Assignment1

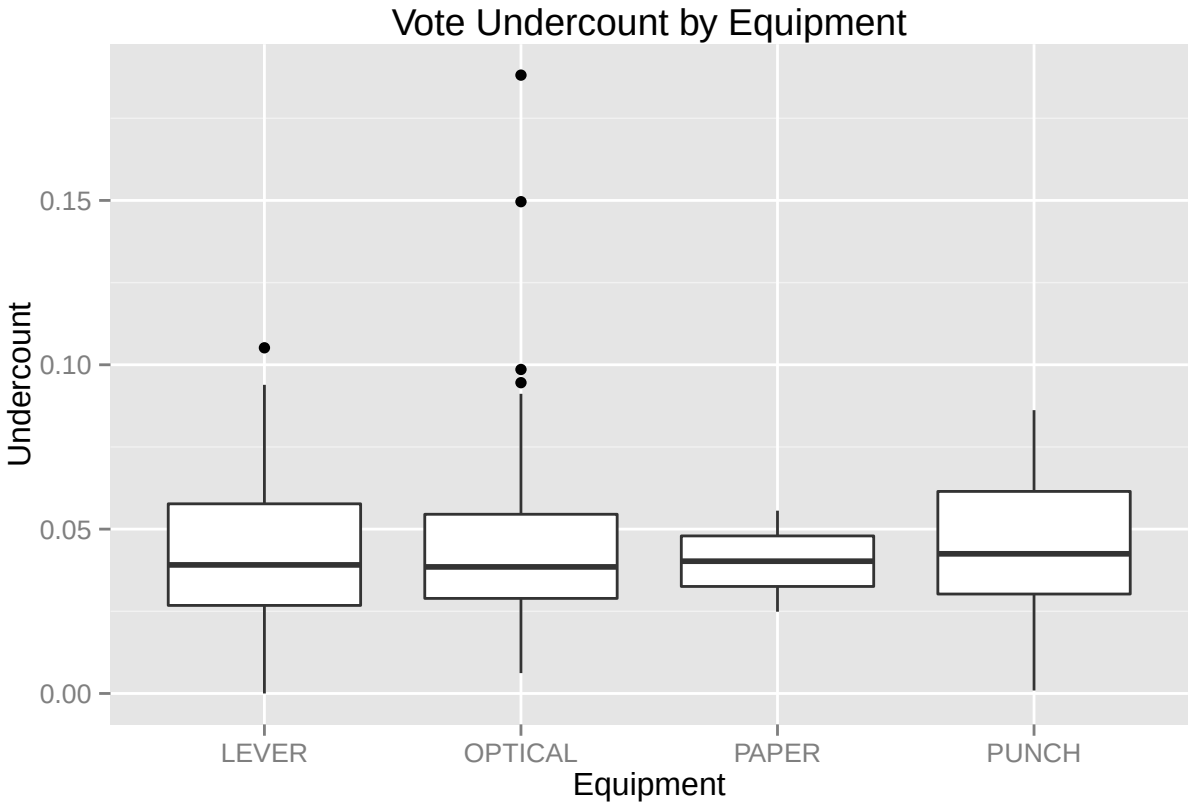
Greg Merchant

August 4, 2015

1 Data visualization

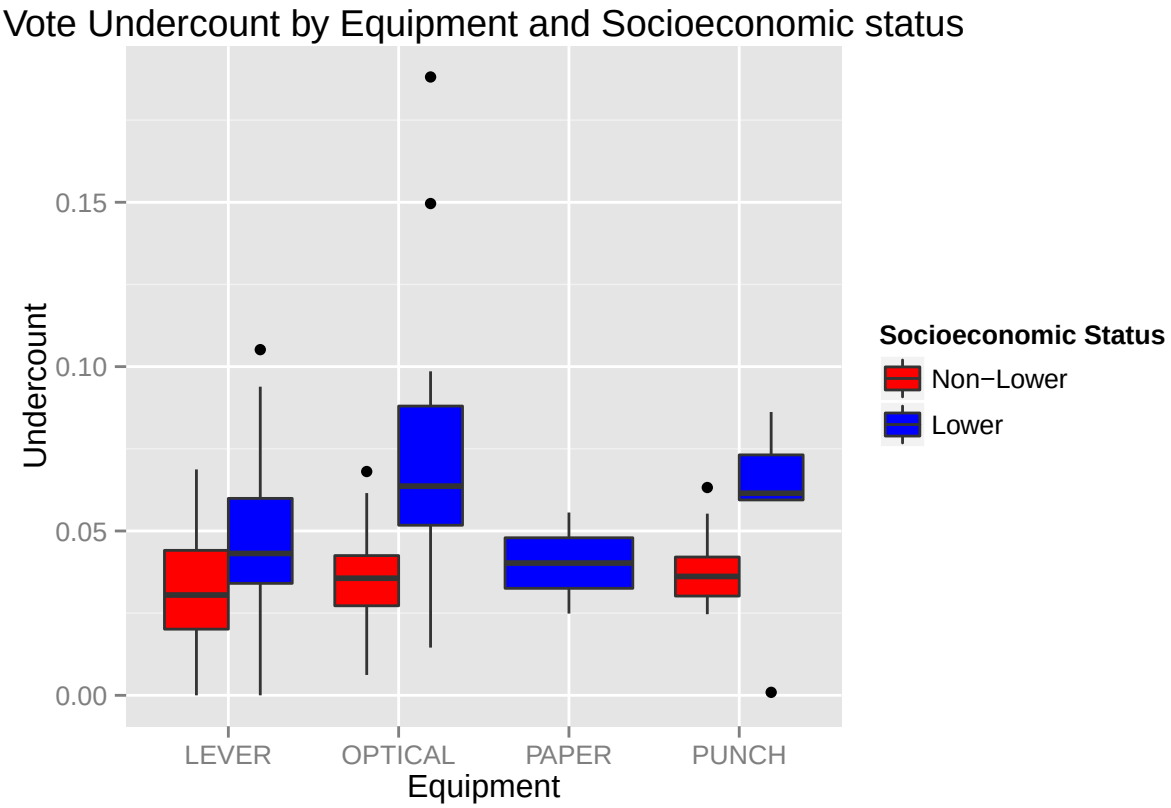
```
library(ggplot2)
library(mosaic)
library(fImport)
library(foreach)
require(gridExtra)

my_favorite_seed = 25
set.seed(my_favorite_seed)
```

```
voting = read.csv('../data/georgia2000.csv', header=TRUE)

voting$under_count = (voting$ballots - voting$votes)/voting$ballots
```



