Exercises 1, STA 380 PART 2

Erik Honore and Michael Lundy

August 8, 2016

Probabilities

Part A)

For this problem, we actually mapped out a probability tree on scratch paper to visualize this.

Here are my assumptions/understanding for the problem:

RC is 30% (.3) of the population (or individuals who perform the survey) LC is 70% (.7) of the population (or individuals who perform the survey)

For RC, we are given that its 50/50 that they answer Yes. So, we said Y = 1, N = 0. (binary with respect to Y giving us any value)

The expected value for one RC person is 1(.50) + 0(.50) = .50

Thus the value of the RC people in a population is: (.3)(.5) = .15

My thinking: We expect 30% of the population to provide us with .50 Yes responses (and .50 No responses), so the total value for that is 15% Yes responses. We are told that the real numbers for the entire survey collection was 65% Y / 35% N. Therefore, the TC folks had to provide .50 Yes responses, and .20 No responses.

we'll describe the tree logic here briefly.

For an imaginary 100 people in the survey, 70 would be TC and 30 would be RC. Of those 30 RC, 15 would respond Yes, and 15 would respond No. That means the TC would have to have 50 respond Yes (for total 65), and the remaining 20 would have responded No (to get to 35 there).

So, the fraction of TCs that answered Yes was 50 out of 70, or (.714).

Part B)

We devised another tree to visualize this. Our goal was to find the probability that someone has the disease given they tested positive. We will define the Bayes calculation below.

Bayes

A = have the disease, **B** = test positive

Want **P(A|B)**, the probability of having the disease given a positive test result.

P(A) = .000025 given by the problem, overall probability of having the disease

P(B|A) = .993 given by the problem, probability of testing positive given you have the disease

P(B|A^c) = .0001 derived from problem, probability of testing positive given you don't have the disease

P(A^c) = .999975 derived from the problem, overall probability of not having the disease in the population

We will now perform the Bayes calculation in R. This calculation will be recreatable with perhaps differing probabilities etc.

```
prob_A = .000025# P(A)
prob_B.A = .993 # P(B|A)
prob_B.notA = .0001 # P(B|not A)
prob_notA = .999975 # P(not A)

prob_A.B = (prob_B.A*prob_A) / ((prob_B.A*prob_A) + (prob_B.notA*prob_notA))

#print P(A|B)
prob_A.B
```

```
## [1] 0.1988824
```

The probability of having the disease given a positive test is about **19.88%**.

According to the information, out of 100,000 people, only 2.5 would actually have the disease. From that 2.5, ~2.48 would test positive for it. However, given the probability we just calculated, a universal test will likely result in false positives. Based on the other side of the tree, we could expect around 9 people in 100,000 to test positive for the disease without actually having it. That is a higher number than the 2.48 people from above who would test positive and actually have it. There won't be many positives, but based on my calculations, the chance of you testing positive and actually having the disease are slim. A testing policy would have to include some sort of multi-testing to figure out false positives, which is probably not cost effective.

Exploratory Analysis on Green Buildings

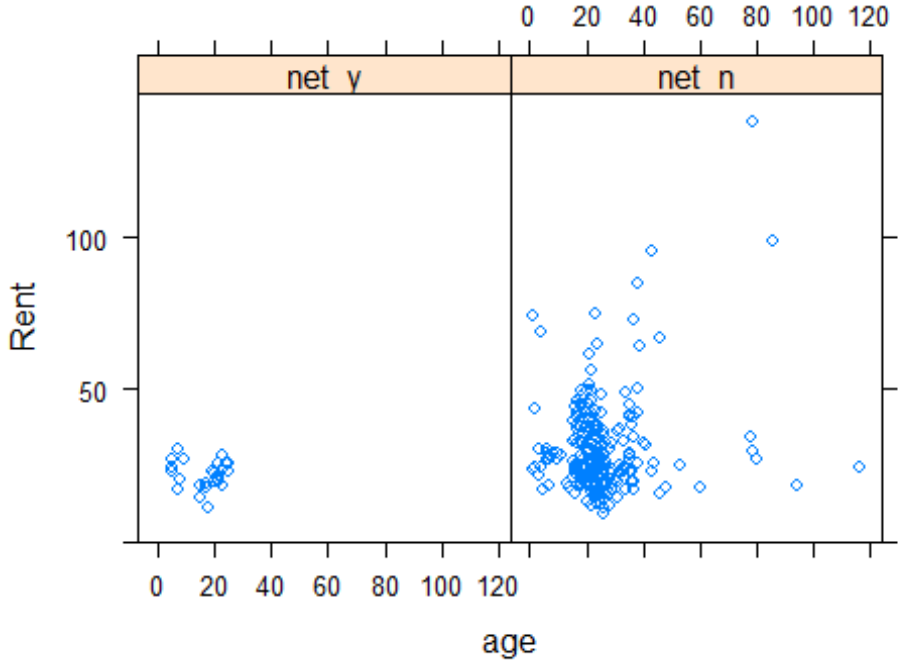
```
rm(list=ls())

library(lattice)

buildings = read.csv('greenbuildings.csv', header = TRUE)

#factorize
buildings$cluster = factor(x = buildings$cluster, 1:685, labels = 1:685)
buildings$renovated = factor(x = buildings$renovated, c(1,0), labels =
```

```
c('renovated_y','renovated_n'))
buildings$class_a = factor(x = buildings$class_a, c(1,0), labels =
c('a_y','a_n'))
buildings$class_b = factor(x = buildings$class_b, c(1,0), labels =
c('b_y','b_n'))
buildings$LEED =  factor(x = buildings$LEED, c(1,0), labels =
c('leed_y','leed_n'))
buildings$Energystar = factor(x = buildings$Energystar, c(1,0), labels =
c('ES_y','ES_n'))
buildings$green_rating = factor(x = buildings$green_rating, c(1,0), labels =
c('rating_y','rating_n'))
buildings$net = factor(x = buildings$net, c(1,0), labels =
c('net_y','net_n'))
buildings$amenities = factor(x= buildings$amenities, c(1,0), labels =
c('amenities_y','amenities_n'))

#must reduce data set down to comparable complexes of 15 stories
mask = (buildings$stories >= 15)
buildings = buildings[mask,] #apply the mask to get only the observations.
mask is basically an index of T,F values. Only want the true

#look at only the green buildings now
mask1 = buildings$green_rating == 'rating_y'
buildings1 = buildings[mask1,]

mask2 = buildings$green_rating == 'rating_n'
buildings2 = buildings[mask2,] #non-green buildings only
#Green buildings >= 15 stories
#non-green buildings >= 15 stories
```

Here, we've created two different data sets for use further down in my exploration. The **buildings** data frame ends up being all observations of buildings that are greater than or equal to 15 stories high. The **buildings1** data frame is then filtered to include only the green-certified buildings in that set. The **buildings2** frame is filtered to include only the non-green buildings. Since the building investment in question is going to be 15 stories and potentially green, we masked out the data set to only look at similar buildings.

