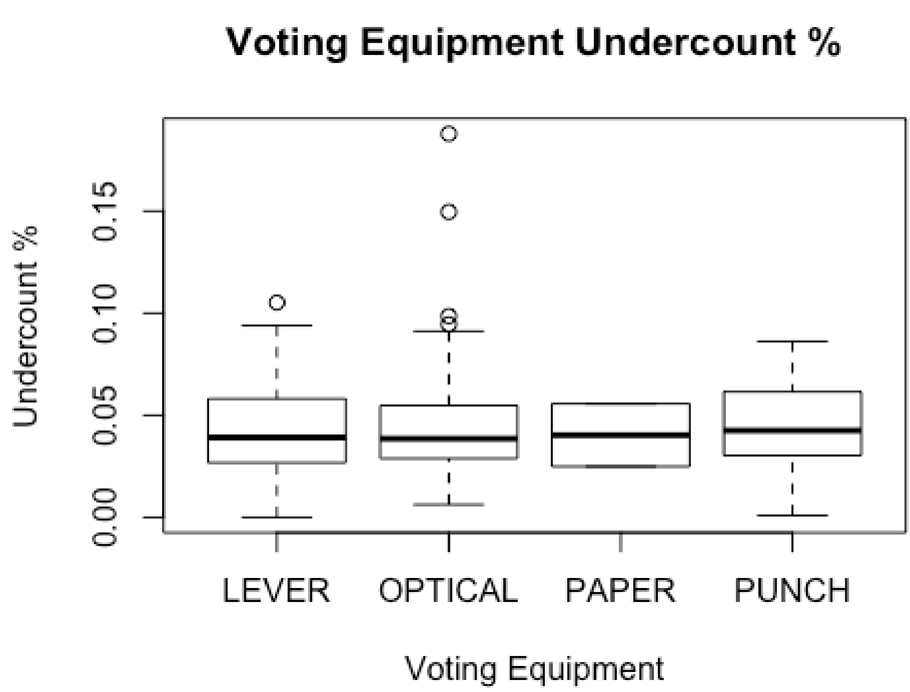
Exploratory Analysis

Let's take a look at Georgia's county-level voting data from the 2000 presidential election and investigate vote undercount.

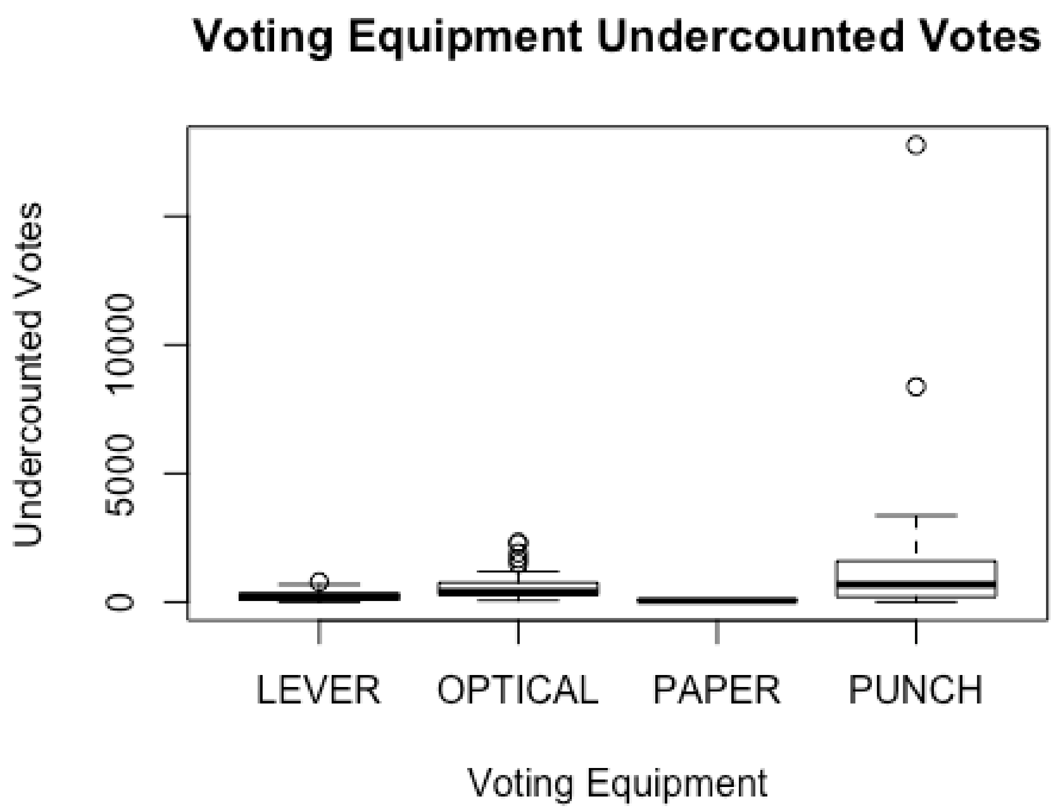
Specifically, let's examine the below issues:

1) Whether voting on certain kinds of voting equipment lead to higher rates of undercount
2) If so, whether we should worry that this effect has a disparate impact on poor and minority communities

```
georgia <- read.csv("../data/georgia2000.csv", row.names=1)

georgia$undercountpct <- 1 - georgia$votes/georgia$ballots
georgia$undercount <- georgia$ballots - georgia$votes

boxplot(undercountpct~equip, data = georgia, main = "Voting Equipment Undercount %", xlab = "Voting Equipment", ylab = "Undercount %")
```



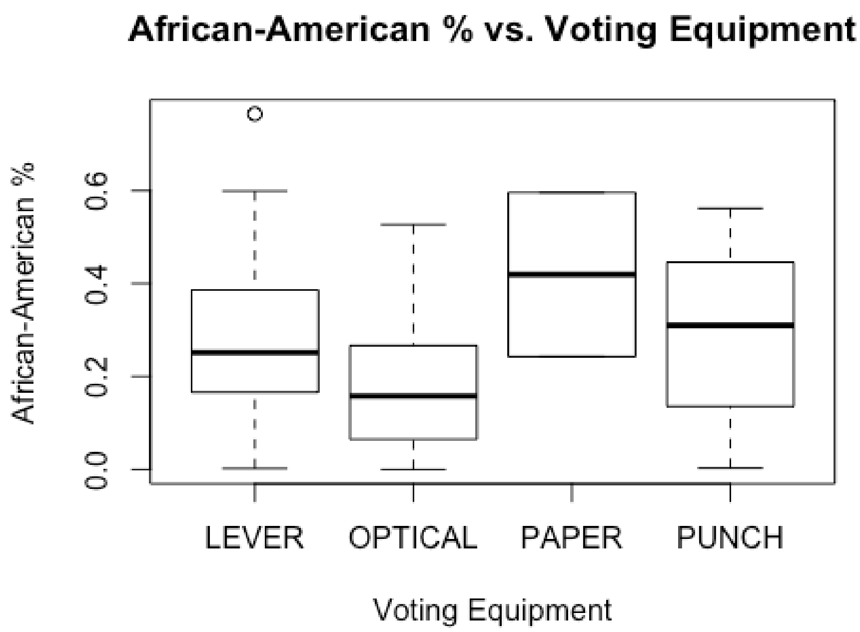
```r
boxplot(undercount~equip, data = georgia, main = "Voting Equipment
Undercounted Votes", xlab = "Voting Equipment", ylab = "Undercounted Votes")
```