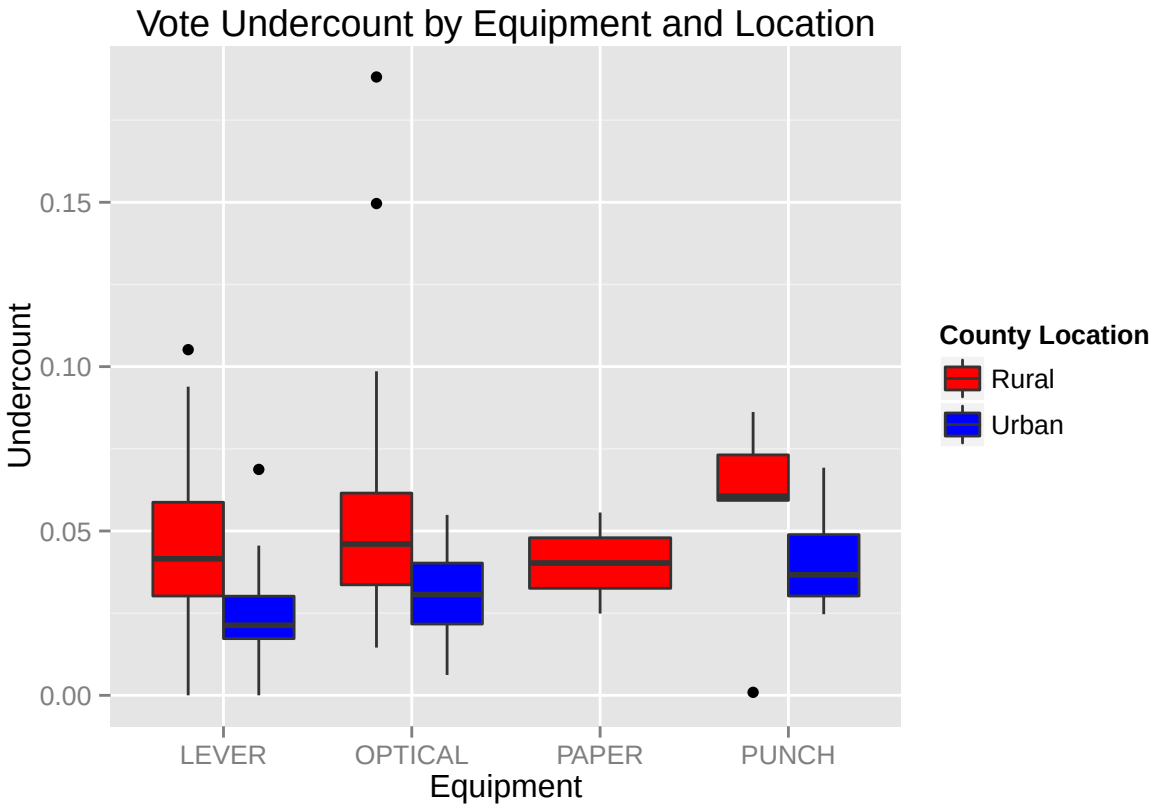
There appears to be a high variation of vote undercount in counties that utilized the optical voting system. There appear to be several outliers greater than 1.5*IQR for this equipment. Punch appears to have the highest 75 percentile point implying that it was the worst voting equipment at producing undercount.

Next we'll look at a boxplot of the equipment as well as its impact on counties with more than 25% of their population lives below 1.5 times the federal poverty line.



As displayed above- it does appear that the those counties with more than 25% of the population living at least 1.5 times below the federal poverty line are disproportionality affected. In all counties their IQR was at a higher percentage of voter undercount. Most tellingly the poorer counties that had utilized punch by and large were more undercounted than any equipment range from a non-poor county.

Next we'll investigate if the same affect occurs in urban or rural areas.



The plot looks very similar to the plot of lower socioeconomic counties vs. equipment on voter undercout. This would imply that several of those counties are both poor and rural (or at least not codified as "urban" in the data set).

Bootstrapping

```
# Import a few stocks
mystocks = c("SPY", "TLT", "LQD", "EEM", "VNQ")
myprices = yahooSeries(mystocks, from='2009-08-05', to='2015-08-05')

# A helper function for calculating percent returns from a Yahoo Series
# Source this to the console first, and then it will be available to use
# (Like importing a library)

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))
}
```

```
my_returns = YahooPricesToReturns(myprices)

#Rf rate taken from Federal Funds Weekly H.15 and utilized 10 year contstanat maturity
# for the week ended 7/31
rf_returns = (1+.0225)^(1/365)-1
rf_annual = 0.0225
#Compute the betas
lm_TLT = lm(my_returns[,2] ~ my_returns[,1])
lm_LQD = lm(my_returns[,3] ~ my_returns[,1])
lm_EEM = lm(my_returns[,4] ~ my_returns[,1])
lm_VNQ = lm(my_returns[,5] ~ my_returns[,1])
#display the betas (linear regressed to SPY)
coef(lm_TLT)
```

```
##     (Intercept) my_returns[, 1]
##    0.0006950643   -0.5304149683
```

```
coef(lm_LQD)
```

```
##     (Intercept) my_returns[, 1]
##    0.0002660935   -0.0339533130
```

```
coef(lm_EEM)
```

