STA 380 Homework 1: Barton, Jace

Jace Barton

August 5, 2015

To begin, I load the libraries I will need throughout my analysis.

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.0.3

##   
## 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

library(mosaic)

## Warning: package 'mosaic' was built under R version 3.0.3

## Loading required package: car

## Warning: package 'car' was built under R version 3.0.3

## Loading required package: lattice  
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.0.3

##   
## Attaching package: 'mosaic'  
##   
## The following object is masked from 'package:car':  
##   
## logit  
##   
## The following objects are masked from 'package:dplyr':  
##   
## do, tally  
##   
## 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)  
library(fImport)

## Warning: package 'fImport' was built under R version 3.0.3

## Loading required package: timeDate

## Warning: package 'timeDate' was built under R version 3.0.3

## Loading required package: timeSeries

## Warning: package 'timeSeries' was built under R version 3.0.3

library(foreach)

## Warning: package 'foreach' was built under R version 3.0.3

library(RCurl)

## Warning: package 'RCurl' was built under R version 3.0.3

## Loading required package: bitops

I will also be performing random draws. So the reader can reproduce my results, I will set the seed as value 722.

set.seed(722)

Now, to the analysis!

# Exploratory Analysis

## County Voting in Georgia for 2000 Election

To begin, I will import the data set, then view a summary of the data.

GeorgiaURLString = getURL("https://raw.githubusercontent.com/jacebarton/STA380/master/data/georgia2000.csv", ssl.verifypeer=0L, followlocation = 1L)  
Georgia = read.csv(text=GeorgiaURLString)  
summary(Georgia)

## county ballots votes equip   
## APPLING : 1 Min. : 881 Min. : 832 LEVER :74   
## ATKINSON: 1 1st Qu.: 3694 1st Qu.: 3506 OPTICAL:66   
## BACON : 1 Median : 6712 Median : 6299 PAPER : 2   
## BAKER : 1 Mean : 16927 Mean : 16331 PUNCH :17   
## BALDWIN : 1 3rd Qu.: 12251 3rd Qu.: 11846   
## BANKS : 1 Max. :280975 Max. :263211   
## (Other) :153   
## poor urban atlanta perAA   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.1115   
## Median :0.0000 Median :0.0000 Median :0.00000 Median :0.2330   
## Mean :0.4528 Mean :0.2642 Mean :0.09434 Mean :0.2430   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.00000 3rd Qu.:0.3480   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :0.7650   
##   
## gore bush   
## Min. : 249 Min. : 271   
## 1st Qu.: 1386 1st Qu.: 1804   
## Median : 2326 Median : 3597   
## Mean : 7020 Mean : 8929   
## 3rd Qu.: 4430 3rd Qu.: 7468   
## Max. :154509 Max. :140494   
##

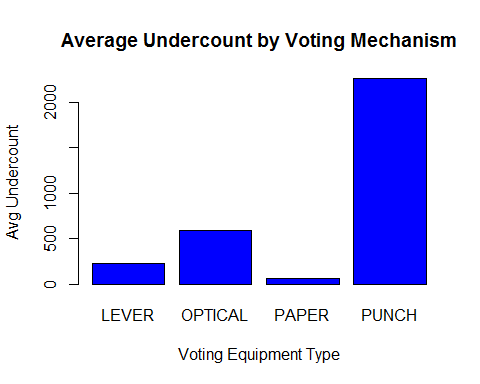
I now want to calculate the undercount for each county and append that to the dataframe. I then want to look at a pivot table of the undercount by machine type.

Georgia$undercount = abs(Georgia$votes - Georgia$ballots)  
UndercountByEquip = group\_by(Georgia, equip)  
UndercountByEquip = summarise(UndercountByEquip, AvgUndercount = mean(Georgia$undercount), SumBallots = sum(Georgia$ballots), SumVotes = sum(Georgia$votes), BallotConversionRate = sum(Georgia$votes)/sum(Georgia$ballots))  
UndercountByEquip

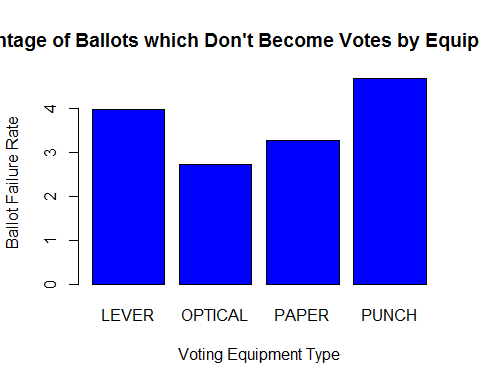
## Source: local data frame [4 x 5]  
##   
## equip AvgUndercount SumBallots SumVotes BallotConversionRate  
## 1 LEVER 229.9459 427780 410764 0.9602225  
## 2 OPTICAL 592.2727 1436159 1397069 0.9727816  
## 3 PAPER 56.5000 3454 3341 0.9672843  
## 4 PUNCH 2262.4706 823921 785459 0.9533183

It appears that the punch equipment type is vastly undercounting. It is the second most used equipment type, but has the lowest conversion rate of ballots to votes at 95%. I want to look at this information graphically though to confirm my suspicions.

barplot(UndercountByEquip$AvgUndercount, names = UndercountByEquip$equip, ylab="Avg Undercount", xlab="Voting Equipment Type", col=4, main="Average Undercount by Voting Mechanism")



barplot((1-UndercountByEquip$BallotConversionRate)\*100, names = UndercountByEquip$equip, ylab="Ballot Failure Rate", xlab="Voting Equipment Type", col=4, main="Percentage of Ballots which Don't Become Votes by Equipment Type")



The punch equipment is the biggest offender. But where are the punch machines located? Are they equally spread across Georgia? Or are they located in areas which are poorer? Or have a higher percentage of minorities?

I begin by looking at a crosstab of county type (poor or rich) versus type of machine.

PoorVsEquip = xtabs(~poor + equip, data=Georgia)  
PoorVsEquip

## equip  
## poor LEVER OPTICAL PAPER PUNCH  
## 0 29 48 0 10  
## 1 45 18 2 7

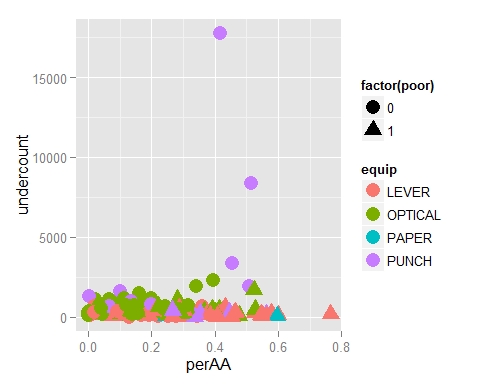
PoorVsEquipProp = prop.table(PoorVsEquip, margin=2)  
PoorVsEquipProp

## equip  
## poor LEVER OPTICAL PAPER PUNCH  
## 0 0.3918919 0.7272727 0.0000000 0.5882353  
## 1 0.6081081 0.2727273 1.0000000 0.4117647

59% of punch machines were located in non-poor counties while 41% were located in poor counties. Thus, from this I can say non-poor counties were more likely to have their votes undercounted. But is that the full story?