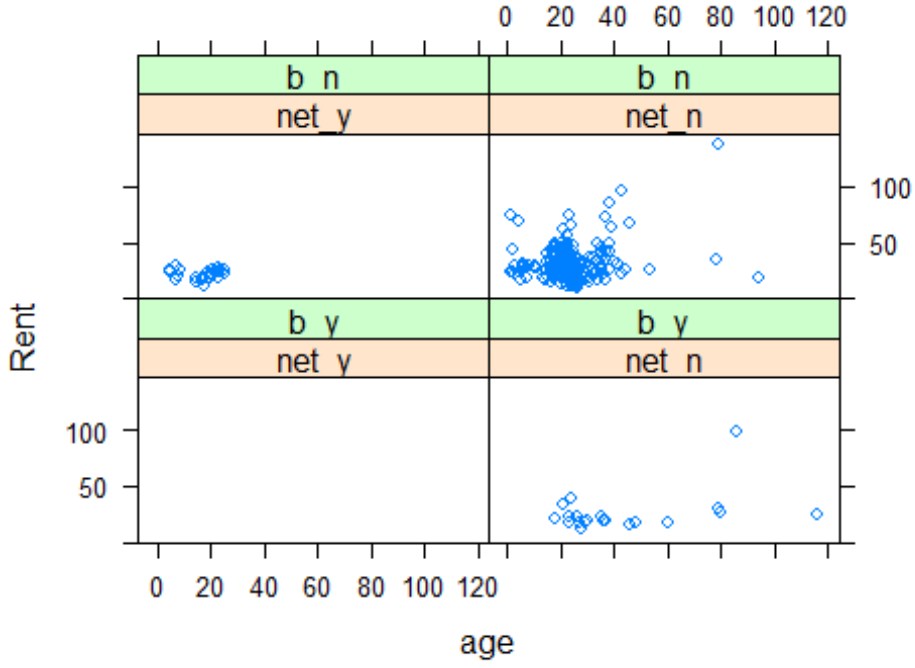
```
plot1 = xyplot(Rent~age | net, data= buildings1, main = 'Plot 1: Rent on Age
for each level of Net')
plot1
```

Plot 1: Rent on Age for each level of Net




```
plot2 = xyplot(Rent~age | net*class_b, data= buildings1, main = 'Plot 2: Rent on Age for each level of Net and B Class')
plot2
```

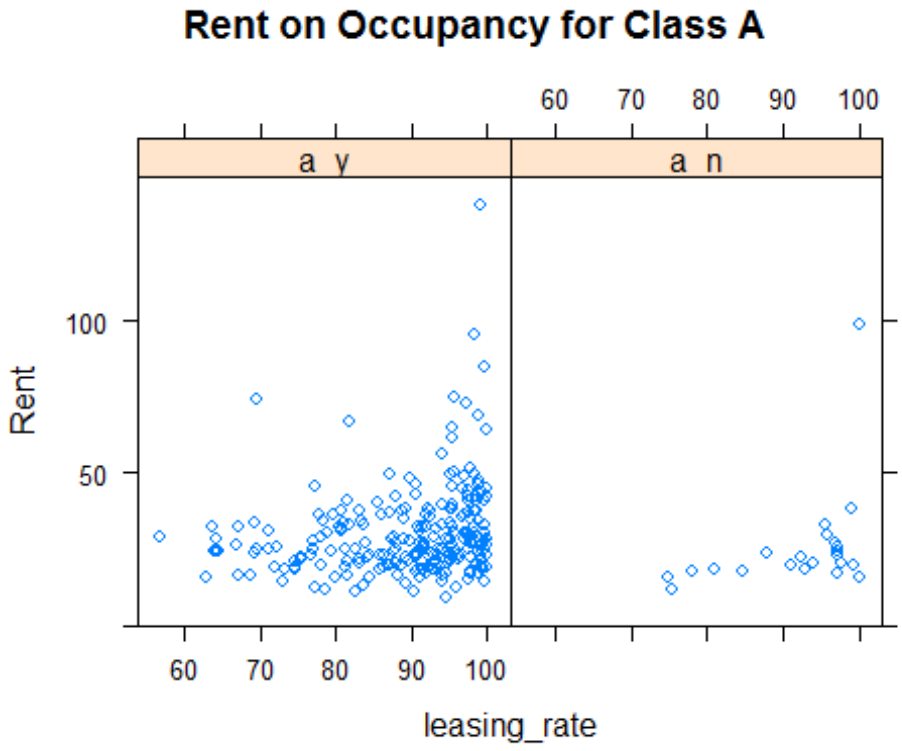
Plot 2: Rent on Age for each level of Net and B Class



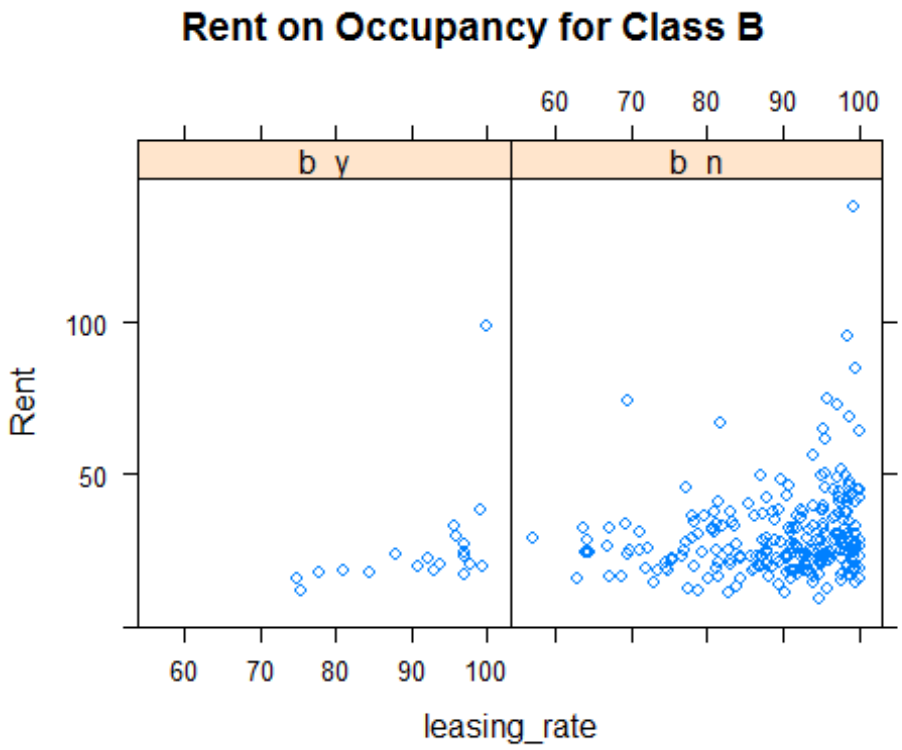
From Plot 1, we're looking at how rent changes as age increases. It is also partitioned based on whether or not the rent is net or not. The analyst's assertions about profits over the time horizon assume that he can maintain that same level of rent upcharge for being green. This plot shows that for rent, especially non-net rent, the variability is very large around the 20 year mark. He anchors his argument based on some single level of profits being captured for "30 years or more". wewould argue that the analyst should consider the vast array of rent charges from the time = 0 to year 30.

