

ML Final HW

2022-08-03

Group Members:

Meeth Yogesh Handa(mh58668), Nicolay Huarancay (nh23865), Jason Nania (jn28878), Nicole Pham-Nguyen (np9967)

GitHub Link:

<https://github.com/meeth414/Intro-to-ML-2-STA-380-Group-Submission>

Probability Practice

Part A

Since the problem states that the expected fraction of random clickers is 0.3, then the fraction of random clickers is a random variable. Thus, the probability of a random clicker is equal to the fraction of random clickers, given that this is a uniform draw. If the fraction of random clickers is a random variable, then the fraction of people who answered yes is also a random variable. We can also say that the expected value of yes is linear because the fraction of truthful clickers and random clickers are random variables. This allows us say that the expected probability of truthful clickers is equal to 1 - 0.3 (the expected probability of random clickers), 0.7. Lastly, the expected probability value of yes is 0.65, because we are certain that the probability of a yes is 65%. Thus, we can solve the expression below, using the rule of total probability, to find the fraction of people who are truthful clickers.

- $E[(RandomClicker)] = 0.3$
- $P(Yes) = 0.65$
- $P(No) = 0.35$

$$E[P(Yes)] = P(Yes \mid TruthfulClicker) \cdot P(TruthfulClicker) + P(Yes \mid RandomClicker) \cdot E[P(RandomClicker)]$$

$$0.65 = (X \cdot 0.7) + (0.5 \cdot 0.3)$$

$$X = 71.44\% \text{ (Or in fraction terms = 0.7144)}$$

Part B

Using the rule of total probability, we were able to conclude that the fraction of people who have the disease, given that they tested positive, was 19.89%.

- $P(\text{Disease} | \text{Positive}) = X$
- $P(\text{Positive} | \text{Disease}) = 0.993$
- $P(\text{Negative} | \text{NoDisease}) = 0.9999$
- $P(\text{Positive} | \text{NoDisease}) = 0.0001$
- $P(\text{Disease}) = 0.000025$
- $P(\text{NoDisease}) = 0.999975$

$$P(\text{Disease} | \text{Positive}) = \frac{P(\text{Disease}) \cdot P(\text{Positive} | \text{Disease})}{(P(\text{Disease}) \cdot P(\text{Positive} | \text{Disease})) + (P(\text{NoDisease}) \cdot P(\text{Positive} | \text{NoDisease}))}$$

$$P(\text{Disease} | \text{Positive}) = \frac{(0.000025 \cdot 0.993)}{(0.000025 \cdot 0.993) + (0.999975 \cdot 0.0001)}$$

$$P(\text{Disease} | \text{Positive}) = 0.198882$$

Wrangling the Billboard Top 100

Part A

Top 10 most popular songs

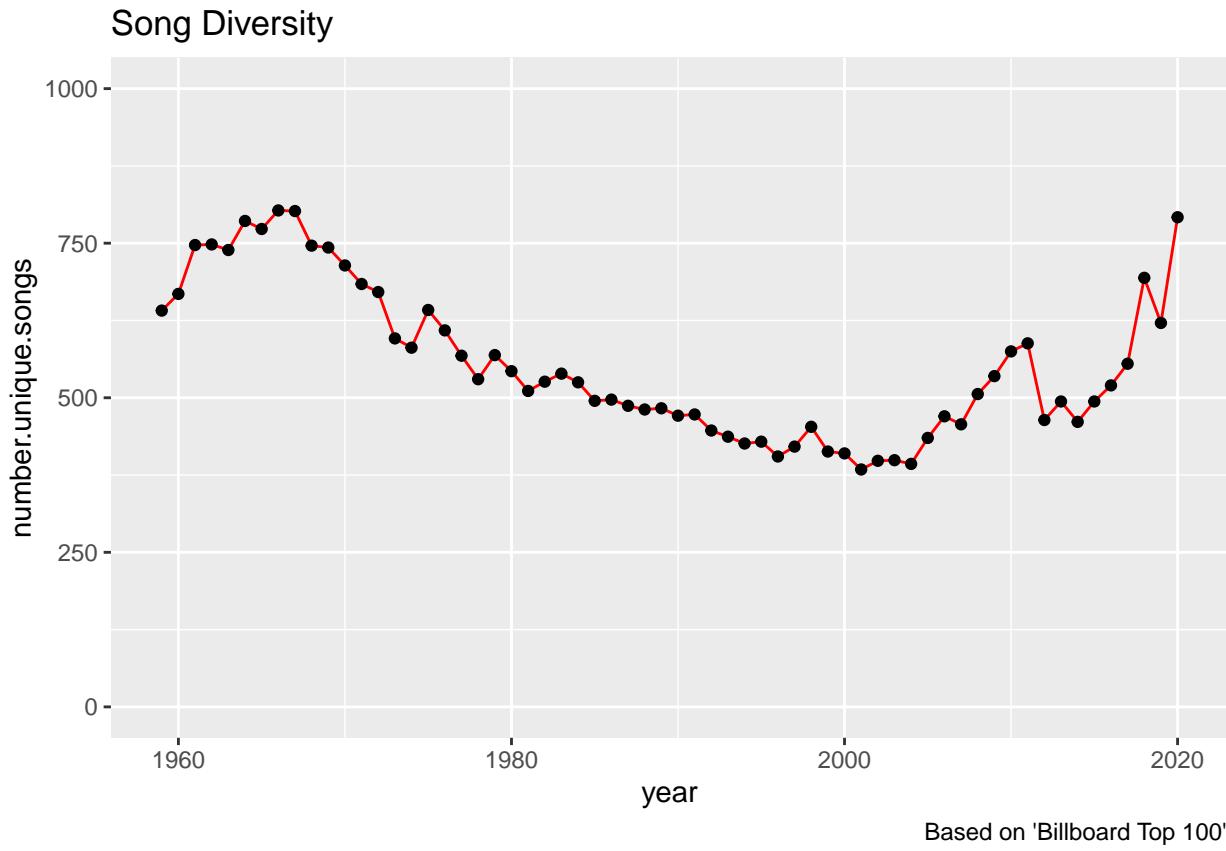
```
## # A tibble: 10 x 3
## # Groups:   performer [10]
##   performer          song     count
##   <chr>            <chr>    <int>
## 1 Imagine Dragons Radioactive      87
## 2 AWOLNATION        Sail       79
## 3 Jason Mraz        I'm Yours    76
## 4 LeAnn Rimes        How Do I Live 69
## 5 LMFAO Featuring Lauren Bennett & GoonRock Party Rock Anthem 68
## 6 OneRepublic        Counting Stars 68
## 7 Adele              Rolling In The Deep 65
## 8 Jewel              Foolish Games/You Were Meant~ 65
## 9 Carrie Underwood Before He Cheats 64
## 10 Lifehouse         You And Me    62
```

Comments:

The top 10 most popular songs since 1958 in the ‘Billboard Top 100’ are: *Radioactive*, *Sail*, *I’m Yours*, *How Do I Live*, *Party Rock Anthem*, *Counting Stars*, *Rolling In The Deep*, *Foolish Games/You Were Meant For Me*, *Before He Cheats*, *You And Me*. The most popular song ever since 1958 has appear in the ‘Billboard Top 100’ up to 87 weeks.