The following plot graphs the percentage of the population in a county which is African American on the x-axis against the undercount in that county on the y-axis. The points are color coded to reflect the voting equipment used in the county. Finally, the poor counties are representing as triangles, while the non-poor counties are represented as octagons.

ggplot(Georgia, aes(x=perAA, y=undercount, color = equip, shape = factor(poor))) + geom\_point (size=5)



I immediately observe that the three counties with the most undercounted ballots were counties with a substantial African American population (greater than 40%). Those counties were also using punch cards. I also note that the right-most counties in the graph (the counties which are most substantially African-American in makeup) are poor.

In conclusion, punch machines see a higher rate of undercounting compared to other machine types. While the impact is spread between non-poor and poor counties about equally, minority counties are more likely to see substantial undercount than non-minority counties.

# Bootstrapping

## Stock market portfolios and levels of risk and return

The goal of this exercise is to see the levels of risk and return across varying compositions of different asset types. The asset types in question are domestic equities, Treasury bonds, corporate bonds, Emerging-market equities, and real estate. These five classes are represented in order by the following Exchange Traded Funds (ETFs). *SPY* TLT *LQD* EEM \*VNQ

I first want to gather five years worth of returns on these five assets. This is accomplished below.

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

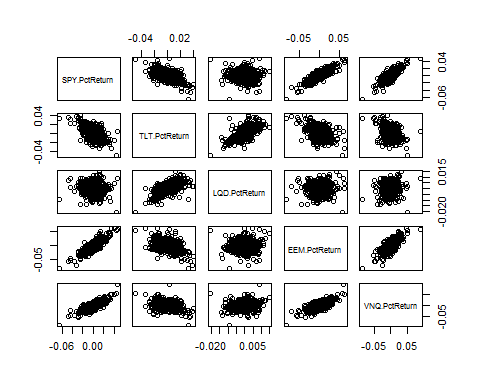
## SPY.Open SPY.High SPY.Low SPY.Close   
## Min. :104.9 Min. :106.0 Min. :104.3 Min. :105.2   
## 1st Qu.:131.7 1st Qu.:132.5 1st Qu.:131.0 1st Qu.:131.8   
## Median :149.9 Median :150.9 Median :149.5 Median :150.1   
## Mean :158.5 Mean :159.3 Mean :157.7 Mean :158.5   
## 3rd Qu.:188.0 3rd Qu.:188.6 3rd Qu.:187.1 3rd Qu.:187.9   
## Max. :213.2 Max. :213.8 Max. :212.9 Max. :213.5   
## SPY.Volume SPY.Adj.Close TLT.Open TLT.High   
## Min. : 42963400 Min. : 95.03 Min. : 88.69 Min. : 89.14   
## 1st Qu.: 98758950 1st Qu.:121.34 1st Qu.:104.94 1st Qu.:105.54   
## Median :131278200 Median :142.94 Median :115.25 Median :115.75   
## Mean :146760627 Mean :151.67 Mean :112.77 Mean :113.33   
## 3rd Qu.:173130100 3rd Qu.:183.42 3rd Qu.:120.80 3rd Qu.:121.39   
## Max. :717828700 Max. :212.62 Max. :136.70 Max. :138.50   
## TLT.Low TLT.Close TLT.Volume TLT.Adj.Close   
## Min. : 88.14 Min. : 88.19 Min. : 987200 Min. : 77.11   
## 1st Qu.:104.48 1st Qu.:105.11 1st Qu.: 5867950 1st Qu.: 98.49   
## Median :114.73 Median :115.29 Median : 7742500 Median :107.27   
## Mean :112.24 Mean :112.79 Mean : 8644372 Mean :105.25   
## 3rd Qu.:120.29 3rd Qu.:120.84 3rd Qu.:10197050 3rd Qu.:114.67   
## Max. :136.66 Max. :138.28 Max. :46221000 Max. :136.27   
## LQD.Open LQD.High LQD.Low LQD.Close   
## Min. :106.7 Min. :107.3 Min. :106.3 Min. :106.8   
## 1st Qu.:112.7 1st Qu.:113.0 1st Qu.:112.4 1st Qu.:112.7   
## Median :116.1 Median :116.3 Median :115.9 Median :116.2   
## Mean :115.9 Mean :116.2 Mean :115.7 Mean :115.9   
## 3rd Qu.:119.4 3rd Qu.:119.6 3rd Qu.:119.2 3rd Qu.:119.4   
## Max. :123.5 Max. :123.9 Max. :123.4 Max. :123.9   
## LQD.Volume LQD.Adj.Close EEM.Open EEM.High   
## Min. : 233400 Min. : 89.53 Min. :33.93 Min. :34.94   
## 1st Qu.: 1054650 1st Qu.: 98.21 1st Qu.:39.88 1st Qu.:40.18   
## Median : 1585400 Median :108.08 Median :41.79 Median :42.01   
## Mean : 1809465 Mean :106.13 Mean :42.10 Mean :42.33   
## 3rd Qu.: 2241350 3rd Qu.:113.33 3rd Qu.:43.85 3rd Qu.:44.02   
## Max. :10863900 Max. :121.63 Max. :50.27 Max. :50.43   
## EEM.Low EEM.Close EEM.Volume EEM.Adj.Close   
## Min. :33.42 Min. :34.36 Min. : 18409100 Min. :31.74   
## 1st Qu.:39.62 1st Qu.:39.88 1st Qu.: 42995550 1st Qu.:38.24   
## Median :41.52 Median :41.79 Median : 53611300 Median :40.01   
## Mean :41.83 Mean :42.10 Mean : 58088010 Mean :39.94   
## 3rd Qu.:43.66 3rd Qu.:43.86 3rd Qu.: 68977100 3rd Qu.:41.82   
## Max. :49.94 Max. :50.20 Max. :191406700 Max. :45.91   
## VNQ.Open VNQ.High VNQ.Low VNQ.Close   
## Min. :47.79 Min. :49.34 Min. :47.10 Min. :48.47   
## 1st Qu.:59.84 1st Qu.:60.26 1st Qu.:59.33 1st Qu.:59.97   
## Median :66.16 Median :66.51 Median :65.72 Median :66.20   
## Mean :66.85 Mean :67.27 Mean :66.37 Mean :66.84   
## 3rd Qu.:73.85 3rd Qu.:74.30 3rd Qu.:73.50 3rd Qu.:73.88   
## Max. :88.83 Max. :89.27 Max. :88.30 Max. :88.65   
## VNQ.Volume VNQ.Adj.Close   
## Min. : 661100 Min. :40.57   
## 1st Qu.: 1839800 1st Qu.:51.33   
## Median : 2478900 Median :60.68   
## Mean : 2849726 Mean :61.20   
## 3rd Qu.: 3410850 3rd Qu.:69.81   
## Max. :11383300 Max. :87.24

As seen, lots of information about the ETFs is returned by this data grab.However, for this theoretical exercise I am only interested in the returns of the assets. Below, I utilize a helper function presented in class by Dr. Scott to obtain the required returns.

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))  
}

I will now calculate the returns, look at the scatter plots of each return type against each other return type, and view summary statistics of the returns.