From Plot 2, we can see that his decision on netting the rent or not is also non-trivial based on this single plot. The netted rent side has vastly fewer data points, but the ones that are there suggest a tighter and perhaps decreasing relationship between age and rent. That means, for buildings with net rent, as age increases, rent likely decreases in some way. That would affect his profitability forecast if the building wanted to net the rent.

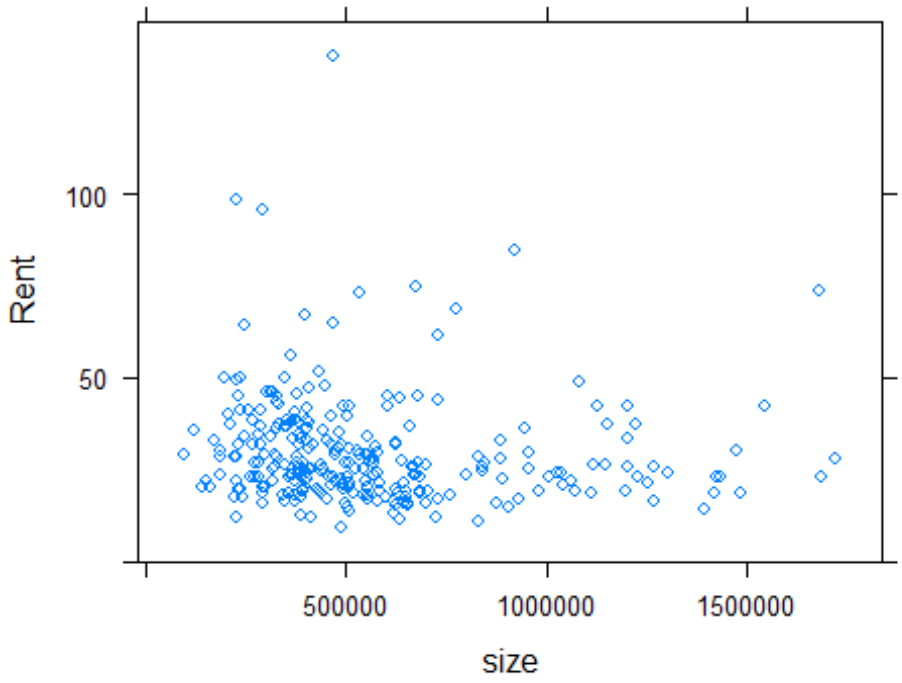
```
plot3 = xyplot(Rent~leasing_rate | class_a, data=buildings1, main= 'Rent on
Occupancy for Class A')
plot3 #for green
```




```
plot4 = xyplot(Rent~leasing_rate | class_b, data=buildings1, main= 'Rent on
Occupancy for Class B')
plot4 #for green
```




```
plot5 = xyplot(Rent~size, data= buildings1)
plot5 #for green
```



```r
median(buildings1$Rent)
```

```
## [1] 25.8
```

```r
mean(buildings1$Rent)
```

```
## [1] 29.63337
```

```r
#final mask for literal sizing
mask3 = buildings1$size >= 250000
buildings4 = buildings1[mask3,] #which is basically green, >=15, >= 250000

median(buildings4$Rent)
```

```
## [1] 25.25
```

```r
mean(buildings4$Rent)
```

```
## [1] 29.14641
```

These two plots related to "Rent on Occupancy" support the previous point about the analyst's surface level assumptions about his rent charge and profitability schedule. These two plots suggest that yes, rent has positive relationship with leasing_rate. This is likely a simple supply-demand relationship often seen in housing markets. However, for the class A properties we can see that when the leasing rate is between 90 and 100 percent, the variability in rent charges is largest in the plot. There are quite a few data points located perhaps lower than the $27/sqft rent charge that he is assuming he can hold onto for the building. The key point here is the visual variance in the data set. wewould say the rent has a non-constant variance based on this plot.

We've essentially shown that a few of the many factors (age, leasing_rate, and property class) reveal great variability in the rent that is charged for buildings greater than or equal to 15 stories that have green certification of some kind. The analyst has not considered how the rent charge can vary depending on market conditions.

Additionally, he has quoted a median rent charge of $27.6/sqft/year. However, we've done a simple median calculation on my subsetted data set on green buildings >= 15 stories that would be basically similar to his new building. That median revealed a charge of $25.8/sqft/year. This metric cuts in to his initial assumptions on payback period on the investment. His rental charges are possibly overly optimistic. At this median rent charge for a similar building, it would take the property longer time to recoup the investment, which already plays into the decision.

We took my analysis a step further on this median rent quantity. With our buildings1 frame, we masked again to only keep buildings that were green, >= 15 stories, and >= 250,000. This adds another dimension of similarity to the proposed building. The median rent charge for that subset of buildings was $25.25/sqft/year. This hurts his argument of asserting a $27.60 charge as feasible/reasonable.

However, as part of the exploratory process, we continued to iterate and evolve on the analysis. We went ahead and changed the data frame identity by masking for buildings exactly equal to 15 stories. We ran all of my plots and quantities down with the exact same logic for green or not green, >= 250,000 in size etc. When we did this, everything changed in the analysis. The median rent charge for green buildings at 15 stories of the same size was actually $36.95/sqft/year. The mean for those buildings was $34.95. That particular median/mean suggests that perhaps the analyst is UNDERESTIMATING the revenues and rent charges he could make. That would help his case. We even went in an remasked to look at >= 20 stories, which resulted in a rent charge in the upper 20 dollar range. All of this being said, we still think that although we were able to reduce the scope of the data frame to basically the exact sizing of the building in question, there were less than 10 data points of 15 story buildings. With all the factors in play in a housing market, judging based on solely on that median for the ten or less 15 story buildings is probably not wise. Therefore, we're holding to the recommendation that the analyst rework his predictions on the rent charges to better reflect the variability. As of now, he has no variability included in his estimate, which is flawed. If we had to make an investment decision, we would simply say no-go until the variability of rent charges can be assessed.

Bootstrapping

```
library(mosaic)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: ggplot2
```

```
## Loading required package: mosaicData
```

```
## Loading required package: Matrix
```

```
##
## The 'mosaic' package masks several functions from core packages in order
to add additional features.
## The original behavior of these functions should not be affected by this.
```

```
##
## Attaching package: 'mosaic'
```

```
## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, 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(fImport)
```

