Exercise 1 for Dr. Scott

J. Zhao (jz5223)

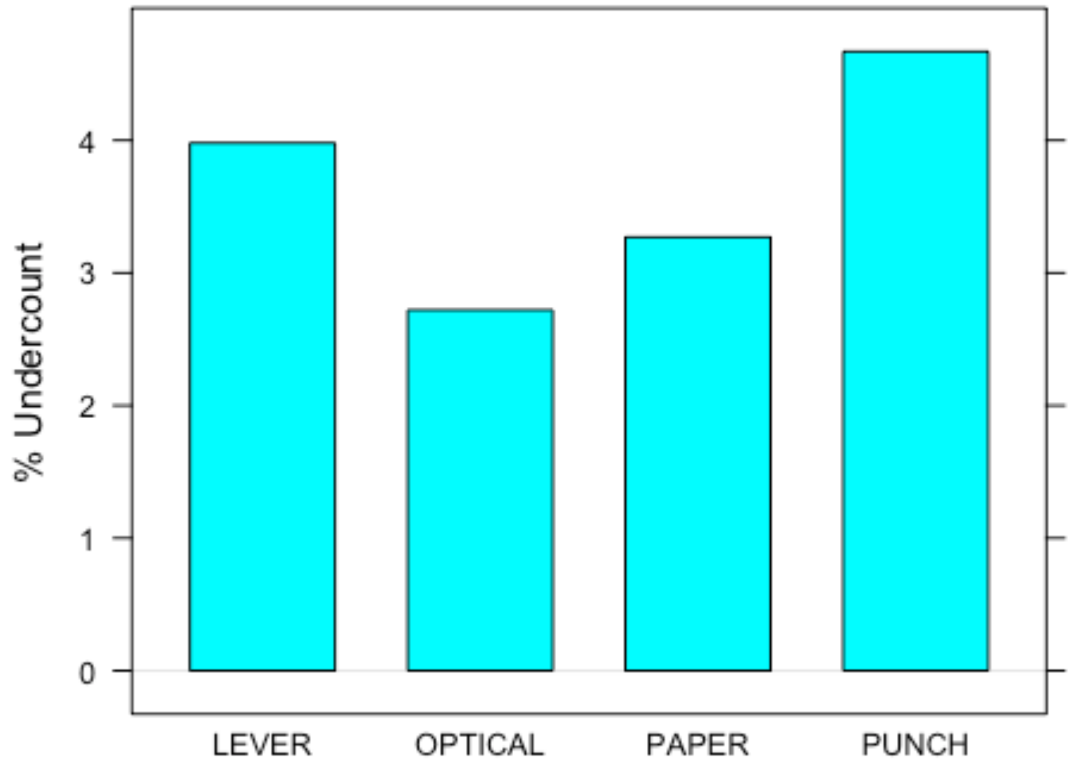
August 5, 2015

```
jingshen_seed = 8148154
set.seed(jingshen_seed)
```

Exploratory analysis

For the georgia2000 (g2k) dataset, I will use the dplyr library to do group-by / pivot tables and the lattice library for bar charts.

```
library(dplyr)
library(lattice)

g2k = read.csv("../data/georgia2000.csv", header=T)
g2k$undercount = g2k$ballots-g2k$votes

pivot = summarise(group_by(g2k, equip),
                  sum_undercount = sum(undercount),
                  sum_ballot = sum(ballots),
                  pct_undercount = round(sum_undercount/sum_ballot*100,
2))
pivot

## Source: local data frame [4 x 4]
##
##     equip sum_undercount sum_ballot pct_undercount
## 1   LEVER          17016     427780           3.98
## 2 OPTICAL          39090    1436159           2.72
## 3   PAPER            113       3454           3.27
## 4   PUNCH          38462     823921           4.67

barchart(pct_undercount~equip, data=pivot, ylab="% Undercount",
origin=0)
```