ETFReturns = YahooPricesToReturns(ETFPrices)  
pairs(ETFReturns)



summary(ETFReturns)

## SPY.PctReturn TLT.PctReturn LQD.PctReturn   
## Min. :-0.0651232 Min. :-0.0504495 Min. :-0.0205232   
## 1st Qu.:-0.0036944 1st Qu.:-0.0057510 1st Qu.:-0.0018105   
## Median : 0.0007426 Median : 0.0007862 Median : 0.0005005   
## Mean : 0.0006210 Mean : 0.0003404 Mean : 0.0002036   
## 3rd Qu.: 0.0053416 3rd Qu.: 0.0065291 3rd Qu.: 0.0022682   
## 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 : 6.504e-05 Mean : 0.0005391   
## 3rd Qu.: 7.747e-03 3rd Qu.: 0.0066850   
## Max. : 6.240e-02 Max. : 0.0910393

What is the spread of these returns? I will look at the Standard Deviation (SD) and Interquartile Range (IQR) of each asset.

ReturnSDs = rep(0,5)  
  
for (i in 1:length(ReturnSDs)){  
 ReturnSDs[i] = sd(ETFReturns[,i])  
}  
  
ReturnIQRs = rep(0,5)  
  
for (i in 1:length(ReturnIQRs)){  
 ReturnIQRs[i] = quantile(ETFReturns[,i], .75) - quantile(ETFReturns[,i], .25)  
}  
  
ReturnSDs

## [1] 0.009351005 0.009768007 0.003581299 0.013727342 0.011520712

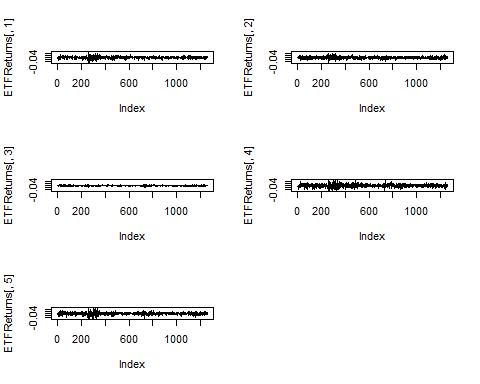
ReturnIQRs

## [1] 0.009035977 0.012280101 0.004078708 0.015486676 0.011740112

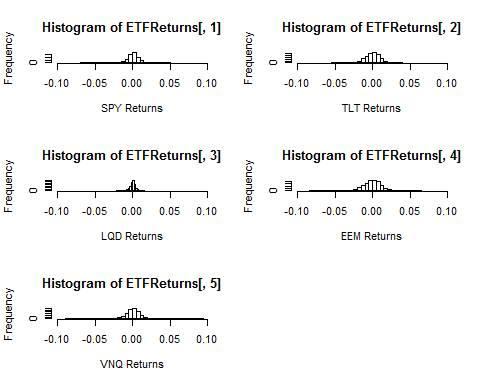
The results are similar. A standard deviation for each return type is about 1% (the exception being LQD) while the middle 50% of returns also varies by about 1%.

What do these returns look like as line graphs? As histograms?

par(mfrow=c(3,2))  
plot(ETFReturns[,1], type='l', ylim=c(-.05, .05))  
plot(ETFReturns[,2], type='l', ylim=c(-.05, .05))  
plot(ETFReturns[,3], type='l', ylim=c(-.05, .05))  
plot(ETFReturns[,4], type='l', ylim=c(-.05, .05))  
plot(ETFReturns[,5], type='l', ylim=c(-.05, .05))  
  
par(mfrow=c(3,2))



hist(ETFReturns[,1], 25, xlim=c(-.10, .10), xlab="SPY Returns")  
hist(ETFReturns[,2], 25, xlim=c(-.10, .10), xlab="TLT Returns")  
hist(ETFReturns[,3], 25, xlim=c(-.10, .10), xlab="LQD Returns")  
hist(ETFReturns[,4], 25, xlim=c(-.10, .10), xlab="EEM Returns")  
hist(ETFReturns[,5], 50, xlim=c(-.10, .10), xlab="VNQ Returns")



While I will begin building my portfolios with an even split, eventually I will be interested in comprising risky and more reserved portfolios. Thus, looking at the volatility of each of the ETFs is informative. First off, I don't observe any seasonality or obvious trends in any of the returns. I do notice that the LQD returns are by far the least volatile, followed by the TLT returns. The SPY returns are closer to the EEM Returns and the VNQ returns, but they do offer a middle ground. The main difference is the latter two ETFs have more instances of extreme returns than the SPY asset. I will keep all of this in mind for when I am choosing which assets to include in my risky and risk-averse portfolios.

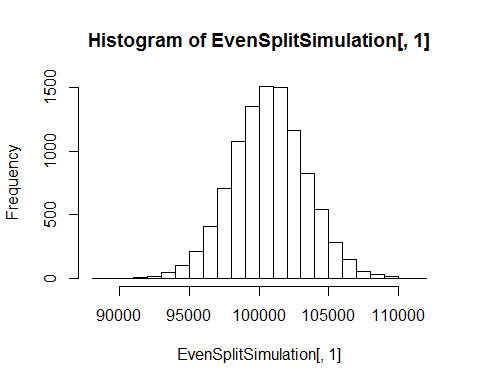
But first, I begin by selecting a portfolio that is an even split of all five assets. I want to get a sense of how well this portfolio would perform in an average month (here defined as 20 trading days) of performance. To do this, I will randomly select 20 days of returns (with replacement) from the five years of return data. I will start with $100,000. Rebalancing my money every day to maintain my desired 20-20-20-20-20 split, I will calculate how much money I possess at the end of the month. This will be one result. I will find 10,000 such results and average them together to get a sense of the true distribution of returns for this portfolio.

Below are the results for the Even Split Simulation.

EvenSplitSimulation = foreach(i=1:10000, .combine='rbind') %do% {  
 totalwealth = 100000  
 weights = c(0.2, 0.2, 0.2, 0.2, 0.2)  
 holdings = weights \* totalwealth  
 n\_days = 20  
 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  
 }  
 totalwealth  
}  
  
summary(EvenSplitSimulation)

## V1   
## Min. : 88718   
## 1st Qu.: 98945   
## Median :100718   
## Mean :100712   
## 3rd Qu.:102459   
## Max. :111523

par(mfrow=c(1,1))  
hist(EvenSplitSimulation[,1], 25)



sd(EvenSplitSimulation)

## [1] 2665.102

In terms of expected return, the center of the even split distribution is about $700. I can expect a standard deviation of about $2700. Maximum and minimum losses are both at about $11,000.

I now want to set up a vector to keep track of the alpha values across each of my simulations. The alpha value of a simulation will tell me what return I can expect at the 5th percentile.