Part B

Musical Diversity

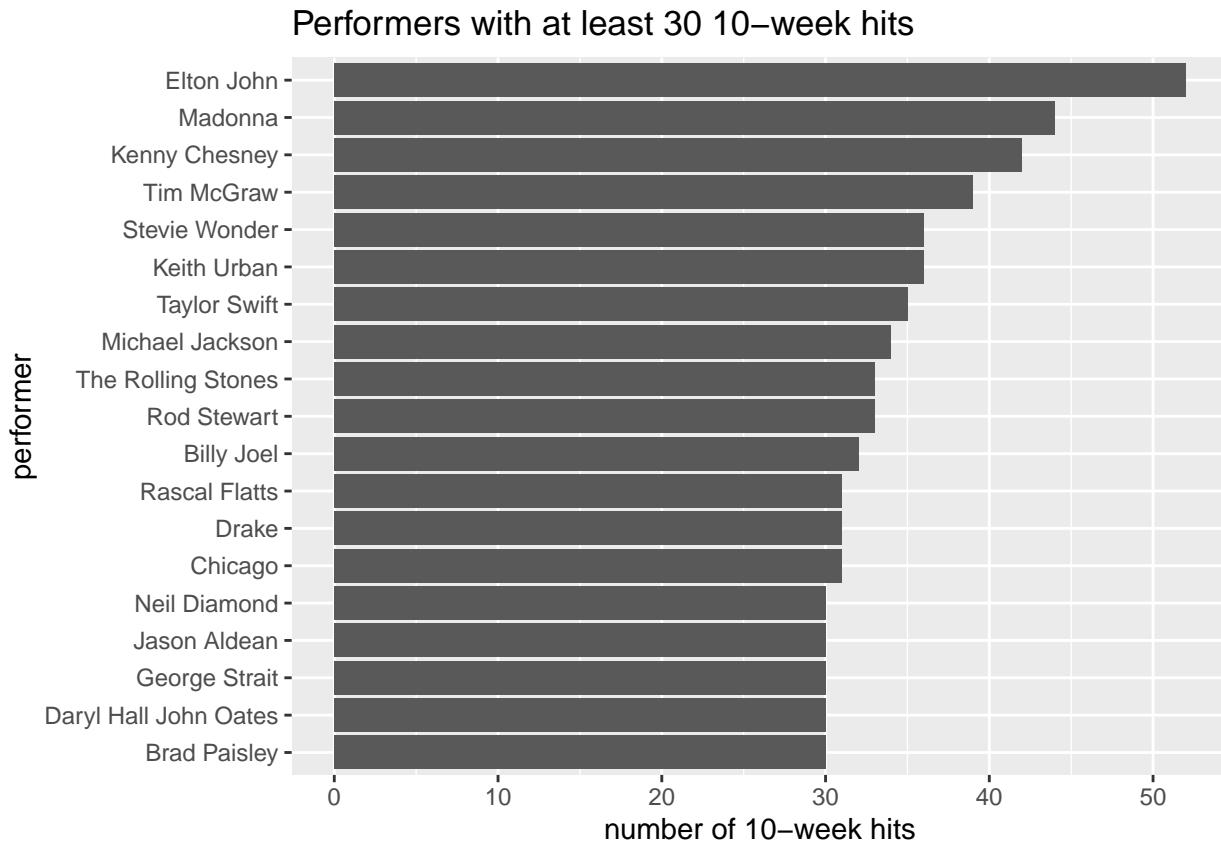


Comments:

In this line graph we can observe that the number of different songs which appear in the 'Billboard Top 100' each year had a decreasing trend from 1966 to 2001. Less diversity means that less songs remains much time in year in the 'Billboard Top 10'. On the other hand, this behavior was reverse from 2001 to 2020 in which there was an increasing trend; more diversity.

Part C

Performers with at least 30 10-week hits



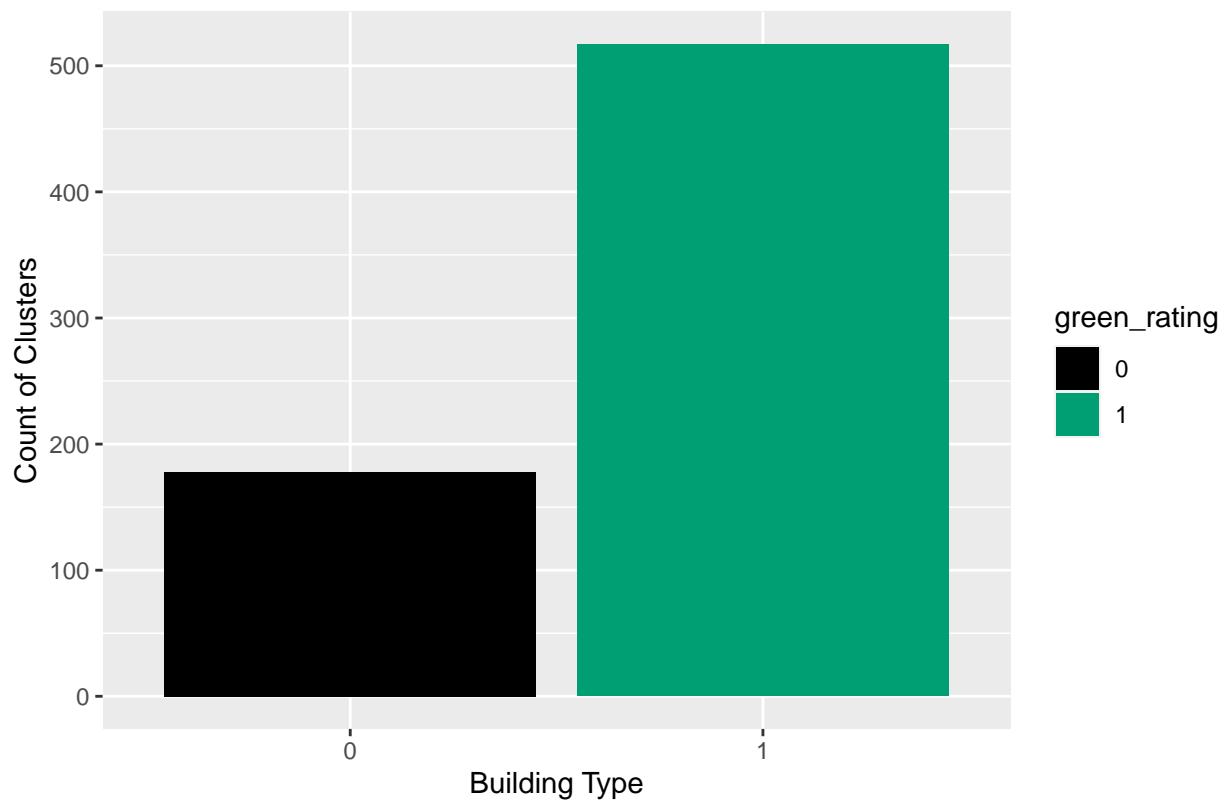
Comments:

This graph shows us the 19 artists in U.S. musical history since 1958 who have had at least 30 songs that were “ten-week hits.” As we can observe, Elton John is the performer with the most number of those songs (52) and he is followed by Madonna with 44 songs.

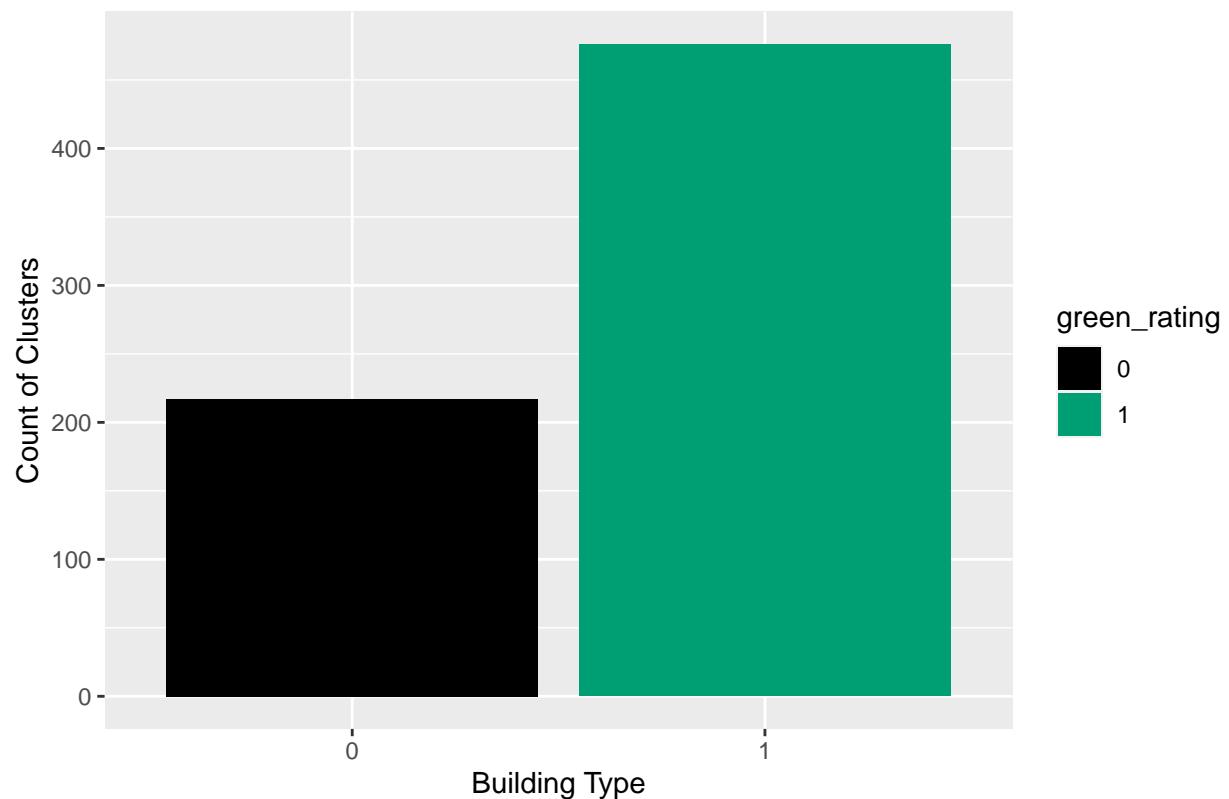
Visual story telling part 1: green buildings

```
## # A tibble: 2 x 2
##   green_rating count
##   <fct>        <int>
## 1 0             212
## 2 1             498
```