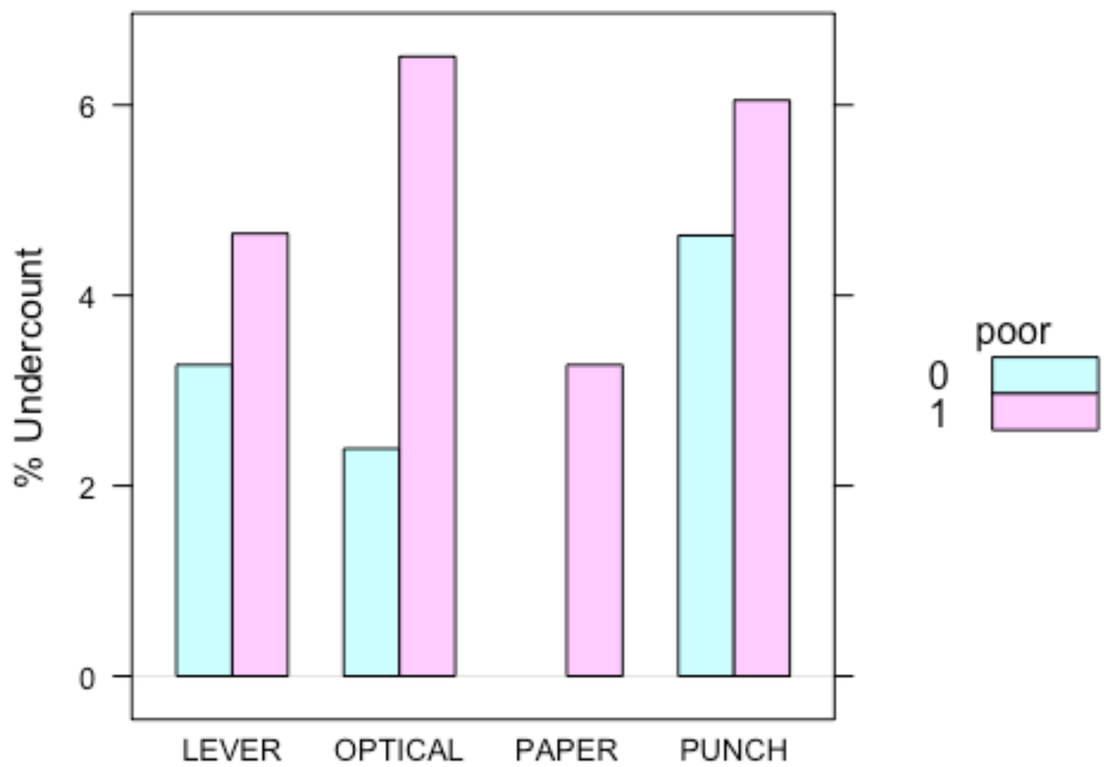


At the state level tally, PUNCH has the highest undercount of 4.67%, followed by LEVER with 3.98%. OPTICAL is associated with the lowest state-wide average vote undercount of 2.72%.

Using the same summarise function from above, I further grouped by "poor" to see if equipment choice has a disparate impact on poor vs. non-poor communities.

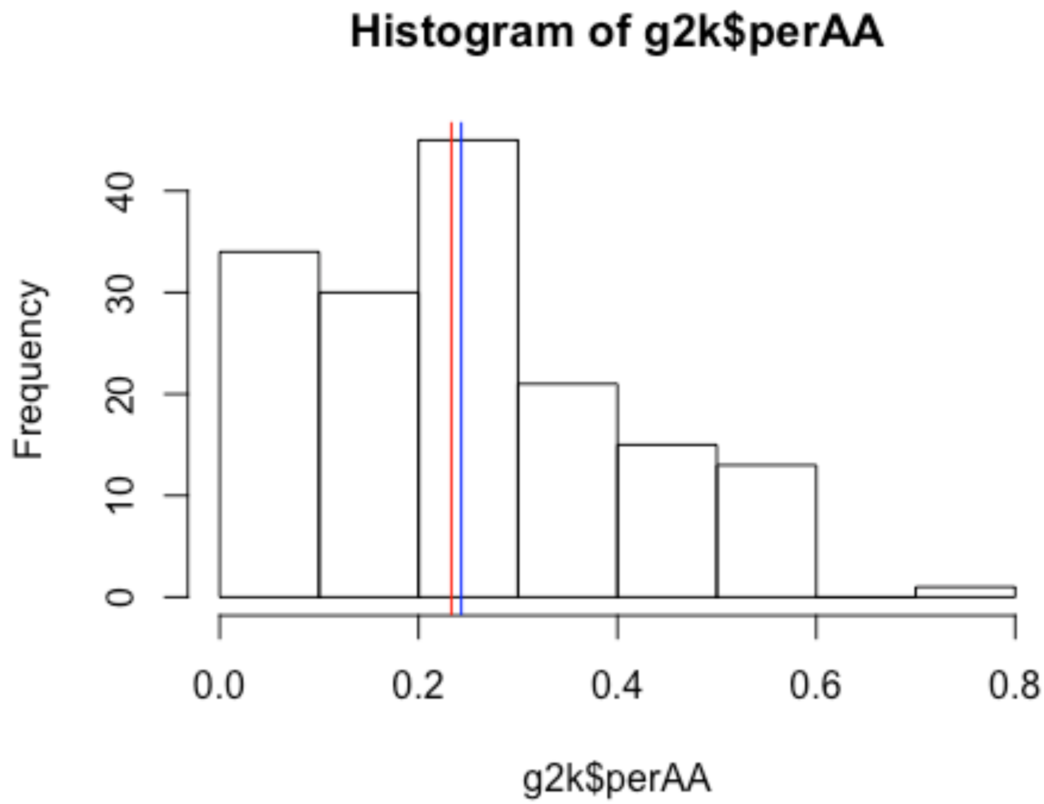
```
## Source: local data frame [7 x 5]
## Groups: equip
##
##     equip poor sum_undercount sum_ballot pct_undercount
## 1   LEVER    0           6816     208526           3.27
## 2   LEVER    1          10200     219254           4.65
## 3 OPTICAL    0          31633    1321694           2.39
## 4 OPTICAL    1           7457     114465           6.51
## 5   PAPER    1            113       3454           3.27
## 6   PUNCH    0          37033     800309           4.63
## 7   PUNCH    1           1429      23612           6.05
```

```
barchart(pct_undercount~equip, data=pivot_poor, groups=poor,
         ylab="% Undercount", origin=0,
         auto.key=list(space="right", title="poor", cex.title=1))
```