AlphaLevels = rep(0,3)  
AlphaLevels[1] = quantile(EvenSplitSimulation,0.05) - 100000  
AlphaLevels[1]

## [1] -3645.636

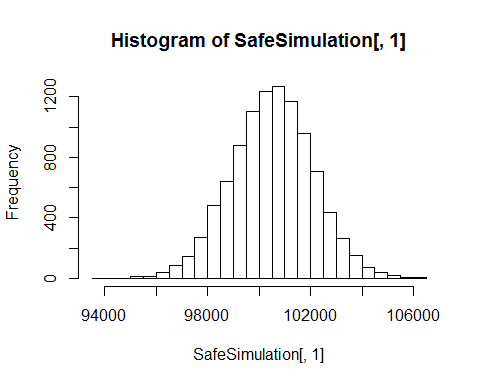
Thus, in 95% of cases, I will do better than a loss of $3600.

I now move on to finding a safer portfolio - one with less spread. From earlier, I remember that by far the least variable asset was LQD. I want the large majority of my portfolio to be in this stock. I will also include the next two least variable assests, TLT and SPY, though in smaller proportions. I arbitrarily choose to put 80% of my portfolio in LQD with 10% each in TLT and SPY. I keep the same parameters as before of 20 days and $100,000 starting value with rebalancing at the end of each day.

set.seed(722)  
SafeSimulation = foreach(i=1:10000, .combine='rbind') %do% {  
 totalwealth = 100000  
 weights = c(0.1, 0.1, 0.8)  
 holdings = weights \* totalwealth  
 n\_days = 20  
 wealthtracker = rep(0, n\_days) # Set up a placeholder to track total wealth  
 for(today in 1:n\_days) {  
 return.today = resample(ETFReturns[,c(1,2,3)], 1, orig.ids=FALSE)  
 holdings = holdings + holdings\*return.today  
 totalwealth = sum(holdings)  
 wealthtracker[today] = totalwealth  
 holdings = weights\*totalwealth  
 }  
 totalwealth  
}  
  
summary(SafeSimulation)

## V1   
## Min. : 93692   
## 1st Qu.: 99463   
## Median :100534   
## Mean :100513   
## 3rd Qu.:101561   
## Max. :106498

par(mfrow=c(1,1))  
hist(SafeSimulation[,1], 25)



sd(SafeSimulation)

## [1] 1583.293

AlphaLevels[2] = quantile(SafeSimulation,0.05) - 100000  
AlphaLevels[2]

## [1] -2120.949

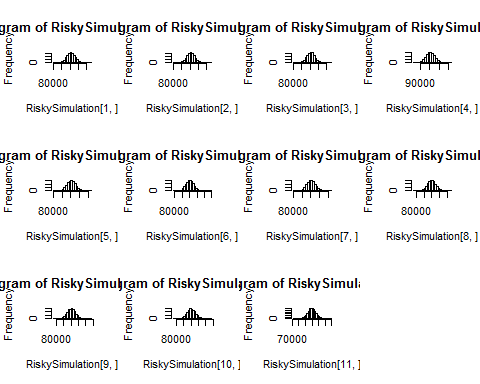
We achieve a much lower standard deviation of about $1600. My minimum loss is about $6000 and my maximum gain is about $6000 as well. My average return is about $500. In 95% of cases, I can expect to do better than a loss of $2100.

Finally, I want to evaluate a risky portfolio. I earlier noted that the two most variable assets of the five were EEM and VNQ. I will thus be building my portfolio around these two assets. However, I want to be more systematic in choosing how I weight the two ETFs. Thus, I will run the risky simulation 11 times, starting with 100% of my money in VNQ and working my way in 10% increments to having all of my money in EEM. For example, in the third run, 80% of my money will be in VNQ and 20% in EEM.

set.seed(722)  
PossibleWeights = seq(0, 1, length = 11)  
ReturnValues = rep(0, 10000)  
RiskySimulation = foreach(i=1:11, .combine = 'rbind') %do% {  
 for(j in 1:10000) {  
 totalwealth = 100000  
 weights = c(PossibleWeights[i], 1-PossibleWeights[i])  
 holdings = weights \* totalwealth  
 n\_days = 20  
 wealthtracker = rep(0, n\_days) # Set up a placeholder to track total wealth  
 for(today in 1:n\_days) {  
 return.today = resample(ETFReturns[,c(4,5)], 1, orig.ids=FALSE)  
 holdings = holdings + holdings\*return.today  
 totalwealth = sum(holdings)  
 wealthtracker[today] = totalwealth  
 holdings = weights\*totalwealth  
 }  
 ReturnValues[j] = totalwealth  
 }  
 ReturnValues  
}

But which set of weights do I choose to be my "risky" portfolio? First, let's look at what the histograms of returns look like.

par(mfrow=c(3, 4))  
hist(RiskySimulation[1,], 25)  
hist(RiskySimulation[2,], 25)  
hist(RiskySimulation[3,], 25)  
hist(RiskySimulation[4,], 25)  
hist(RiskySimulation[5,], 25)  
hist(RiskySimulation[6,], 25)  
hist(RiskySimulation[7,], 25)  
hist(RiskySimulation[8,], 25)  
hist(RiskySimulation[9,], 25)  
hist(RiskySimulation[10,], 25)  
hist(RiskySimulation[11,], 25)

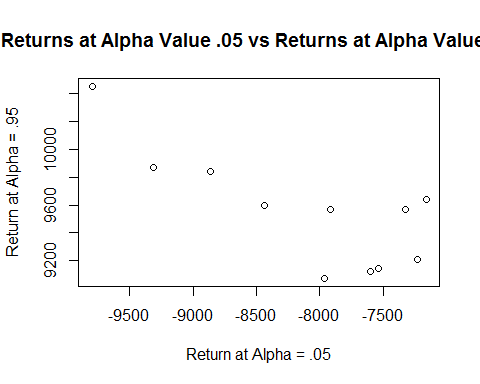


It's hard to see much difference here. Perhaps the alpha levels will be revealing. I will plot my return at alpha = .05 for each portfolio against that same portfolio's return at alpha = .95.

AlphaVsOneMinusAlpha = matrix(0, nrow=11, ncol=2)  
  
for (i in 1:length(AlphaVsOneMinusAlpha[,1])) {  
 AlphaVsOneMinusAlpha[i,1] = quantile(RiskySimulation[i,],0.05) - 100000  
}  
  
for (i in 1:length(AlphaVsOneMinusAlpha[,2])) {  
 AlphaVsOneMinusAlpha[i,2] = quantile(RiskySimulation[i,],0.95) - 100000  
}  
  
AlphaVsOneMinusAlpha

## [,1] [,2]  
## [1,] -7323.255 9570.039  
## [2,] -7159.853 9641.565  
## [3,] -7227.039 9205.495  
## [4,] -7537.103 9139.401  
## [5,] -7600.073 9124.357  
## [6,] -7962.509 9071.581  
## [7,] -7919.841 9565.188  
## [8,] -8438.530 9599.067  
## [9,] -8860.268 9843.345  
## [10,] -9310.423 9868.803  
## [11,] -9797.346 10453.997

par(mfrow=c(1, 1))  
plot(AlphaVsOneMinusAlpha, main="Returns at Alpha Value .05 vs Returns at Alpha Value .95", xlab="Return at Alpha = .05", ylab = "Return at Alpha = .95")