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

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

```
library(foreach)
```

```
# Import a few stocks
mystocks = c("SPY", "TLT", "LQD", "EEM", "VNQ")

#Get the info for these assets
myprices = yahooSeries(mystocks, from='2000-01-01', to='2016-07-30')

# The first few rows
head(myprices)
```

```
## GMT
##            SPY.Open SPY.High  SPY.Low SPY.Close SPY.Volume SPY.Adj.Close
## 2000-01-03 148.2500 148.2500 143.8750  145.4375    8164300      106.9868
## 2000-01-04 143.5312 144.0625 139.6406  139.7500    8089800      102.8029
## 2000-01-05 139.9375 141.5312 137.2500  140.0000   12177900      102.9869
## 2000-01-06 139.6250 141.5000 137.7500  137.7500    6227200      101.3317
## 2000-01-07 140.3125 145.7500 140.0625  145.7500    8066500      107.2167
## 2000-01-10 146.2500 146.9062 145.0312  146.2500    5741700      107.5845
##            TLT.Open TLT.High TLT.Low TLT.Close TLT.Volume TLT.Adj.Close
## 2000-01-03       NA       NA      NA        NA         NA            NA
## 2000-01-04       NA       NA      NA        NA         NA            NA
## 2000-01-05       NA       NA      NA        NA         NA            NA
## 2000-01-06       NA       NA      NA        NA         NA            NA
## 2000-01-07       NA       NA      NA        NA         NA            NA
## 2000-01-10       NA       NA      NA        NA         NA            NA
##            LQD.Open LQD.High LQD.Low LQD.Close LQD.Volume LQD.Adj.Close
## 2000-01-03       NA       NA      NA        NA         NA            NA
```

```
## 2000-01-04       NA       NA       NA         NA          NA             NA
## 2000-01-05       NA       NA       NA         NA          NA             NA
## 2000-01-06       NA       NA       NA         NA          NA             NA
## 2000-01-07       NA       NA       NA         NA          NA             NA
## 2000-01-10       NA       NA       NA         NA          NA             NA
##            EEM.Open EEM.High EEM.Low EEM.Close EEM.Volume EEM.Adj.Close
## 2000-01-03       NA       NA       NA         NA          NA             NA
## 2000-01-04       NA       NA       NA         NA          NA             NA
## 2000-01-05       NA       NA       NA         NA          NA             NA
## 2000-01-06       NA       NA       NA         NA          NA             NA
## 2000-01-07       NA       NA       NA         NA          NA             NA
## 2000-01-10       NA       NA       NA         NA          NA             NA
##            VNQ.Open VNQ.High VNQ.Low VNQ.Close VNQ.Volume VNQ.Adj.Close
## 2000-01-03       NA       NA       NA         NA          NA             NA
## 2000-01-04       NA       NA       NA         NA          NA             NA
## 2000-01-05       NA       NA       NA         NA          NA             NA
## 2000-01-06       NA       NA       NA         NA          NA             NA
## 2000-01-07       NA       NA       NA         NA          NA             NA
## 2000-01-10       NA       NA       NA         NA          NA             NA
```

```
# 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))
}

# Compute the returns from the closing prices
myreturns = YahooPricesToReturns(myprices)

# The sample correlation matrix
cor(myreturns)
```

```
##               SPY.PctReturn TLT.PctReturn LQD.PctReturn EEM.PctReturn
## SPY.PctReturn     1.0000000    -0.4280388    0.10337897     0.8722087
## TLT.PctReturn    -0.4280388     1.0000000    0.42804537    -0.3685517
## LQD.PctReturn     0.1033790     0.4280454    1.00000000     0.1150226
## EEM.PctReturn     0.8722087    -0.3685517    0.11502264     1.0000000
## VNQ.PctReturn     0.7729570    -0.2626517    0.06448117     0.6831243
##               VNQ.PctReturn
## SPY.PctReturn    0.77295703
## TLT.PctReturn   -0.26265173
```

```
## LQD.PctReturn    0.06448117
## EEM.PctReturn    0.68312428
## VNQ.PctReturn    1.00000000
```

Here is where we found how the returns from each of these five asset classes are correlated to each other. For the safer portfolio, we chose to invest approximately equally into the SPY, TLT, and LQD assets, as they have low correlation, and they have the lowest standard deviations among the five, therefore they are the least risky. We chose to allocate 45% of our assets into EEM and VNQ for the risky portfolio, as well as 10% into SPY, because it is highly correlated to the other two. EEM and VNQ also have high standard deviations and are highly correlated to each other, so this model has the potential to have either extremely high returns or high losses.

```
#Get the mean and standard deviation of
mu_SPY = mean(myreturns[,1]) #.000379
sigma_SPY = sd(myreturns[,1])#.01233768
mu_SPY

## [1] 0.0003795515

sigma_SPY

## [1] 0.01233768

mu_TLC = mean(myreturns[,2])#.000344
sigma_TLC = sd(myreturns[,2])#.00896
mu_TLC

## [1] 0.0003437063

sigma_TLC

## [1] 0.008962194

mu_LQD = mean(myreturns[,3])#.00023
sigma_LQD = sd(myreturns[,3])#.00519
mu_LQD

## [1] 0.0002253977

sigma_LQD

## [1] 0.005185087

mu_EEM = mean(myreturns[,4])#.00049
sigma_EEM = sd(myreturns[,4])#.0203
mu_EEM

## [1] 0.0004933146

sigma_EEM

## [1] 0.02027941
```

```
mu_VNQ = mean(myreturns[,5])#.00061
sigma_VNQ = sd(myreturns[,5])#.0211
mu_VNQ
```

```
## [1] 0.0006113327
```

```
sigma_VNQ
```

```
## [1] 0.02106797
```

```
set.seed(54)
# Now loop over two trading weeks
n_days = 20
wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
```

Here we simulated the potential outcomes of many different sets of 20 trading day periods in order to gain a full range of the possible outcomes that could occur for each portfolio and their likelihood. Here, we did this the evenly split portfolio. One of the histograms shown below displays the range of returns expected for this portfolio, while the next one is the same, less the initial $100,000 investment, therefore is the expected range of profits from this portfolio. As can be seen in the histogram, the mean expected return is very low, however the potential losses don't grow too large either. Lastly, we see the value at risk at the 5% level for this portfolio. These steps are repeated for the next two portfolios as well.

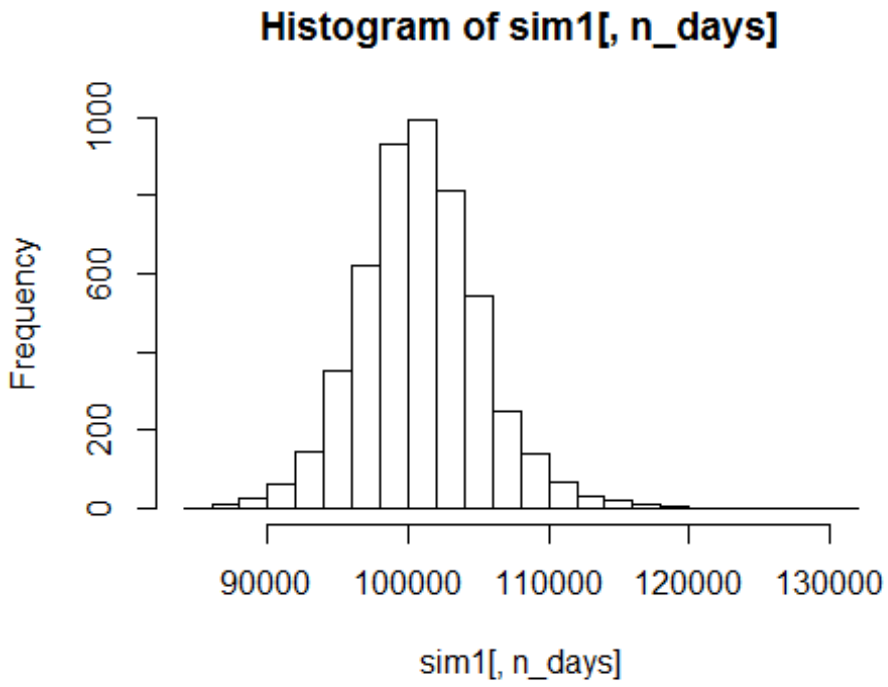
```
# Now simulate many different possible trading years!
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
  }
  wealthtracker
}
```

```
head(sim1)
```

