Final Exam

Boran Sheu & Noah Shimizu

2022-08-05

https://github.com/NoahCSCS/Stat-380-Exercises:

Probability Practice

Part A

Let Y refer to clicking yes, and RC and TC refer to the events of being a random or truthful clicker, respectively. Then by basic laws of probability we have :

$$\begin{aligned}
P(TC \cap Y) + P(RC \cap Y) &= P(Y) \\
P(TC) * P(Y|TC) + P(RC) * P(Y|RC) &= P(Y) \\
0.7 * P(Y|TC) + 0.3 * 0.5 &= 0.65 \\
P(Y|TC) &= \frac{0.65 - 0.3 * 0.5}{0.7} \\
P(Y|TC) &= \frac{5}{7}
\end{aligned}$$

5/7 of truthful clickers answered yes.

Part B

Let T and D denote the events of testing positive and being positive for the disease, respectively. Then by Baye's rule:

$$\begin{aligned}
P(D|T) &= \frac{P(D \cap T)}{P(T)} \\
P(D|T) &= \frac{P(T|D) * P(D)}{P(T|D) * P(D) + P(T|D^C) * P(D^C)} \\
P(D|T) &= \frac{0.993 * 0.000025}{0.993 * 0.000025 + 0.0001 * 0.9975} \\
P(D|T) &\approx 0.1992
\end{aligned}$$

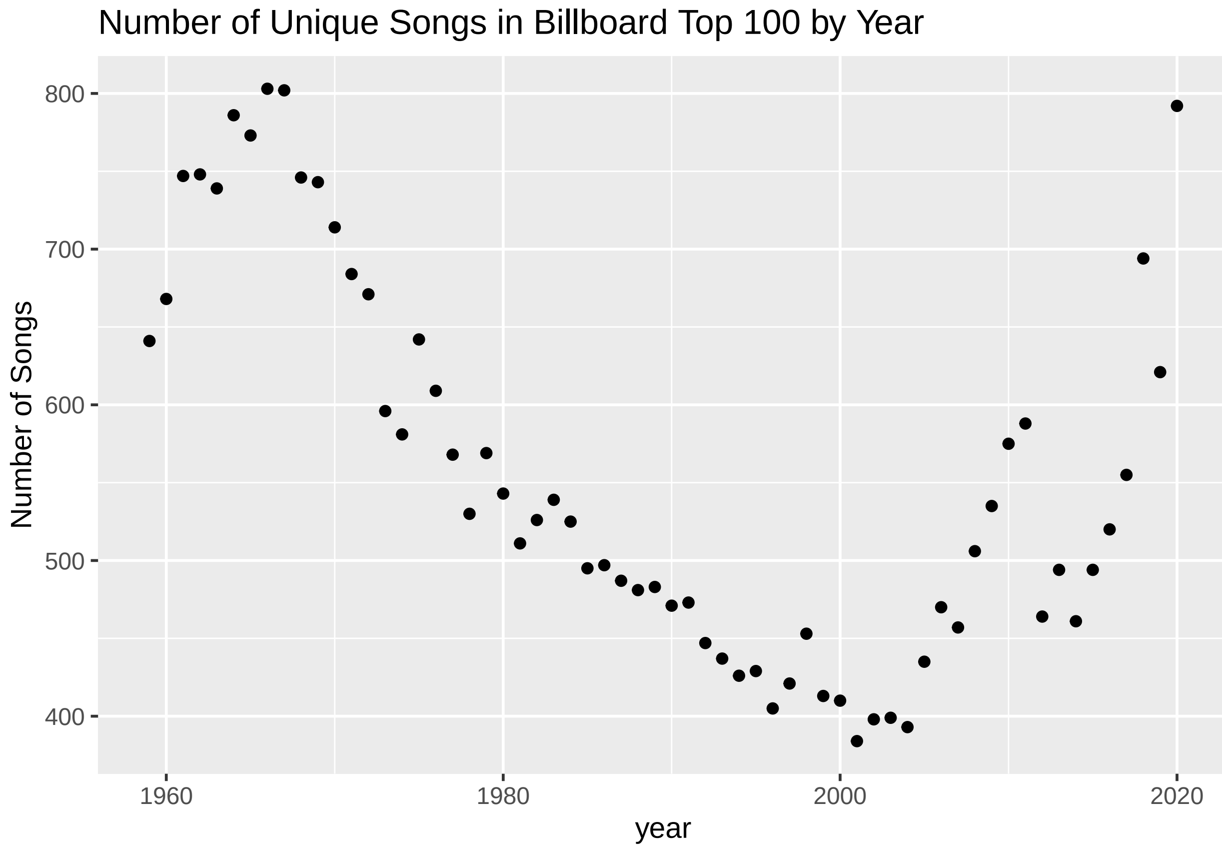
or 19.92 % chance of having the diesase among those who test positive.

Wrangling the Billboard Top 100

A

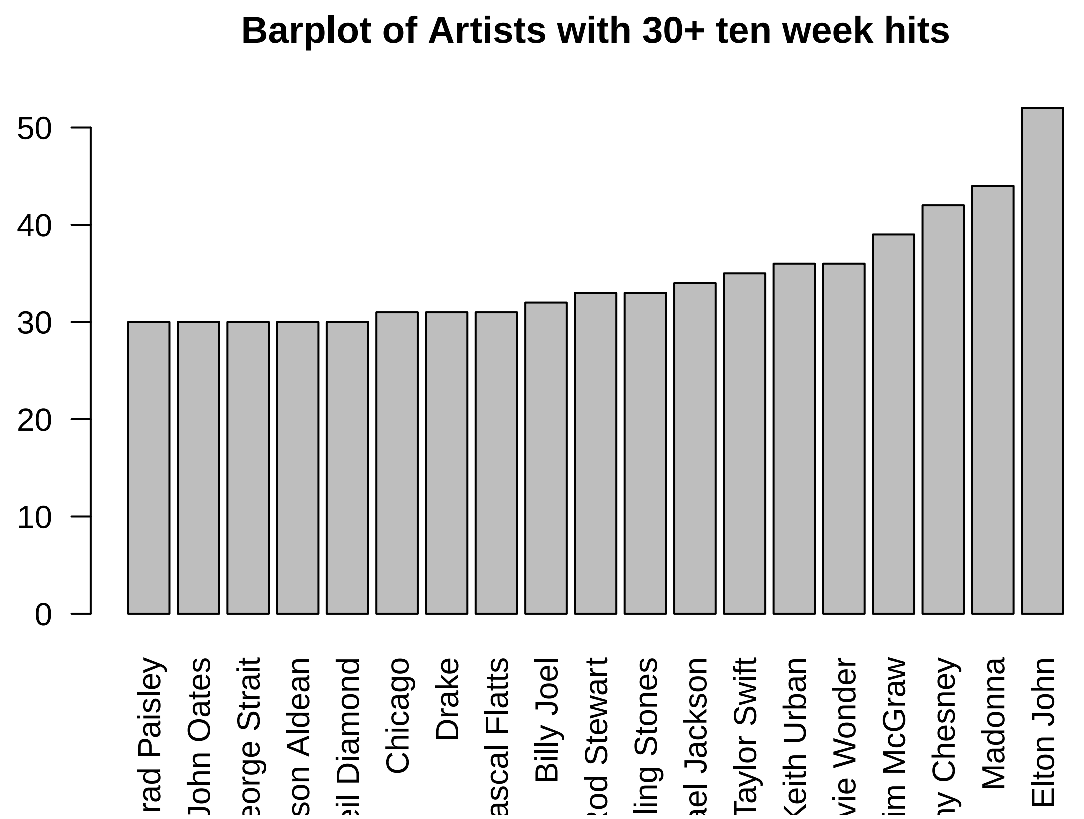
```
## # A tibble: 20 x 3
## # Groups:   song [20]
##    song                                performer                           count
##    <chr>                               <chr>                               <int>
##  1 Radioactive                         Imagine Dragons                      3828
##  2 Sail                                AWOLNATION                           3160
##  3 Blinding Lights                     The Weeknd                           2926
##  4 I'm Yours                           Jason Mraz                           2926
##  5 How Do I Live                       LeAnn Rimes                          2415
##  6 Counting Stars                      OneRepublic                          2346
##  7 Party Rock Anthem                   LMFAO Featuring Lauren Bennett & G~  2346
##  8 Foolish Games/You Were Meant For Me Jewel                                2145
##  9 Rolling In The Deep                 Adele                                2145
## 10 Before He Cheats                    Carrie Underwood                     2080
## 11 Ho Hey                              The Lumineers                        1953
## 12 I Hope                              Gabby Barrett Featuring Charlie Pu~  1953
## 13 You And Me                          Lifehouse                            1953
## 14 Circles                             Post Malone                          1891
## 15 Demons                              Imagine Dragons                      1891
## 16 Macarena (Bayside Boys Mix)         Los Del Rio                          1830
## 17 Need You Now                        Lady Antebellum                      1830
## 18 All Of Me                           John Legend                          1770
## 19 Somebody That I Used To Know        Gotye Featuring Kimbra               1770
## 20 How To Save A Life                  The Fray                             1711
```

B



We wanted to determine if musical diversity (ie. the number of unique songs in a given year) has changed with time. We have thus plotted the number of unique songs for each year. We see that the number of unique songs in 1959 is about 640. It then generally increases to 800 at around 1967, only to then begin decreasing drastically, till around 2001, where it bottoms out below 400. Finally, starting at around 2003, it increases rapidly, until the most recent year of 2020, where it is again almost at 800.