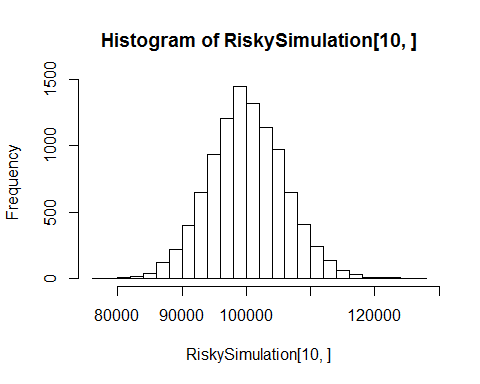


It thus looks like the "riskiest" portfolio is portfolio 11, which has 100% of the money in EEM. Since this violates the spirit of having a portfolio comprised of two assets, I will instead choose portfolio 10, which has the next lowest return at alpha = .05. Here is the summary of Portfolio 10, which is 90% EEM and 10% VNQ.

summary(RiskySimulation[10,])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 77430 96230 99960 100100 104000 126300

par(mfrow=c(1,1))  
hist(RiskySimulation[10,], 25)



sd(RiskySimulation[10,])

## [1] 5853.336

We achieve a much higher standard deviation of about $5850. My minimum loss is about $23,000 and my maximum gain is about $26,000. My average return is about $100, though my median return is to lose $40. In 95% of cases, I can expect to do better than a loss of $9300.

I now need to update my alpha vector to include the risky portfolio.

AlphaLevels[3] = quantile(RiskySimulation[10,],0.05) - 100000  
  
AlphaLevels

## [1] -3645.636 -2120.949 -9310.423

Which portfolio I recommend depends entirely upon the riskiness of the indivdual investor. Personally as a risk-averse individual, I would opt for the "safe" portfolio, but I know I'm not going to have the chance to make 20% returns with this portfolio - the absolute best case scenario is 6% and the most likely is 0.5%. The even-split portfolio represents a nice middle ground between the two extremes of risk. My best case scenario bumps up to a 10% gain with a most likely return of about 0.7%. The risky portfolio is too volatile for my money. The standard deviation is around double that of the even split portfolio. I know there is risk inherent in the stock market, but the risky portfolio seems too much like gambling for my taste.

# Clustering and PCA

## Characteristics of Wine from Northern Portugal

Given only chemical properties, can I distinguish whether a wine is red or white? More challenging, can I distinguish the quality of the wine from its chemical characteristics?

I begin my analysis by loading in the data.

WineURLString = getURL("https://raw.githubusercontent.com/jacebarton/STA380/master/data/wine.csv", ssl.verifypeer=0L, followlocation = 1L)  
Wine = read.csv(text=WineURLString)  
summary(Wine)

## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. : 3.800 Min. :0.0800 Min. :0.0000 Min. : 0.600   
## 1st Qu.: 6.400 1st Qu.:0.2300 1st Qu.:0.2500 1st Qu.: 1.800   
## Median : 7.000 Median :0.2900 Median :0.3100 Median : 3.000   
## Mean : 7.215 Mean :0.3397 Mean :0.3186 Mean : 5.443   
## 3rd Qu.: 7.700 3rd Qu.:0.4000 3rd Qu.:0.3900 3rd Qu.: 8.100   
## Max. :15.900 Max. :1.5800 Max. :1.6600 Max. :65.800   
## chlorides free.sulfur.dioxide total.sulfur.dioxide  
## Min. :0.00900 Min. : 1.00 Min. : 6.0   
## 1st Qu.:0.03800 1st Qu.: 17.00 1st Qu.: 77.0   
## Median :0.04700 Median : 29.00 Median :118.0   
## Mean :0.05603 Mean : 30.53 Mean :115.7   
## 3rd Qu.:0.06500 3rd Qu.: 41.00 3rd Qu.:156.0   
## Max. :0.61100 Max. :289.00 Max. :440.0   
## density pH sulphates alcohol   
## Min. :0.9871 Min. :2.720 Min. :0.2200 Min. : 8.00   
## 1st Qu.:0.9923 1st Qu.:3.110 1st Qu.:0.4300 1st Qu.: 9.50   
## Median :0.9949 Median :3.210 Median :0.5100 Median :10.30   
## Mean :0.9947 Mean :3.219 Mean :0.5313 Mean :10.49   
## 3rd Qu.:0.9970 3rd Qu.:3.320 3rd Qu.:0.6000 3rd Qu.:11.30   
## Max. :1.0390 Max. :4.010 Max. :2.0000 Max. :14.90   
## quality color   
## Min. :3.000 red :1599   
## 1st Qu.:5.000 white:4898   
## Median :6.000   
## Mean :5.818   
## 3rd Qu.:6.000   
## Max. :9.000

There are 1600 red wines and 4900 white wines represented in the data. The quality of all the wines ranges from 3-9 on a 1-10 scale with an average of about 6.

### Clustering

In order to perform clustering analysis, I will need to scale and center this wine data. In this process, I will also remove the quality and color features as these are outputs I will eventually attempt to predict. I will keep track of the means and standard deviations of each feature in case I want to convert back to unstandardized data.

WineScaled = scale(Wine[,-(c(12,13))], center=TRUE, scale=TRUE)  
  
mu = attr(WineScaled,"scaled:center")  
sigma = attr(WineScaled,"scaled:scale")

In class, we discussed both hierarchical clustering and K-means clustering. For this data set, I know I'm not going to want many clusters as ultimately I'm going to be interested in predicting only a handful of output classes (Red vs Wine and a number from 3-9 for color and quality, respectively). With only a few clusters, hierarchical clustering is not an ideal candidate as I will get the overwhelming majority of the data points in one cluster and then several much smaller clusters. Thus, I will pursue K-Means clustering.

But how many K's shall I choose? To answer this, I will use the CH(k) matrix we discussed in class which attempts to balance inter-cluster distance with intra-cluster distance. To calculate CH(k), I will loop through K values from 2 to 20 looking for the maximum CH(k).

PossibleKs = matrix(0, nrow=19, ncol=2)  
PossibleKs[,1] = 2:20  
  
set.seed(722)  
for(i in 1:19){  
 WineKMeanClusters = kmeans(WineScaled, i, nstart=50)  
 CHk = ((WineKMeanClusters$betweenss/(i-1))/(WineKMeanClusters$tot.withinss/(nrow(WineScaled)-i)))  
 PossibleKs[i,2] = CHk   
}

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 324850)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

plot(PossibleKs, xlab="# of Clusters", Ylab="CH(k)", main="Choosing K to Maximize CH(k)")

## Warning in plot.window(...): "Ylab" is not a graphical parameter

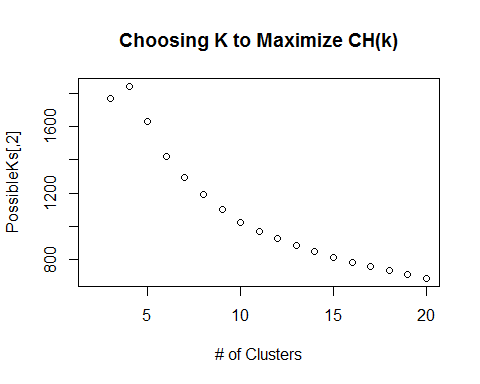
## Warning in plot.xy(xy, type, ...): "Ylab" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "Ylab" is not  
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "Ylab" is not  
## a graphical parameter

## Warning in box(...): "Ylab" is not a graphical parameter

## Warning in title(...): "Ylab" is not a graphical parameter



CH(k) peaks at k=4, so I will have four clusters of wine.