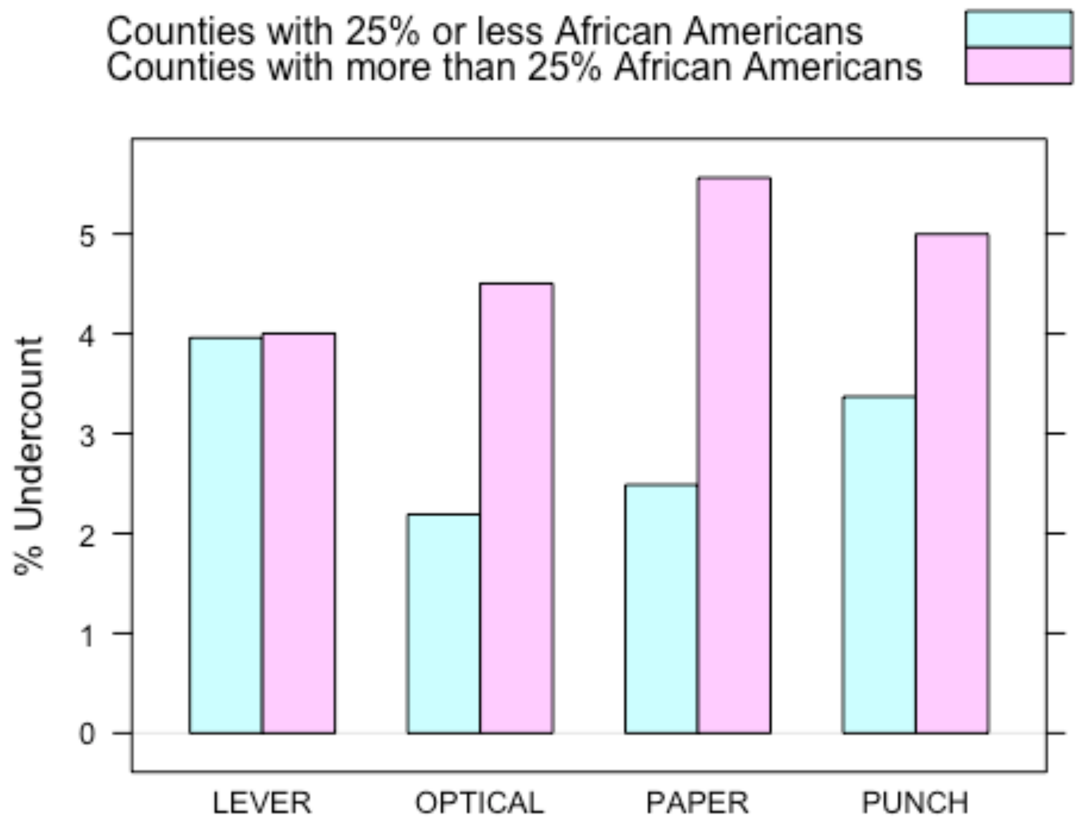
OPTICAL seems to discriminate against poor communities. This bar chart suggests that PAPER or LEVER should be used for poor communities.

```
hist(g2k$perAA)
abline(v=mean(g2k$perAA),col="blue")
abline(v=median(g2k$perAA),col="red")
```



Based on the histogram, I decided on a perAA cut-off of 25%, meaning that counties with more than 25% African Americans will be categorized as a "minority community" for the purposes of this exercise.

I used the same barchart function from lattice to make the following:



Overall, minority communities have higher undercount (minority being roughly defined as "more African American population than average"). PAPER and OPTICAL produce a large difference between minority and non-minority communities, whereas LEVER is non-discriminatory by this criterion.

Bootstraping

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

tickers = c("SPY", "TLT", "LQD", "EEM", "VNQ")
prices = yahooSeries(tickers, from='2010-08-01', to='2015-07-31')

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))
}
returns = YahooPricesToReturns(prices)

mean(returns[,1]) #SPY

## [1] 0.000620997

sd(returns[,1])

## [1] 0.009351005

mean(returns[,2]) #TLT

## [1] 0.0003403541

sd(returns[,2])

## [1] 0.009768007

mean(returns[,3]) #LQD

## [1] 0.0002036266

sd(returns[,3])

## [1] 0.003581299

mean(returns[,4]) #EEM

## [1] 6.504442e-05

sd(returns[,4])

## [1] 0.01372734

mean(returns[,5]) #VNQ

## [1] 0.0005390817

sd(returns[,5])