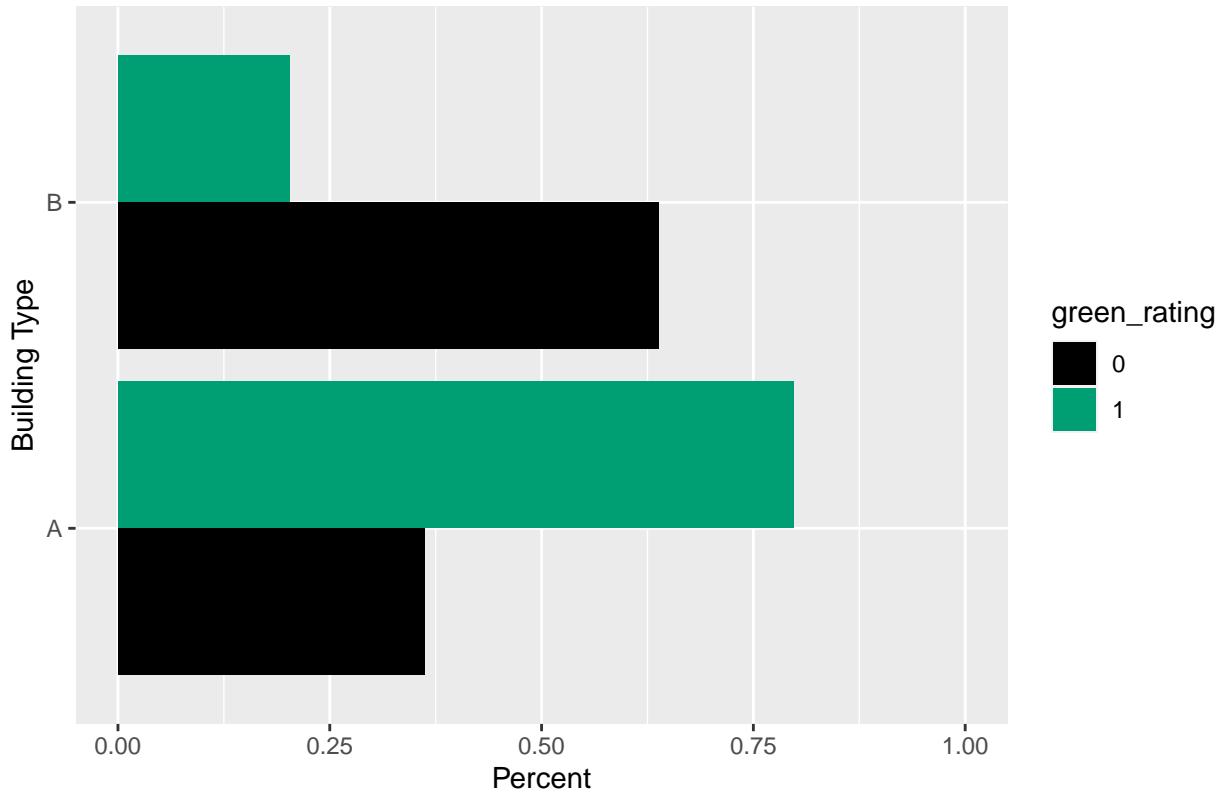
Max Average Occupancy by Cluster



Conservative Average Profit by Cluster



Green Building Percentage by Building Quality Type



Comments: We agree with the analyst's opinion that a green building would be the best investment for the company. However, we disagree with the analysis to reach that conclusion. In his analysis, he groups all of the houses together without taking into account the different markets across Austin. In addition, since the analysis was based solely off of the expected numbers given, and not accounting for fluctuation, we felt that his analysis was not every strong. Thus, we conducted our own analysis accounting for these variables.

In plot 1, "Max Average Occupancy by Cluster," we can see that the occupancy rates across Austin are primarily in green buildings, allowing us to make the initial conclusion that Austinites tend to prefer green buildings over non-green buildings.

In plot 2, "Max Average Profit," we accounted for the building types and the maximum average profit by cluster. In order to create this plot, we assumed that all the green buildings paid for the utility costs of the tenant, to get an extremely conservative amount of profit for green buildings vs. non-green buildings. This allows us to conservatively compare the lower-end of profit landlords receive from green buildings compared to non-green buildings. Thus, the profit was calculated as: Rents received - total utility costs. Even with this conservative factor, the average profit by green buildings vastly outperformed the non-green buildings.

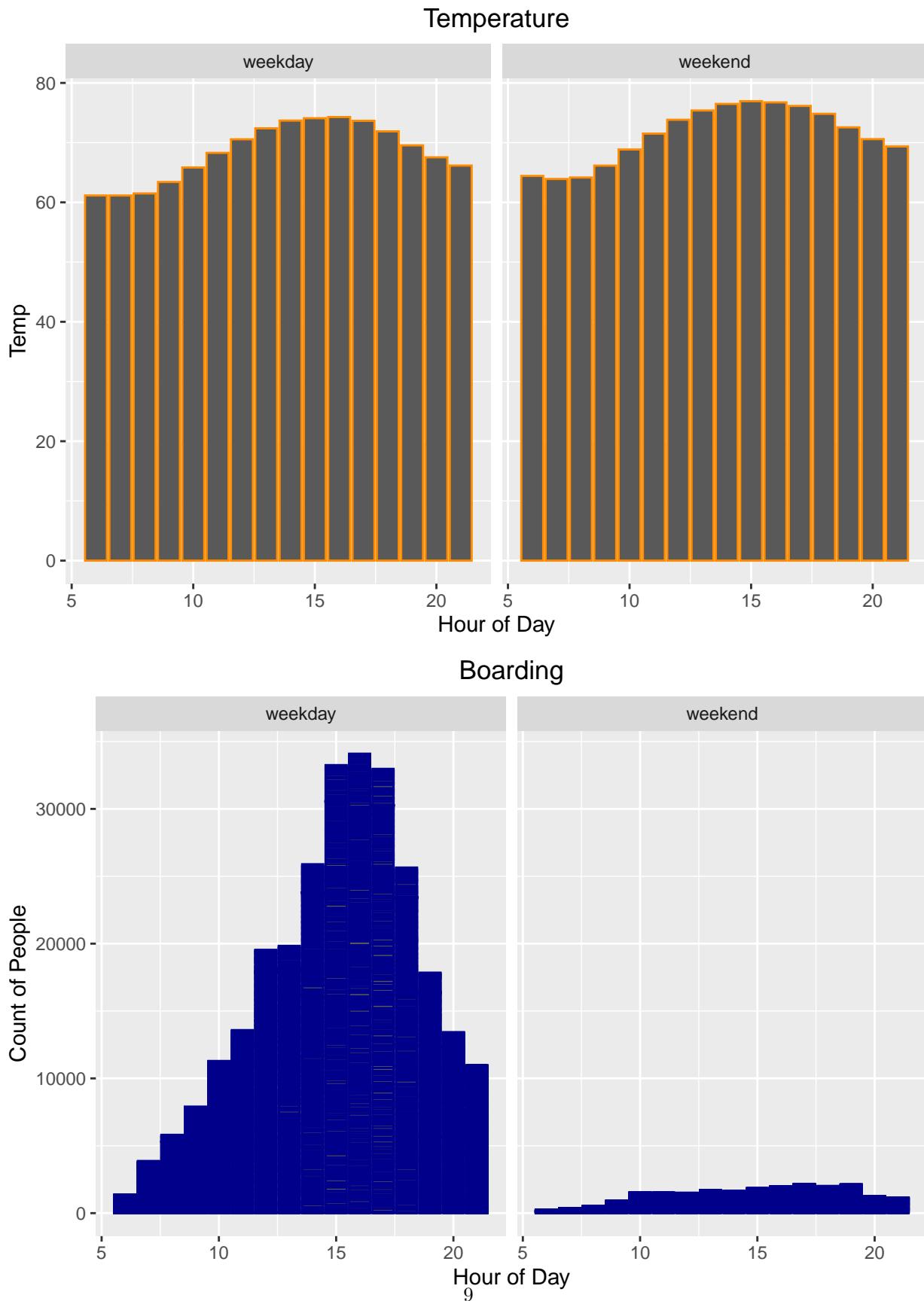
Finally, we wanted to evaluate the building quality, as company brand and longevity is extremely important. In plot 3, "Green Building Percentage by Building Quality Type" we looked at the percentage of green buildings vs. non-green buildings and their respective building quality types. The plot suggests that green buildings have an extremely high ratio of a label of a "good" building quality type compared to the non-green buildings.

Although we found higher utility costs associated with a few green buildings which was surprising, the average conservative profit generated by each green building cluster was still significantly higher than the non-green buildings.

Thus, we can conclude that the company should invest in a green building, despite the extra initial cost, since green buildings in this dataset lead to a higher average occupancy, a higher average profit, and a higher

building quality rating.