I now will add which cluster each wine is assigned to on the original data set.

set.seed(722)  
WineKMeanClusters = kmeans(WineScaled, 4, nstart=50)  
Wine$cluster = factor(WineKMeanClusters$cluster)  
WineKMeanClusters$centers

## fixed.acidity volatile.acidity citric.acid residual.sugar chlorides  
## 1 1.9272263 0.4658003 0.97461378 -0.5621815 1.2651916  
## 2 0.0310115 1.6264134 -1.24743592 -0.6171303 0.6338616  
## 3 -0.1946570 -0.3568490 0.26831468 1.2164487 -0.1039414  
## 4 -0.3381025 -0.4400768 0.02868768 -0.4336995 -0.4480556  
## free.sulfur.dioxide total.sulfur.dioxide density pH  
## 1 -0.88068209 -1.21063354 0.9263596 -0.09894986  
## 2 -0.77099529 -1.10241627 0.4510104 0.96385049  
## 3 0.85498261 0.96137057 0.7633800 -0.38505854  
## 4 -0.07507221 0.04818817 -0.8602004 -0.06191271  
## sulphates alcohol  
## 1 1.3525304 0.02318728  
## 2 0.3994917 -0.21910641  
## 3 -0.2580984 -0.79520297  
## 4 -0.2894397 0.57819988

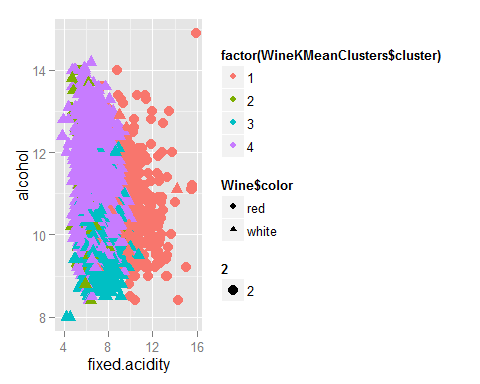
summary(factor(WineKMeanClusters$cluster))

## 1 2 3 4   
## 687 1007 1873 2930

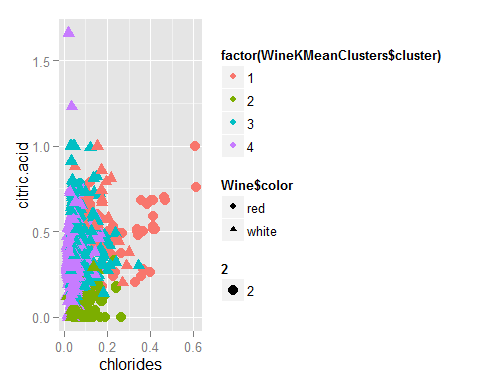
Cluster 4 is the biggest, followed by Clusters 3, 2, and 1 in descending order. Clusters 1 and 2 have high acidity, while cluster 4 has the highest alcohol level and cluster three the most sugar. Note that the centers are given in Z-scores since this analysis was run on the centered and scaled data.

Below, two sample plots are shown. The X and Y axis in each plot are different features of the wine. The wine cluster is indicated by color while the wine color is indicated by shape. These plots are only meant to get a sense of how well we can judge the clusters, though since we can only visualize two dimensions at a time, this sense will be dulled.

qplot(fixed.acidity, alcohol, data = Wine, color=factor(WineKMeanClusters$cluster), shape=Wine$color, size=2)



qplot(chlorides, citric.acid, data = Wine, color=factor(WineKMeanClusters$cluster), shape=Wine$color, size=2)



Most importantly, how well do my clusters differentiate between wine colors?

ClusterVsColor = xtabs(~cluster + color, data=Wine)  
ClusterVsColor

## color  
## cluster red white  
## 1 638 49  
## 2 917 90  
## 3 4 1869  
## 4 40 2890

ClusterVsColorProp = prop.table(ClusterVsColor, margin=1)  
ClusterVsColorProp

## color  
## cluster red white  
## 1 0.928675400 0.071324600  
## 2 0.910625621 0.089374379  
## 3 0.002135611 0.997864389  
## 4 0.013651877 0.986348123

Clusters 3 and 4 are the "White" clusters, and they perform best with a 99% classification rate. Clusters 1 and 2 are the "red" clusters, and they don't perform quite as well, but both still ahve classifcation rates above 90%.

How well can the clusters judge quality compared to the baseline percentages?

ClusterVsQuality = xtabs(~cluster + quality, data=Wine)  
ClusterVsQuality

## quality  
## cluster 3 4 5 6 7 8 9  
## 1 5 20 229 285 136 12 0  
## 2 7 71 503 367 53 6 0  
## 3 10 44 797 833 157 31 1  
## 4 8 81 609 1351 733 144 4

ClusterVsQualityProp = prop.table(ClusterVsQuality, margin=1)  
ClusterVsQualityProp

## quality  
## cluster 3 4 5 6 7  
## 1 0.0072780204 0.0291120815 0.3333333333 0.4148471616 0.1979621543  
## 2 0.0069513406 0.0705064548 0.4995034757 0.3644488580 0.0526315789  
## 3 0.0053390283 0.0234917245 0.4255205553 0.4447410571 0.0838227443  
## 4 0.0027303754 0.0276450512 0.2078498294 0.4610921502 0.2501706485  
## quality  
## cluster 8 9  
## 1 0.0174672489 0.0000000000  
## 2 0.0059582920 0.0000000000  
## 3 0.0165509877 0.0005339028  
## 4 0.0491467577 0.0013651877

#Baseline percentages  
QualityCounts = summary(factor(Wine$quality))  
QualityCountsProp = QualityCounts/sum(QualityCounts)  
QualityCountsProp

## 3 4 5 6 7 8   
## 0.004617516 0.033246114 0.329074958 0.436509158 0.166076651 0.029706018   
## 9   
## 0.000769586

Not very well. None of the percentages in the cluster stand out as vastly different from the corresponding percentage in the baseline. Put another way, if you tell me a wine is in Cluster 3, I will not be able to tell you with any more certainty what quality wine it is versus just telling you what I could glean from the unclustered data (i.e., a quality of 6 is most)

Performance is somewhat good. For example, in the baseline, 43% of wines are quality 6 while 33% are quality 5. But in cluster 4, the difference is more pronounced - 47% of cluster 4 wines are quality 6 while 21% are cluster 5. Conversely, in cluster 2, 50% of wines are quality 5 while 36% are cluster 6. So I can do a little better than just guessing the most common quality if the wine is in cluster 2.

### Principal Component Analysis (PCA)

Overall, I was pretty happy with the performance of my clusters. Will I be able to top it using PCA?