## [1] 0.01152071
```

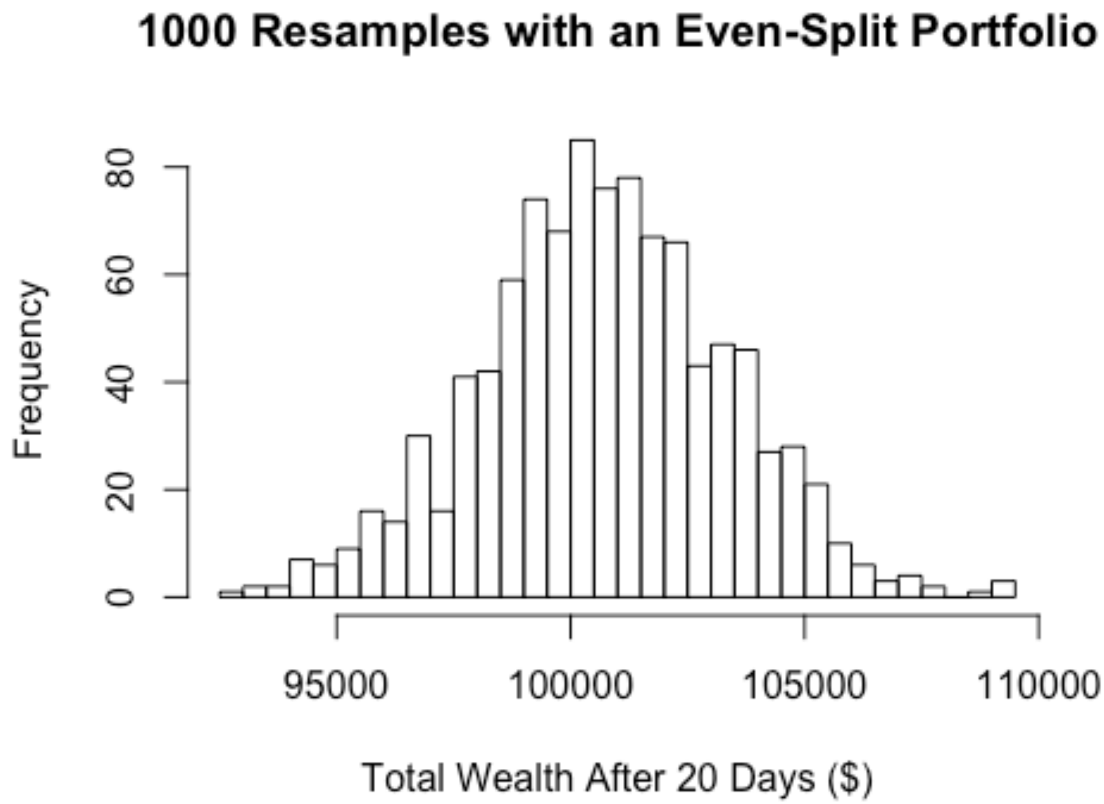
LQD seemed to be low-return and low-risk based on mean and standard deviation. LQD and SPY have lower standard deviations than even the US Treasury Bonds (TLT), so I will consider them low-risk, especially LQD. The EEM ETF had the lowest average returns and the highest standard deviation, which makes sense because it is following the emerging markets. VNQ had the second highest standard deviation, and thus I will pick EEM and VNQ for my high-risk portfolio.

Next, I wanted to compare the latter four ETF's against the movement of the S&P 500 (SPY), to find out whether the other ETF's move with or against the market, and to what degree.

```
coef(lm(returns[,2]~returns[,1])) #TLT

##  (Intercept) returns[, 1]
##  0.000691201 -0.564973561

coef(lm(returns[,3]~returns[,1])) #LQD

##   (Intercept)  returns[, 1]
##  0.0002321012 -0.0458530951

coef(lm(returns[,4]~returns[,1])) #EEM

##  (Intercept) returns[, 1]
## -0.000699388  1.230975946

coef(lm(returns[,5]~returns[,1])) #VNQ

##   (Intercept)  returns[, 1]
## -5.289625e-05  9.532702e-01
```

The coefficients from the linear models further support my initial choice of LQD as low-risk. The TLT seems to move against the market somewhat, making it a good choice for my low-risk portfolio to counteract the S&P 500. When the market is going up or down, VNQ and EEM tend to move in the same direction but with a higher magnitude, which further support them as higher-risk ETF's.

```
initial_funding = 100000
n_days = 20
set.seed(jingshen_seed)
bootstrap_even = foreach(i=1:1000, .combine='rbind') %do% {
  totalwealth = initial_funding
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        wealthtracker[today] = totalwealth
    holdings = weights * totalwealth
    }
    wealthtracker
}
hist(bootstrap_even[,n_days], breaks=25, main="1000 Resamples with an
Even-Split Portfolio", xlab="Total Wealth After 20 Days ($)")
```




```
quantile(bootstrap_even[,n_days] - initial_funding, 0.25)

##      25%
## -978.959

quantile(bootstrap_even[,n_days] - initial_funding, 0.5)

##      50%
## 692.8948

quantile(bootstrap_even[,n_days] - initial_funding, 0.75)

##      75%
## 2435.264

quantile(bootstrap_even[,n_days] - initial_funding, 0.75) -
quantile(bootstrap_even[,n_days] - initial_funding, 0.25) #IQR

##      75%
## 3414.223
```

With the even-split portfolio, the average sample returned $700. Below I will run the same chunck of script on my low-risk and high-risk portfolios.

```
quantile(bootstrap_lowrisk[,n_days] - initial_funding, 0.25)
```

```
##       25%
## -407.2044
```

```r
quantile(bootstrap_lowrisk[,n_days] - initial_funding, 0.5)
```

```
##      50%
## 772.6977
```

```r
quantile(bootstrap_lowrisk[,n_days] - initial_funding, 0.75)
```

```
##      75%
## 1930.027
```

```r
quantile(bootstrap_lowrisk[,n_days] - initial_funding, 0.75) -
quantile(bootstrap_lowrisk[,n_days] - initial_funding, 0.25) #IQR
```

```
##      75%
## 2337.232
```

The 50th sample quantile increased by a mere $80 in the conversative portfolio, which was comprised of 40% LQD, and 30% each of SPY and TLT. The inter-quartile range decreased by about $1,080, signifying that it is indeed a safer bet.