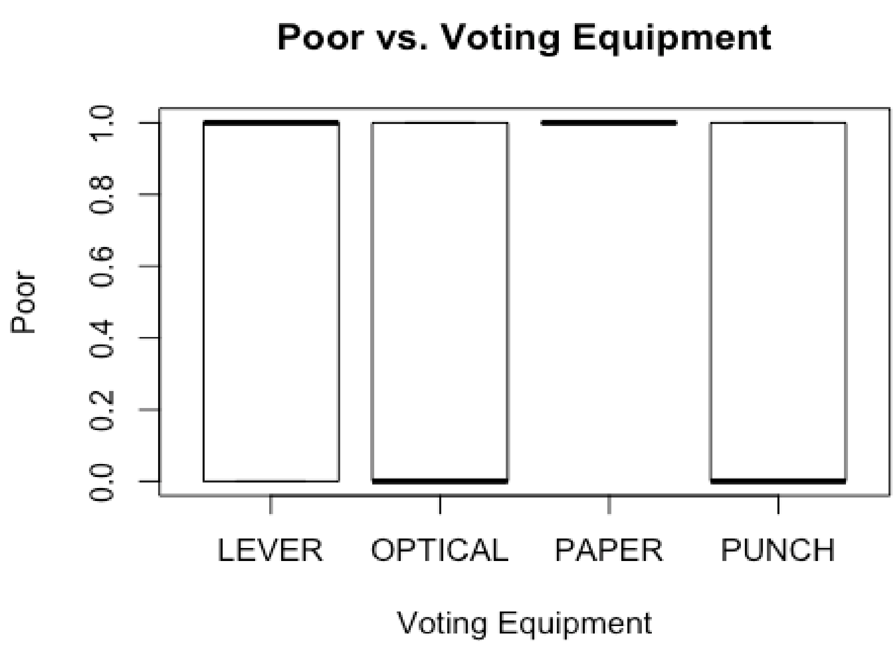
Looking at these boxplots, we see that the undercount % is fairly consistent across voting equipments. However, when we look at undercounted votes as a number, there are two counties with a large number of undercounted votes, and both of these counties used "Punch" equipment.

```r
aggregate(georgia[, 4], list(georgia$equip), mean)

##   Group.1         x
## 1   LEVER 0.6081081
## 2 OPTICAL 0.2727273
## 3   PAPER 1.0000000
## 4   PUNCH 0.4117647

aggregate(georgia[, 7], list(georgia$equip), mean)

##   Group.1         x
## 1   LEVER 0.2762432
## 2 OPTICAL 0.1860455
## 3   PAPER 0.4195000
## 4   PUNCH 0.2984706
```

```
boxplot(perAA~equip, data = georgia, main = "African-American % vs. Voting
Equipment", xlab = "Voting Equipment", ylab = "African-American %")
```




```
boxplot(poor~equip, data = georgia, main = "Poor vs. Voting Equipment", xlab
= "Voting Equipment", ylab = "Poor")
```



When we look at the boxplots and means for African-American % and Poor vs. Voting Equipment, we do not see a disproportionate representation of these two variables in counties using "Punch" equipment.

Therefore, while there may be slight variations in vote undercounting among voting equipment, we should not worry that this effect has a disparate impact on poor and minority communities.

Bootstrapping

Considering the below five asset classes:

- US domestic equities (SPY: the S&P 500 stock index)
- US Treasury bonds (TLT)
- Investment-grade corporate bonds (LQD)
- Emerging-market equities (EEM)
- Real estate (VNQ)

Let's take a look at the risk/return properties. To estimate variability, we use CAPM and standard deviations.

```
library(mosaic)

library(fImport)

library(foreach)
etfs = c("SPY", "TLT", "LQD", "EEM", "VNQ")
etfprices = yahooSeries(etfs, from='2010-01-01', to='2015-07-30')

# A helper function for calculating percent 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
etfreturns = YahooPricesToReturns(etfprices)

# Standard deviations of the asset classes
sigma_SPY = sd(etfreturns[,1])
sigma_TLT = sd(etfreturns[,2])
```

```r
sigma_LQD = sd(etfreturns[,3])
sigma_EEM = sd(etfreturns[,4])
sigma_VNQ = sd(etfreturns[,5])

# First fit the market model to each asset class
lm_TLT = lm(etfreturns[,2] ~ etfreturns[,1])
lm_LQD = lm(etfreturns[,3] ~ etfreturns[,1])
lm_EEM = lm(etfreturns[,4] ~ etfreturns[,1])
lm_VNQ = lm(etfreturns[,5] ~ etfreturns[,1])

# The estimated beta for each asset class based on daily returns
coef(lm_TLT); coef(lm_LQD); coef(lm_EEM); coef(lm_VNQ)

##     (Intercept) etfreturns[, 1]
##    0.0007008087   -0.5476286621

##     (Intercept) etfreturns[, 1]
##    0.0002551472   -0.0381982702

##     (Intercept) etfreturns[, 1]
##   -0.0006397898    1.2434317239

##     (Intercept) etfreturns[, 1]
##    4.319518e-05    1.029497e+00

# The standard deviations for each asset class based on daily returns
sigma_SPY; sigma_TLT; sigma_LQD; sigma_EEM; sigma_VNQ

## [1] 0.00977304

## [1] 0.009694163

## [1] 0.003548176

## [1] 0.01427438

## [1] 0.01263622
```

From the CAPM, we see that both treasury bonds (TLT) and investment-grade corporate bonds (LQD) have slightly negative betas, meaning that they tend to go down when the market goes up, and vice versa. This means that the expected return on these investments is less than the riskfree rate. We also see that emerging-market equities (EEM) is more volatile than the total market. Finally, real estate (VNQ) follows the market closely and can be considered a representative stock.

Looking at the standard deviations, we rank the asset classes in order of least volatile to most volatile:

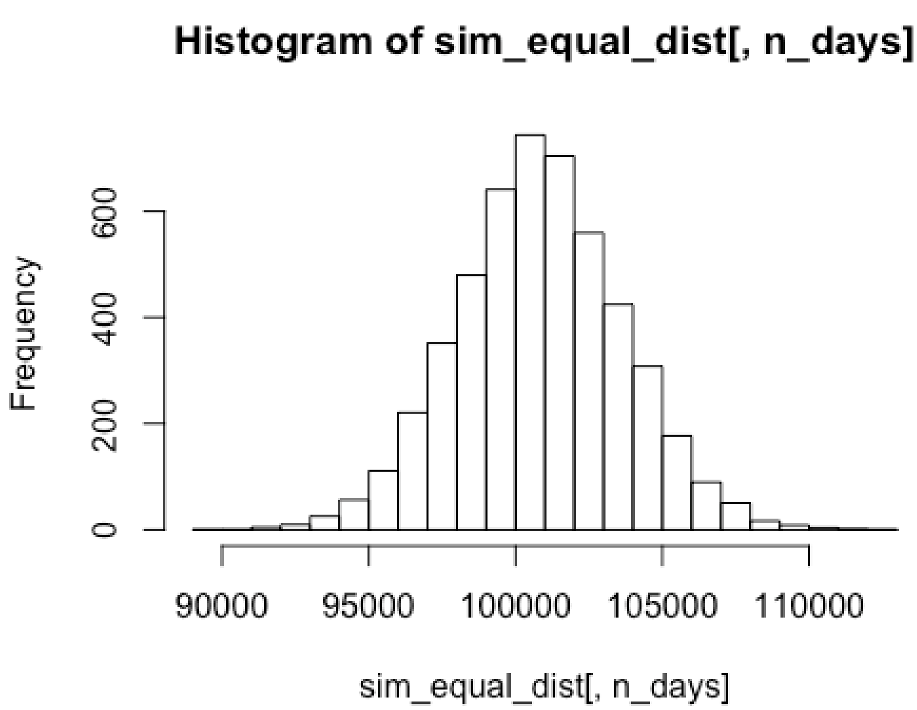
1) LQD
2) TLT
3) SPY
4) VNQ

5) EEM

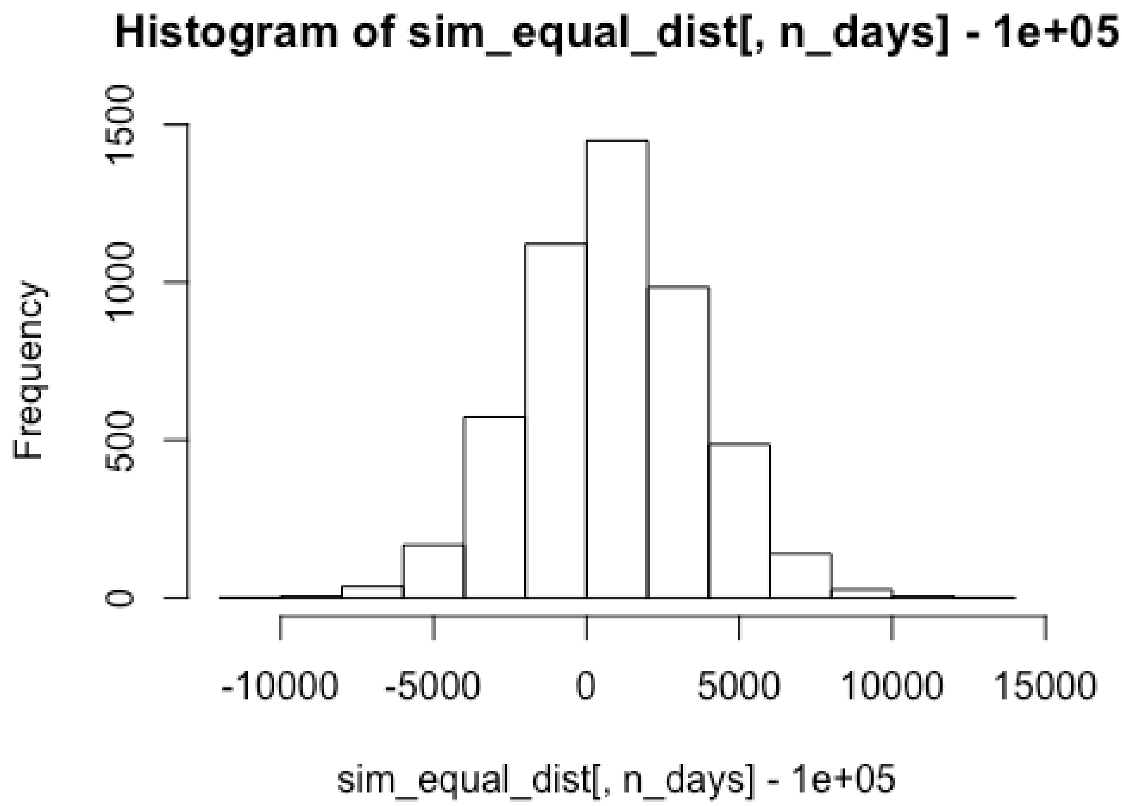
What if we consider these asset classes together in a single portfolio?

First, let's assume an even split (20% of assets in each of the five ETFs above).

```
n_days = 20

set.seed(2015)

sim_equal_dist = 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(etfreturns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        holdings = weights * totalwealth
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}

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



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




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

##       5%
## 3772.192
```

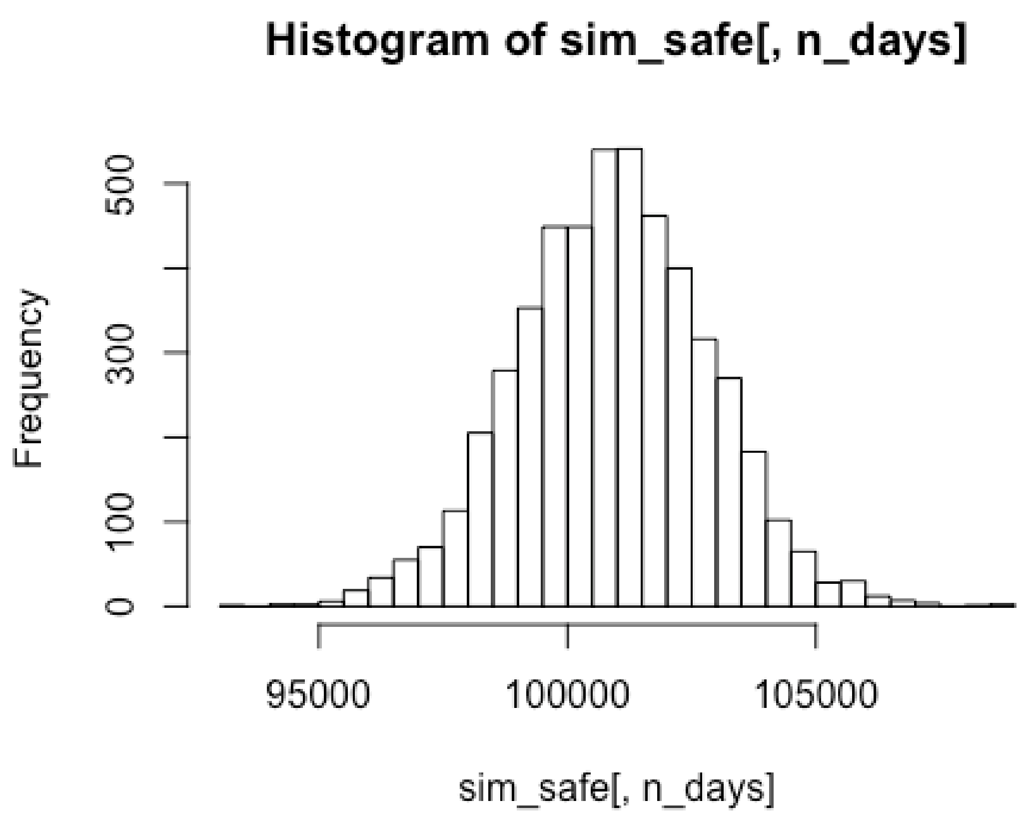
We see that the value at risk (VaR) at the 5% level for this portfolio is $3,772. In other words, we can say with a 95% confidence level that the most money this portfolio will lose over a 4-week trading period is $3,772.

What if we want a portfolio that's "safer"? Let's allocate more of the portfolio to LQD and TLT and remove EEM.