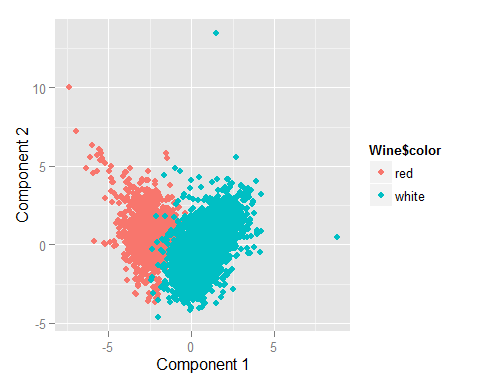
WinePrincipalComponent = prcomp(WineScaled)  
loadings = WinePrincipalComponent$rotation  
scores = WinePrincipalComponent$x  
loadings[,1:2]

## PC1 PC2  
## fixed.acidity -0.23879890 0.33635454  
## volatile.acidity -0.38075750 0.11754972  
## citric.acid 0.15238844 0.18329940  
## residual.sugar 0.34591993 0.32991418  
## chlorides -0.29011259 0.31525799  
## free.sulfur.dioxide 0.43091401 0.07193260  
## total.sulfur.dioxide 0.48741806 0.08726628  
## density -0.04493664 0.58403734  
## pH -0.21868644 -0.15586900  
## sulphates -0.29413517 0.19171577  
## alcohol -0.10643712 -0.46505769

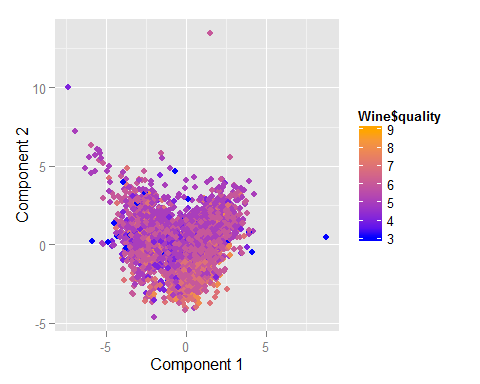
head(scores[,1:2])

## PC1 PC2  
## [1,] -3.205749 0.4164913  
## [2,] -3.038817 1.1073769  
## [3,] -3.071657 0.8788968  
## [4,] -1.571141 2.1123820  
## [5,] -3.205749 0.4164913  
## [6,] -3.011934 0.3893675

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



qplot(scores[,1], scores[,2], color=Wine$quality, xlab='Component 1', ylab='Component 2') + scale\_color\_gradient(low="blue", high="orange")



Looking at the first two components, I cannot determine quality with any accuracy. However, the reds and whites are very nicely split, and that's looking almost solely across component 1. I am unaware of how to quantify a classification rate based on PCA, but just looking at the picture, I prefer PCA to Clustering for distingushing red wines from white wines.

# Market Segmentation

## Using Social Media Data to Find Similar Customers

As always, the first step is to read in the data.

SocialMediaURLString = getURL("https://raw.githubusercontent.com/jacebarton/STA380/master/data/social\_marketing.csv", ssl.verifypeer=0L, followlocation = 1L)  
SocialMedia = read.csv(text=SocialMediaURLString)  
summary(SocialMedia)

## X chatter current\_events travel   
## 123pxkyqj: 1 Min. : 0.000 Min. :0.000 Min. : 0.000   
## 12grikctu: 1 1st Qu.: 2.000 1st Qu.:1.000 1st Qu.: 0.000   
## 12klxic7j: 1 Median : 3.000 Median :1.000 Median : 1.000   
## 12t4msroj: 1 Mean : 4.399 Mean :1.526 Mean : 1.585   
## 12yam59l3: 1 3rd Qu.: 6.000 3rd Qu.:2.000 3rd Qu.: 2.000   
## 132y8f6aj: 1 Max. :26.000 Max. :8.000 Max. :26.000   
## (Other) :7876   
## photo\_sharing uncategorized tv\_film sports\_fandom   
## Min. : 0.000 Min. :0.000 Min. : 0.00 Min. : 0.000   
## 1st Qu.: 1.000 1st Qu.:0.000 1st Qu.: 0.00 1st Qu.: 0.000   
## Median : 2.000 Median :1.000 Median : 1.00 Median : 1.000   
## Mean : 2.697 Mean :0.813 Mean : 1.07 Mean : 1.594   
## 3rd Qu.: 4.000 3rd Qu.:1.000 3rd Qu.: 1.00 3rd Qu.: 2.000   
## Max. :21.000 Max. :9.000 Max. :17.00 Max. :20.000   
##   
## politics food family home\_and\_garden   
## Min. : 0.000 Min. : 0.000 Min. : 0.0000 Min. :0.0000   
## 1st Qu.: 0.000 1st Qu.: 0.000 1st Qu.: 0.0000 1st Qu.:0.0000   
## Median : 1.000 Median : 1.000 Median : 1.0000 Median :0.0000   
## Mean : 1.789 Mean : 1.397 Mean : 0.8639 Mean :0.5207   
## 3rd Qu.: 2.000 3rd Qu.: 2.000 3rd Qu.: 1.0000 3rd Qu.:1.0000   
## Max. :37.000 Max. :16.000 Max. :10.0000 Max. :5.0000   
##   
## music news online\_gaming shopping   
## Min. : 0.0000 Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 0.0000 1st Qu.: 0.000 1st Qu.: 0.000 1st Qu.: 0.000   
## Median : 0.0000 Median : 0.000 Median : 0.000 Median : 1.000   
## Mean : 0.6793 Mean : 1.206 Mean : 1.209 Mean : 1.389   
## 3rd Qu.: 1.0000 3rd Qu.: 1.000 3rd Qu.: 1.000 3rd Qu.: 2.000   
## Max. :13.0000 Max. :20.000 Max. :27.000 Max. :12.000   
##   
## health\_nutrition college\_uni sports\_playing cooking   
## Min. : 0.000 Min. : 0.000 Min. :0.0000 Min. : 0.000   
## 1st Qu.: 0.000 1st Qu.: 0.000 1st Qu.:0.0000 1st Qu.: 0.000   
## Median : 1.000 Median : 1.000 Median :0.0000 Median : 1.000   
## Mean : 2.567 Mean : 1.549 Mean :0.6392 Mean : 1.998   
## 3rd Qu.: 3.000 3rd Qu.: 2.000 3rd Qu.:1.0000 3rd Qu.: 2.000   
## Max. :41.000 Max. :30.000 Max. :8.0000 Max. :33.000   
##   
## eco computers business outdoors   
## Min. :0.0000 Min. : 0.0000 Min. :0.0000 Min. : 0.0000   
## 1st Qu.:0.0000 1st Qu.: 0.0000 1st Qu.:0.0000 1st Qu.: 0.0000   
## Median :0.0000 Median : 0.0000 Median :0.0000 Median : 0.0000   
## Mean :0.5123 Mean : 0.6491 Mean :0.4232 Mean : 0.7827   
## 3rd Qu.:1.0000 3rd Qu.: 1.0000 3rd Qu.:1.0000 3rd Qu.: 1.0000   
## Max. :6.0000 Max. :16.0000 Max. :6.0000 Max. :12.0000   
##   
## crafts automotive art religion   
## Min. :0.0000 Min. : 0.0000 Min. : 0.0000 Min. : 0.000   
## 1st Qu.:0.0000 1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 0.000   
## Median :0.0000 Median : 0.0000 Median : 0.0000 Median : 0.000   
## Mean :0.5159 Mean : 0.8299 Mean : 0.7248 Mean : 1.095   
## 3rd Qu.:1.0000 3rd Qu.: 1.0000 3rd Qu.: 1.0000 3rd Qu.: 1.000   
## Max. :7.0000 Max. :13.0000 Max. :18.0000 Max. :20.000   
##   
## beauty parenting dating school   
## Min. : 0.0000 Min. : 0.0000 Min. : 0.0000 Min. : 0.0000   
## 1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 0.0000   
## Median : 0.0000 Median : 0.0000 Median : 0.0000 Median : 0.0000   
## Mean : 0.7052 Mean : 0.9213 Mean : 0.7109 Mean : 0.7677   
## 3rd Qu.: 1.0000 3rd Qu.: 1.0000 3rd Qu.: 1.0000 3rd Qu.: 1.0000   
## Max. :14.0000 Max. :14.0000 Max. :24.0000 Max. :11.0000   
##   
## personal\_fitness fashion small\_business spam   
## Min. : 0.000 Min. : 0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.: 0.000 1st Qu.: 0.0000 1st Qu.:0.0000 1st Qu.:0.00000   
## Median : 0.000 Median : 0.0000 Median :0.0000 Median :0.00000   
## Mean : 1.462 Mean : 0.9966 Mean :0.3363 Mean :0.00647   
## 3rd Qu.: 2.000 3rd Qu.: 1.0000 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :19.000 Max. :18.0000 Max. :6.0000 Max. :2.00000   
##   
## adult   
## Min. : 0.0000   
## 1st Qu.: 0.0000   
## Median : 0.0000   
## Mean : 0.4033   
## 3rd Qu.: 0.0000   
## Max. :26.0000   
##