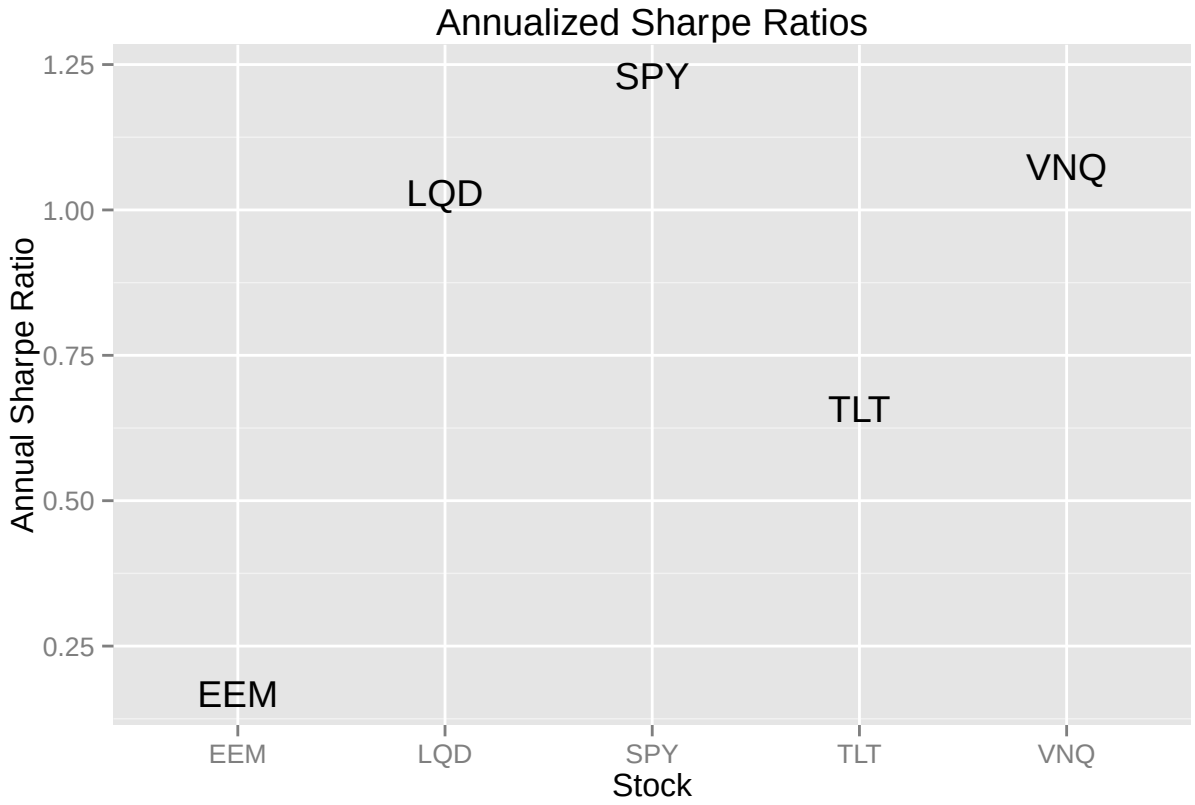
```
##     (Intercept) my_returns[, 1]
##   -0.0005959517    1.2607785826
```

```
coef(lm_VNQ)
```

```
##     (Intercept) my_returns[, 1]
##    4.910787e-05    1.081425e+00
```

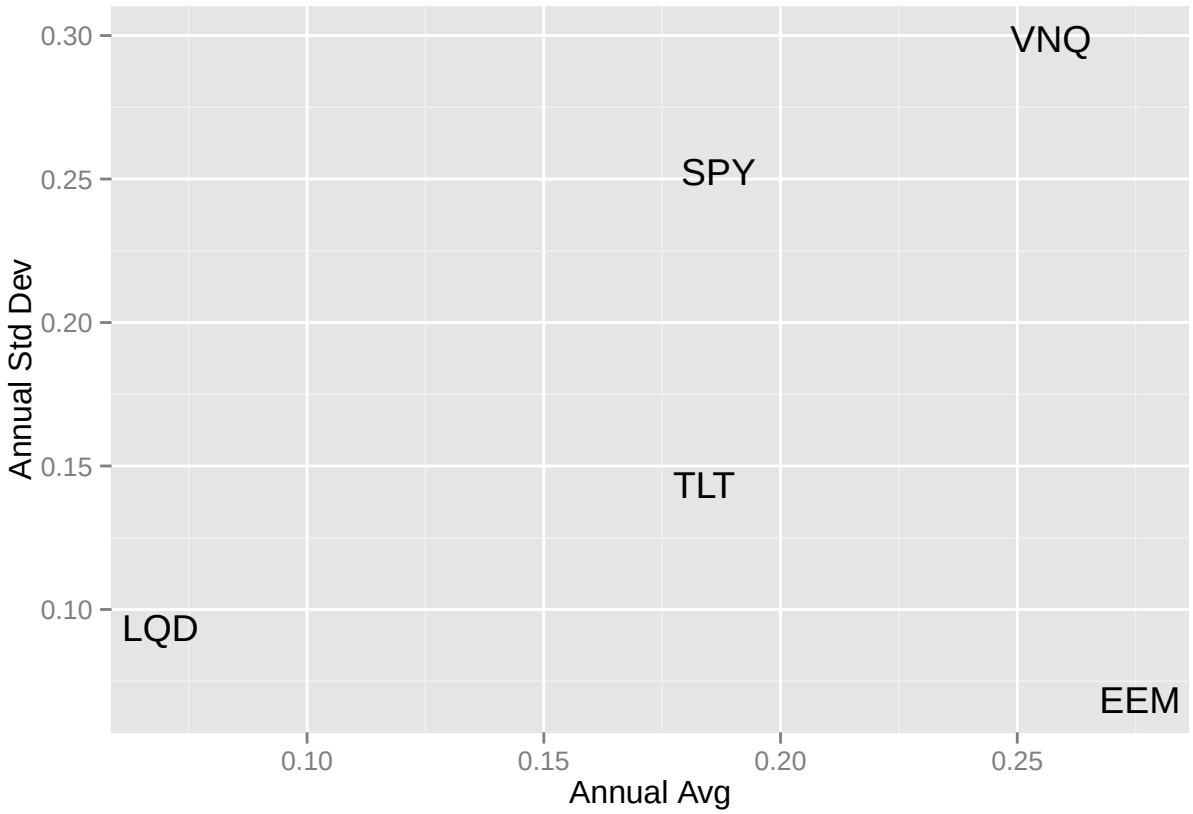
```
#Calculate the average daily returns, standard deviations, and sharpe ratio
##SPY
mu_SPY = mean(my_returns[,1])
sigma_SPY = sd(my_returns[,1])
sharpe_SPY = (mu_SPY-rf_returns)/sigma_SPY

##TLT
mu_TLT = mean(my_returns[,2])
sigma_TLT = sd(my_returns[,2])
sharpe_TLT = (mu_TLT-rf_returns)/sigma_TLT

##LQD
mu_LQD = mean(my_returns[,3])
sigma_LQD = sd(my_returns[,3])
sharpe_LQD = (mu_LQD-rf_returns)/sigma_LQD

##EEM
mu_EEM = mean(my_returns[,4])
```

```
sigma_EEM = sd(my_returns[,4])
sharpe_EEM = (mu_EEM-rf_returns)/sigma_EEM

##VNQ
mu_VNQ = mean(my_returns[,5])
sigma_VNQ = sd(my_returns[,5])
sharpe_VNQ = (mu_VNQ-rf_returns)/sigma_VNQ

#create a list of mus, sigmas, sharpes, and index and pass into a dataframe.
mus <- c(mu_SPY, mu_TLT, mu_LQD, mu_EEM, mu_VNQ)
mus_annual <- (mus + 1)^365 -1
sigmas <- c(sigma_SPY, sigma_TLT, sigma_LQD, sigma_EEM, sigma_VNQ)
sigmas_annual <- sigmas*sqrt(365)

sharpes <- c(sharpe_SPY, sharpe_TLT, sharpe_LQD, sharpe_EEM, sharpe_VNQ)
sharpes_annual <- (mus_annual-rf_annual)/sigmas_annual
index_names <- c('SPY', 'TLT', 'LQD', 'EEM', 'VNQ')
risk_ret <- data.frame(mus, sigmas, sharpes)
rownames(risk_ret) <- index_names

risk_ret_annual <- data.frame(mus_annual, sigmas_annual, sharpes_annual)
rownames(risk_ret_annual) <- index_names
```