```r
quantile(bootstrap_highrisk[,n_days] - initial_funding, 0.5)
```

```
##      50%
## 464.9022
```

```r
quantile(bootstrap_highrisk[,n_days] - initial_funding, 0.25);
quantile(bootstrap_highrisk[,n_days] - initial_funding, 0.75)
```

```
##       25%
## -2725.239
```

```
##      75%
## 3873.354
```

As expected, the high-risk portfolio of 50% EEM and 50% VNQ exhibited high potential gains and losses, with a 50th percentile of merely $465. I don't believe in luck, so I would advise investors to play it safe with my low-risk portfolio proposal.

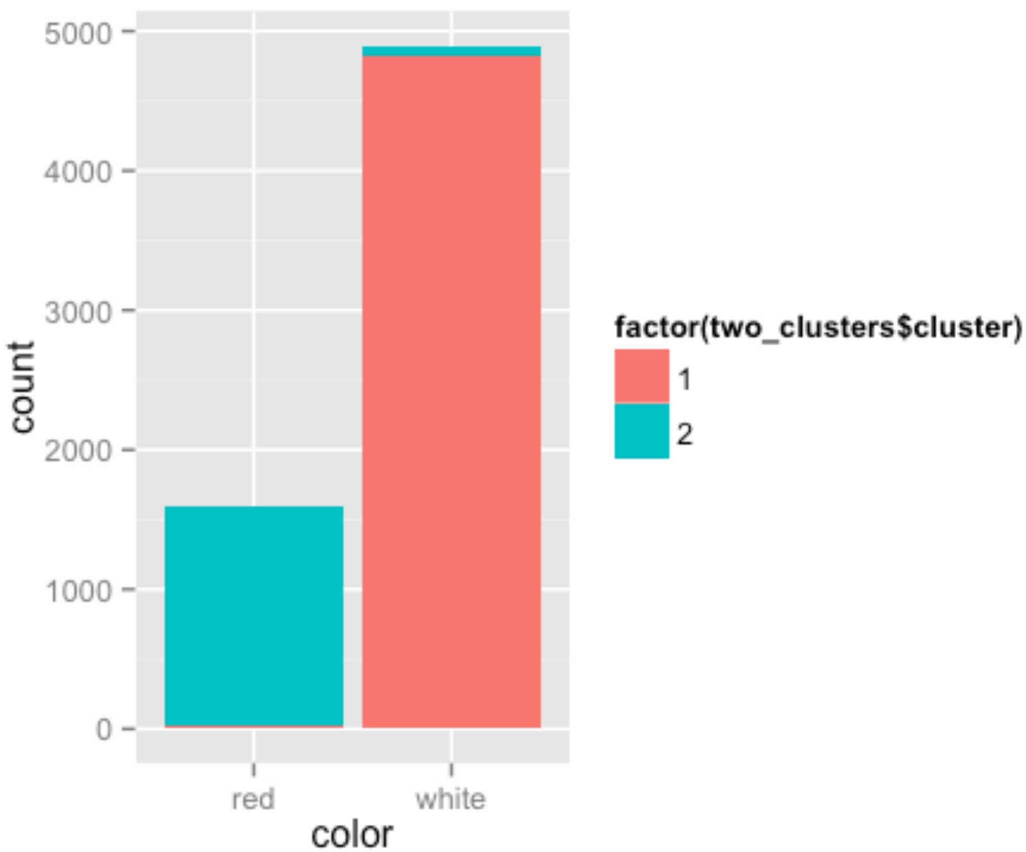
Clustering and PCA

```r
library(ggplot2)
set.seed(jingshen_seed)
wine = read.csv("../data/wine.csv", header=T)
winex = scale(wine[,1:11],center=TRUE, scale=TRUE)
two_clusters = kmeans(winex, 2, nstart=500)
prop.table(table(wine$color, two_clusters$cluster), margin = 1)
```

```
##
##                   1          2
```

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

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

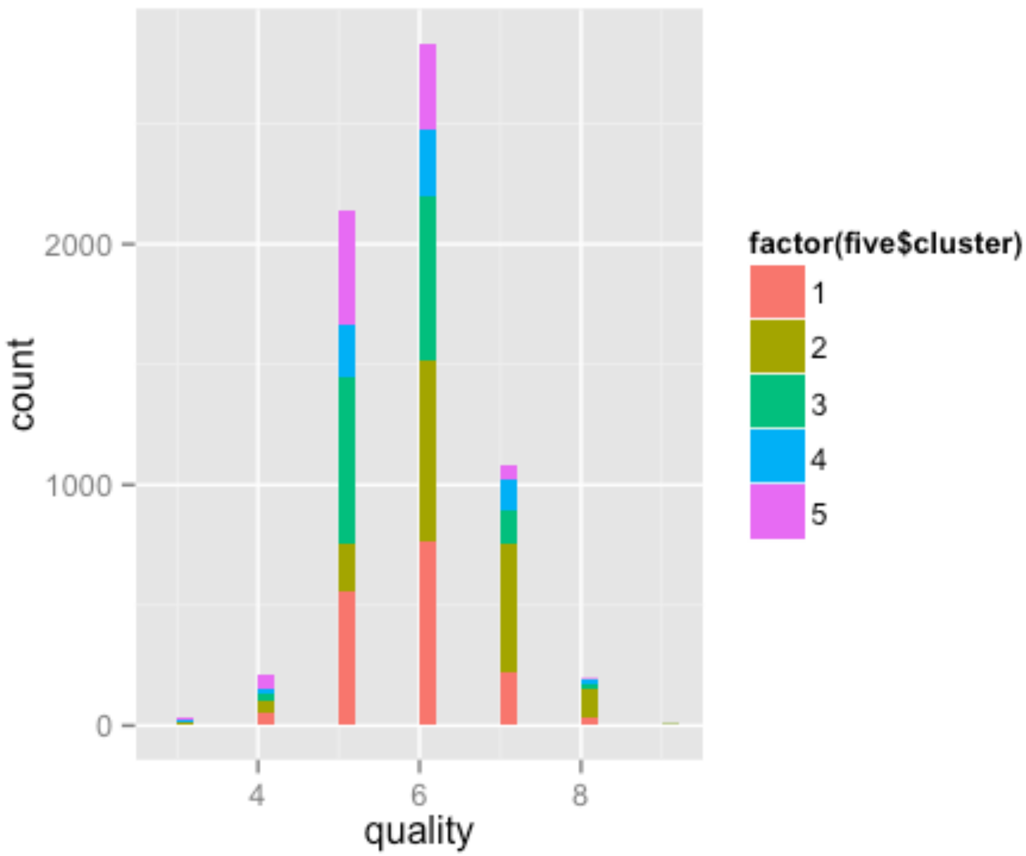