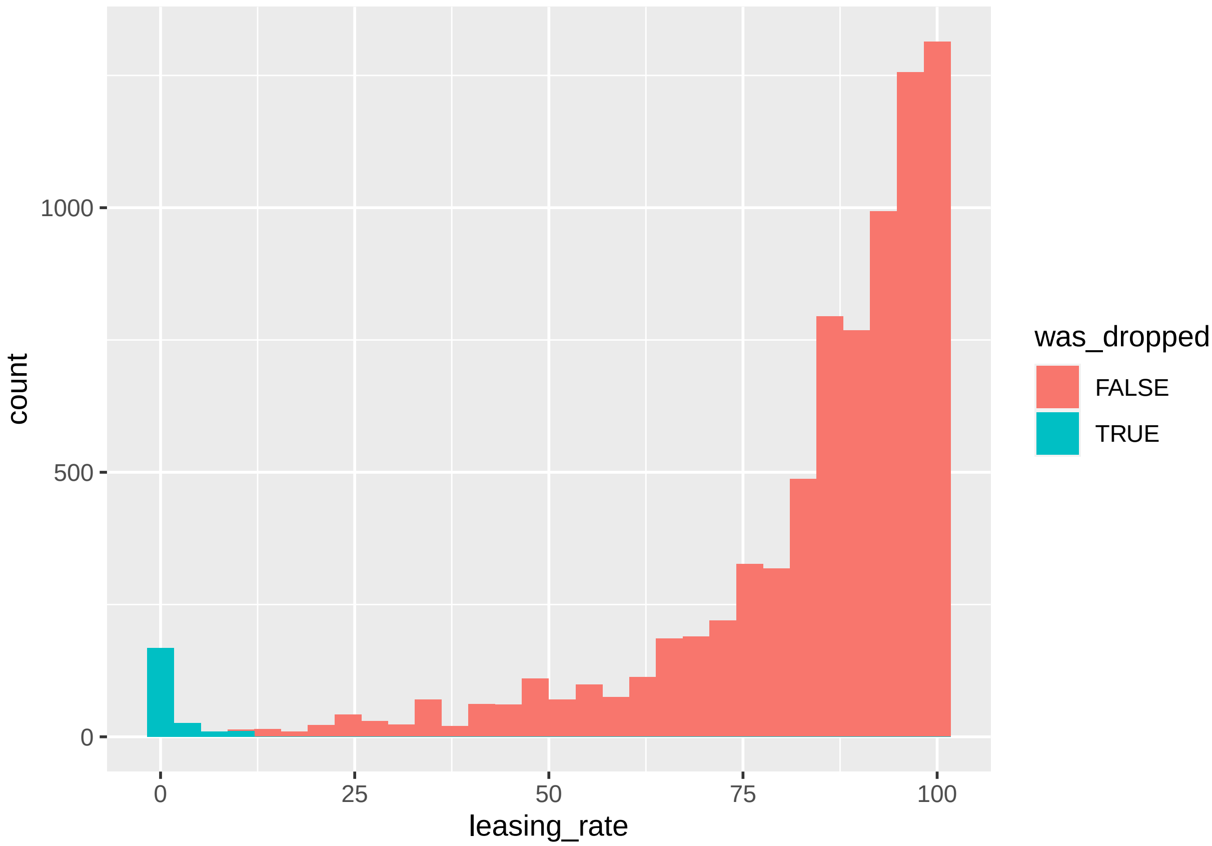
C



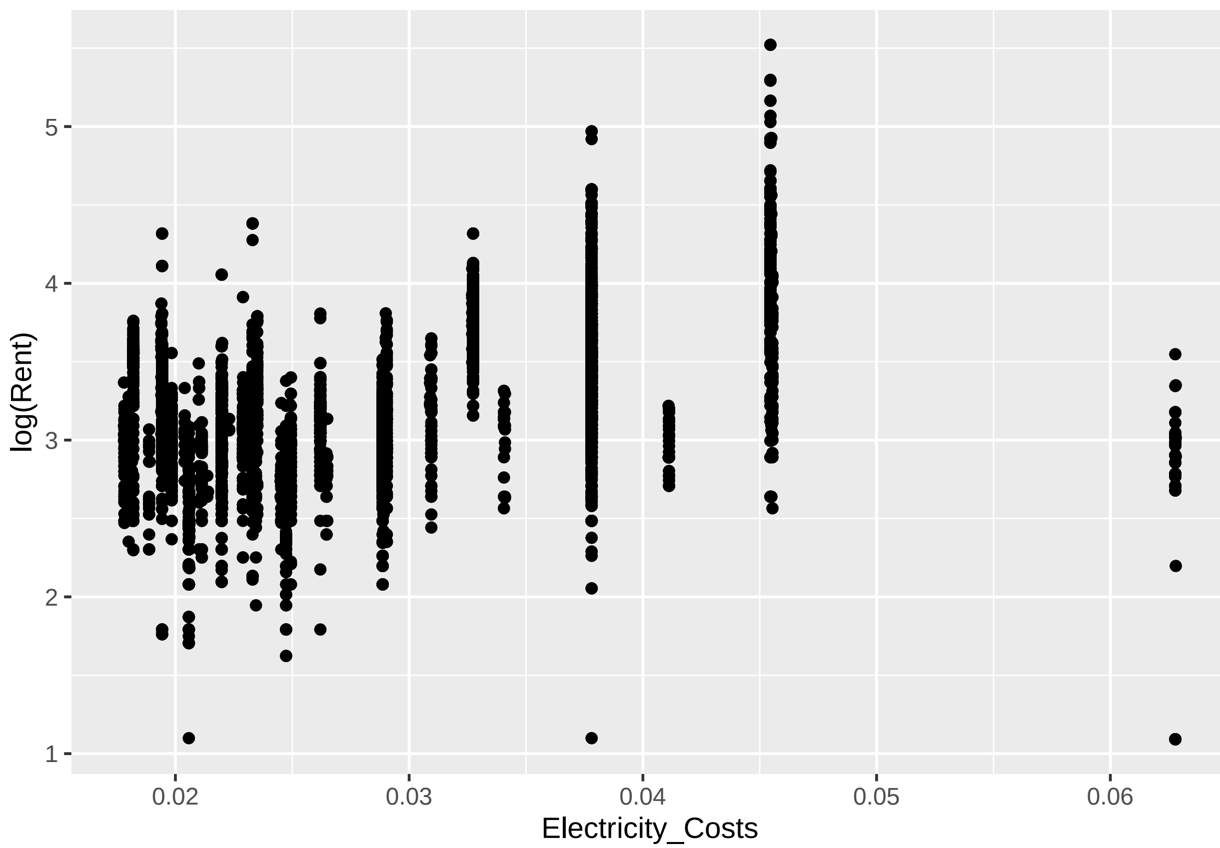
Well, Elton John has obviously the highest number of songs that have ten weeks hit. And total of 19 singers have 30 or more ten weeks hit songs. Except Elton John, Tim McGraw, Michael Jackson and Madonna are also among the top tier of the list. Quite interesting is that, I never heard of Tim McGraw in Taiwan!!

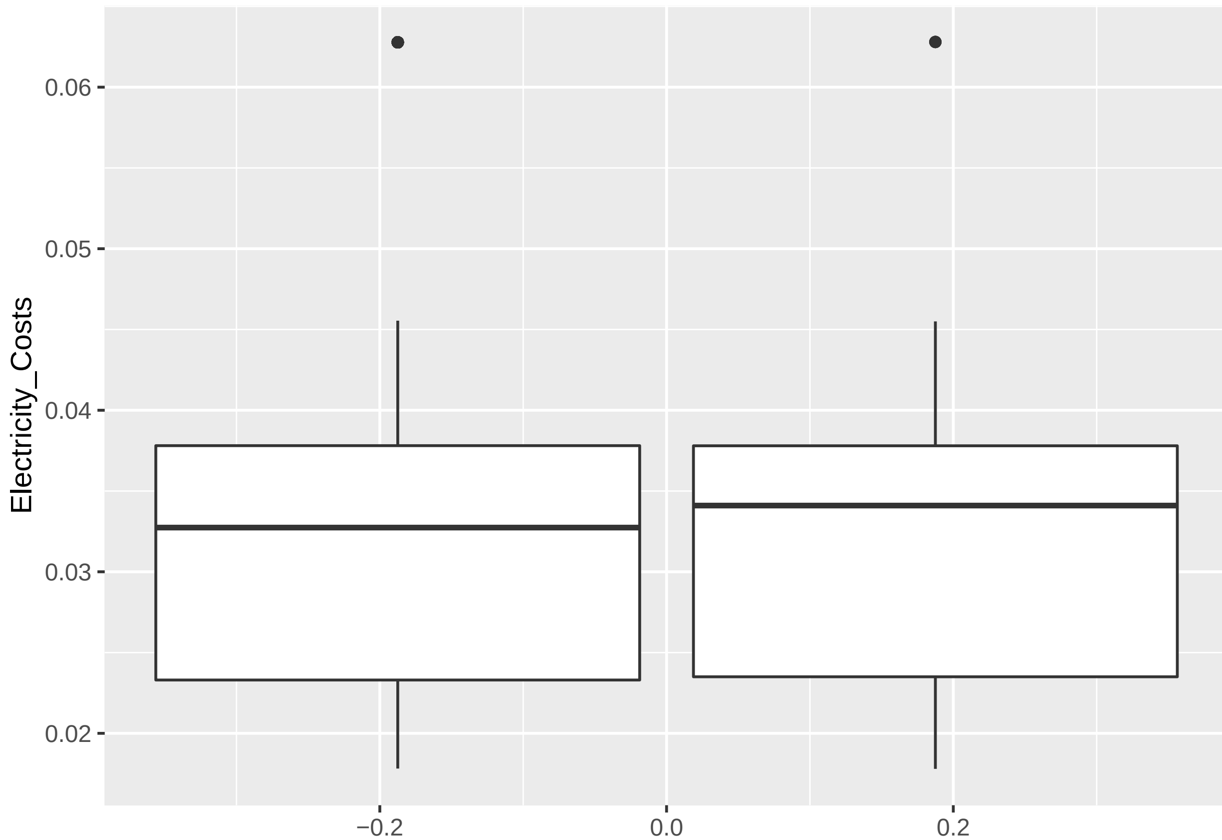
Visual Story Telling Part 1: Green Buildings

We first begin with dropping the low occupancy building. While the below histogram does agree that there appears to be an outlying cluster of low occupancy buildings. These are likely un-representative of our new building, as it is being built in Austin, where housing is lacking. However, we choose not to omit them, as we prefer to have as much data in our analysis as possible, and we cannot be sure that they will hurt our analysis. We can model only on the high occupancy buildings, and see if that performs the accuracy of our model, yet this again assumes that we know ahead of time that our building won't be one of these low occupancy buildings, which we cannot be 100% certain that it will be.



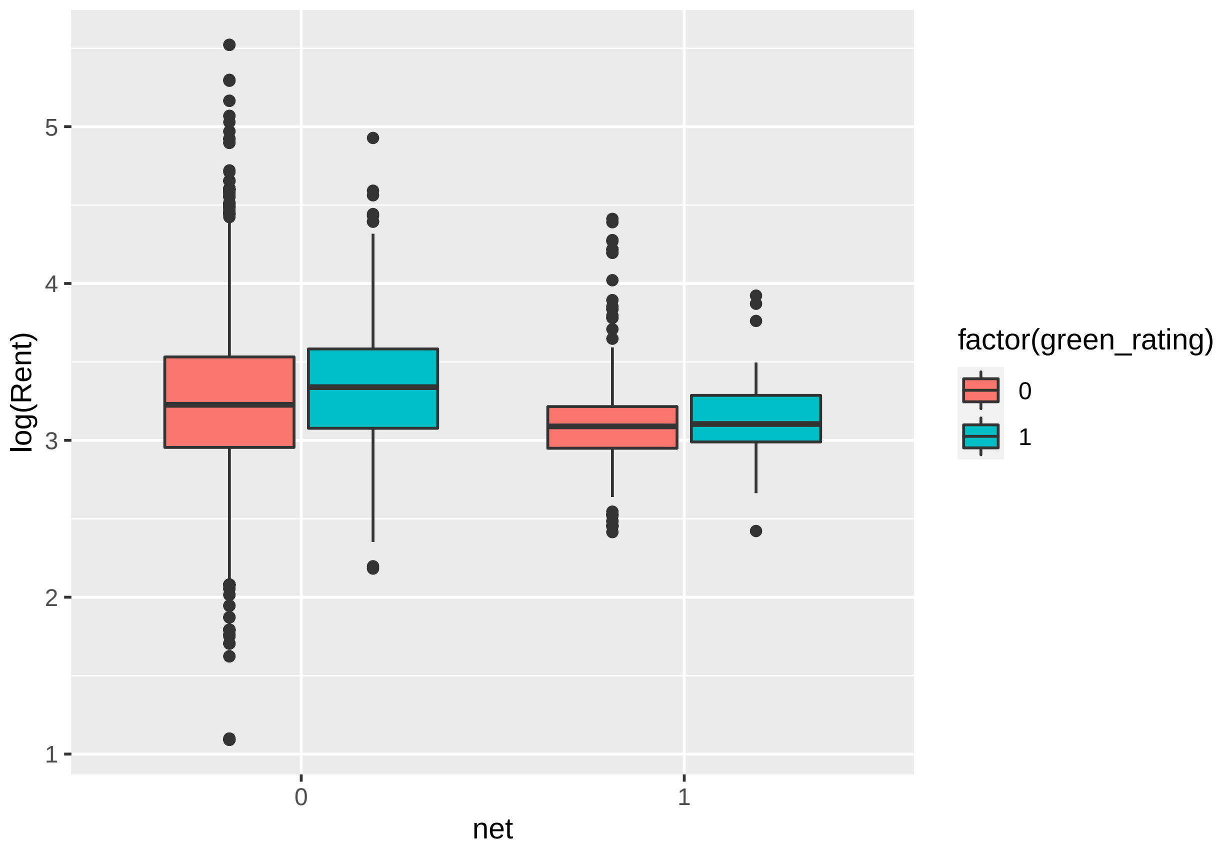
Next, we move on to the stats guru's rent predictions. He first assumes that there are no confounding variables between the interaction between rent and green status. For instance, since one of the main advantages of green buildings is lower energy usage, we might expect that green buildings are more enticing to builders in high energy cost areas. We see this in the boxplot below, where green rating buildings have on median a greater electricity cost then non-green-rated buildings. High electricty cost buildings also tend to have greater rents however, as seen in the dotplot below, perhaps because these costs are passed down to consumers, or just that prices tend to be higher in higher energy cost areas, due to general wealth.