After evaluating the Sharpe Ratios of all of the data points- plotted each stock with its corresponding Sharpe ration. As shown the highest Sharpe ratio stock was actually the market (SPY), followed by Real estate (VNQ), and the corporate grade bonds (LQD).

Of the 5 ETFs listed The following plot shows the risk-reward tradeoff in terms of daily average return and daily standard deviation.



As shown, VNQ had the highest average return, but had the highest standard deviation. EEM actually had the lowest return, but it had one of the highest standard deviations. EEM is a suboptimal ETF to be held. It has a low average return, a high standard deviation and has a beta of 0.85 to the market- so it doesn't even provide a large diversification benefit. For the safety stock I would prefer LQD because the standard deviation is much lower than TLT. TLT's standard deviation is about in line with the market, but with a lower return. It's true benefit is in the near 0 correlation with the market. As such I will include it in some of the portfolios.

Portfolio 1- Even Split

```
n_days= 20
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(my_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        wealthtracker[today] = totalwealth
        holdings = weights * totalwealth
    }
    wealthtracker
}
```

```
profit1 = qplot(sim1[,n_days]-100000, binwidth=30) +
        geom_bar( fill = "white",color="black") +
        ggtitle("Equal Weight Portfolio Profit") +
        xlab("Profit") + ylab('') + xlim(-20000,30000)
portfolio1_var = quantile(sim1[,n_days], 0.05) - 100000
print(portfolio1_var)
```

```
##        5%
## -3936.572
```

The equal weighted portfolio has a VAR of roughly $4,000 ($3,937 exactly). The histogram looks relatively normally distributed, which I would expect from a diversified portfolio.

Portfolio 2- Safe

```
sim2 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.5, 0.2, 0.3, 0, 0)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(my_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        wealthtracker[today] = totalwealth
        holdings = weights * totalwealth
    }
    wealthtracker
}

profit2 = qplot(sim2[,n_days]-100000, binwidth=30) +
        geom_bar( fill = "white",color="black") +
        ggtitle("Safe Weight Portfolio Profit") +
        xlab("Profit") + ylab('') + xlim(-20000,30000)

portfolio2_var = quantile(sim2[,n_days], 0.05) - 100000
print(portfolio2_var)
```

```
##        5%
## -2424.837
```

This portfolio I weighted with approximately 50% in the market, and 20% in treasuries, and 30% in corporate bonds. I thought that about a 50-50 split would be relatively safe, and in fact it was. The VAR is about $1500 lower than the equal weight portfolio. The histograms also looked normal, but were generally less spread out than the equal weight portfolio. The center of the distribution to actually be slightly further to the right than the equal weight portfolio- which is impressive since it has a lower VAR and an overall tighter distribution.

Portfolio 3- Aggressive

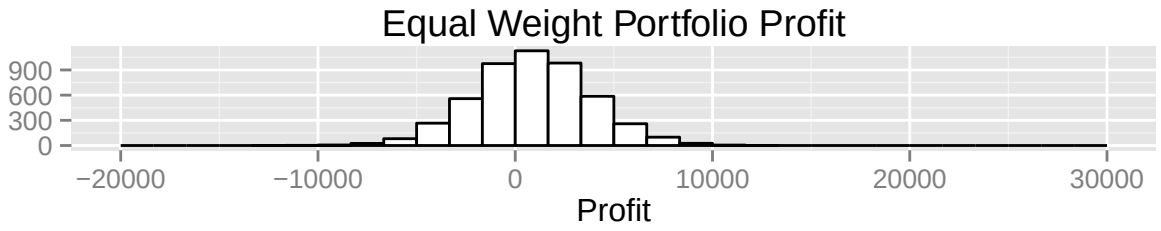
```
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.2, 0, 0, 0.4, 0.4)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(my_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        wealthtracker[today] = totalwealth
        holdings = weights * totalwealth
    }
    wealthtracker
}

profit3 = qplot(sim3[,n_days]-100000, binwidth=30) +
        geom_bar( fill = "white",color="black") +
        ggtitle("Aggressive Weight Portfolio Profit") +
        xlab("Profit")+ylab('')+xlim(-20000,30000)

portfolio3_var = quantile(sim3[,n_days], 0.05) - 100000
print(portfolio3_var)
```