Visual story telling part 2: Capital Metro data



Comments:

The first plot measures the temperature vs. hour of day. We can see that the highest temperatures occur from 3-5P.M. Interestingly, this trend is also seen in the bus boarding rates across each hour, with maximum boarding rates also occurring around 3-5P.M. In order to account for E.O.D. traffic (for after-work commuters), we also looked at the weekend temperatures and the weekend boarding rates. Although not as visually significant as the weekday, this trend still occurs during the weekend.

This allows us to make the conclusion that boarding is seen to have a positive correlation with temperature because as the temperature increases during the day, so does the number of people who want to take the bus. This pattern is probably due to people taking the bus as opposed to walking or taking the car to avoid the extreme heat, as cars can overheat too or be unpleasant to ride in when hot.

Portfolio modeling

```
## [1] "AOK"   "JPST"  "GOVT"  "VBK"   "SCHG"  "DRN"   "SOXX"

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/AOK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/AOK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/JPST?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/JPST?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GOVT?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GOVT?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VBK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/VBK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SCHG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SCHG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/DRN?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/DRN?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

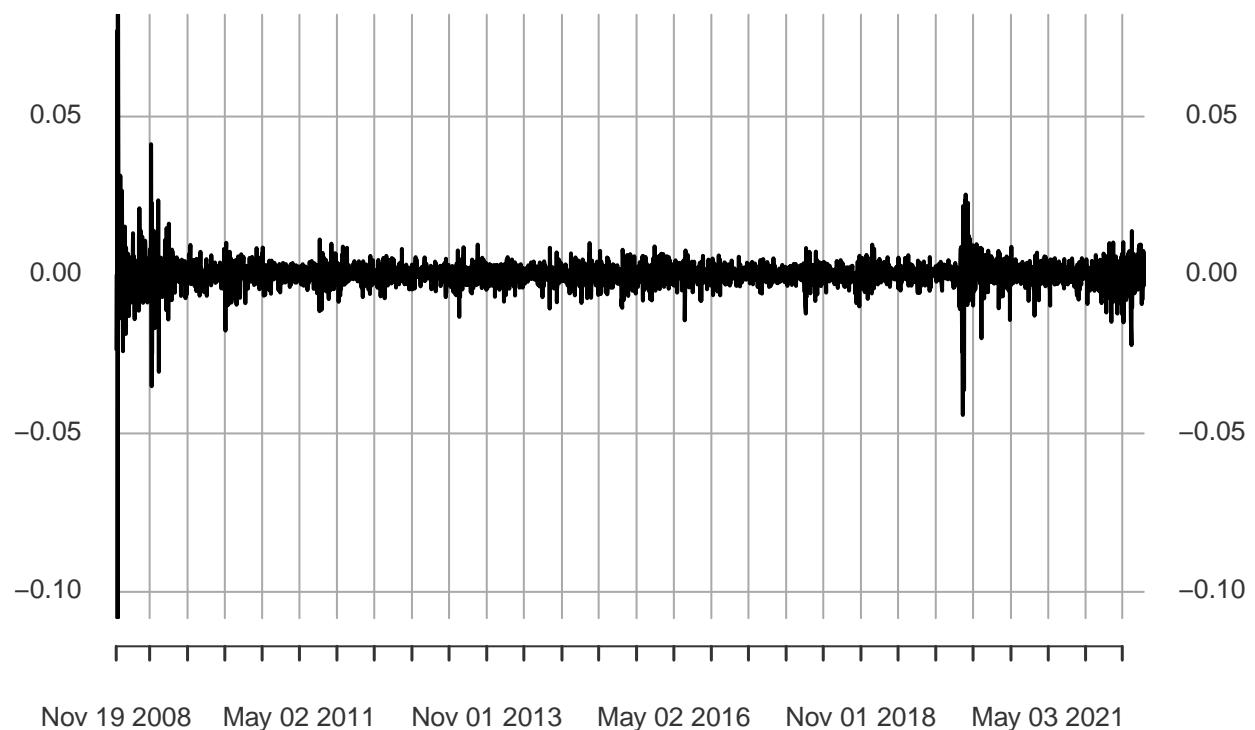
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SOXX?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SOXX?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

```

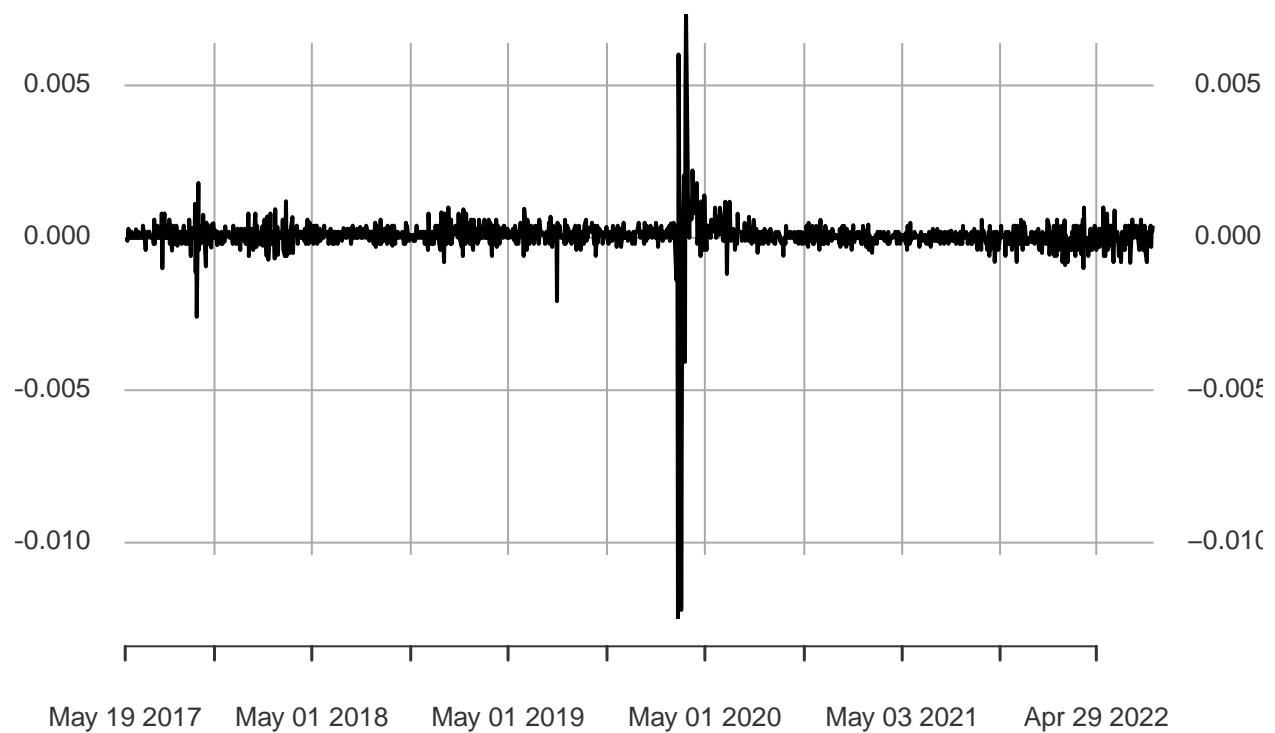
CICI(AOKa)

2008-11-19 / 2022-08-12



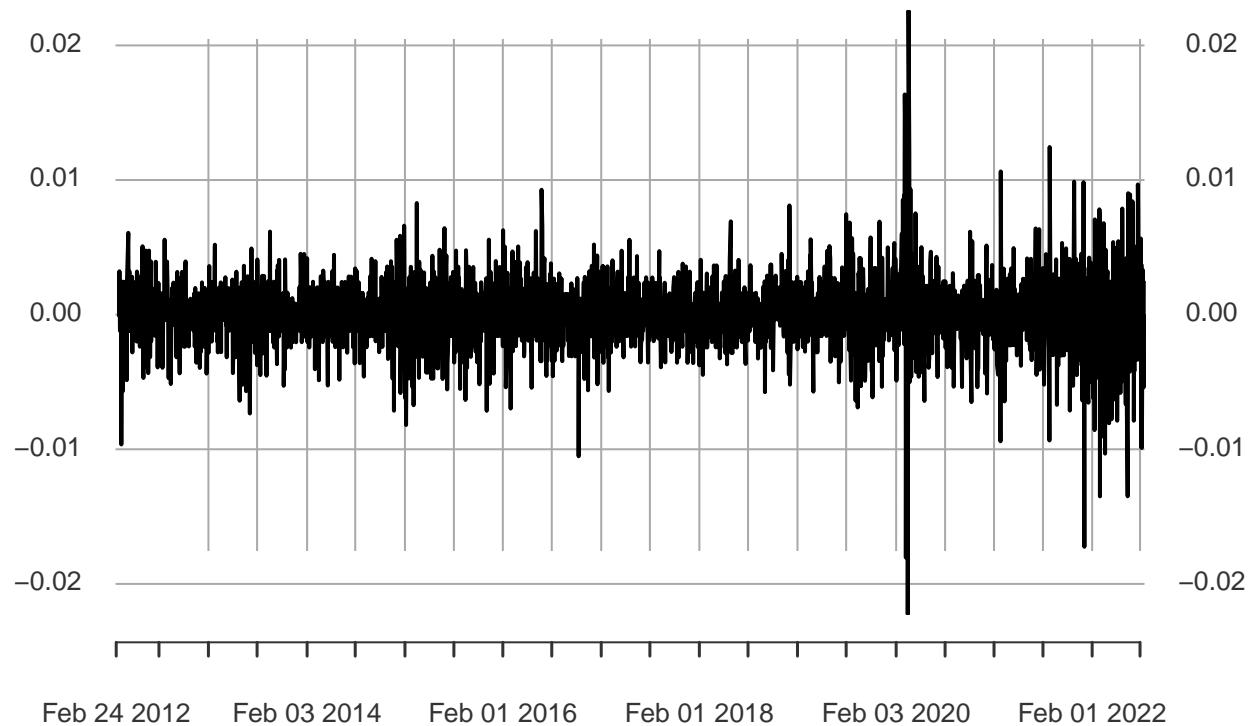
CICI(JPSTa)