log(Rent)
5
4
3
2
1
0.02
0.03
0.04
0.05
0.06
Electricity_Costs



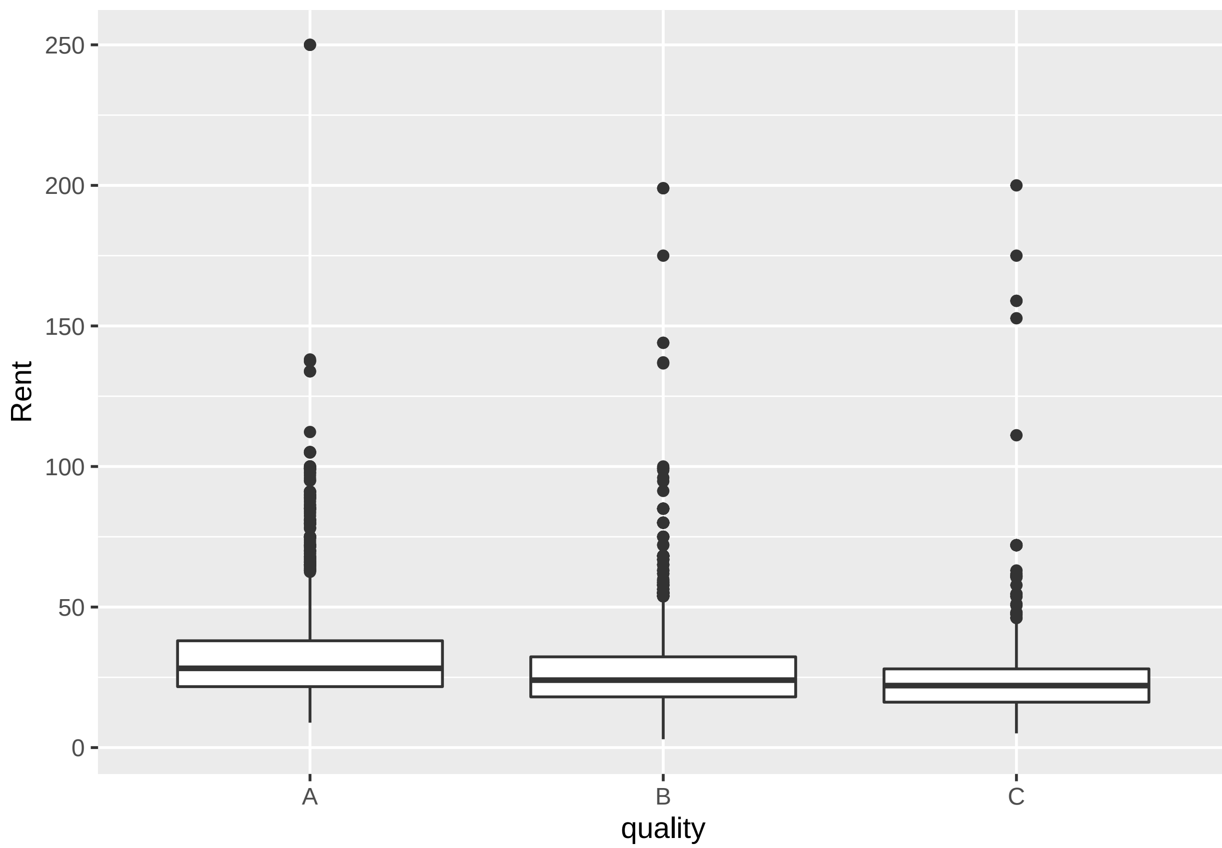
We could try and take this variable into account in a more advance regression model. We could also account for facts about the area by predicting rent-cluster_rent, or in other words how much higher rent green buildings are than non-green_buildings in each given area.

Another important fact unaccounted for by the model is that some buildings charge their individuals for utilities, while others do not. Charging for utilities separate from rent is called a net contract. We see in the below boxplot that green_rating and net contracts have some interaction, (ie. In buildings where utilties are charged for, rent is on median the same, regardless of whether the building is green rated. However in non net contract apartments, green rating buidlings come at a premium). We should use this to note that perhaps green rated buildings are more profitable when they are not on a net contract, and we can perhaps expect a higher premium if we place users on a net contract.



Another area where we could see other variables taken into account is building quality. As we see, green buildings are disproportionately good, and good buildings have higher rents. This variable however is difficult to fully take into account, as a building being green contributes to the quality of the building.

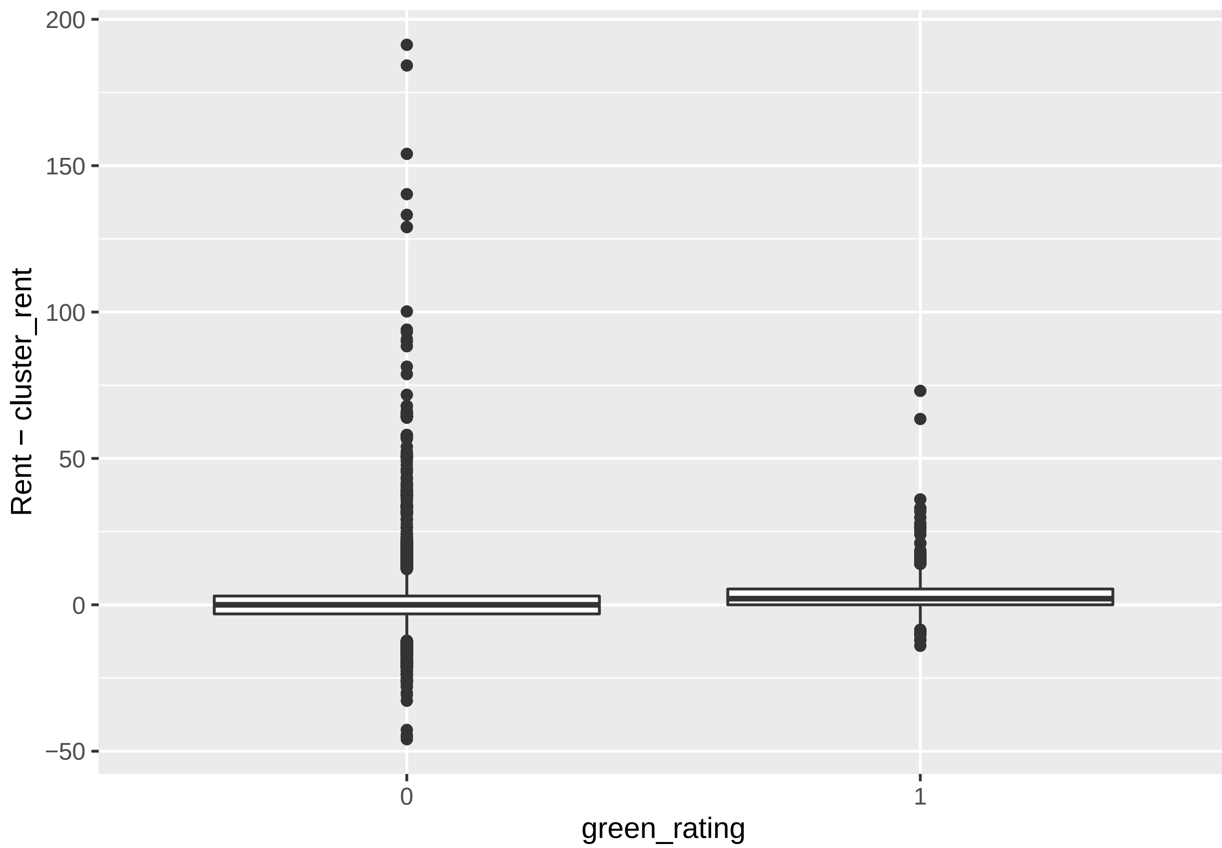
```
##
##          0    1
##   A 2611  546
##   B 3495  132
##   C 1103    7
```



One way we can try to account for these things is rather than looking at how much rent

We also note that the stats wizard makes that rent and occupancy rate are not related. If rents are too high or low, it might lead to changes in occupancy rate. It would be perhaps better if we predicted revenue, ie. rent * occupancy * square footage, rather than just rent, and asusming occupancy is fixed.

Overall, it is difficult to fix many of these issues without running advanced statistical models. However, one quick fix is assuming that non-green factors a buildings is a function of proximity, ie. if one thing is affecting a building's rent, then it is likely effecting the rent of nearby buildings as well. Thus, our modeling for how much green buildings may charge in extra rent is how much more rent it charges then the average rent of those building's in it's cluster. Note that each cluster contains 1 green certified building, and many non-green certified building, meaning that we need not worry about our clustering placing all green buildings together, and thus not detecting the green-building premium. The results of this clustering is seen below.

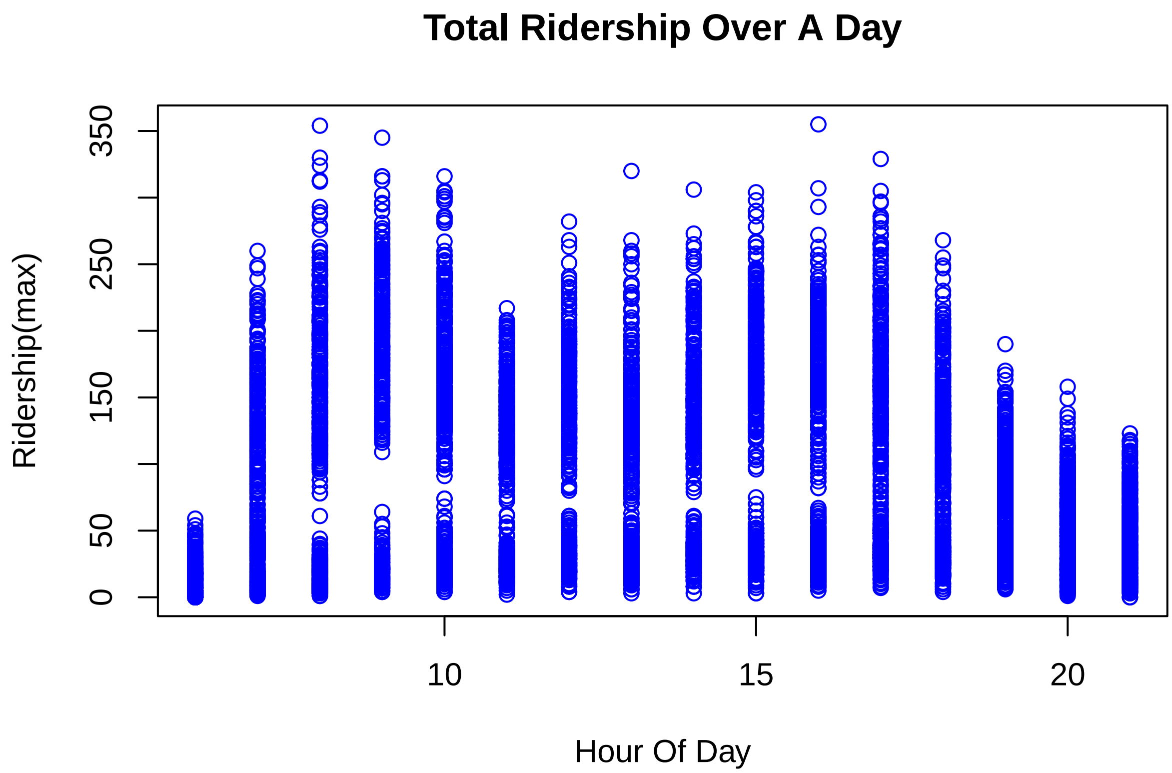



```
## [1] "Excess Rent: 2.41245805577578"

## # A tibble: 2 x 4
##   green_rating  Rent cluster_rent excess_rent
##   <fct>        <dbl>        <dbl>       <dbl>
## 1 0             28.3         27.6       0.712
## 2 1             30.0         26.9       3.12
```

Here, we see green rating has an excess rent of 3.1244 in dollars per square root per year, as compared to the default 0.712's excess rent. We thus replace the $2.6 number with $2.41. Repeating their arithmatic is left as an excercize to the grader :)

Capital Metro Data

For each hour in the day, we look at the ridership.