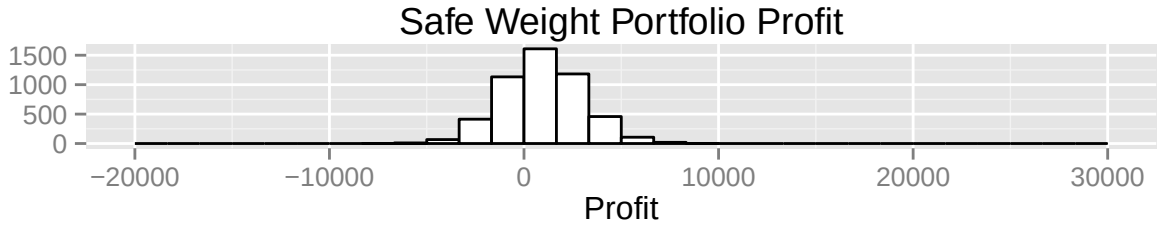
```
##        5%
## -7714.342
```

For the aggressive portfolio I weighted 20% in SPY, 40% in EEM and 40% in VNQ. I thought that this allocation was very aggressive and would result in a higher risk higher reward scenario. The average return is actually close to 0 (and is sub-optimal to both the other portfolios), and the distribution is much more spread out. Additionally the VAR shoots up to $7,714. This is likely due to the heavy weight given to EEM (which has a lower average return and a high standard deviation).

The following graphs show a histogram of the profits forma simulated year of 20 trading days. The spreads of the three show the tradeoff of risk and return. Portfolio one has a decent allocation of histogram counts, with a most-likely profit slightly above 0. Portfolio two also has a most likely profit slightly above 0, but has a lower upside potential (and a lower downside potential). Portfolio 3 has a most likely profit centered at 0, but has a wide range of outcomes both on the positive and negative side.

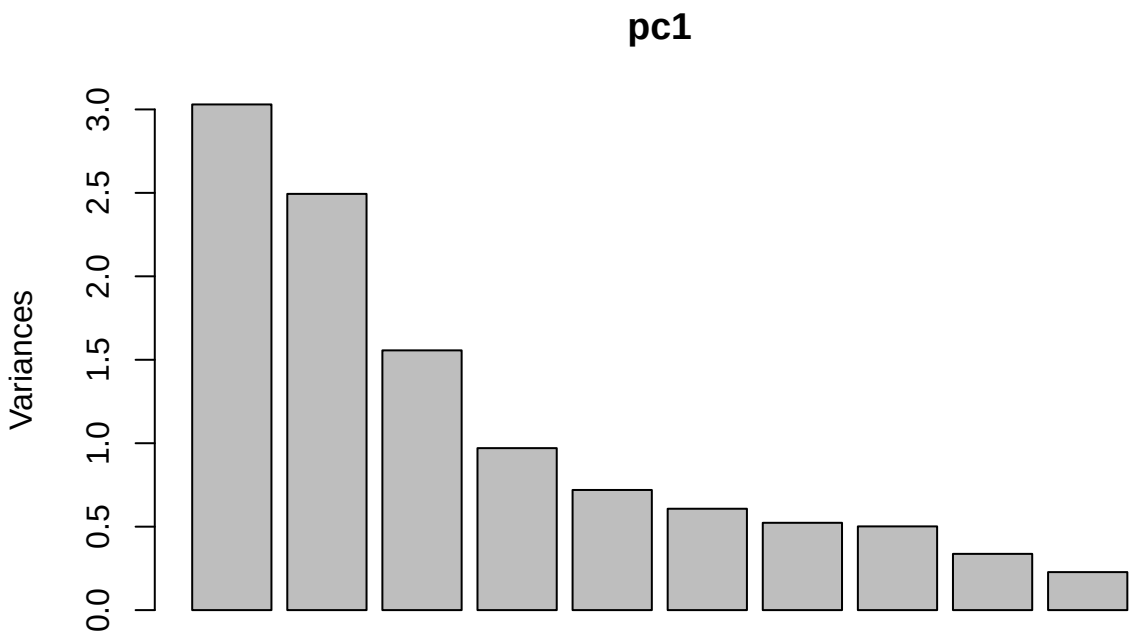
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