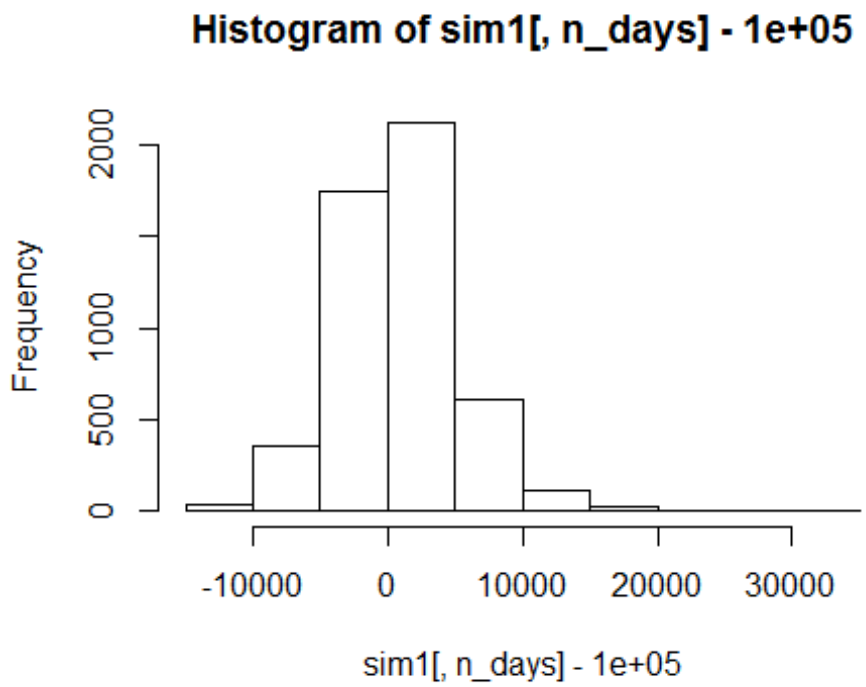
```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## result.1  99949.95 100024.57  99733.25  99543.06  97894.77  97647.02
## result.2 100394.32 100005.11 100354.21  99972.29  97965.77  97863.04
## result.3 100987.50 100726.39 100871.42 100467.75 100579.18 100331.21
## result.4  99913.91 100158.03 100125.76  98030.13  98580.77  98235.72
## result.5  98950.62  99199.03 100810.07 100441.67 100633.56 100444.79
## result.6 101112.50  99657.44  96337.81  95054.74  94720.87  94427.94
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1  98225.14  97801.93  98066.91  97675.71  98377.87  98976.69
## result.2  98205.48  96917.81  97308.26  97719.24  98109.65  98711.72
```

```
## result.3 100850.14 100653.79 100008.02 100678.99 100668.35 100806.18
## result.4  99144.01  98487.22  96012.99  95664.94  95576.68  96366.32
## result.5  99818.22 100000.95 101288.32 101783.94 102213.46 102174.75
## result.6  93572.11  94036.67  94484.34  94090.63  95606.28  96442.52
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1  98770.49  98023.46  97127.76  96659.23  97093.50  97206.01
## result.2  98829.31  98666.43  98709.23  98972.48 100177.46 101157.74
## result.3 101512.72 100384.29 100261.81 100322.11 100282.84  97436.66
## result.4  96083.08  91353.34  90724.33  90802.86  91277.13  90956.07
## result.5 101760.52 102486.13 101203.79 101149.49 102054.50 101943.56
## result.6  96931.66  96764.96  96223.46  96642.69  96705.36  97834.82
##              [,19]     [,20]
## result.1  96770.30  97536.30
## result.2 102132.24 101988.24
## result.3  97892.31  98180.23
## result.4  90392.75  90106.88
## result.5 100520.32 100570.84
## result.6  97526.79  98099.86
```

```
hist(sim1[,n_days], 25)
```




```
# Profit/loss
hist(sim1[,n_days]- 100000)
```




```
# Calculate 5% value at risk
quantile(sim1[,n_days], 0.05) - 100000

##        5%
## -5868.372
```

Here we created the simulation for the safer portfolio. The histogram for this model is even smaller than for the original portfolio, with a lower expected mean as well as smaller potential losses. The assets in this portfolio were the most stable of the five, with low standard deviations indicating that their returns were more likely to be close to where they are expected. Also, the fact that these assets are less correlated to each other indicates that one asset could potentially make up for higher than expected losses of another asset in the portfolio.

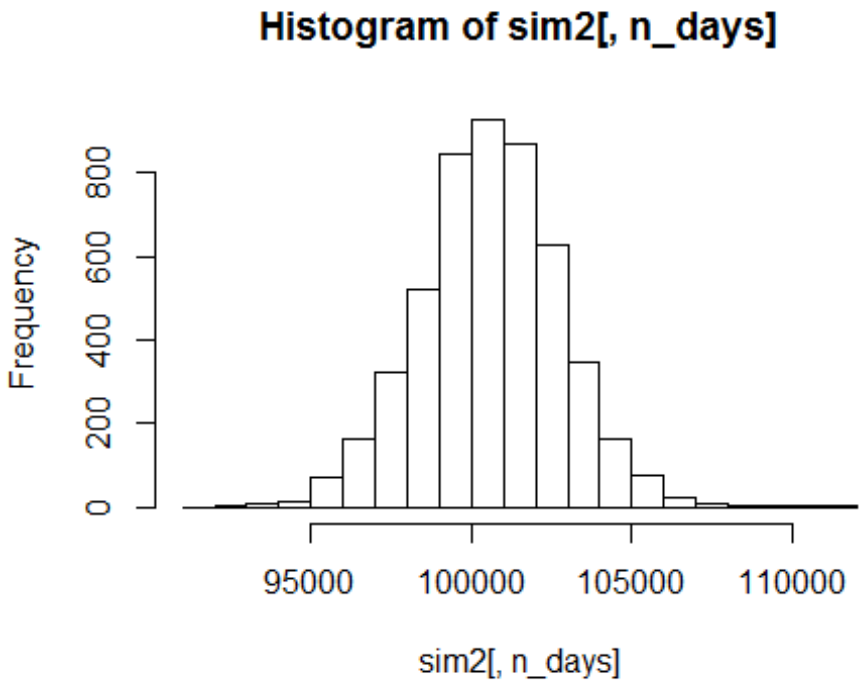
```
# Now simulate many different possible trading years!
sim2 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.33, 0.33, 0.34, 0, 0)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
  }
  wealthtracker
```

```
}
```