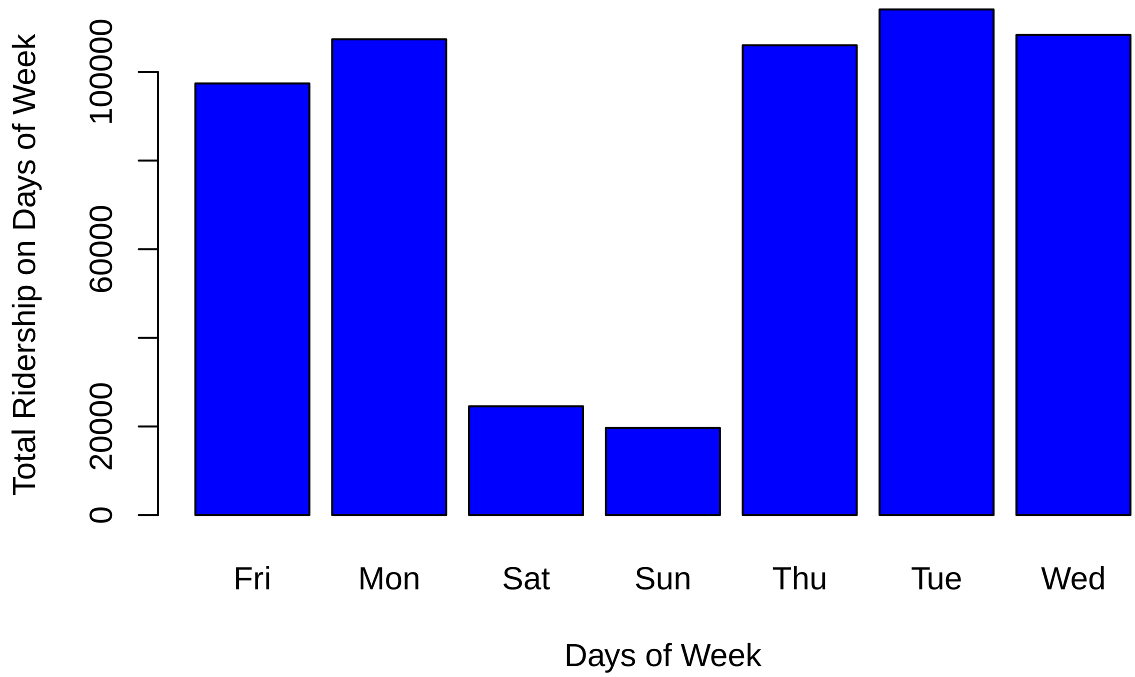


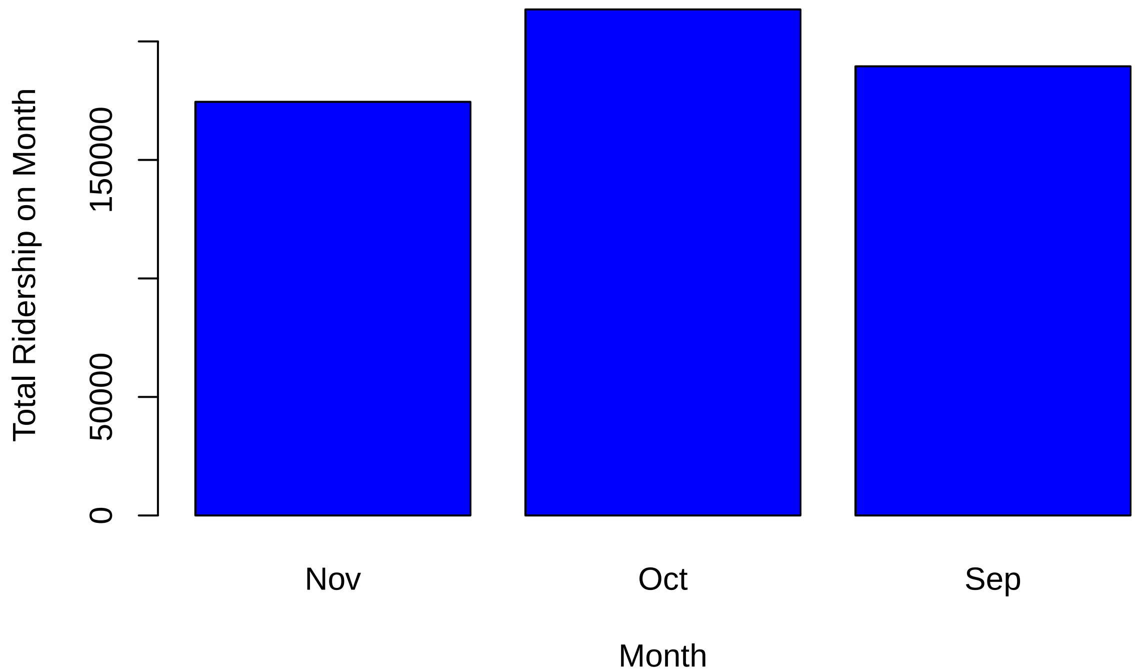
From the plot we can see that the ridership were initially low at the beginning of a day (6am), as the commute time starts, which is around 7am - 10am, the ridership goes up fast. And then it starts to drop after 10 am, which most students have all gone to school. At 4pm, students are off school and the ridership increases till 6pm.

Next, combine the month and day of month.

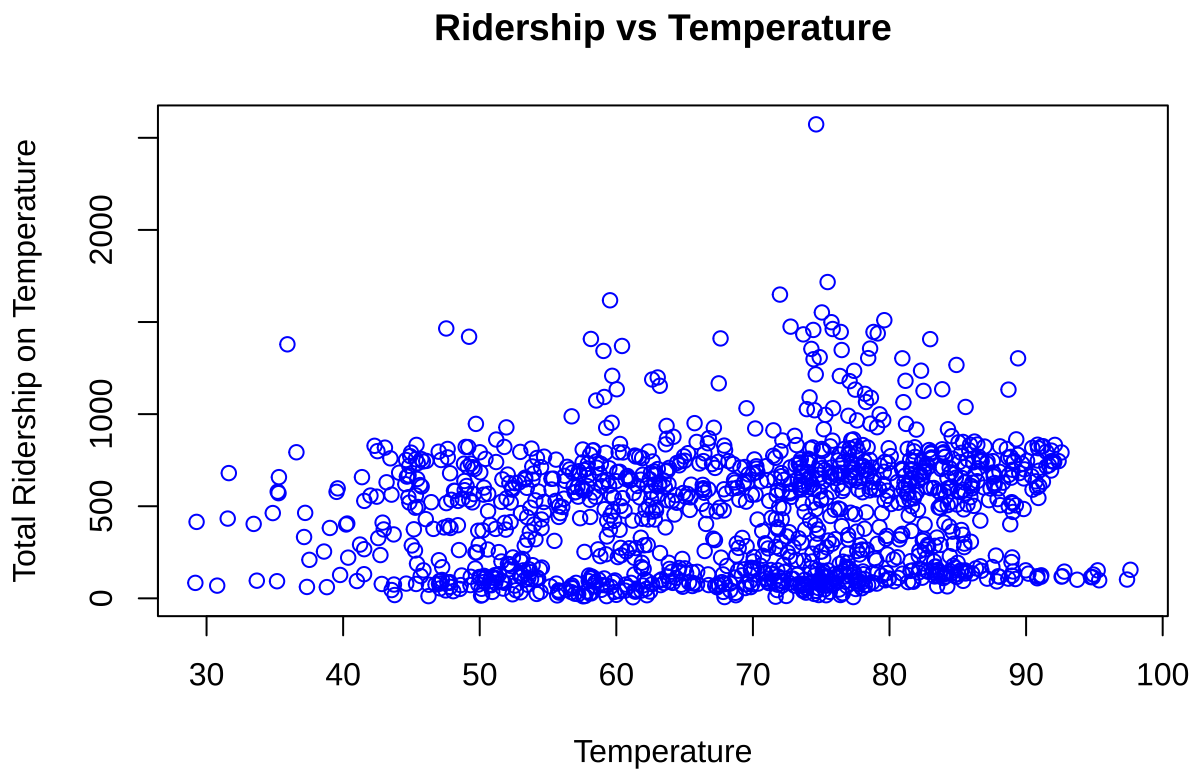
```
##    df2$day_of_week df2$month df2$Total_ridership
## 1              Fri       Nov              32162
## 2              Mon       Nov              34055
## 3              Sat       Nov               6890
## 4              Sun       Nov               5608
## 5              Thu       Nov              35001
## 6              Tue       Nov              32824
## 7              Wed       Nov              27969
## 8              Fri       Oct              31898
## 9              Mon       Oct              44724
## 10             Sat       Oct               8123
## 11             Sun       Oct               6643
## 12             Thu       Oct              34682
## 13             Tue       Oct              44229
## 14             Wed       Oct              43190
## 15             Fri       Sep              33343
## 16             Mon       Sep              28593
## 17             Sat       Sep               9543
## 18             Sun       Sep               7424
## 19             Thu       Sep              36340
## 20             Tue       Sep              37051
```

```
## 21             Wed       Sep               37204

##   df2$month df2$Total_ridership
## 1       Nov              174509
## 2       Oct              213489
## 3       Sep              189498

##    df2$month df2$day_of_week df2$Total_ridership.x df2$Total_ridership.y
## 1        Nov             Fri                 32162                174509
## 2        Nov             Mon                 34055                174509
## 3        Nov             Sat                  6890                174509
## 4        Nov             Sun                  5608                174509
## 5        Nov             Thu                 35001                174509
## 6        Nov             Tue                 32824                174509
## 7        Nov             Wed                 27969                174509
## 8        Oct             Fri                 31898                213489
## 9        Oct             Mon                 44724                213489
## 10       Oct             Sat                  8123                213489
## 11       Oct             Sun                  6643                213489
## 12       Oct             Thu                 34682                213489
## 13       Oct             Tue                 44229                213489
## 14       Oct             Wed                 43190                213489
## 15       Sep             Fri                 33343                189498
## 16       Sep             Mon                 28593                189498
## 17       Sep             Sat                  9543                189498
## 18       Sep             Sun                  7424                189498
## 19       Sep             Thu                 36340                189498
## 20       Sep             Tue                 37051                189498
## 21       Sep             Wed                 37204                189498
```


Total Ridership on Days of Week
0
20000
60000
100000
Fri
Mon
Sat
Sun
Thu
Tue
Wed
Days of Week



From the two bar plots we wouldn't say that there is obvious patterns on the three months we have. We can say that October has the highest total ridership while November has the lowest.But there is clearly a pattern on weekdays. Weekdays have much more riderships than weekends and Fridays have fewer ridership compare to other days. That might be some colleges, like us MSBA program, don't have lectures on Friday.



A plot showing no clear trend between ridership numbers and temperature

There is no obvious pattern between temperature and total ridership.