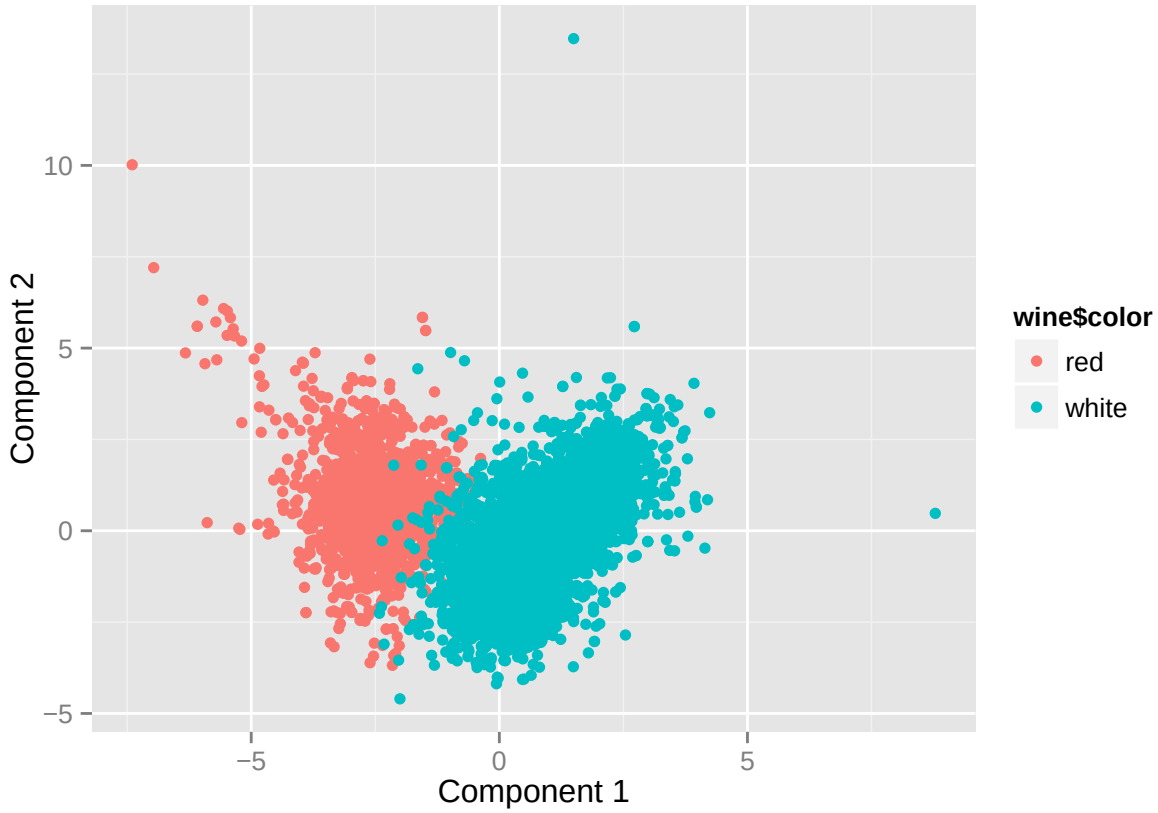


Clustering and PCA

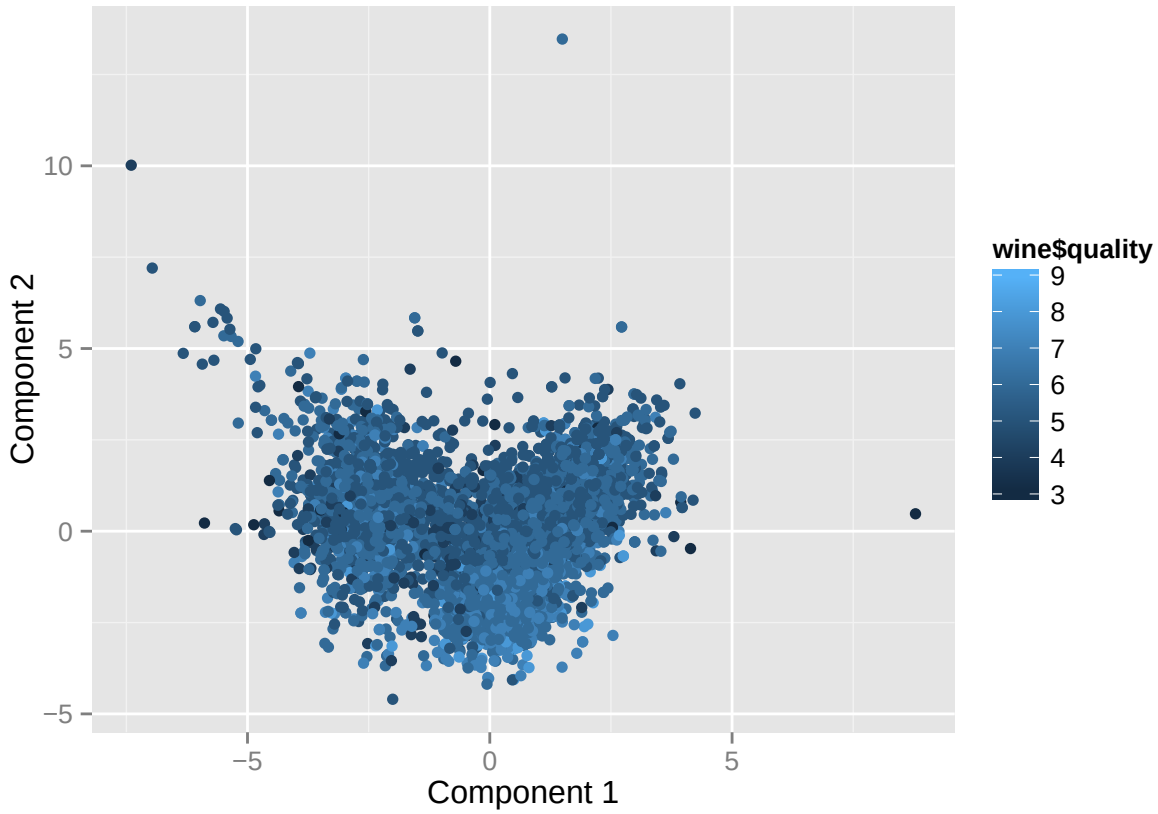
```
wine = read.csv('../data/wine.csv', header=TRUE)

wine_unsup <- wine[,-c(12,13)]
wine_centered <- scale(wine_unsup, center=TRUE, scale=FALSE)

pc1 = prcomp(wine_unsup, scale. =TRUE)

plot(pc1)
```




```
loadings = pc1$rotation
scores = pc1$x
```