2017–05–19 / 2022–08–12



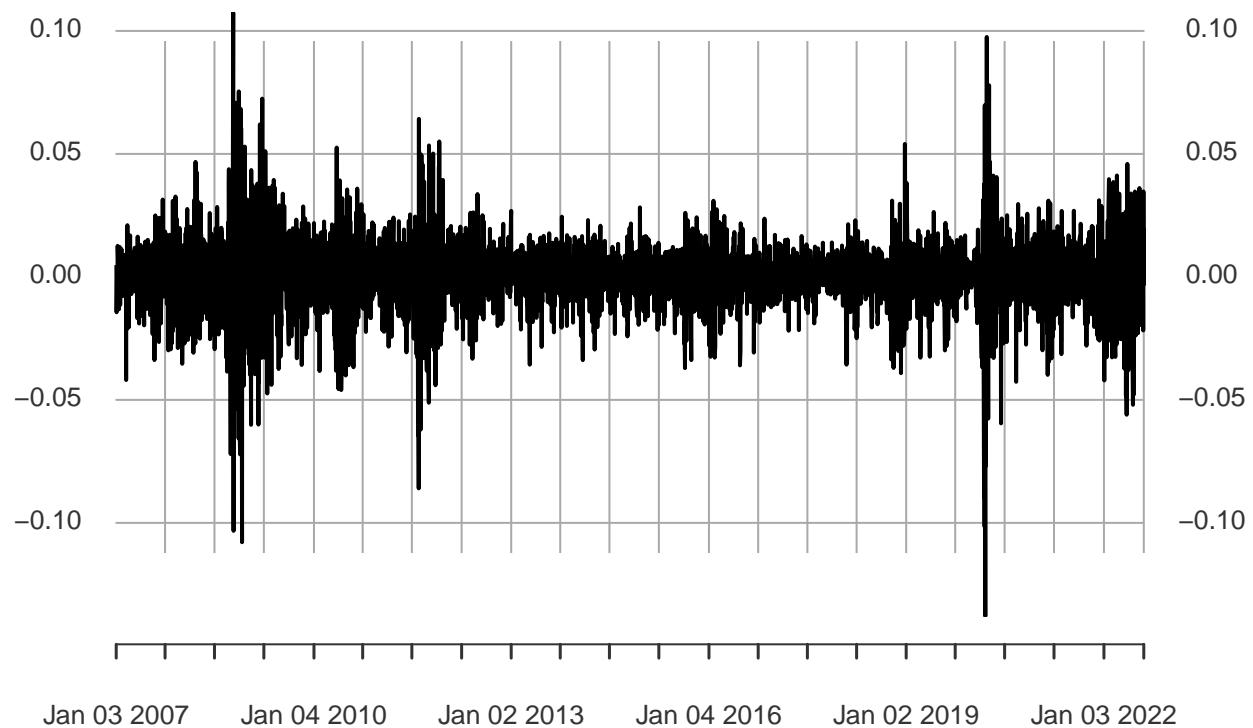
CICI(GOVTa)

2012-02-24 / 2022-08-12



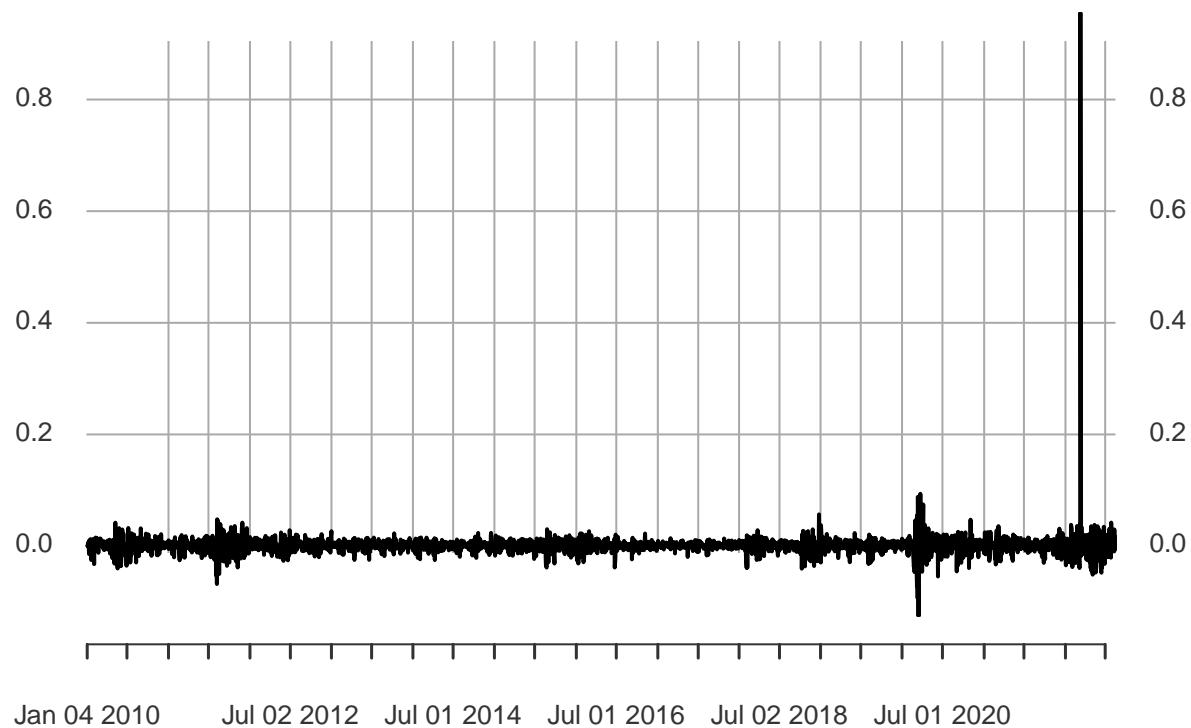
CICI(VBKa)

2007–01–03 / 2022–08–12



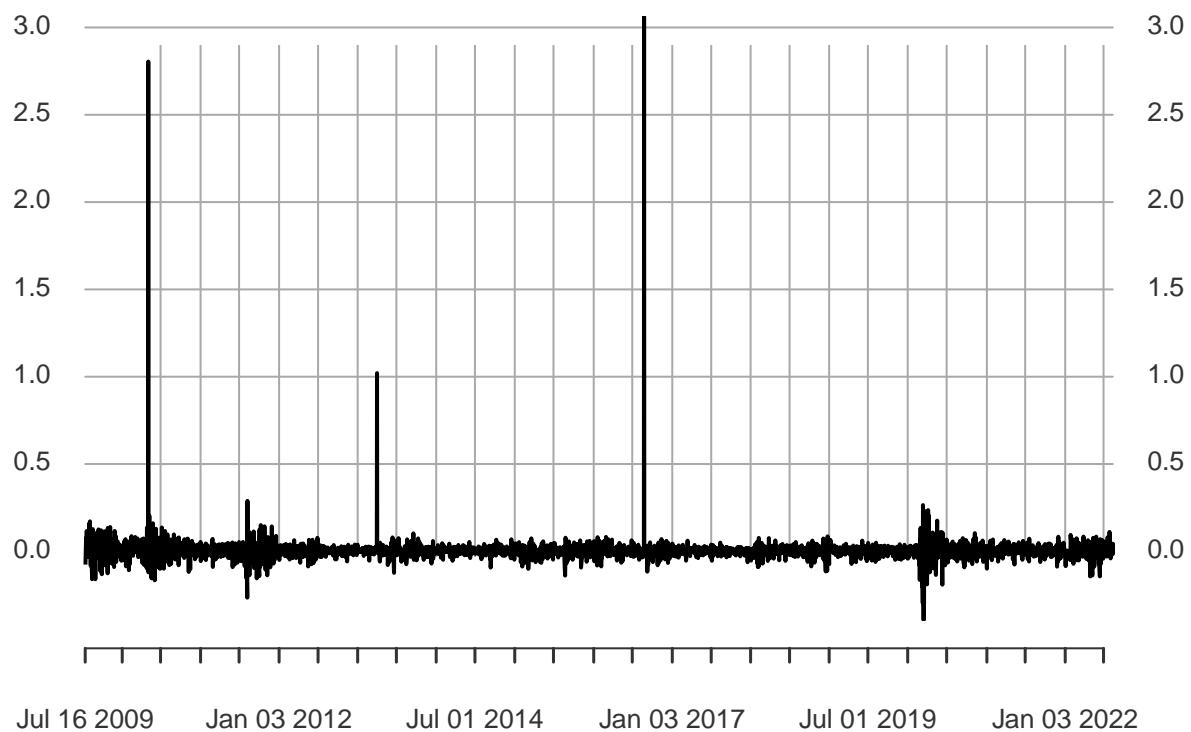
CICI(SCHa)