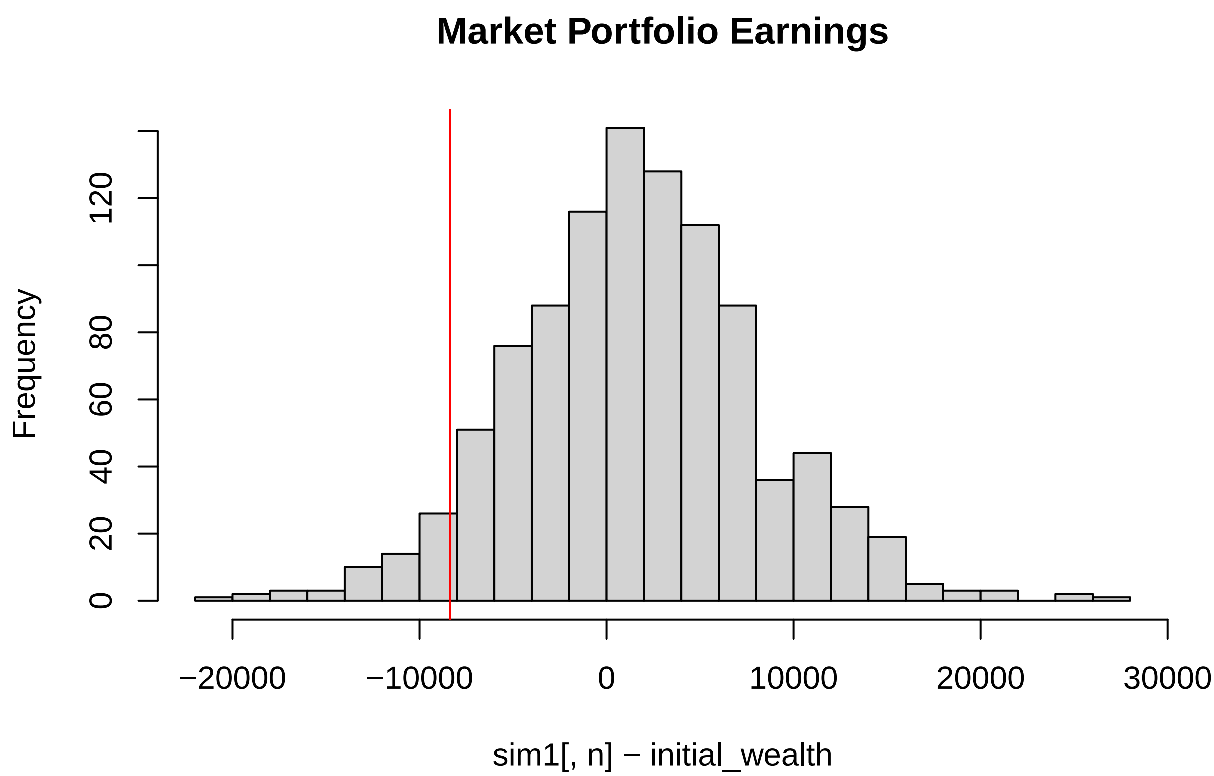
Portfolio Modeling

Portfolio 1: Market

Portfolio 1 is built to act like the market, using etf's that each are meant to track the market.

- **33% SPY** ETF designed to track S&P 500
- **33% QQQ** ETF that tracks NASDAQ-100
- **33% IWM** iShares Russell 2000 ETF, tracks 2000 small cap companies

We know plot the bootstrap distribution of change in wealth, as well as the VaR of 0.05



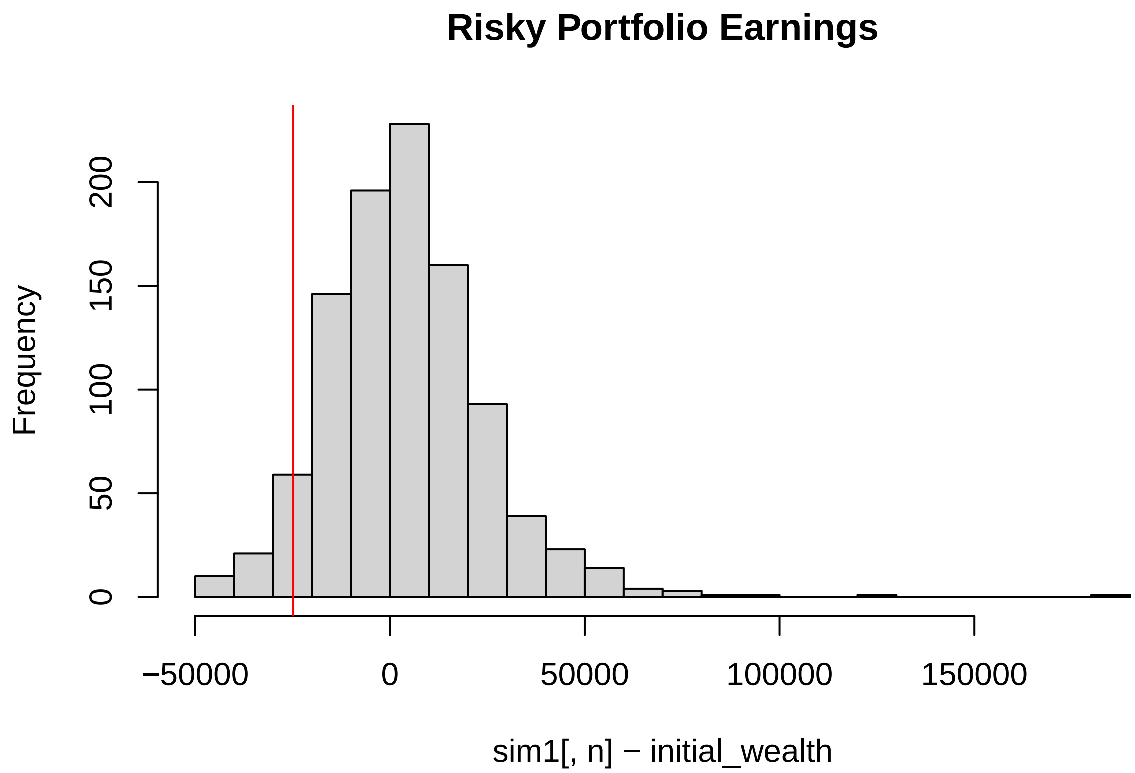

```
##        5%
## -8381.143
```

Here, VAR of 5% is a loss of 8381$, or about an 8.4% negative return.

Portfolio 2: "High Betas"

Portfolio 2 takes some of the highest beta etf's to maximize risk and return.

- **25% TQQQ** ETF designed to triple returns of NASDAQ-100
- **25% OIH** Oil ETF
- **25% FAS** Designed for 3x return of RUssell 1000
- **25% LABU** Designed to perform 3x S&P Biotech.




```
##       5%
## -24801.75
```

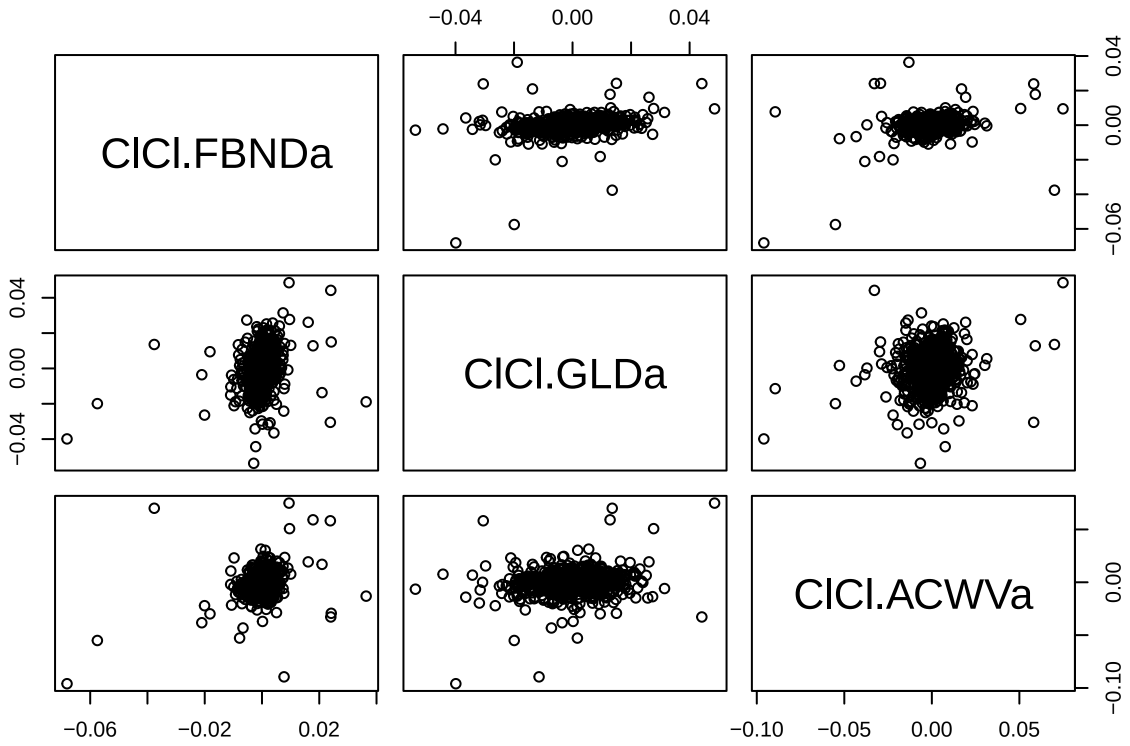
Here, losses are much larger, at negative 24,800

Portfolio 3: Low Beta ETF

Lastly, we use ver low beta etfs to try and underperform the market at lower risk

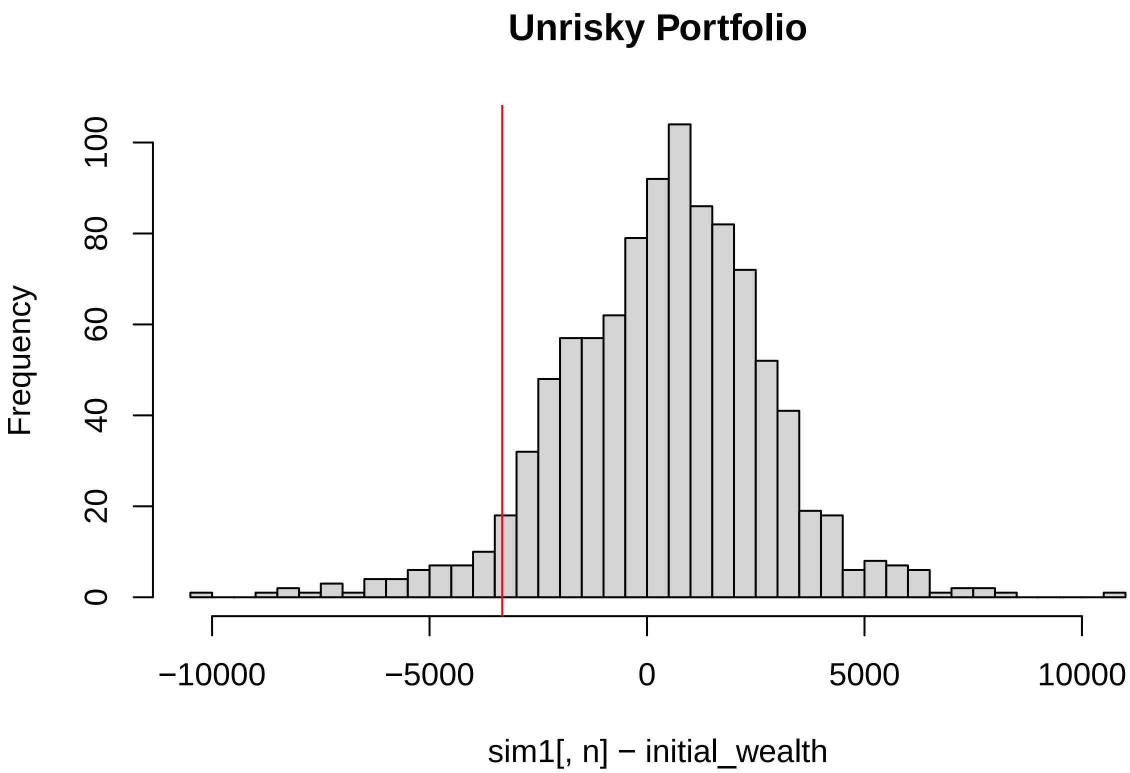
- **33% FBND** ETF of many bonds
- **33% GLD** Gold Shares
- **33% ACWV** Global equity market ETF

```
## [1] "FBND"
## [1] "GLD"
## [1] "ACWV"
```



From the pairs correlation matrix, we see that these Bond ETFs are less correlated than the ETFs in Portfolio 1 but higher than Portfolio 2.

Then, we simulate the 20-day trading period of this portfolio.