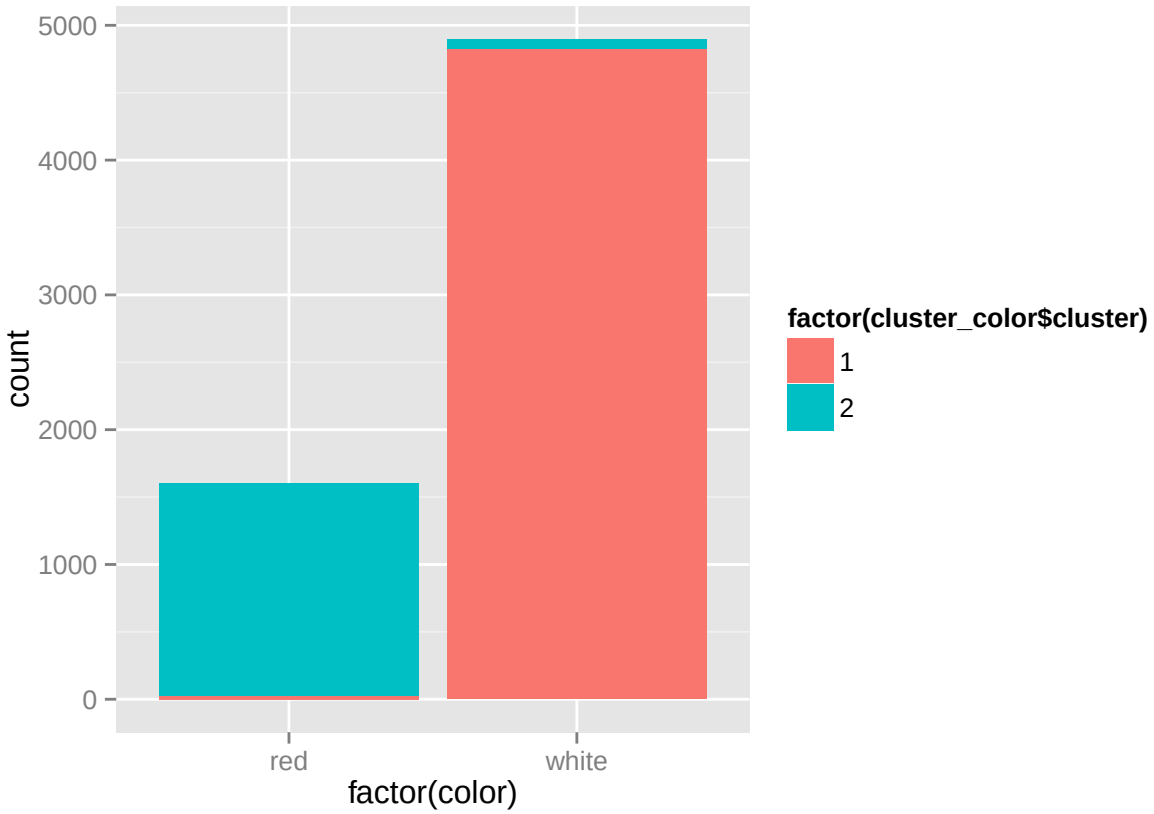
As shown above, PCA is particularly good at determining whether wine is red or white.



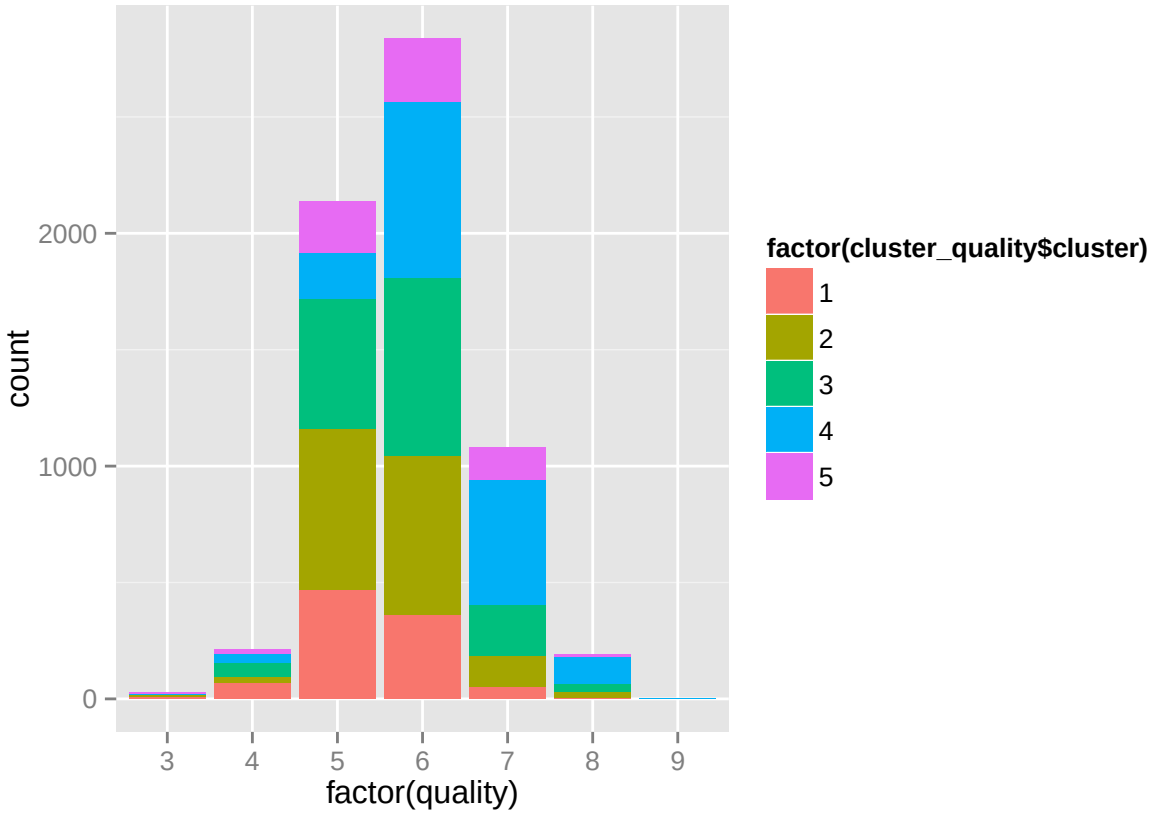
As shown above, PCA isn't particularly good at determining the quality of the wine.

Next let's take a look at Kmeans clustering.

```
wine_unsup_scaled <- scale(wine_unsup, center=TRUE, scale=TRUE)
cluster_color <- kmeans(wine_unsup_scaled, centers=2, nstart=500)
cluster_quality <- kmeans(wine_unsup_scaled, centers=5, nstart=500)
```



As shown above Kmeans accurately classified wines as either white or red. There was a very small amount of error observable.



K-means with a cluster of 5 didn't show any meaningful clusters around quality. This makes intuitive sense.

Both PCA and Kmeans accurately predicted whether a wine was red or white, both struggled with mapping the quality. This makes intuitive sense to me because a wine often isn't judged on it's chemical traits for quality. Often times unseen factors can impact a persons perception of quality. Vintage, price, vineyard etc. Even if the taste was a blind test (which would be difficult with that many wines) it would be difficult to characterize some other factors that affect taste and quality rating. Differences in processes may not yield chemical differences, but could still impact taste and as result rating.

Between the two, both appear to accurately characterize red vs. white, and appear equally bad at assessing quality. As a result I would lean towards Kmeans simply on an interpretability factor. Intuitively kmeans seems more simple than PCA.