2010–01–04 / 2022–08–12



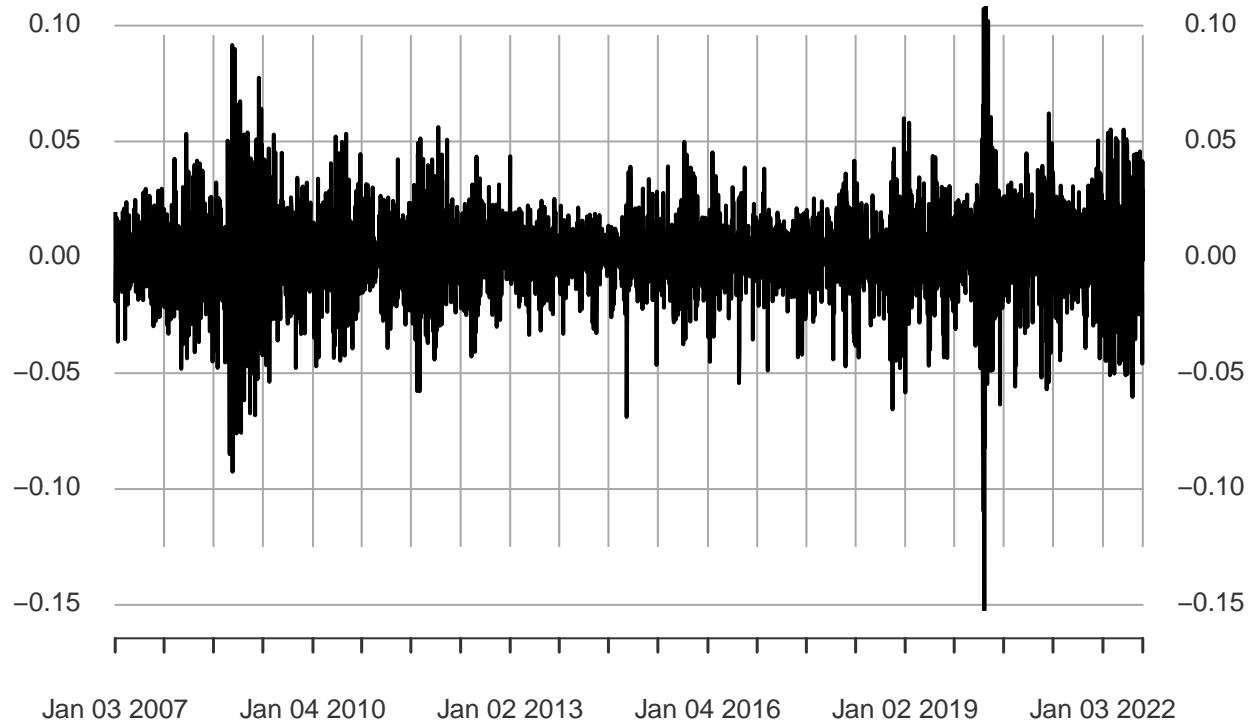
CICl(DRNa)

2009-07-16 / 2022-08-12

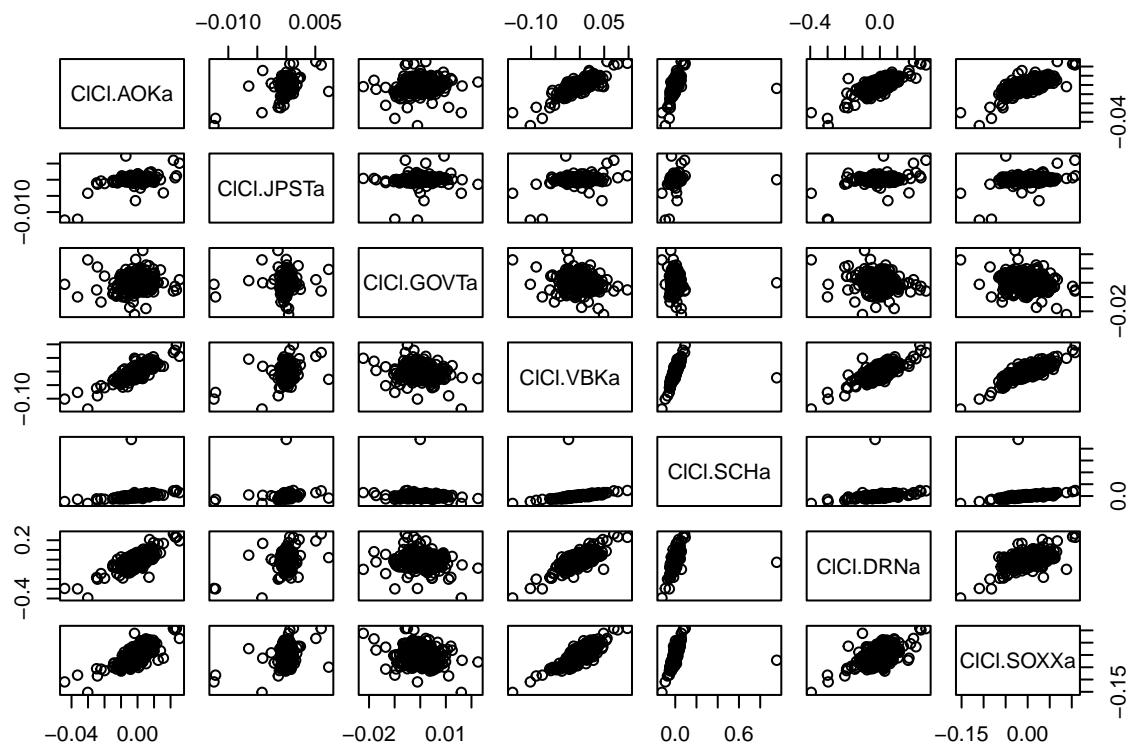


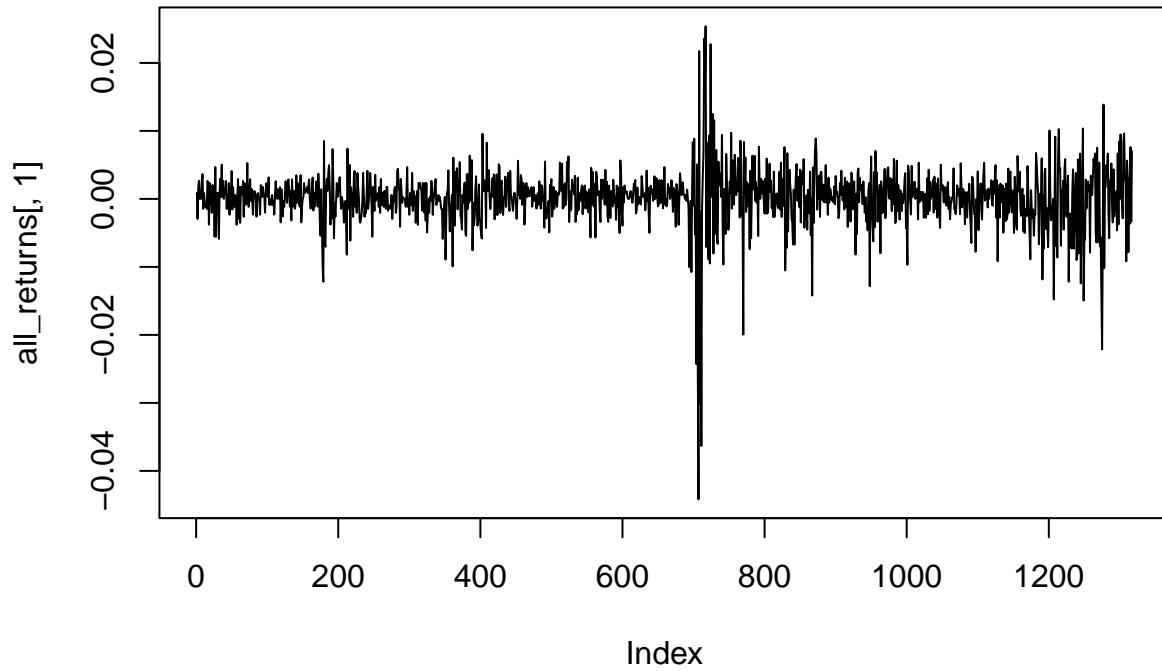
CICI(SOXXa)