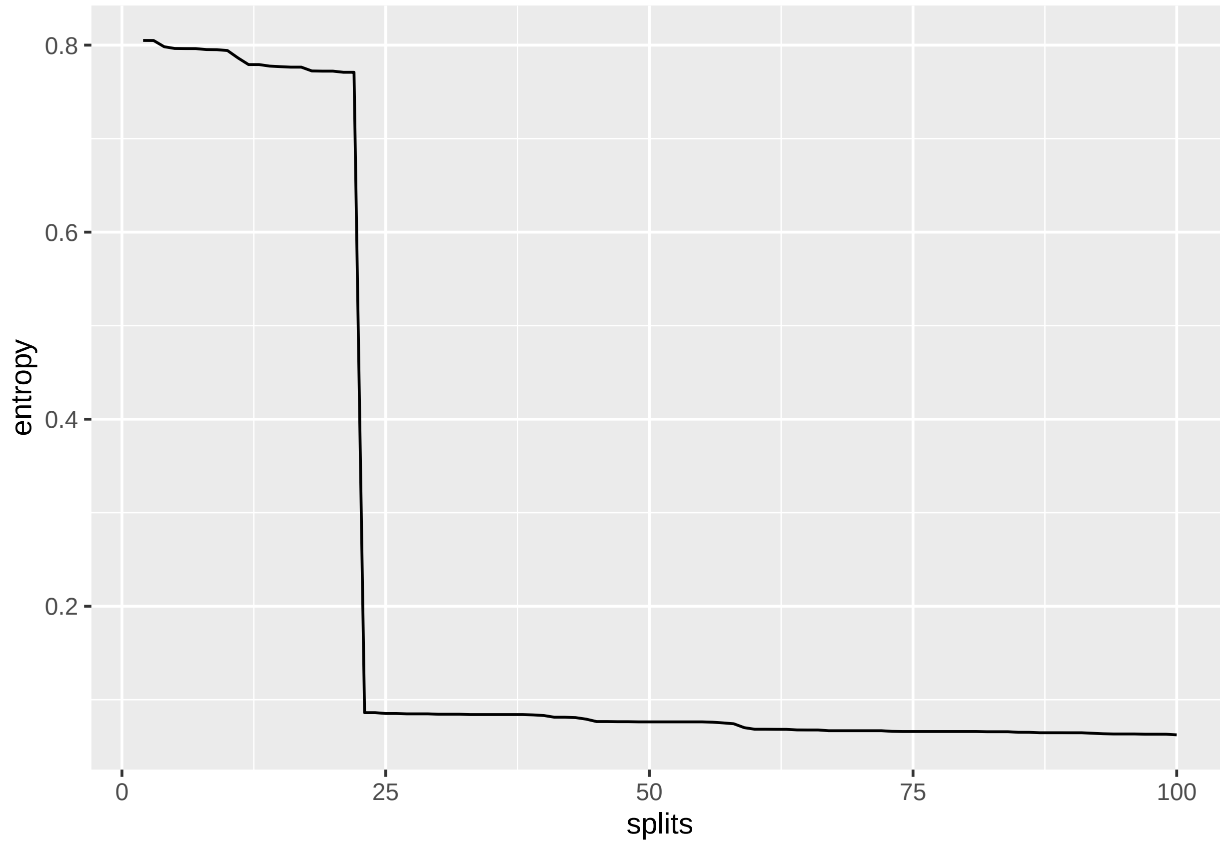

```
##        5%
## -3328.81
```

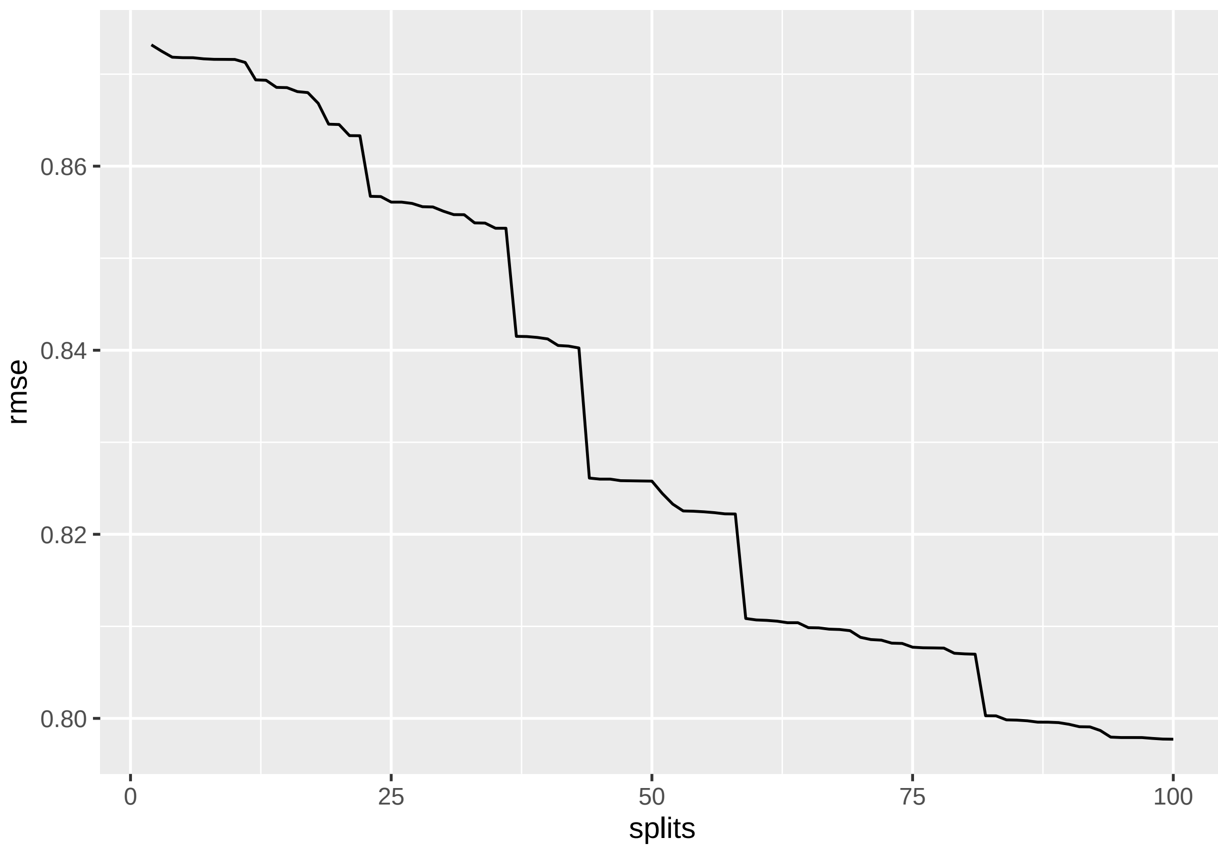
Here, we see the smallest loss of -3328.81

In conclusion, we saw what we expected. The low beta etf of bonds and gold had very low VAR, at a 3.23% loss, the very risky portfolio of high leverage etf's had a much higher VAR of about a 25% loss, and the in market indices lied somehwere in between at around an 8.4% loss.

Clustering and PCA

We first attempt hierarchical clustering. To judge how well the hierarchical clustering and how it's picked up on the wine and beer, look at the rmse of taste and the entropy of type of wine in the hierarchical tree at various numbers of splits, as seen below.

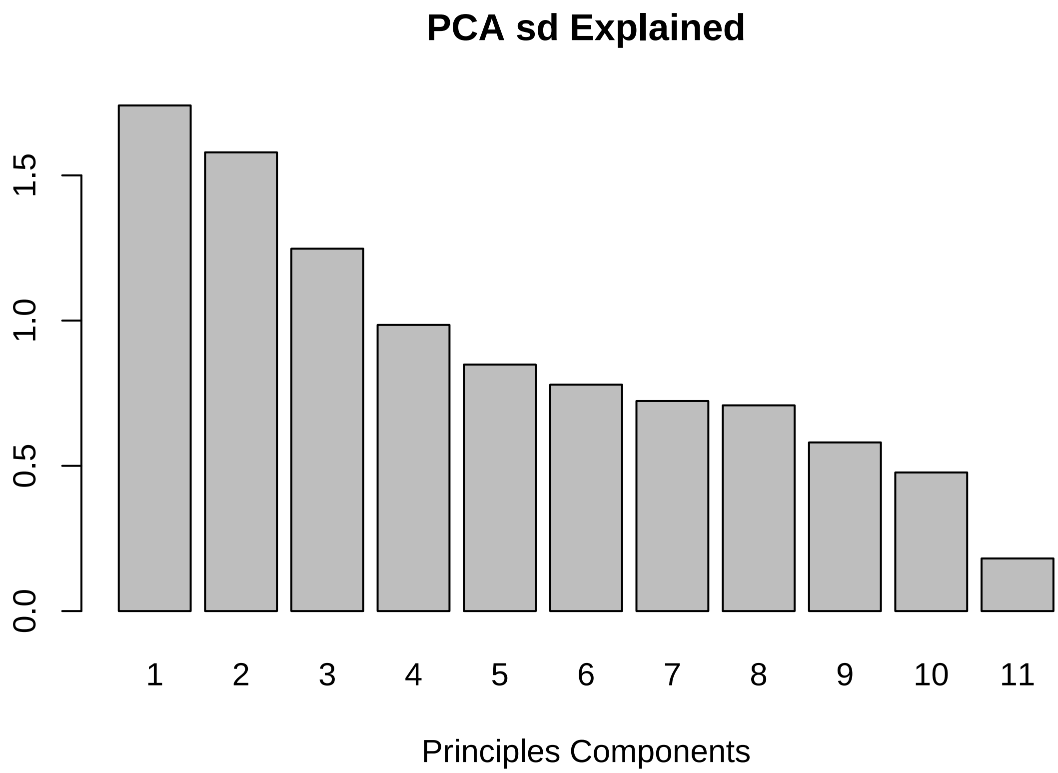





```
## [1] 6497
```

Here, we see that there is a significant drop in entropy at the 22nd split, implying that this split divides a large number of red and white wines. This is unlikely to be simply overfitting, as it is a rather significant single drop, and occurs at the 22nd split in a 6000+ entry list. Alternatively, we see a few gentle declines in rmse for quality, implying it is at various splits detecting differences in quality.

We next attempt PCA. We look at how much variance the principal components each explain.



We see that the first 4 principle components do most in explaining the data, while the other's do much so, and we thus choose to only view the first 4 principal components.

```
##                              PC1         PC2         PC3         PC4
## fixed.acidity        -0.23879890  0.33635454 -0.43430130  0.16434621
## volatile.acidity     -0.38075750  0.11754972  0.30725942  0.21278489
## citric.acid           0.15238844  0.18329940 -0.59056967 -0.26430031
## residual.sugar        0.34591993  0.32991418  0.16468843  0.16744301
## chlorides            -0.29011259  0.31525799  0.01667910 -0.24474386
## free.sulfur.dioxide   0.43091401  0.07193260  0.13422395 -0.35727894
## total.sulfur.dioxide  0.48741806  0.08726628  0.10746230 -0.20842014
## density              -0.04493664  0.58403734  0.17560555  0.07272496
## pH                   -0.21868644 -0.15586900  0.45532412 -0.41455110
## sulphates            -0.29413517  0.19171577 -0.07004248 -0.64053571
## alcohol              -0.10643712 -0.46505769 -0.26110053 -0.10680270
```

Here, the first principal components seems to be detecting high sulfur dioxide wines. This would explain why these wines are lower ph, as sulfur dioxide is acidic according to google. This also perhaps explains why these wines are lower in other types of acidity, as they are more acidic due to the sulfur dioxide.

Principal component 2 appears to show very dense very low acohol content wines.

Principal component 3 seems to see low citric acid high ph wines. Citric acid according to google citric acid is relatively weak, and thus likely explains why it has higher ph than other wines.

Lastly, PC 4 seems to detect low sulphate low ph wines, which is detecting what I just said.