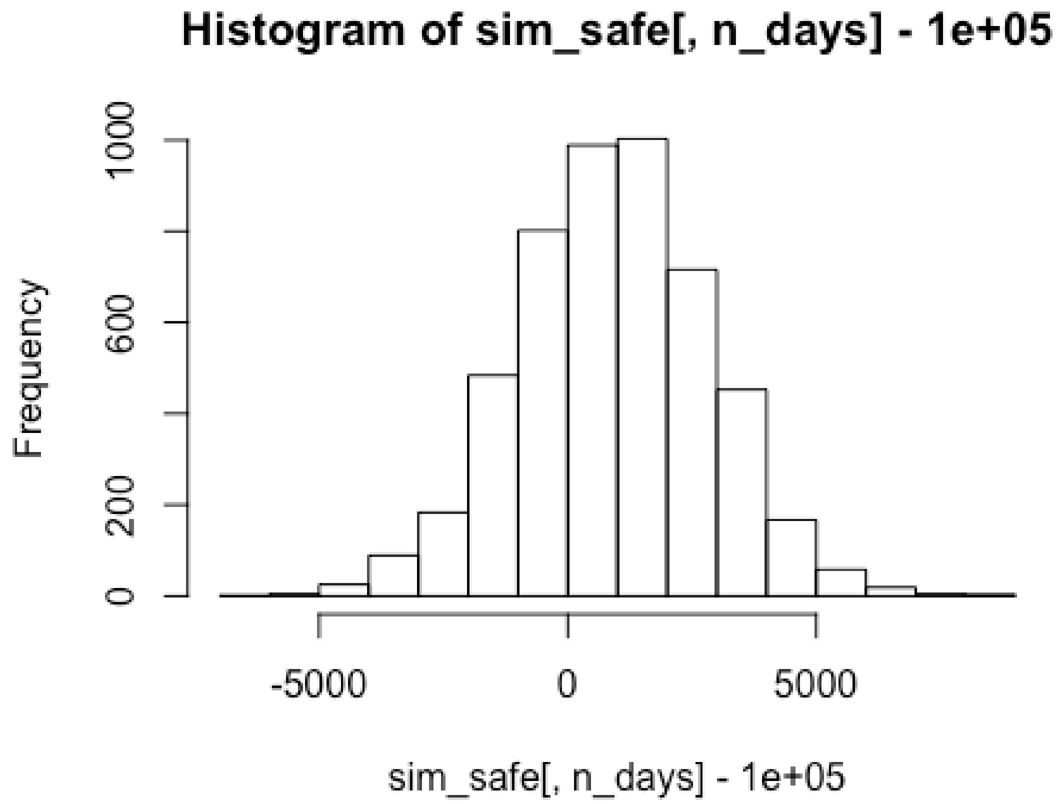
Let's take a look at the VaR associated with this portfolio.

```r
set.seed(2015)

sim_safe = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.4, 0.3, 0.25, 0, 0.05)
    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(etfreturns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        holdings = weights * totalwealth
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}

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




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

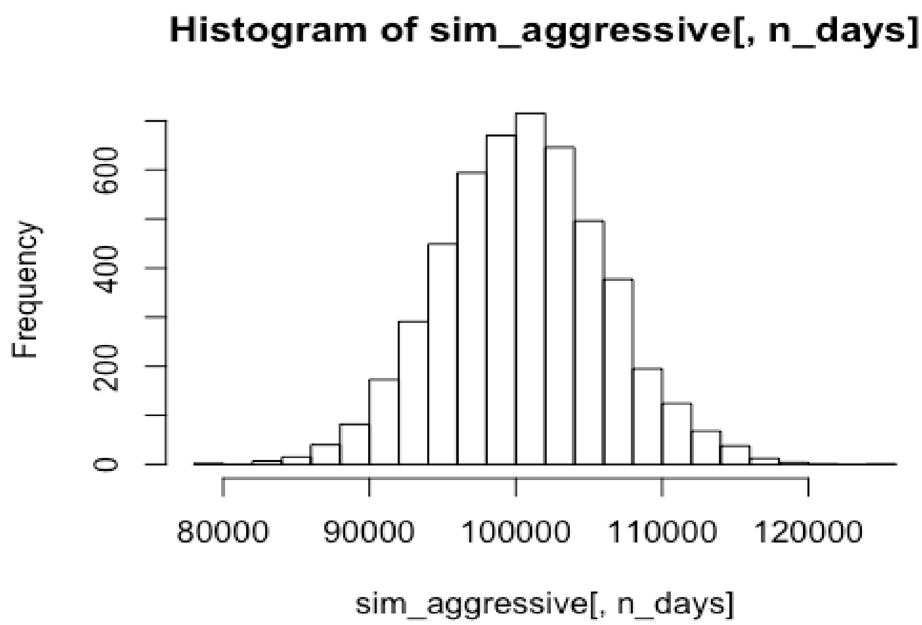



```
# Calculate 5% value at risk
abs(quantile(sim_safe[,n_days], 0.05) - 100000)

##       5%
## 2201.501
```

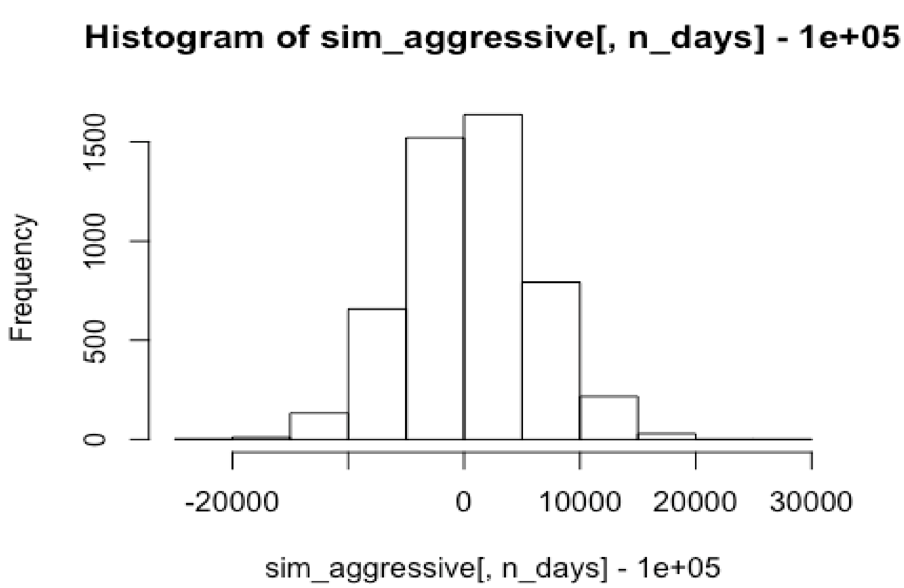
With this "safer" allocation, the VaR at the 5% level is $2,202. What about a portfolio that's considered more "aggreesive"? Let's allocate the portfolio to only include SPY and EEM, with 70% weight on EEM.

```
set.seed(2015)

sim_aggressive = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.3, 0, 0, 0.7, 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(etfreturns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        holdings = weights * totalwealth
```

```
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}


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




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



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

```
##       5%
## 8681.392
```

Now, our VaR is much higher: $8,681.