2007-01-03 / 2022-08-12



```
##          C1C1.AOKa C1C1.JPSTa C1C1.GOVTa      C1C1.VBKa C1C1.SCHa C1C1.DRNa
## 2007-01-03      NA        NA        NA        NA        NA        NA
## 2007-01-04      NA        NA        NA  0.004758204      NA        NA
## 2007-01-05      NA        NA        NA -0.014359945      NA        NA
## 2007-01-08      NA        NA        NA  0.003874768      NA        NA
## 2007-01-09      NA        NA        NA  0.002161603      NA        NA
## 2007-01-10      NA        NA        NA  0.001078401      NA        NA
##          C1C1.SOXXa
## 2007-01-03      NA
## 2007-01-04  0.019562682
## 2007-01-05 -0.019026121
## 2007-01-08  0.003944806
## 2007-01-09  0.003601784
## 2007-01-10  0.017128858
```





```

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VBK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/VBK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SCHG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SCHG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/DRN?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

```

```

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/DRN?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SOXX?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SOXX?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

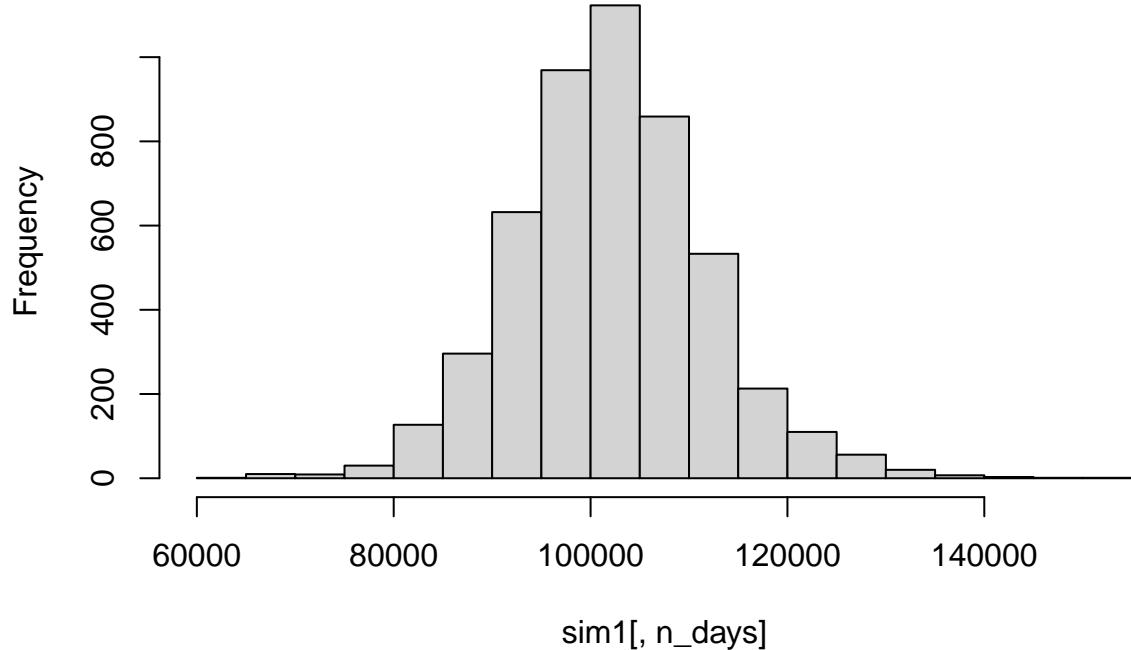
##          VBK.Open VBK.High  VBK.Low  VBK.Close VBK.Volume VBK.Adjusted
## 2017-06-01 139.1914 141.1713 139.1041 141.1713     124100    141.1713
## 2017-06-02 141.4139 142.3941 141.1713 142.0933     110900    142.0933
## 2017-06-05 142.0448 142.1127 141.2975 141.4333      94300    141.4333
## 2017-06-06 140.7637 141.8507 140.4531 141.2004     124000    141.2004
## 2017-06-07 141.3072 141.5692 140.6375 141.1810      69800    141.1810
## 2017-06-08 141.1034 142.2777 140.7249 142.0545     106000    142.0544

##          C1C1.VBKa      C1C1.SCHa C1C1.DRNa C1C1.SOXXa
## 2010-01-04       NA           NA       NA       NA
## 2010-01-05       NA  0.0026465028       NA       NA
## 2010-01-06       NA -0.0007541478       NA       NA
## 2010-01-07       NA  0.0030188679       NA       NA
## 2010-01-08       NA  0.0063957863       NA       NA
## 2010-01-11       NA  0.0003738318       NA       NA

##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 100548.44 100750.24 101248.83 100657.63 99209.73 99115.89 101132.86
## result.2 102035.74 101496.08 101831.65 102094.38 100994.10 102720.64 104767.22
## result.3 100302.17 95567.23 97710.48 98419.42 98299.55 101551.67 101891.70
## result.4 100361.48 100514.58 101719.68 100619.61 100239.18 100538.64 100905.96
## result.5  99809.67 96554.11 93922.91 96959.83 97317.01 95323.41 94600.65
## result.6  99524.00 100506.03 101197.64 101491.64 102875.99 103778.14 103991.06
##          [,8]      [,9]      [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  98949.24 99450.93 100436.24 101945.2 103061.22 104636.60 105712.35
## result.2 106165.89 107429.76 105634.69 103508.7 99184.80 96033.45 96700.88
## result.3 101510.60 101545.71 99497.66 100144.5 97776.87 96673.38 97823.76
## result.4 101380.63 102123.94 104416.27 103358.9 102307.67 101746.17 101695.36
## result.5  94535.66 94099.71 91385.24 83433.3 83420.27 83582.36 83611.68
## result.6 105742.83 106241.97 108459.73 107369.4 108429.75 106231.10 105900.41
##          [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 106699.92 107366.39 108059.18 109558.05 110385.80 110204.38
## result.2  92779.33 93268.44 93376.55 93328.01 93265.58 93568.94
## result.3  94670.03 95345.15 95890.87 97265.75 100048.08 100032.83
## result.4 103811.59 103412.99 102906.22 102268.66 102393.92 102714.02
## result.5  82293.09 81859.08 83943.96 83833.08 84995.18 84691.85
## result.6 106141.36 106582.46 106561.40 105830.34 102914.89 105947.99

```

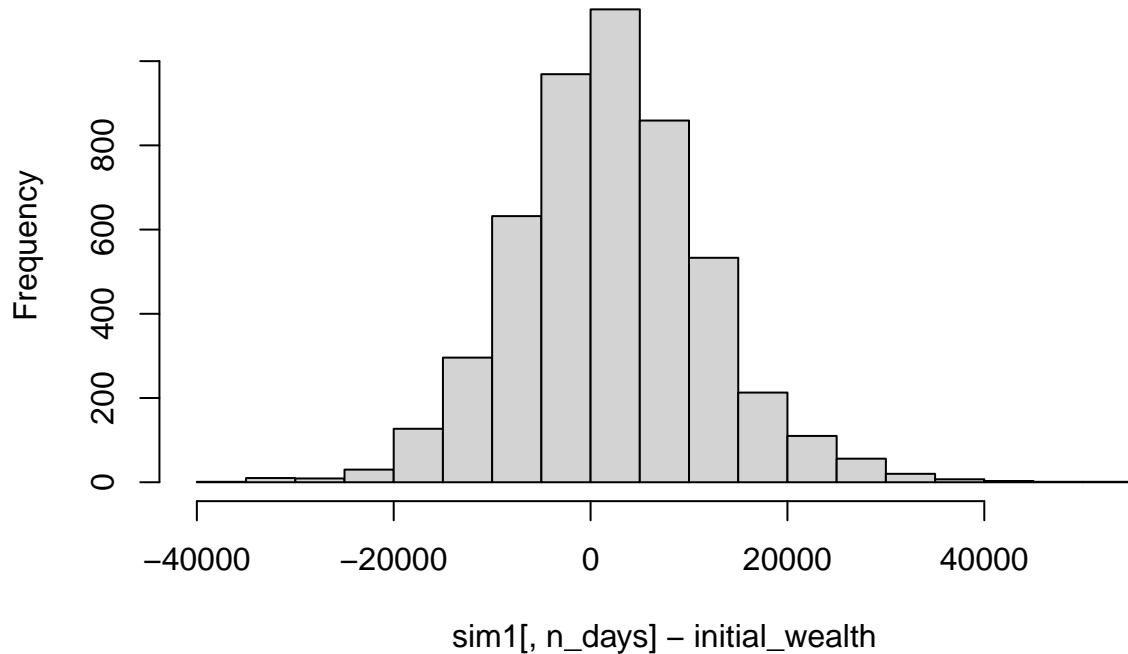
Histogram of sim1[, n_days]



```
## [1] 102105.7
```

```
## [1] 2105.711
```