```r
set.seed(jingshen_seed)
five = kmeans(winex, 5, nstart=100)
prop.table(table(wine$quality, five$cluster), margin = 1)
```

```
##
##               1          2          3          4          5
##   3 0.13333333 0.16666667 0.26666667 0.20000000 0.23333333
##   4 0.26388889 0.18981481 0.13888889 0.09722222 0.31018519
##   5 0.26286249 0.09214219 0.32179607 0.10336763 0.21983162
##   6 0.26939351 0.26586742 0.24153738 0.09626234 0.12693935
##   7 0.20574606 0.49582947 0.12233550 0.12789620 0.04819277
##   8 0.17098446 0.61139896 0.12953368 0.06217617 0.02590674
##   9 0.00000000 0.80000000 0.20000000 0.00000000 0.00000000
```

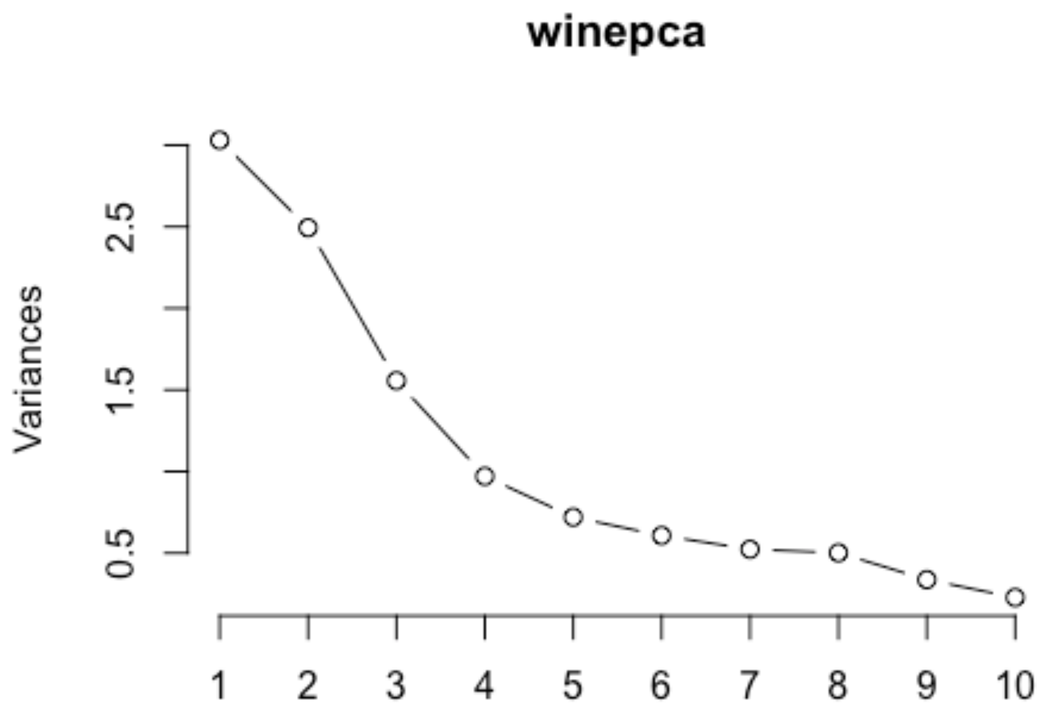
```r
qplot(quality, data=wine, fill=factor(five$cluster))
```

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

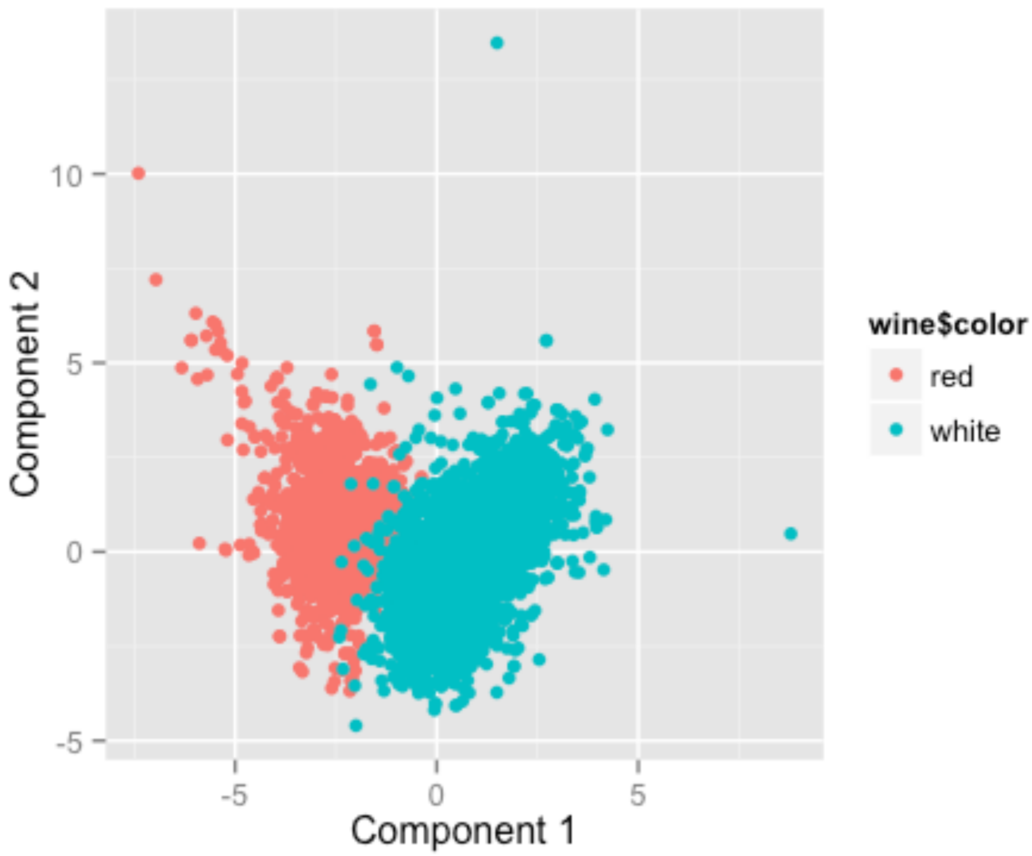


Two k-means clusters superimposed almost perfectly (~98.5%) on top of the actual red and white shows that it is not difficult to distinguish red wines from the white using the chemical properties given in the dataset. On the other hand, increasing the number of clusters could not differentiate wines by quality, as demonstrated by the very colorful graph above.

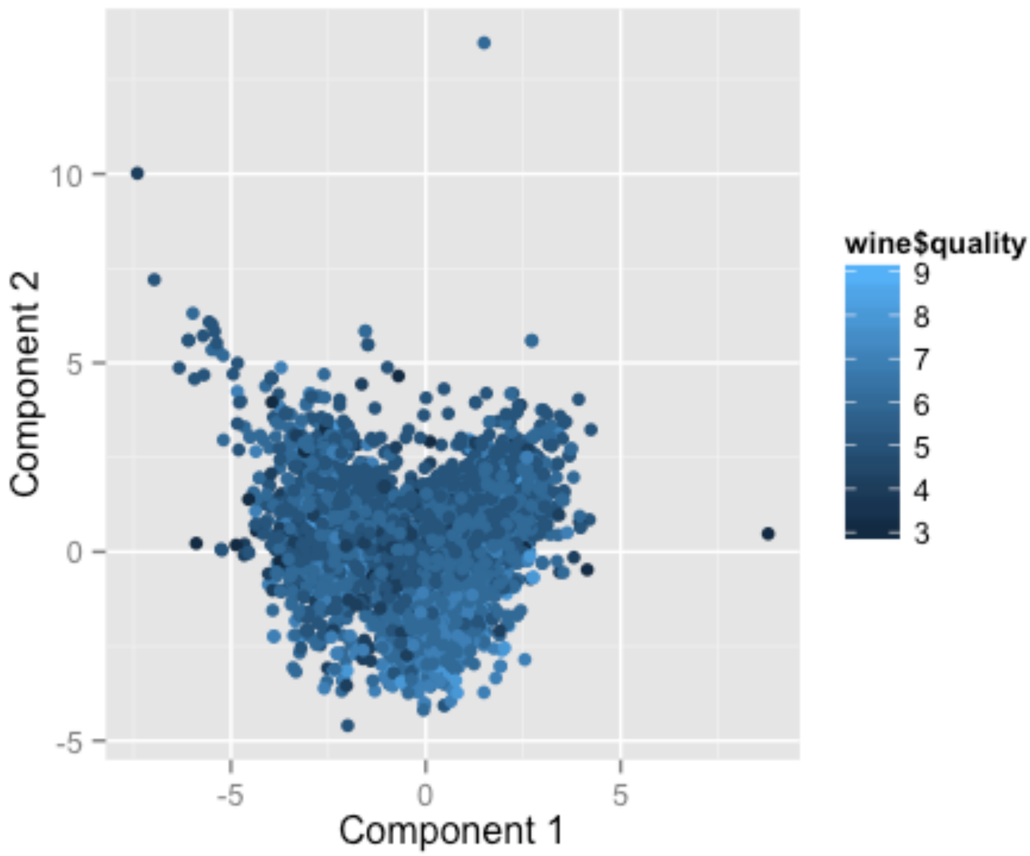
```
winepca = prcomp(winex)
plot(winepca, type="lines")
```