So what does this all mean? Well, depending on your risk tolerance, one of these portfolios is more preferable than the other two.

```r
quantile(sim_aggressive[,n_days], 0.025); quantile(sim_aggressive[,n_days],
0.975)
```

```
##     2.5%
## 89673.57
```

```
##    97.5%
## 111942.1
```

```r
quantile(sim_safe[,n_days], 0.025); quantile(sim_safe[,n_days], 0.975)
```

```
##     2.5%
## 97043.26
```

```
##    97.5%
## 104676.8
```

```r
quantile(sim_equal_dist[,n_days], 0.025); quantile(sim_equal_dist[,n_days],
0.975)
```

```
##     2.5%
## 95286.13
```

```
##    97.5%
## 106458.6
```

Taking a look at the 90% confidence intervals for each of these portfolios, we see that in the span of a 4-week trading period, the value of...

- the aggressive portfolio can range from as little as $89,674 to as large as $111,942
- the safe portfolio can range from as little as $97,043 to as large as $104,677
- the equally distributed portfolio can range from as little as $95,286 to as large as $106,459

Which would you choose?

Clustering and PCA

We have a dataset of 6,500 different bottles of vinho verde wine from Portugal that contains information on 11 chemical properties for each wine, whether the wine is red or white, and the quality score of the wine.

Let's run PCA and a clustering algorithm on this dataset to see what we can learn.

```
library(ggplot2)
vinho_full <- read.csv("../data/wine.csv")
vinho = vinho_full[,(1:11)]

# Running PCA, basic plotting and summary methods
pca_vinho = prcomp(vinho, scale = TRUE)

pca_vinho

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

```
## residual.sugar        0.1080032 -0.001372773 -0.4497650778
## chlorides             0.2964437 -0.196630217 -0.0434375867
## free.sulfur.dioxide   0.3666563  0.480243340  0.0002125351
## total.sulfur.dioxide -0.3206955 -0.713663486  0.0626848131
## density               0.1128800 -0.003908289  0.7151620723
## pH                    0.1278367 -0.141310977 -0.2063605036
## sulphates            -0.2077642  0.045959499 -0.0772024671
## alcohol               0.2518903 -0.205053085  0.3357018784
```

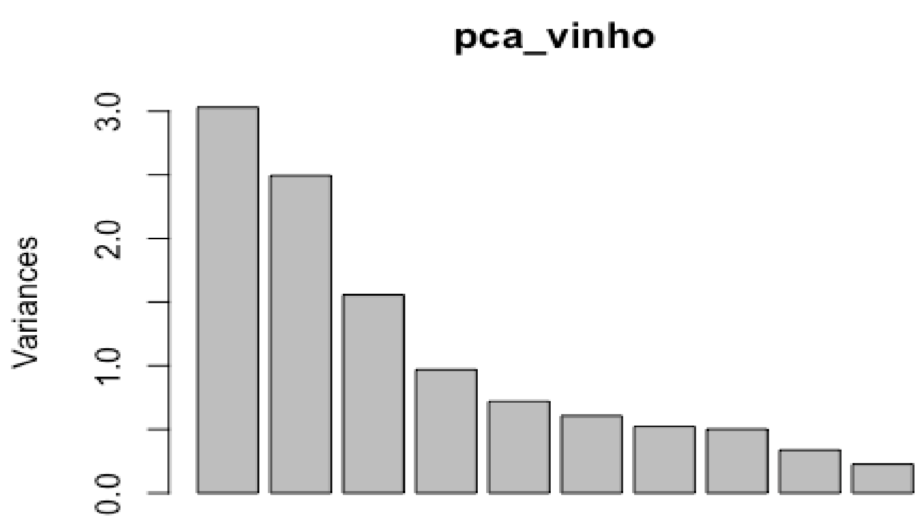
```
summary(pca_vinho)
```

```
## 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
```

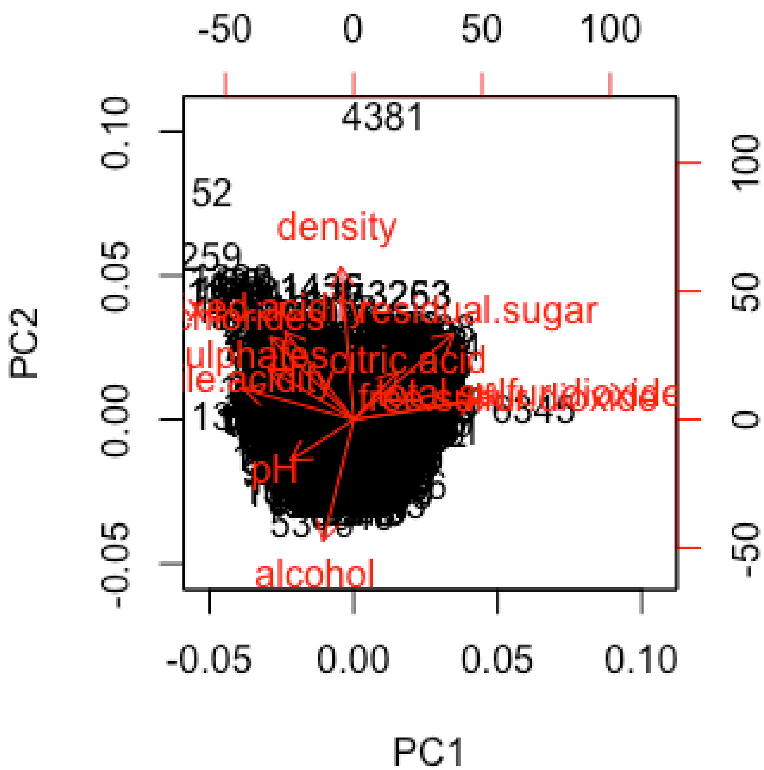
```
sum((pca_vinho$sdev)^2)
```