Market segmentation

```
survey = read.csv('../data/social_marketing.csv',header=TRUE)
#sclae the survey data dropping username, chatter, uncategorized, adult and spam
Z = survey[,-c(1,2,6,36,37)]
Zscale = scale(Z, center=TRUE, scale=TRUE)

#create a cluster with 5 centers of the scaled data (Zscale)
cluster_5 = kmeans(Zscale, centers=5, nstart = 250)
#get the index of all members of each cluster
clus1 = which(cluster_5$cluster == 1)
clus2 = which(cluster_5$cluster == 2)
clus3 = which(cluster_5$cluster == 3)
clus4 = which(cluster_5$cluster == 4)
clus5 = which(cluster_5$cluster == 5)

#Going to get the mean zscore for each topic
means_all = colMeans(Zscale)
means1 = colMeans(Zscale[clus1,])
means2 = colMeans(Zscale[clus2,])
means3 = colMeans(Zscale[clus3,])
means4 = colMeans(Zscale[clus4,])
means5 = colMeans(Zscale[clus5,])

#turn numerics into vectors and transpose
means1t = t(as.vector(means1))
means2t = t(as.vector(means2))
means3t = t(as.vector(means3))
means4t = t(as.vector(means4))
means5t = t(as.vector(means5))

#create a data frame of the aveage z score on each topic of each cluster
topics = data.frame(rbind(means1t, means2t, means3t, means4t, means5t))
#add the column names back in
colnames(topics) <- names(Z)

#drop topics without a z score above 1 standard deviation
topics <- topics[,c(2,3,5,6,7,8,11,14,17,19,21,23,25,26,27,29,30,31)]
```

After processing all of the data and getting the Zscores of tweets about specific topics we learned the following about the 5 clusters.

Cluster 1 - The religous, family-oriented, foodie, sports-fan Cluster

```
print(topics[1,])
```

```
##     travel photo_sharing sports_fandom politics      food     family
## 1 1.739988   -0.06432708     0.1859279 2.327682 0.0260945 0.05199343
##       news health_nutrition   cooking computers  outdoors automotive
## 1 1.911851       -0.2014448 -0.213231  1.540729 0.1169822   1.091946
##      religion     beauty   parenting      school personal_fitness
## 1 -0.04249063 -0.1788191 0.004639843 -0.04444393       -0.1909396
##      fashion
## 1 -0.1724965
```

As evidenced by the above Zscores, users in this cluster seem to be family-oriented. They tweet about their family, religions, school, food, parenting and sports the most. Could be a subset of "weekend warriors" that NutrientH20 wants to target to try it's new nutrient-filled water. Additionally, family values appear to be important- so targeting the wholesomeness of NutrientH20's product may be most beneficial strategy to these users.

Cluster 2 - The informed, automotive-loving, world travelers

```
print(topics[2,])
```

```
##        travel photo_sharing sports_fandom   politics       food     family
## 2 -0.03826632      1.229675    -0.2052086 -0.1316701 -0.1759344 0.04360903
##          news health_nutrition  cooking  computers   outdoors automotive
## 2 -0.07757994      -0.07483581 2.502315 0.07689752 0.03083788 0.05505111
##     religion   beauty   parenting    school personal_fitness  fashion
## 2 -0.1284606 2.330876 -0.09065778 0.1707048      -0.04989235 2.415136
```

As evidenced by the above zscores, users in this cluster seems to be product-loving globe trotters. They tweet about traveling, the news/world events, computers/technology, and automotives. They keep up to date on the latest trends and products and love testing the out as they satisfy their wanderlust. Advertisements about a sense of discovery and adventure or focusing on the features will attract these users to tweet about NutrientH20.

Cluster 3 - Cooking, fashionable, social butterfly

```
print(topics[3,])
```

```
##        travel photo_sharing sports_fandom   politics     food   family
## 3 -0.09695717   -0.02621943       1.98073 -0.2027484 1.769888 1.430606
##          news health_nutrition    cooking  computers    outdoors
```

```
## 3 -0.07545654       -0.1558357 -0.1146945 0.07883704 -0.06089195
##   automotive religion    beauty parenting   school personal_fitness
## 3  0.1608093 2.167868 0.2880661  2.046731 1.623061       -0.1091672
##      fashion
## 3 0.01229769
```

As evidenced by the above zscores, users in this cluster seems to be fans of aesthetically pleasing things. They tweet about the latest fashion trends, the latest and greatest beauty products and supplies, and they love sharing all the wonderful things they've found and photographed. These users tend to show what they love as much as they talk about. NutrientH20 may want to target some well designed advertisements to these users so that they'll pass them on to their followers.

Cluster 4 - Fit, healthy, outdoorsmen and women

```
print(topics[4,])
```

```
##       travel photo_sharing sports_fandom  politics      food      family
## 4 -0.1504449   0.004279626    -0.2044569 -0.180972 0.4039272 -0.06339472
##          news health_nutrition   cooking   computers outdoors automotive
## 4 -0.05089773         2.068883 0.3670649 -0.08504485 1.580729 -0.1171052
##     religion     beauty  parenting     school personal_fitness    fashion
## 4 -0.1774467 -0.2138177 -0.1121847 -0.1508712         2.034182 -0.1118489
```

As evidenced by the above zscores, users in this cluster tweet about being healthy and fit. They tweet about health/nutrition, the outdoors and personal fitness significantly more than the average. These users are the "health-conscious" cluster. NutrientH20 should focus marketing activity talking to the health benefits and healthy-ingredient mix to this subset of users.

Cluster 5- Followers and all-purpose tweeple

```
print(topics[5,])
```

```
##       travel photo_sharing sports_fandom   politics       food     family
## 5 -0.2064105    -0.1423618     -0.284165 -0.2579974 -0.3473547 -0.2336028
##         news health_nutrition    cooking  computers   outdoors automotive
## 5 -0.2494244       -0.3344641 -0.3367019 -0.2328336 -0.3159808 -0.1711924
##   religion     beauty  parenting     school personal_fitness    fashion
## 5 -0.29589 -0.2736231 -0.3004796 -0.2498951       -0.3400587 -0.2600028
```

As evidenced by the above zscores, these users don't tweet about anything more than average. They likely follow several accounts, but hardly ever tweet. Or if they do tweet it tends to not be about one particular subject.