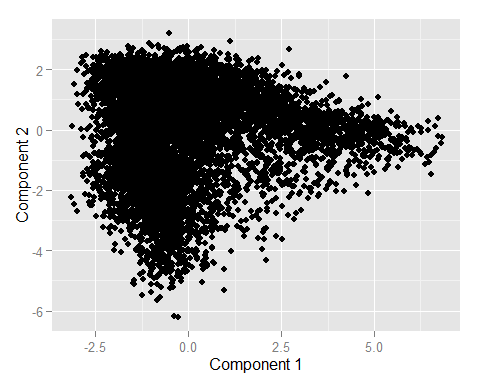
Now, I need to find the frequency of the content types for each user rather than the count.

# Normalize phrase counts to phrase frequencies  
SocialMediaFrequencies = SocialMedia[,-1]/rowSums(SocialMedia[,-1])

And now, since I'm feeling wild, I'll perform PCA analysis *first* instead of cluster analysis. I know, try to contain your excitement.

### PCA for Social Media Data

SocialMediaPCA = prcomp(SocialMediaFrequencies, scale=TRUE)  
SMLoadings = SocialMediaPCA$rotation  
SMScores = SocialMediaPCA$x  
  
qplot(SMScores[,1], SMScores[,2], xlab='Component 1', ylab='Component 2')



Unlike the wine data, there's not an output variable for me to look at on a plot of Component 1 vs Component 2. Instead, I can try looking at the features which score highest on each component, starting with component 1.

Component1Ordered = order(SMLoadings[,1])  
colnames(SocialMediaFrequencies)[tail(Component1Ordered,5)]

## [1] "school" "food" "parenting" "sports\_fandom"  
## [5] "religion"

Religion scores highest, followed by sports\_fandom and parenting (note that the highest score is the last entry).

Component2Ordered = order(SMLoadings[,2])  
colnames(SocialMediaFrequencies)[tail(Component2Ordered,5)]

## [1] "automotive" "shopping" "travel" "politics" "chatter"

In Component 2, chatter, politics, and travel score highest.

Now, I'll see if some of these patterns hold when I look at the clustered data.

### Social Media Clusters

Similar to the wine problem, I will use CH(k) to determine what number of clusters to use.

PossibleKsSocial = matrix(0, nrow=19, ncol=2)  
PossibleKsSocial[,1] = 2:20  
set.seed(722)  
for(i in 1:19){  
 SocialKMeanClusters = kmeans(SocialMediaFrequencies, i, nstart=50)  
 CHk = ((SocialKMeanClusters$betweenss/(i-1))/(SocialKMeanClusters$tot.withinss/(nrow(SocialMediaFrequencies)-i)))  
 PossibleKsSocial[i,2] = CHk   
}

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 394100)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

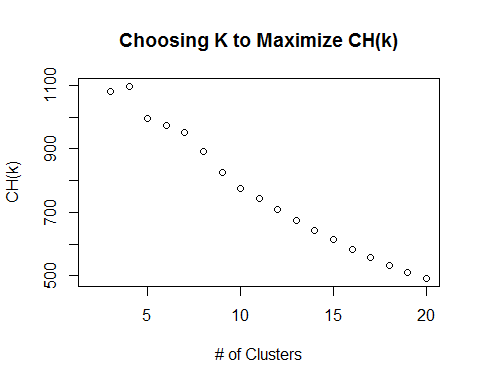
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

plot(PossibleKsSocial, xlab="# of Clusters", ylab="CH(k)", main="Choosing K to Maximize CH(k)")



Yet again, four clusters is the optimal choice. I will add the cluster values to the original social media data.

set.seed(722)  
SocialKMeanClusters = kmeans(SocialMediaFrequencies, 4, nstart=50)  
SocialMedia$cluster = factor(SocialKMeanClusters$cluster)  
  
summary(factor(SocialKMeanClusters$cluster))

## 1 2 3 4   
## 1417 3437 807 2221

Now, I'll look at the five most significant features for each cluster.

sort(SocialKMeanClusters$centers[1,], decreasing=TRUE)[1:5]

## health\_nutrition personal\_fitness chatter cooking   
## 0.21192108 0.10608338 0.07639196 0.05458727   
## outdoors   
## 0.04303909

sort(SocialKMeanClusters$centers[2,], decreasing=TRUE)[1:5]

## chatter politics sports\_fandom college\_uni travel   
## 0.07714131 0.06247846 0.05596013 0.05293592 0.05251692

sort(SocialKMeanClusters$centers[3,], decreasing=TRUE)[1:5]

## cooking photo\_sharing fashion chatter beauty   
## 0.18541240 0.09484052 0.09099594 0.07366347 0.05922135

sort(SocialKMeanClusters$centers[4,], decreasing=TRUE)[1:5]

## chatter photo\_sharing shopping current\_events travel   
## 0.23012456 0.11417367 0.06582203 0.05626872 0.03543507

The only feature from the first component to appear in the clusters' most significant features is sports\_fandom. From the second component, chatter appears in all 4 clusters, while politics appears in 1, travel appears in 2, shopping appears in 1, and automotive appears in 0.

Given the difficulty in interpreting PCA in this case, I will base my analysis off of the clusters. The clusters also make sense. For example, health-nutrition, personal fitness, and outdoors all appear in a cluster together, while sports-fandom and college-uni also appear in a cluster together. Additionally, fashion and beauty appear in a cluster. These are three good market segments to begin to target amongst the company's customers.