```
## [1] 11
```

```
plot(pca_vinho)
```

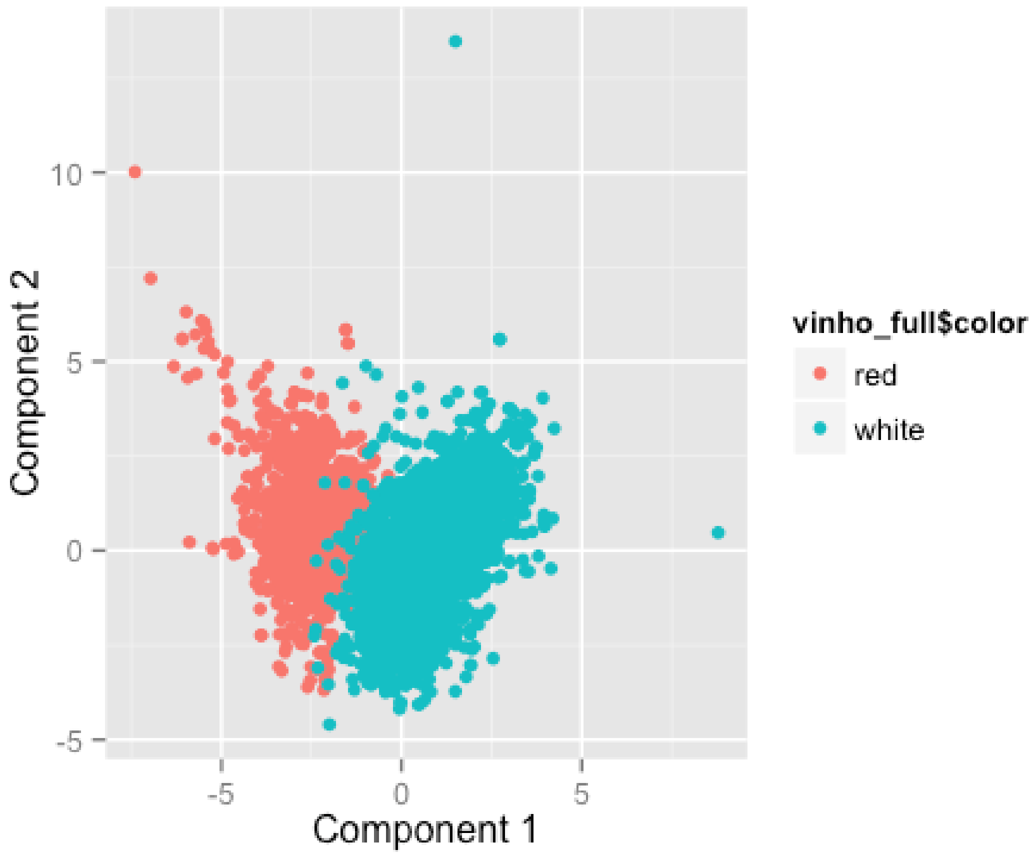



```
biplot(pca_vinho)
```

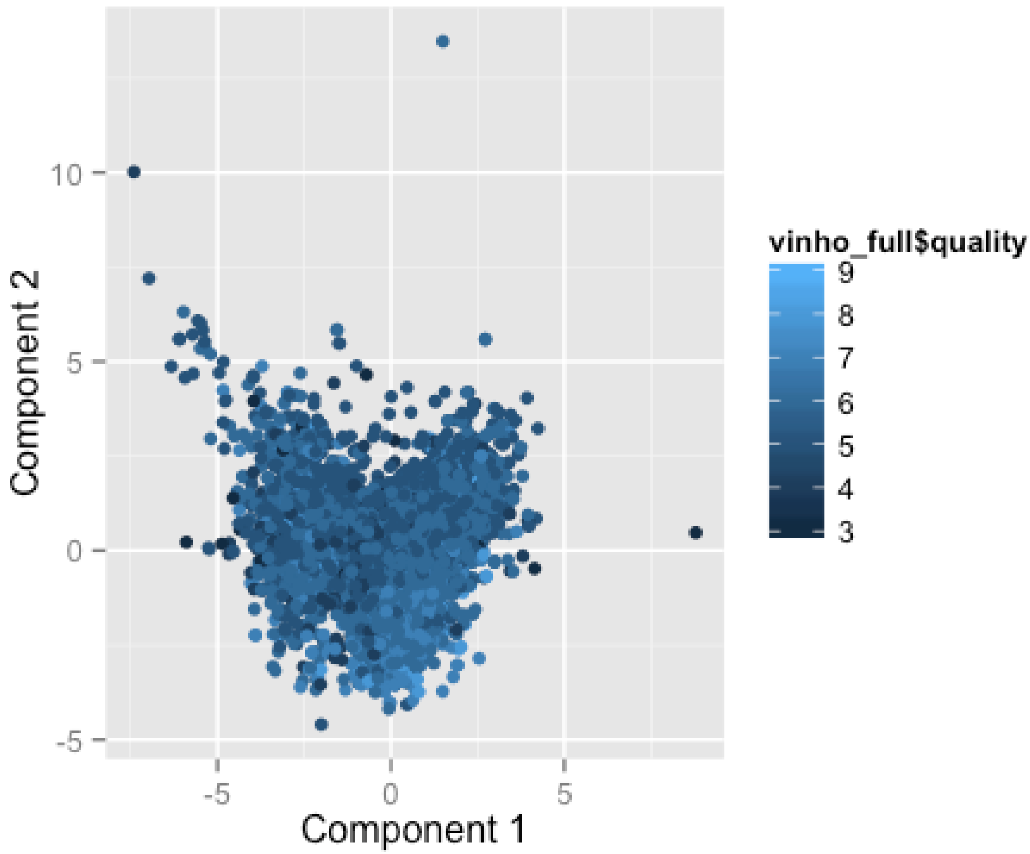


Can we use PCA to distinguish reds from whites? What about to sort the higher from the lower quality wines?

```
# Looking to see if the first two principal components can distinguish reds from whites and sort higher from lower quality wines
loadings = pca_vinho$rotation
scores = pca_vinho$x
qplot(scores[,1], scores[,2], color=vinho_full$color, xlab='Component 1',
ylab='Component 2')
```



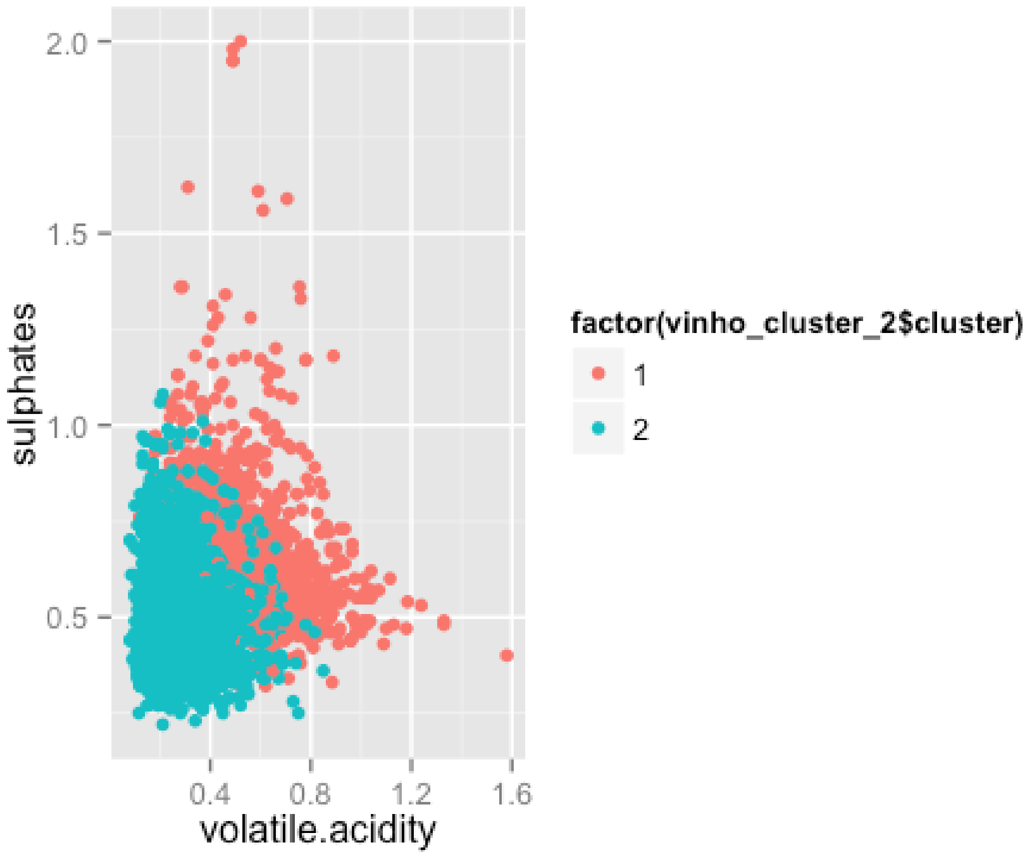

```
qplot(scores[,1], scores[,2], color=vinho_full$quality, xlab='Component 1',
ylab='Component 2')
```



It looks like we can distinguish reds from whites pretty well, but not higher quality from lower quality.

Now, let's run k-means clustering on the dataset, and see if we can use this method to distinguish reds from whites.