We check how much these PC detect quality and different qualities by seeing how well they can be used as a linear model for quality and a logistic regression for wine taste.

```
##
## Call:
## lm(formula = quality ~ pca$x[, 1] + pca$x[, 2] + pca$x[, 3] +
##     pca$x[, 4])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6624 -0.5093 -0.0559  0.5324  3.5263
##
## Coefficients:
##              Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  5.818378   0.009822  592.39 < 0.0000000000000002 ***
## pca$x[, 1]   0.038202   0.005643    6.77       0.000000000014 ***
## pca$x[, 2]  -0.174166   0.006220  -28.00 < 0.0000000000000002 ***
## pca$x[, 3]  -0.150821   0.007874  -19.16 < 0.0000000000000002 ***
## pca$x[, 4]  -0.146175   0.009970  -14.66 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7917 on 6492 degrees of freedom
## Multiple R-squared:  0.1786, Adjusted R-squared:  0.1781
## F-statistic: 352.9 on 4 and 6492 DF,  p-value: < 0.00000000000000022
```

```
## [1] 0.987071
```

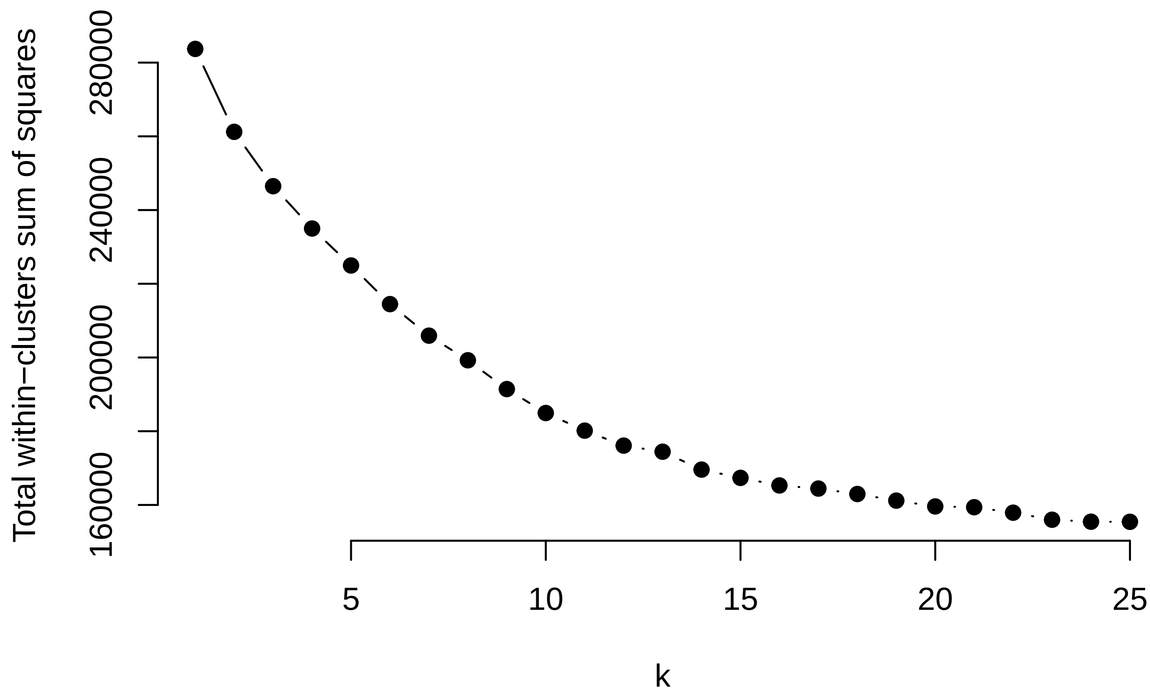
Here, we see that we see that our residual square error is relatively high still at 0.781, with R-square of only 0.1786. However, this is not that much better than our other data. We see that it is very good at accessing color however, at 98.7/% accuracy using only the first 4 pc.

Ultimately, we believe PCA makes more sense on our data. This is for the primary reason that many of the measures are very highly correlated with one another. The amount of free vs total sulfur dioxide will obviously be related, as one is a subset of the other, and the amount of each acid will affect the ph of the wine. While these issues might hurt clustering, causing some extraneous factors effect creating useless sparsness in clustering, PCA is designed for these exact scenarios in mind.

Market segmentation

Our method will be using kmeans to cluster the data, and observe the results. We first begin by cleaning our data, scaling it and reindexing.

We continue with our data ### Elbow method



The Elbow Plot above indicates a kink at $k = 14$. Although perhaps just a random drop, we choose to select it as our choice k, since it is a point at which wthin sum squares does not appear to decrease much more from further itterating.

Running K-Means with 14 clusters

We now look at the centers of some of the largest clusters.

```
##         chatter current_events       travel photo_sharing uncategorized
## 6   0.07205880     0.27684711  0.288727032   -0.09071828    0.11259854
## 13  1.03583481     0.08209725 -0.044325901   -0.01234994    0.75732750
## 11 -0.01951185    -0.01663479 -0.003362117   -0.20979829    0.23838226
## 3  -0.05609136     0.33853271  0.355535359   -0.01396405    0.45730692
## 1  -0.16479790     0.18220668 -0.006172873   -0.13487751    0.89756950
## 10 -0.03520613     0.25745234  0.028581119    0.14091908   -0.06207133
##         tv_film sports_fandom    politics        food      family
## 6  -0.11619101    0.14065669  0.15052740  0.04049422 -0.05999555
## 13 -0.11117616   -0.15165194 -0.14184117 -0.15346615 -0.10053936
## 11 -0.25368123   -0.13653580 -0.21256295 -0.05838928  0.07013474
## 3   2.76851340   -0.15780317  0.04853353  0.23755311 -0.11029086
## 1   2.13059224   -0.04523218 -0.24417590 -0.05994916 -0.15654764
## 10  0.02067083    2.90549609 -0.12109863  2.60687493  2.03615164
```

```
##    home_and_garden       music          news online_gaming   shopping
## 6       0.23510191  0.01418264 -0.0006901999    0.08935906 -0.2378226
## 13      0.59162058 -0.11600476 -0.1211191143   -0.06961621 -0.1010846
## 11      0.06288968 -0.05894335 -0.1591778016   -0.00482014 -0.2067087
## 3       0.38896740  0.09303865  0.0818640860   -0.19426608  0.1244383
## 1       0.23530864  2.61174292 -0.1511209468   -0.09575482 -0.1038649
## 10      0.35342583  0.20985681  0.0468349705    0.05605539  0.1424614
##    health_nutrition college_uni sports_playing     cooking        eco
## 6       0.050860594  0.12732753     -0.1112904 -0.05898219  0.4479995
## 13     -0.079020000 -0.06397636      0.3197372 -0.13258116  0.1252254
## 11     -0.172020156 -0.14162330     -0.1532389 -0.17080548  0.1782169
## 3      -0.113656707 -0.09678584     -0.1117945 -0.09432379  0.1953218
## 1      -0.208321800  1.14206171      0.4004778 -0.23119930 -0.0596169
## 10      0.002488701  0.01337708      0.2526282  0.10913460  0.4595234
##      computers   business    outdoors      crafts automotive         art
## 6   0.29753290 -0.3460090  0.29780310  0.21793373  0.1245356  0.33167537
## 13  0.01645205  0.4248374  0.07184545  0.41970713 -0.1817836 -0.01789660
## 11  0.08775547 -0.1199295  0.26918780  0.01213724  0.1132163 -0.10000172
## 3  -0.13149986  0.2935722 -0.18224348  1.07451555 -0.1988306  4.20323013
## 1  -0.10741060  0.4773231  0.08713668 -0.01027882 -0.2250684 -0.21581184
## 10  0.23380558  0.2571776  0.01216739  0.98476458  0.3470012  0.08865192
##        religion      beauty   parenting       dating      school
## 6   0.120701794 -0.10070195  0.18658414 -0.009528244  0.09244824
## 13  0.004101468  0.28343247  0.07342746  4.900724879  1.31148861
## 11 -0.170963536 -0.07297489  0.10633422 -0.083600176  0.03933017
## 3  -0.068701593 -0.01688381 -0.19721392 -0.121684662  0.09410364
## 1   0.106066993 -0.01399486 -0.22633690 -0.128772294 -0.28808986
## 10  3.071666149  0.59857470  3.04953832 -0.002792854  2.37092108
##    personal_fitness      fashion small_business        spam       adult
## 6        0.12183236 -0.020449872      0.3142883 12.41886450  3.75022215
## 13      -0.05653418  0.834146963      0.3790624 -0.07768727 -0.08155232
## 11      -0.04852091 -0.147543491      0.4232040 -0.07768727  4.78604639
## 3       -0.12030991 -0.006912407      0.6383398 -0.07768727 -0.06738593
## 1       -0.18590216 -0.124653364      0.6993820 -0.07768727 -0.16068118
## 10       0.10680362  0.184181019      0.2066116 -0.07768727 -0.09085662
```

Our first cluster appears to be overwhelmingly spam, with an overwhelmingly large spam value. Our second cluster has a high dating value, and a somewhat large chatter value, implying it contains users focused around dating. The third cluster has an overwhelming large adult value, implying it is users focused on looking at… "adult" things. Cluster 4 has larg values in tv_film, art, and crafts, implying this is used by users who primarily want to discuss various media they consume and look at cool crafts. The fifth cluster appears to be college students, talking mostly about college and tv_film.The sixth is again similar, again college student users, but ones who care more about music. Both clusters 5 and 6 care some about business, perhaps because they are college students looking for jobs.

The other clusters are not discussed for the purpose of brevity, although they would also be shown in a broader analysis.

The Reuters corpus

Our goal is to predict author of article based upon the article itself. We believe this to be a useful question, as it could perhaps be used as a tool in detecting anonymously written articles or plagarism.

First, we read in the 50 training articles for each of the 50 different authors. Then set training Corpus.

After reading in the data, we pre-processed the text in the articles. * Converting all text to lowercase * Remove numbers * Remove punctuation * Remove excess white space

```
## <<DocumentTermMatrix (documents: 2500, terms: 33472)>>
## Non-/sparse entries: 628611/83051389
## Sparsity           : 99%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 33373)>>
## Non-/sparse entries: 545286/82887214
## Sparsity           : 99%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3448)>>
## Non-/sparse entries: 428509/8191491
## Sparsity           : 95%
## Maximal term length: 43
## Weighting          : term frequency (tf)
```

After these four steps, we're down to **2500 documents** with **32,669 terms.** * Remove stop and filler words, based on the "basic English" stop words

After removing filler words, we're down to **32,570 terms.** * Removed words that have count 0 in > 99% of documents

Thus cuts the long tail significantly to only **3393 terms.** * Finally, we converted the raw counts of words in each document to TF-IDF weights.

Then, we replicated the same process to read in the 50 testing articles for the authors. There are 3448 terms in the testing data, compared to only 3393 terms in the training data. We will deal with this in the later procedure.