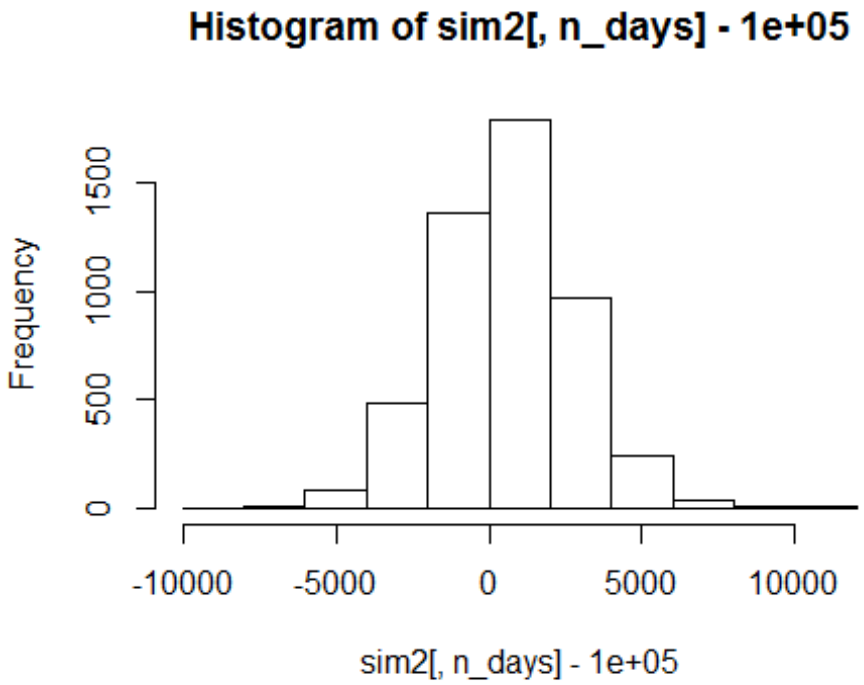
```
head(sim2)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## result.1 100732.74 101288.23 102030.04 102299.79 102075.71 103116.83
## result.2 100251.03 100161.96  99343.39  99375.70  99683.97  99065.89
## result.3 100387.56 100448.42 100386.83 100542.92 101271.98 101614.70
## result.4 100412.39 100434.08 100122.46  99948.43  99859.75  99650.42
## result.5  99929.63  99752.04  99548.23  99803.63  99876.75  99809.93
## result.6  99332.89  99781.21 100321.46 100738.03 100923.97 101087.41
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1 103556.82 102972.81 102471.66 102878.07 103955.94 103754.56
## result.2  99367.10  99358.45  99351.51  99802.00  99675.01  99277.66
## result.3 101348.18 101533.38 101846.41 102201.17 102391.55 102752.49
## result.4 100072.55  99433.43  99227.39  99876.31  99925.29 100260.11
## result.5  99594.17 100140.65  99945.97  99338.66  99527.45  99773.82
## result.6 100193.97  99956.75 100247.51  99853.27  99787.93  99632.86
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1 104392.01 104505.93 104561.73 104478.66 104898.29 103818.52
## result.2  99157.76 100019.87  99078.18  99182.95  98588.44  97567.68
## result.3 102633.93 102889.15 103229.60 103338.42 103230.01 103398.78
## result.4 100098.85  99861.63  99731.16  99214.29  99428.93  99710.95
## result.5  99446.43  99172.13  98621.38  99292.98  99579.19 105789.82
## result.6  99932.15  99451.89  99351.90  98941.82  98969.91  98887.68
##              [,19]     [,20]
## result.1 103887.32 104037.55
## result.2  97449.43  97920.78
## result.3 103330.58 104386.40
## result.4  99554.54 100077.83
## result.5 105714.68 105918.24
## result.6  98241.28  98534.13
```

```
hist(sim2[,n_days], 25)
```




```r
# Profit/loss
hist(sim2[,n_days]- 100000)
```



```
# Calculate 5% value at risk
quantile(sim2[,n_days], 0.05) - 100000
```

```
##       5%
## -3075.88
```

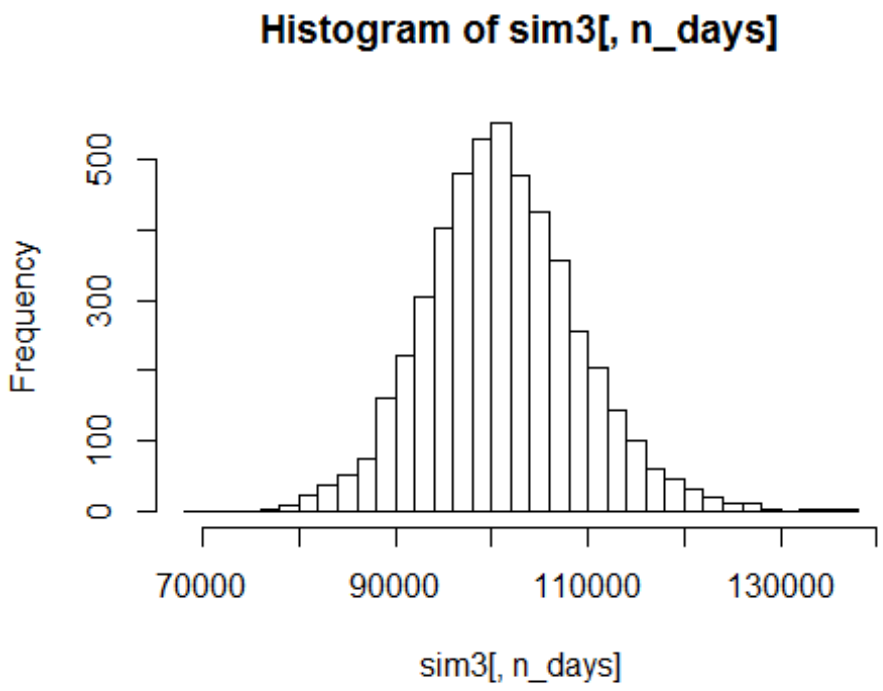
Here, we ran the simulation for the risky portfolio. This portfolios width of potential returns is much wider, as the high standard deviations lead to potential deviations into both higher profits and bigger losses. These profits or losses are made even more heightened due to the fact that these assets are highly correlated, meaning if one asset goes up or down in value, the others are more likely to come with it, making this portfolio much less diversified and more risky, with a much higher value at risk at the 5% level.

```
# Now simulate many different possible trading years!
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.1, 0, 0, 0.45, 0.45)
  holdings = weights * totalwealth
  wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
  }
  wealthtracker
}

head(sim3)
```