```
# K-means clustering
vinho_scaled <- scale(vinho)
vinho_cluster_2 <- kmeans(vinho_scaled, centers=2, nstart=50)

qplot(volatile.acidity, sulphates, data=vinho_full,
color=factor(vinho_cluster_2$cluster))
```




```
color_table = table(vinho_full$color, vinho_cluster_2$cluster)
prop.table(color_table, margin = 1)

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

It does! What about distinguishing quality?

```
vinho_cluster_3 <- kmeans(vinho_scaled, centers=3, nstart=50)

quality_table = table (vinho_cluster_3$cluster, vinho_full$quality)
prop.table(quality_table, margin = 1)

##
##                3            4            5            6            7
##   1 0.0026586906 0.0332336324 0.2120305749 0.4556331007 0.2459288800
##   2 0.0063324538 0.0253298153 0.4242744063 0.4448548813 0.0828496042
##   3 0.0062774639 0.0426867546 0.4369114878 0.3904582549 0.1142498431
##
##                8            9
##   1 0.0491857760 0.0013293453
```

```
##   2 0.0158311346 0.0005277045
##   3 0.0094161959 0.0000000000
```

Oh no, not at all. The three clusters are not very helpful in terms of distinguishing different quality wines.

For this data, using k-means clustering with k = 2 makes more sense than PCA because we are trying to group the wines into 2 clusters (red and white), and we are able to successfully do so. 98.5% of the red wines are in Cluster 1, and 98.6% of the white wines are Cluster 2. However, we can consider using PCA before clustering.

Market Segmentation

Given social media conversation data gathered from the followers of "NutrientH20," let's take a look at the data and try to identify any interesting market segments. What can we do with these market segments? We can use them to better tailor the social content strategy to engage with the Nutrient H20 community.

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

tweetsdf <- as.data.frame(tweets)

tweetsdf_clean <- tweetsdf[,-c(1, 4, 5, 35, 36)]
tweetsdf_clean_scaled <- scale(tweetsdf_clean)

tweet_clusters <- kmeans(tweetsdf_clean_scaled, centers=6, nstart=50)

mu = attr(tweetsdf_clean_scaled,"scaled:center")
sigma = attr(tweetsdf_clean_scaled,"scaled:scale")

colSums(tweetsdf)
```

```
##          chatter   current_events           travel    photo_sharing
##            34671            12030            12493            21256
##    uncategorized          tv_film    sports_fandom         politics
##             6408             8436            12564            14098
##             food           family  home_and_garden            music
##            11015             6809             4104             5354
##             news    online_gaming         shopping health_nutrition
##             9502             9528            10951            20235
##      college_uni   sports_playing          cooking              eco
##            12213             5038            15750             4038
##        computers         business         outdoors           crafts
##             5116             3336             6169             4066
##       automotive              art         religion           beauty
##             6541             5713             8634             5558
##        parenting           dating           school personal_fitness
##             7262             5603             6051            11524
```

```
##         fashion   small_business             spam           adult
##            7855             2651               51            3179
```

```r
#Multi-faceted
rbind(tweet_clusters$centers[1,],tweet_clusters$centers[1,]* sigma + mu)
```

```
##      current_events    travel     tv_film sports_fandom   politics
## [1,]    -0.06068533 -0.213106 -0.04263903    -0.2861432 -0.2562403
## [2,]     1.44925934  1.097944  0.99955782     0.9756799  1.0119390
##            food     family home_and_garden      music       news
## [1,] -0.3537408 -0.2550026      -0.1128540 -0.1213488 -0.2448602
## [2,]  0.7694008  0.5750608       0.4375415  0.5542781  0.6911342
##      online_gaming    shopping health_nutrition college_uni sports_playing
## [1,]    -0.2334385 -0.05159407       -0.3283642  -0.2246898     -0.2286812
## [2,]     0.5814725  1.29604245        1.0908689   0.8985187      0.4160955
##         cooking        eco  computers   business   outdoors     crafts
## [1,] -0.3371854 -0.1557282 -0.2325845 -0.1233630 -0.3147823 -0.1849265
## [2,]  0.8416980  0.3924386  0.3747513  0.3378289  0.4019456  0.3648021
##      automotive         art   religion     beauty  parenting      dating
## [1,] -0.1793921 -0.06503471 -0.2953001 -0.2731218 -0.3029224 -0.09467412
## [2,]  0.5847889  0.61883706  0.5299580  0.3424718  0.4623038  0.54211806
##          school personal_fitness    fashion small_business
## [1,] -0.2432415       -0.3334818 -0.2682625    -0.09916295
## [2,]  0.4786646        0.6599602  0.5060800     0.27503869
```

```r
#Outdoorsy health nuts
rbind(tweet_clusters$centers[2,],tweet_clusters$centers[2,]* sigma + mu)
```

```
##      current_events     travel   tv_film sports_fandom  politics
## [1,]    -0.00465003 -0.0124447 0.4650048    -0.1126063 -0.160907
## [2,]     1.52036199  1.5565611 1.8416290     1.3506787  1.300905
##             food     family home_and_garden      music       news
## [1,] -0.06586381 0.1841234       0.1346961 0.3772801 -0.1861461
## [2,]  1.28054299 1.0723982       0.6199095 1.0678733  0.8144796
##      online_gaming     shopping health_nutrition college_uni
## [1,]      3.077550 -0.005129062       -0.1910742    3.072237
## [2,]      9.479638  1.380090498        1.7081448   10.450226
##      sports_playing    cooking         eco   computers   business
## [1,]       1.997974 -0.1459169 -0.03950252 -0.06309127 0.03897203
## [2,]       2.588235  1.4977376  0.48190045  0.57466063 0.45022624
##        outdoors     crafts automotive       art   religion     beauty
## [1,] -0.1046404 0.09691369 0.05332457 0.2896561 -0.1313516 -0.1953830
## [2,]  0.6561086 0.59502262 0.90271493 1.1968326  0.8438914  0.4457014
##       parenting      dating     school personal_fitness     fashion
## [1,] -0.1541265 0.003553956 -0.1986304       -0.1902262 -0.05628343
## [2,]  0.6877828 0.717194570  0.5316742        1.0045249  0.89366516
##      small_business
## [1,]      0.2501255
## [2,]      0.4909502
```

```r
#College students and online gamers
rbind(tweet_clusters$centers[3,],tweet_clusters$centers[3,]* sigma + mu)
```