```
summary(winepca)

## 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

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




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



As shown above, the first two principal components can predict color like k-means can, but PCA, too, cannot differentiate the wines' quality.

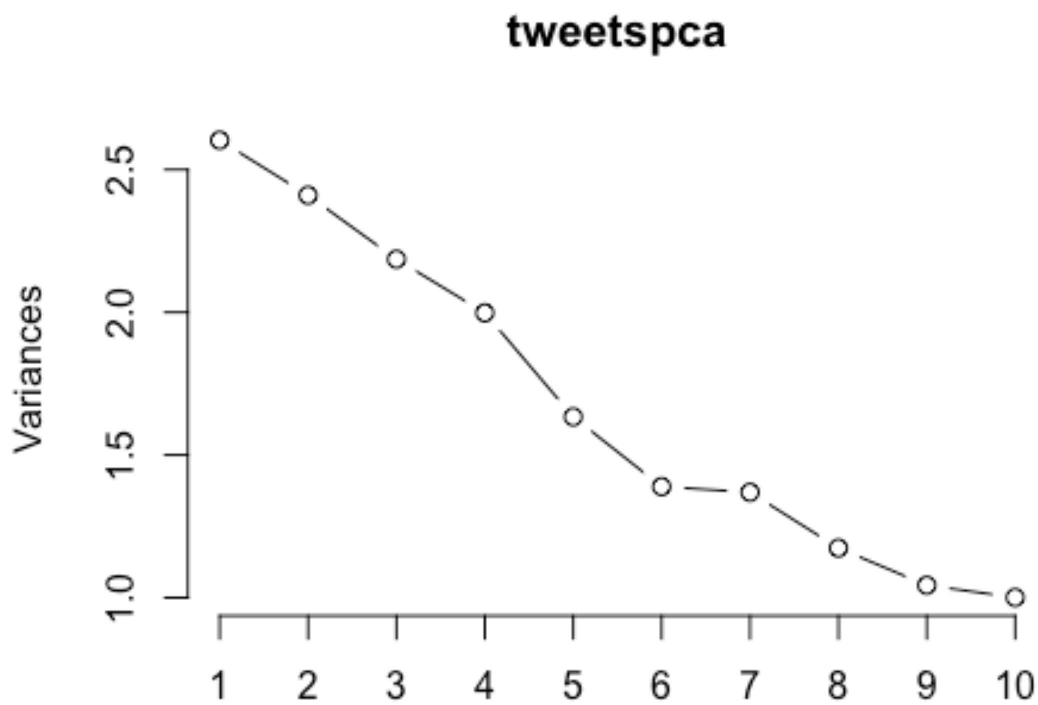
Market segmentation

First, I removed the four bad categories as specified in the assignment, as well as "photo sharing" which I do not want to count as a theme because it is too broad in terms of content.

```
tweets = read.csv("../data/social_marketing.csv", row.names=1)
tweets = tweets[,-c(1,4,5,35,36)]
```

Then, I ran PCA on the user profiles, which are proportions of how much of each user's tweets are in each of the categories.

```
profiles = tweets/rowSums(tweets)
tweetspca = prcomp(profiles, scale=TRUE)
scores = tweetspca$x
loadings = tweetspca$rotation
plot(tweetspca, type="lines")
```




```
summary(tweetspca)

## Importance of components:
##                           PC1     PC2     PC3     PC4    PC5
PC6
## Standard deviation     1.61336 1.55248 1.47853 1.41356 1.2781
1.17865
## Proportion of Variance 0.08397 0.07775 0.07052 0.06446 0.0527
0.04481
## Cumulative Proportion  0.08397 0.16171 0.23223 0.29669 0.3494
0.39420
##                           PC7     PC8     PC9    PC10    PC11
PC12
## Standard deviation     1.17021 1.08350 1.02156 1.00037 0.99584
0.98771
## Proportion of Variance 0.04417 0.03787 0.03366 0.03228 0.03199
0.03147
## Cumulative Proportion  0.43837 0.47624 0.50991 0.54219 0.57418
0.60565
##                          PC13    PC14    PC15    PC16    PC17
PC18
## Standard deviation     0.97774 0.96843 0.94089 0.92389 0.92117
0.8803
```

```
## Proportion of Variance 0.03084 0.03025 0.02856 0.02753 0.02737
0.0250
## Cumulative Proportion  0.63649 0.66674 0.69530 0.72283 0.75021
0.7752
##                           PC19    PC20    PC21    PC22    PC23
PC24
## Standard deviation     0.86177 0.85007 0.84312 0.82094 0.81053
0.78812
## Proportion of Variance 0.02396 0.02331 0.02293 0.02174 0.02119
0.02004
## Cumulative Proportion  0.79916 0.82247 0.84540 0.86714 0.88833
0.90837
##                          PC25    PC26    PC27    PC28    PC29   PC30
## Standard deviation     0.7854 0.76017 0.70007 0.65425 0.61186 0.5944
## Proportion of Variance 0.0199 0.01864 0.01581 0.01381 0.01208 0.0114
## Cumulative Proportion  0.9283 0.94691 0.96272 0.97652 0.98860 1.0000
##                             PC31
## Standard deviation     3.472e-15
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

The first eight principal components explain roughly half of the variance, and thus I report eight market segments as follows. Each market segment is described by the three highest-loading categories in each segment.

```
colnames(profiles)[tail(order(loadings[,1]),3)]

## [1] "parenting"     "religion"      "sports_fandom"

colnames(profiles)[tail(order(loadings[,2]),3)]

## [1] "food"      "parenting" "religion"

colnames(profiles)[tail(order(loadings[,3]),3)]

## [1] "cooking" "beauty"  "fashion"

colnames(profiles)[tail(order(loadings[,4]),3)]

## [1] "sports_playing" "online_gaming"  "college_uni"

colnames(profiles)[tail(order(loadings[,5]),3)]

## [1] "tv_film"        "current_events" "shopping"

colnames(profiles)[tail(order(loadings[,6]),3)]

## [1] "dating"    "travel"    "computers"

colnames(profiles)[tail(order(loadings[,7]),3)]

## [1] "current_events" "eco"            "shopping"

colnames(profiles)[tail(order(loadings[,8]),3)]
```

```
## [1] "home_and_garden" "school"          "dating"
```