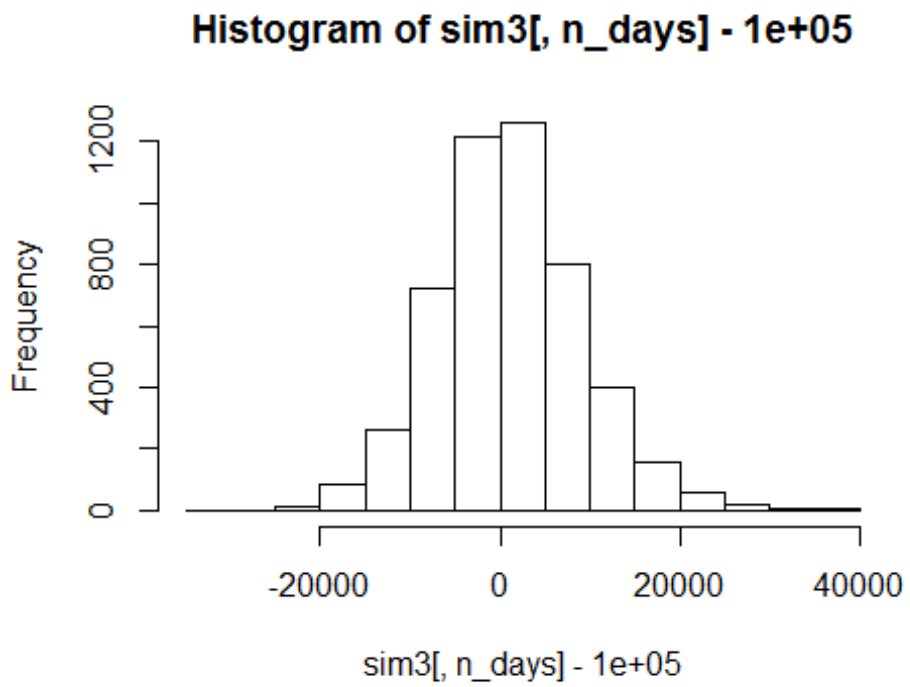
```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## result.1 101267.73 100291.89  96193.03  95041.49  94185.26  95090.35
## result.2 100408.62 103241.94  99760.07 100400.36  99513.56  99890.26
## result.3 103027.03 102896.26 103055.39 101137.22 100218.25 100682.27
## result.4 102882.00 100126.28 100506.54 100171.47 100832.91 101416.51
## result.5 107913.56 109501.09 116797.59 117233.89 118539.46 119263.98
## result.6  99239.19  96182.83  96044.33  96129.99  94549.29  94938.11
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1  95041.92  93599.72  93193.31  93336.62  93310.86  90833.52
## result.2 100897.62 101552.17 101605.39 102016.32 103025.78 104925.68
## result.3  99616.86 100513.71 100444.92 101929.55 101977.69 102427.18
## result.4  99996.31  97869.33 100106.11  98713.95  99488.34  98916.42
## result.5 110954.97 109837.90 110775.28 110082.33 110670.55 110199.04
## result.6  96719.80  97236.94  96186.19  96302.49  97288.22  98390.83
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1  90390.53  92733.17  93957.02  95579.25  95522.34  95506.36
## result.2 105140.19 105197.31 105184.85 106153.29 107693.56 108735.81
## result.3 102126.95 102490.53 102962.04 103589.66 104417.75 105140.18
## result.4 100415.10  99727.11 100237.28  99226.44 100667.78  99917.21
## result.5 110913.44 113228.17 113748.58 114797.05 113823.70 116841.42
```

```
## result.6  99633.98 101250.88 102049.00  98876.37  98294.95  98268.23
##              [,19]     [,20]
## result.1  95155.71  95383.70
## result.2 109716.13 106832.27
## result.3 105797.51 106320.06
## result.4  99633.40  95124.81
## result.5 117705.75 119203.04
## result.6  99433.51  99380.52
```

```
hist(sim3[,n_days], 25)
```




```
# Profit/loss
hist(sim3[,n_days]- 100000)
```




```
# Calculate 5% value at risk
quantile(sim3[,n_days], 0.05) - 100000

##       5%
## -11260.2
```

Market Segmentation

```
library(ggplot2)

twitterdata = read.csv('social_marketing.csv')

twitterdata = twitterdata[,-1]

#are there any topics that appear to be grouped together?

Z = scale(twitterdata, center= TRUE, scale=TRUE)

kmeans_tweets = kmeans(Z, 6, nstart = 50)

df = data.frame(Z)
df$cluster = factor(kmeans_tweets$cluster)

print(apply(kmeans_tweets$centers,1,function(x) colnames(Z)[order(x,
decreasing=TRUE)[1:7]]))
```

```
##      1                2                3                4
## [1,] "politics"       "religion"       "adult"          "online_gaming"
## [2,] "news"           "parenting"      "spam"           "college_uni"
## [3,] "travel"         "sports_fandom"  "chatter"        "sports_playing"
## [4,] "computers"      "food"           "tv_film"        "tv_film"
## [5,] "automotive"     "school"         "shopping"       "music"
## [6,] "business"       "family"         "art"            "art"
## [7,] "small_business" "crafts"        "current_events" "small_business"
##      5                6
## [1,] "cooking"        "health_nutrition"
## [2,] "fashion"        "personal_fitness"
## [3,] "beauty"         "outdoors"
## [4,] "photo_sharing"  "eco"
## [5,] "music"          "food"
## [6,] "uncategorized"  "cooking"
## [7,] "shopping"       "dating"
```

From the cluster analysis using K means, we were able to identify six very clearly distinguished clustered around the twitter topics. We also used a ggplot to visually confirm that these clusters were somewhat uniform on certain topics that were related to them individually.

The first cluster reminds us of what could a typical American family with children. Sort of like a regular household. The top categories associated with this cluster were: religion, parenting, sports fandom, food, school, family, crafts. This are very evident representations of a classic household.

The second cluster is most likely a young people cluster made up of college students and/or young professionals. The categories strongly associated with this cluster were: online gaming, college, sports playing, tv/film, art, music, small business.

The third cluster is likely a more feminine set of users. They were interested in topics such as cooking, fashion, beauty, photo sharing, music, and shopping.

The fourth cluster was somewhat random without a clear thread. This one contained all of the 'junk' found in the data, such as adult, spam, and chatter.

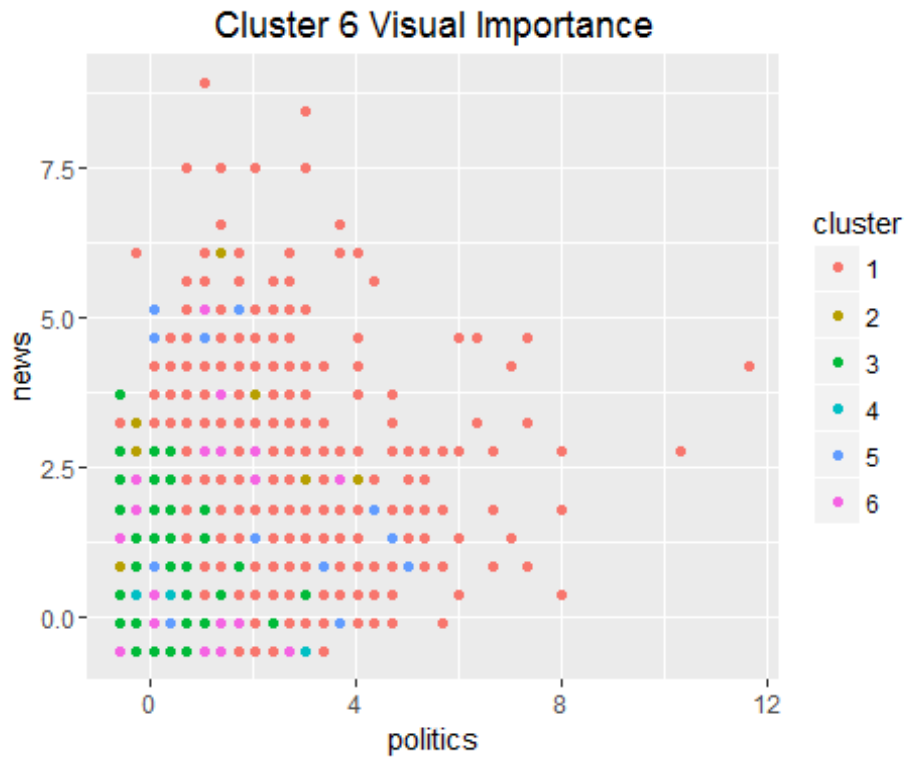
The fifth cluster most likely represents another side of the millennial generation. They were particularly involved with health and nutrition, personal fitness, outdoors, eco-friendly things, food, cooking, and dating. These represent the 'granola', outdoors-type people that may enjoy a city like Austin.