```
##      current_events     travel      tv_film sports_fandom   politics
## [1,]      0.1189579 -0.1031972 -0.005041735      2.003580 -0.2054111
## [2,]      1.6772069  1.3491436  1.061923584      5.923584  1.1660079
##         food   family home_and_garden      music        news online_gaming
## [1,] 1.78927 1.439395       0.1731272 0.06707076 -0.08215687   -0.07917604
## [2,] 4.57444 2.494071       0.6482213 0.74835310  1.03293808    0.99604743
##        shopping health_nutrition college_uni sports_playing    cooking
## [1,] 0.0498717       -0.1569304  -0.1205377      0.1024611 -0.1051163
## [2,] 1.4795784        1.8616601   1.2002635      0.7391304  1.6376812
##            eco  computers  business    outdoors    crafts automotive
## [1,] 0.2005394 0.06853509 0.1270346 -0.07739242 0.7136648  0.1631118
## [2,] 0.6666667 0.72990777 0.5111989  0.68906456 1.0988142  1.0527009
##             art religion    beauty parenting     dating   school
## [1,] 0.09286884 2.185003 0.3103454  2.074241 0.05651762 1.638026
## [2,] 0.87615283 5.279315 1.1172596  4.064559 0.81159420 2.714097
##      personal_fitness    fashion small_business
## [1,]       -0.1132285 0.02132924      0.1123700
## [2,]        1.1897233 1.03557312      0.4057971
```

```r
#Political and news-interested travelers
rbind(tweet_clusters$centers[4,],tweet_clusters$centers[4,]* sigma + mu)
```

```
##      current_events     travel     tv_film sports_fandom   politics
## [1,]     0.01408452 -0.1523202 -0.05517042    -0.2050859 -0.1772404
## [2,]     1.54413408  1.2368715  0.97877095     1.1508380  1.2513966
##          food      family home_and_garden      music       news
## [1,] 0.416738 -0.06823015       0.1744041 0.05646613 -0.0441181
## [2,] 2.137430  0.78659218       0.6491620 0.73743017  1.1128492
##      online_gaming   shopping health_nutrition college_uni sports_playing
## [1,]    -0.1317530 0.04602798          2.09573  -0.2031600    -0.02870754
## [2,]     0.8547486 1.47262570         11.98994   0.9608939     0.61117318
##        cooking       eco   computers   business outdoors     crafts
## [1,] 0.3800218 0.5377902 -0.07192148 0.06486851 1.612500 0.09480003
## [2,] 3.3016760 0.9262570  0.56424581 0.46815642 2.732961 0.59329609
##      automotive       art   religion     beauty  parenting    dating
## [1,] -0.1224577 0.0152829 -0.1746962 -0.2028739 -0.1080918 0.1891801
## [2,]  0.6625698 0.7497207  0.7608939  0.4357542  0.7575419 1.0480447
##          school personal_fitness    fashion small_business
## [1,] -0.1448908         2.060419 -0.1038445    -0.05607079
## [2,]  0.5955307         6.417877  0.8067039     0.30167598
```

```r
#Fashionistas
rbind(tweet_clusters$centers[5,],tweet_clusters$centers[5,]* sigma + mu)
```

```
##      current_events   travel    tv_film sports_fandom politics       food
## [1,]      0.1148271 1.760370 0.07959171     0.1878778 2.348126 0.02192452
## [2,]      1.6719653 5.608382 1.20231214     2.0000000 8.906069 1.43641618
##          family home_and_garden      music     news online_gaming
```

```
## [1,] 0.04874622       0.1327789 -0.0351506 1.940355    -0.1363161
## [2,] 0.91907514       0.6184971  0.6430636 5.281792     0.8424855
##        shopping health_nutrition college_uni sports_playing    cooking
## [1,] 0.01961876       -0.2029778  -0.0764354    -0.01231167 -0.2038201
## [2,] 1.42485549        1.6546243   1.3280347     0.62716763  1.2991329
##            eco computers  business  outdoors    crafts automotive
## [1,] 0.1229399  1.554606 0.3550546 0.1103906 0.1539572   1.109329
## [2,] 0.6069364  2.482659 0.6690751 0.9161850 0.6416185   2.345376
##             art    religion     beauty  parenting    dating     school
## [1,] 0.01368073 -0.03699587 -0.1621102 0.01280996 0.2043846 -0.0270559
## [2,] 0.74710983  1.02456647  0.4898844 0.94075145 1.0751445  0.7355491
##      personal_fitness    fashion small_business
## [1,]       -0.1915067 -0.1585677      0.2624279
## [2,]        1.0014451  0.7066474      0.4985549
```

```
#Family first, religion and sports second
rbind(tweet_clusters$centers[6,],tweet_clusters$centers[6,]* sigma + mu)
```

```
##      current_events      travel     tv_film sports_fandom   politics
## [1,]      0.1649388 -0.05979689 -0.02547995    -0.2157259 -0.1405788
## [2,]      1.7355517  1.44833625  1.02802102     1.1278459  1.3625219
##           food    family home_and_garden     music        news
## [1,] -0.205126 0.0119561       0.1252624 0.5341213 -0.08949935
## [2,]  1.033275 0.8774081       0.6129597 1.2294221  1.01751313
##      online_gaming  shopping health_nutrition college_uni sports_playing
## [1,]   -0.05620051 0.2504434      -0.08136767 -0.02705292      0.1885559
## [2,]    1.05779335 1.8423818       2.20140105  1.47110333      0.8231173
##        cooking        eco computers  business   outdoors    crafts
## [1,]  2.574942 0.00562663 0.0287747 0.2461837 0.01606654 0.1060087
## [2,] 10.830123 0.51663748 0.6830123 0.5936953 0.80210158 0.6024518
##      automotive       art   religion   beauty parenting    dating
## [1,] 0.01044364 0.1269550 -0.1449444 2.416619 -0.084465 0.1278336
## [2,] 0.84413310 0.9316988  0.8178634 3.914186  0.793345 0.9387040
##         school personal_fitness  fashion small_business
## [1,] 0.1630726      -0.05813289 2.495112      0.2123521
## [2,] 0.9614711       1.32224168 5.558669      0.4676007
```

```
tweet_clusters$size
```

```
## [1] 4523  442  759  895  692  571
```

After removing some prevalent and not very informative conversation topics such as chatter, photo sharing, spam, etc and running k-means clustering with a k of 6, we some interesting segments of the Nutrient H20 followers.

For example, 11% of the followers are in the "Outdoorsy Health Nuts" cluster, meaning that they enjoy talking about fitness, health, and the outdoors. This is not too surprising given the probable target market for H20, but clustering quantifies the proportion of its followers that are especially into these topics--the "health nuts."

Given that health/nutrition is the top conversation category (besides photo sharing and chatter), there is strong evidence that social content with a health/nutrition/fitness focus would resonate well with the Nutrient H20 community. For example, Nutrient H20 could post some healthy recipes or workout routines on their social platforms.

Additionally, we see 10% of the followers fall into the "Family First, Religion and Sports Second" cluster. These individuals enjoy participating in social conversation about family-oriented topics like parenting, school, religion, and food. They can also be classified as sports fans. Knowing this, we can envision a mom or dad who is constantly juggling work, raising a family, and enjoying watching sports. Nutrient H20 can offer value by delivering social content that highlights the role of the product in a hectic lifestyle both for parents and their children.

One way we could improve this analysis is to include follower counts for each individual in the analysis. This way, we could identify influentials (those with high follower counts) that Nutrient H20 could leverage as brand advocates. This is a great first step though.