For the testing data, we did the same pre-processing steps as the training data.

```
## <<DocumentTermMatrix (documents: 2500, terms: 33472)>>
## Non-/sparse entries: 628611/83051389
## Sparsity           : 99%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 33373)>>
## Non-/sparse entries: 545286/82887214
## Sparsity           : 99%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3448)>>
## Non-/sparse entries: 428509/8191491
## Sparsity           : 95%
## Maximal term length: 43
## Weighting          : term frequency (tf)
```

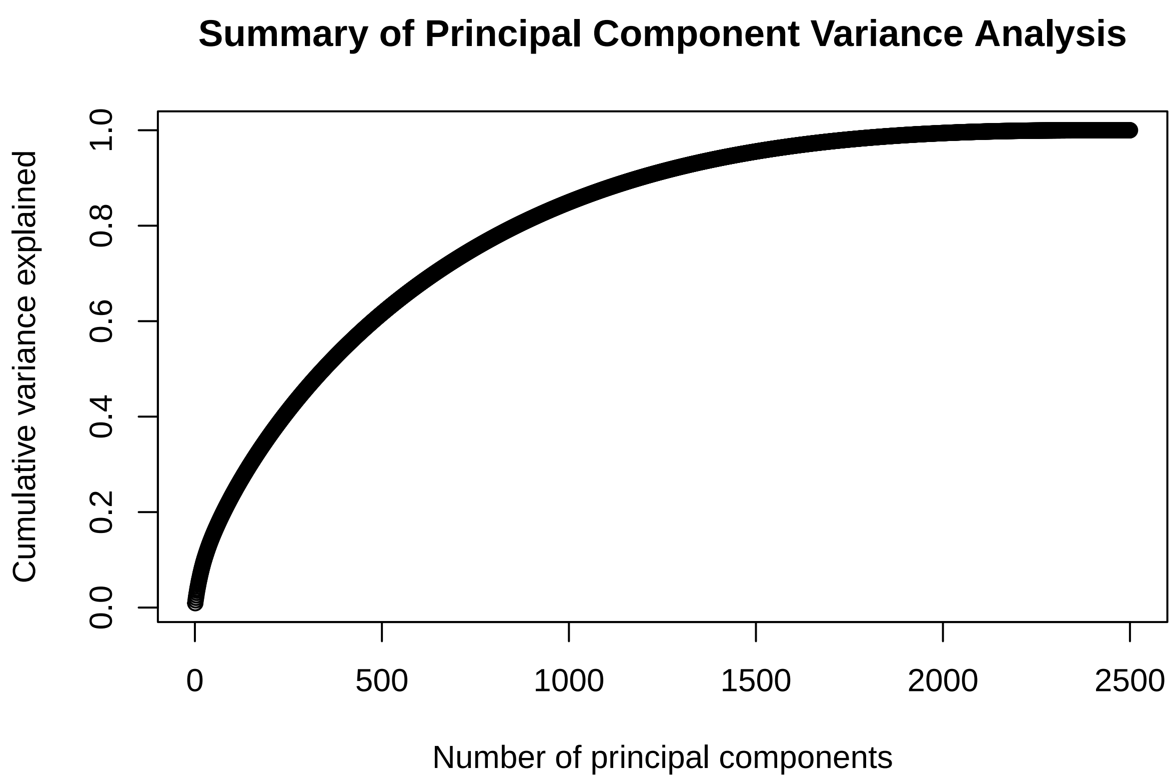
We ignored words that are in the testing set and but not in the training set as below

```
## <<DocumentTermMatrix (documents: 2500, terms: 3448)>>
## Non-/sparse entries: 388509/8231491
## Sparsity           : 95%
## Maximal term length: 43
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

This removes the 55 "new" terms from the training data, less than 2% of the training terms. After this procedure, both of the training and testing groups have 3393 terms.

We now use PCA to simplify the predictors. * We remove columns that have zero entries. * We use only intersecting columns of the train and testing data.

PCA process:



We stop at 1000 principal components because it already can explain 80% of the variance as shown in the chart above.

We now can move on to the models. We chose to Naive Bayes and Random Forest to do so.

Random Forest

The random forest model with mtry = $=\sqrt(n_{features}) \approx 50$

```
## [1] 0.8316
```

The Random Forest accuracy is 83.16%.

Naive Bayes

We then used a Naive Bayes model to predict the testing data from a training data.

```
## [1] 0.9684
```

The Naive Bayes accuracy is 96.8%, outperforming random forest.

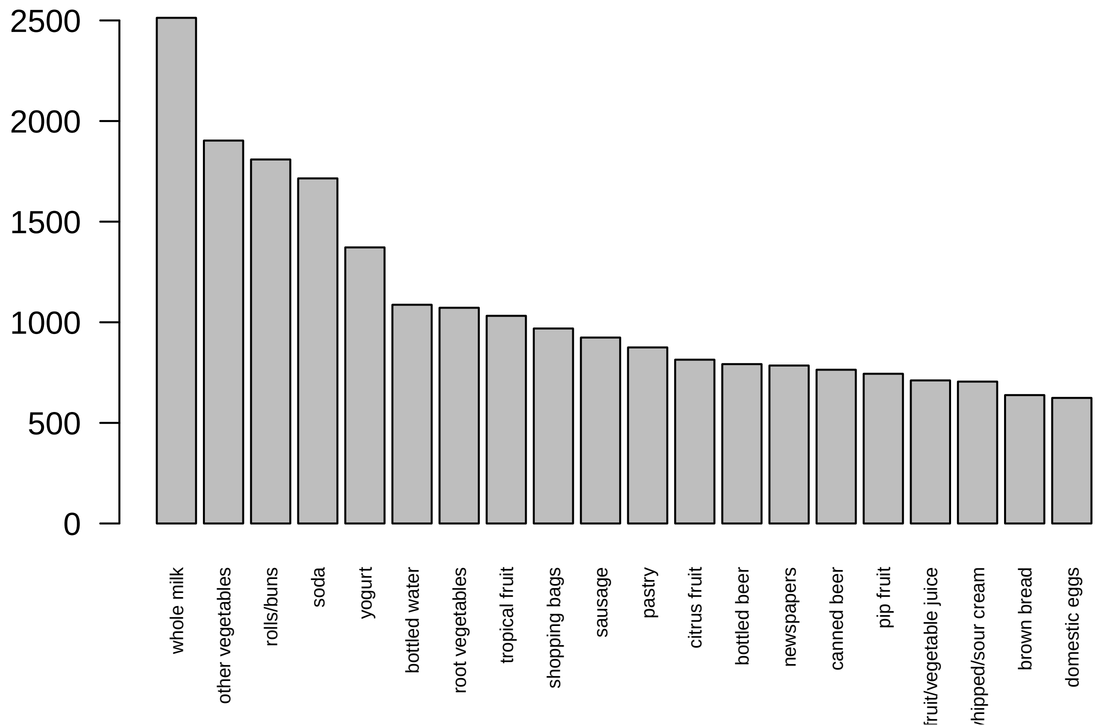
In summary, the Naive Bayes recieved the best accuracy of 96.8%, which is remarkably strong.
.

Association rule mining

```
## [[1]]
## [1] "V1" "V2" "V3" "V4"
```

```
##       User          value
##  Min.   :    1   Length:43367
##  1st Qu.: 3814   Class :character
##  Median : 7620   Mode  :character
##  Mean   : 7650
##  3rd Qu.:11482
##  Max.   :15296
```

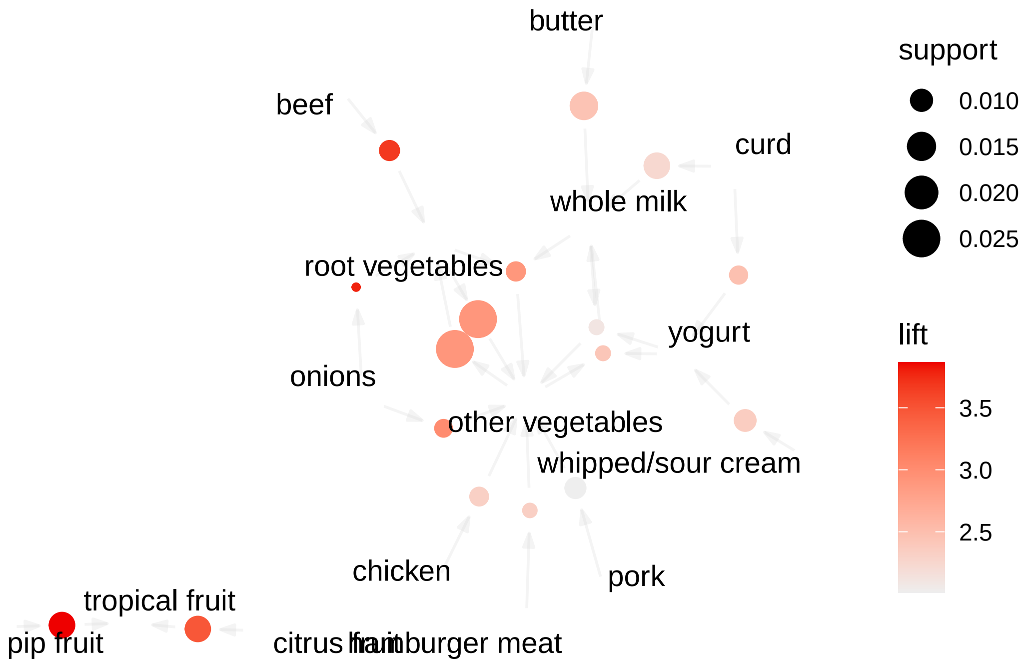
```
##    Length     Class      Mode
##     43367 character character
```




```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target  ext
##       4  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
## Absolute minimum support count: 76
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 15296 transaction(s)] done [0.00s].
## sorting and recoding items ... [101 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [118 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
##      lhs                               rhs                support     confidence
## [1]  {onions}                       => {root vegetables}  0.005295502 0.2655738
## [2]  {onions}                       => {other vegetables} 0.007452929 0.3737705
## [3]  {hamburger meat}               => {other vegetables} 0.006210774 0.2905199
## [4]  {chicken}                      => {other vegetables} 0.007975941 0.2890995
## [5]  {beef}                         => {root vegetables}  0.008695084 0.2577519
## [6]  {curd}                         => {yogurt}           0.007649059 0.2232824
## [7]  {curd}                         => {whole milk}       0.012617678 0.3683206
## [8]  {butter}                       => {whole milk}       0.014382845 0.4036697
## [9]  {pork}                         => {other vegetables} 0.009283473 0.2504409
## [10] {whipped/sour cream}           => {yogurt}           0.009741109 0.2113475
## [11] {pip fruit}                    => {tropical fruit}   0.012683054 0.2607527
## [12] {citrus fruit}                 => {tropical fruit}   0.012486925 0.2346437
## [13] {root vegetables}              => {other vegetables} 0.025366109 0.3619403
## [14] {other vegetables}             => {root vegetables}  0.025366109 0.2038886
## [15] {root vegetables, whole milk} => {other vegetables} 0.008172071 0.3612717
## [16] {other vegetables, yogurt}    => {whole milk}       0.006341527 0.3991770
## [17] {whole milk, yogurt}          => {other vegetables} 0.006341527 0.2614555
##      coverage   lift     count
## [1]  0.01993985 3.789381  81
## [2]  0.01993985 3.004306 114
## [3]  0.02137814 2.335151  95
## [4]  0.02758891 2.323734 122
## [5]  0.03373431 3.677774 133
## [6]  0.03425732 2.489306 117
## [7]  0.03425732 2.241875 193
## [8]  0.03563023 2.457036 220
## [9]  0.03706851 2.013003 142
## [10] 0.04609048 2.356248 149
## [11] 0.04864017 3.864800 194
## [12] 0.05321653 3.477820 191
## [13] 0.07008368 2.909216 388
## [14] 0.12441161 2.909216 388
## [15] 0.02262029 2.903842 125
## [16] 0.01588651 2.429690  97
## [17] 0.02425471 2.101536  97
```



As a general heuristic, we chose a lift value of 2. We saw that lower lift values such as 1.5 gave some odd rules, such as onion therefore milk, or curd therefore other vegetables, and higher lift values gave perhaps less rules than one might desire, with 2.5 going down to 18 rules. In general, we also thought that in general doubling the likelihood of buying an item was a good baseline for if an item increased your likelihodo of purchasing. We still saw some odd rules at a lift of 2, such as sausage implies citrus fruit, and believe these to be since we have yet to adjust for confidence. Looking at the data, a confidence threshold of about 0.2 seemed to clean up these issues. It's also a good general value, as if the likelihood of something is less than 20%, we generally consider it very unlikely. We stuck with general round values to prevent overtuning the parameters to get the exact data we wanted. This leaves us with 17 rules.

These rule sets seem to make sense, with types of vegetables, fruit, and dairy implying types of vegetables, fruit and dairy, respectively. It also seems to be picking up on the existence of meals, with chicken, pork, and beef both implying vegetable purchases. Some rules do not entirely make sense, such as whole milk and yogurt implying other vegetables, although this could be picking up on the existence of families, who tend to buy milk and yogurt, but also want healthy children and therefore buy vegetables.