The final cluster contains individuals that are global in their mindset. The dealt with topics related to politics, news, travel, business, and small business. These are people that like to keep up with the state of the world.

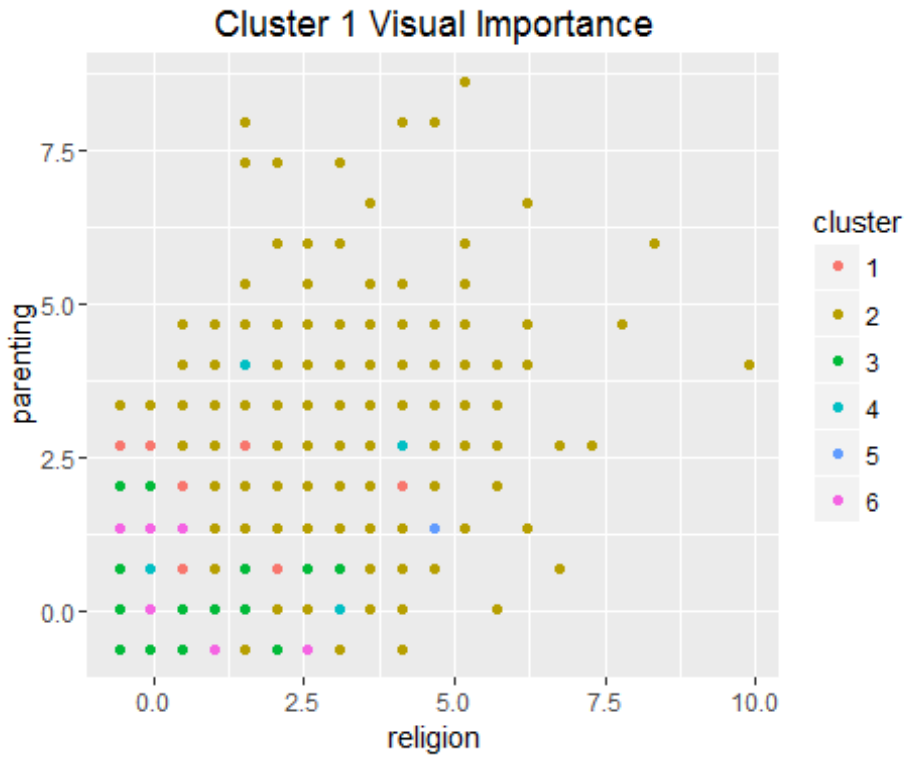
Overall, 5 of these 6 clusters have very identifiable people within them. It would be fairly simply for the company to target their marketing efforts to these groups with their characteristics in mind.

Below, we will show a ggplot for each cluster (except cluster 4 which is the junk/noise cluster) that accentuates interests that are strong within them. Essentially, each graph will show an overwhelming amount of observations for a given cluster corresponding to its color in the legend.

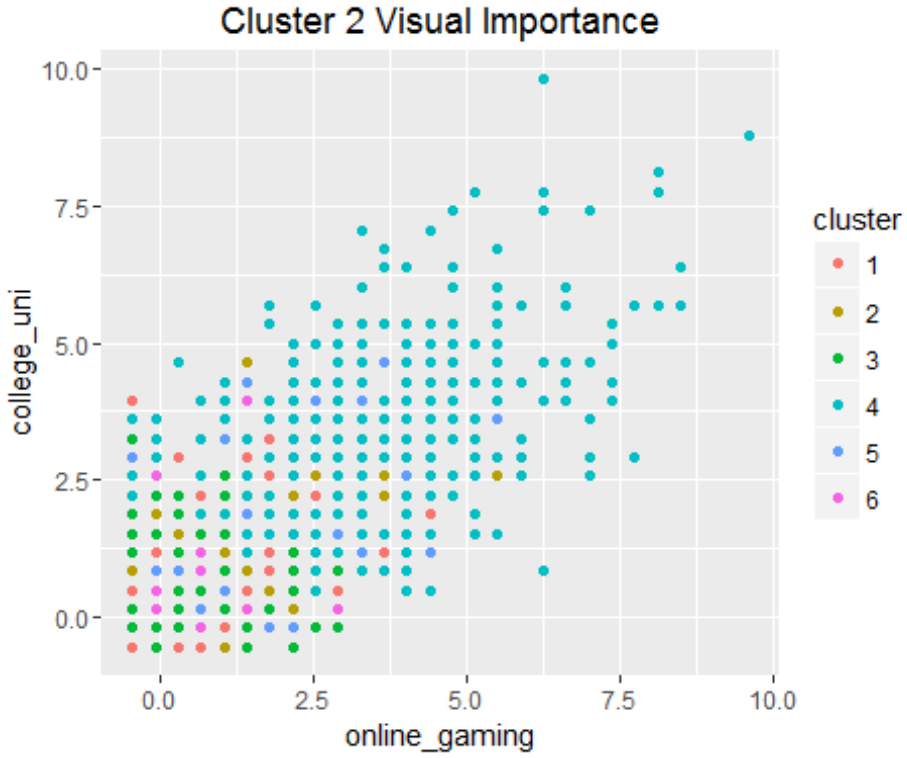
```
p1 = ggplot(data= df, aes(x=politics, y=news, color=cluster)) + geom_point()
p1 + labs(title = 'Cluster 6 Visual Importance')
```




```
p2 = ggplot(data= df, aes(x=religion, y=parenting, color=cluster)) +
geom_point()
p2 + labs(title = 'Cluster 1 Visual Importance')
```




```
p3 = ggplot(data= df, aes(x=online_gaming, y=college_uni, color=cluster)) +
geom_point()
p3 + labs(title = 'Cluster 2 Visual Importance')
```



```
p4 = ggplot(data= df, aes(x=fashion, y=beauty, color=cluster)) + geom_point()
p4 + labs(title = 'Cluster 3 Visual Importance')
```


```
p5 = ggplot(data= df, aes(x=personal_fitness, y=outdoors, color=cluster)) +
geom_point()
p5 + labs(title = 'Cluster 5 Visual Importance')
```


Cluster 5 Visual Importance
outdoors
7.5
5.0
2.5
0.0
0
2
4
6
personal_fitness
cluster
1
2
3
4
5
6