Histogram of sim1[, n_days] – initial_wealth



```
##      5%
## -13281.28

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/AOK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/AOK?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/JPST?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/JPST?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
```

```

## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GOVT?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/GOVT?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

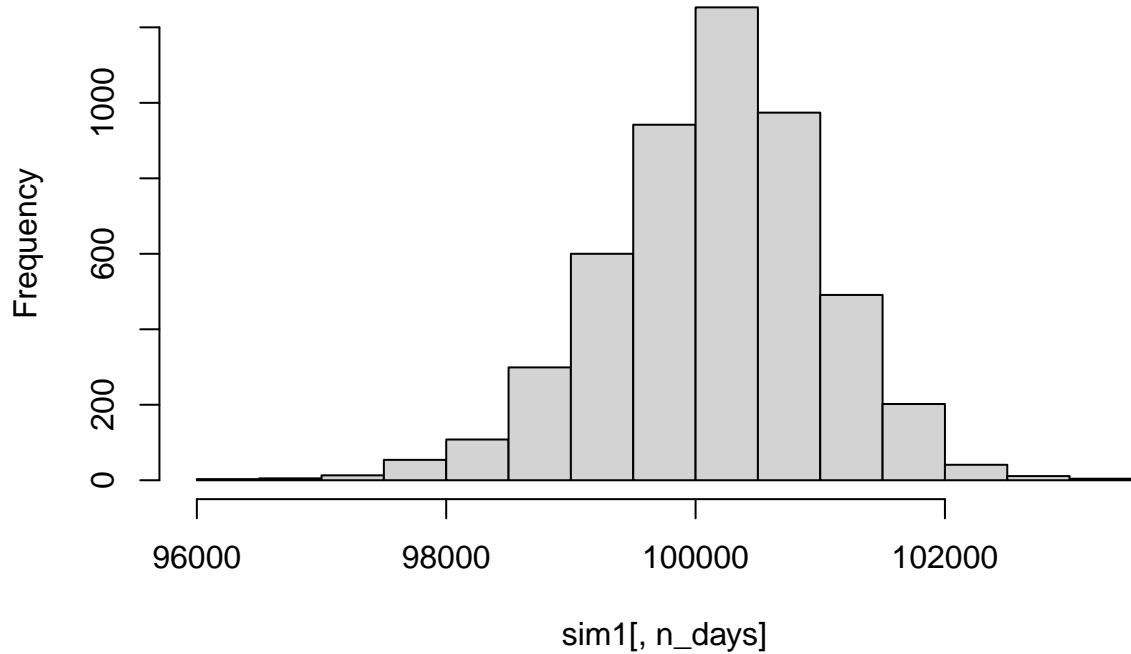
## GOVT.Open GOVT.High GOVT.Low GOVT.Close GOVT.Volume GOVT.Adjusted
## 2017-06-01 23.13438 23.15270 23.10692 23.15270 677900 23.15271
## 2017-06-02 23.19847 23.23509 23.19847 23.21678 697500 23.21677
## 2017-06-05 23.18931 23.19847 23.18016 23.18931 672900 23.18931
## 2017-06-06 23.24424 23.24424 23.20762 23.24424 432000 23.24424
## 2017-06-07 23.21678 23.22593 23.18016 23.18931 393400 23.18931
## 2017-06-08 23.18931 23.18931 23.14354 23.18016 742600 23.18016

## ClC1.AOKa ClC1.JPSTa ClC1.GOVTa
## 2017-06-02 0.0036464743 9.984019e-05 0.0027678923
## 2017-06-05 -0.0014649575 0.000000e+00 -0.0011830047
## 2017-06-06 0.0008802523 2.003396e-05 0.0023686932
## 2017-06-07 -0.0002932571 0.000000e+00 -0.0023630958
## 2017-06-08 -0.0011729326 2.796309e-04 -0.0003947888
## 2017-06-09 -0.0005871990 0.000000e+00 -0.0007899289

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## result.1 100026.6 99937.82 99893.72 99883.15 100008.36 100140.06 101227.81
## result.2 100173.2 100370.31 100216.68 99882.24 99851.04 99535.08 99531.80
## result.3 99930.9 99974.34 100073.33 100079.55 100125.48 100018.65 99921.87
## result.4 100021.5 99703.07 99653.77 99861.36 99732.79 99764.73 99723.46
## result.5 100053.1 100132.13 100058.53 100186.59 100262.09 100186.00 100209.35
## result.6 100074.1 100184.65 100232.03 100175.67 100048.29 100195.51 100161.90
## [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## result.1 100871.69 100896.85 100776.97 100916.45 101125.87 101088.94 101202.95
## result.2 98835.67 98998.61 98891.22 98432.54 98739.97 98742.37 99015.78
## result.3 99814.16 99851.38 99997.28 100002.26 99935.81 99691.46 99837.19
## result.4 99720.03 99712.55 99724.31 99712.19 99858.41 99870.03 99948.57
## result.5 100277.79 100529.37 100461.43 100570.96 100553.82 100507.41 100524.32
## result.6 100307.55 100368.48 100438.38 100550.20 100517.99 100512.20 100500.00
## [,15] [,16] [,17] [,18] [,19] [,20]
## result.1 101083.91 101078.43 101104.34 101119.99 101154.95 101255.23
## result.2 99216.51 99114.24 99242.71 99167.48 99298.03 99548.28
## result.3 99977.96 100041.52 99974.95 99914.97 100060.16 100022.00
## result.4 99987.43 100216.35 100344.87 100196.48 100319.76 100412.72
## result.5 100694.83 100734.27 100808.39 100979.26 101035.40 101068.48
## result.6 100593.69 100741.39 100928.02 101123.98 101114.81 101340.69

```

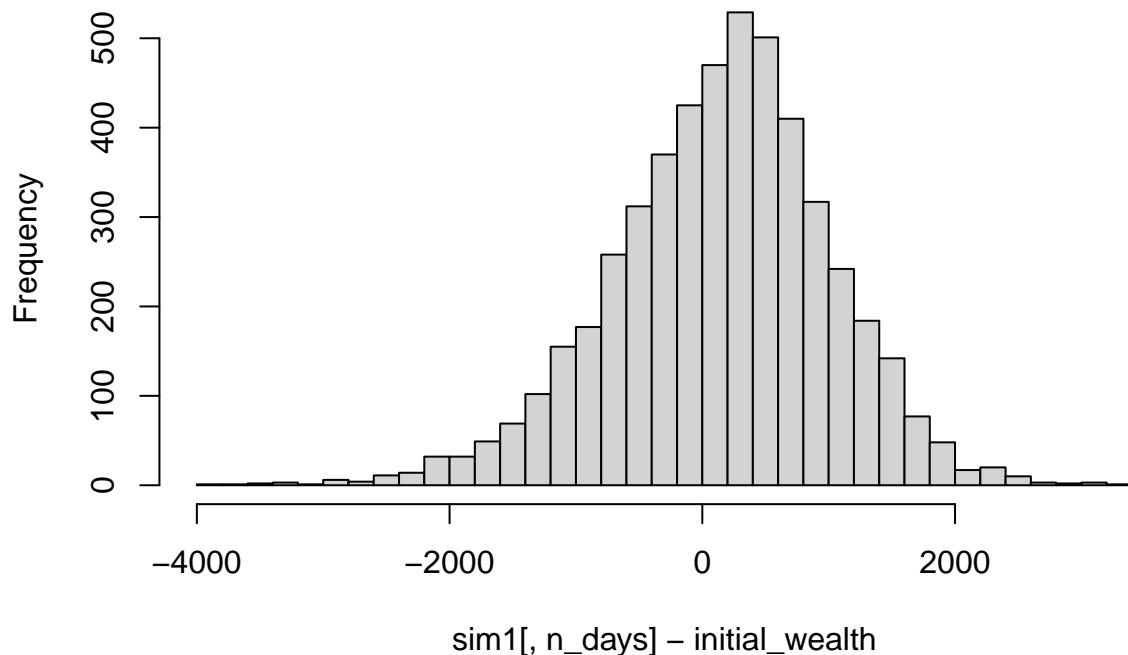
Histogram of sim1[, n_days]



```
## [1] 100143.3
```

```
## [1] 143.3204
```

Histogram of sim1[, n_days] – initial_wealth



```
##      5%
## -1342.545
```

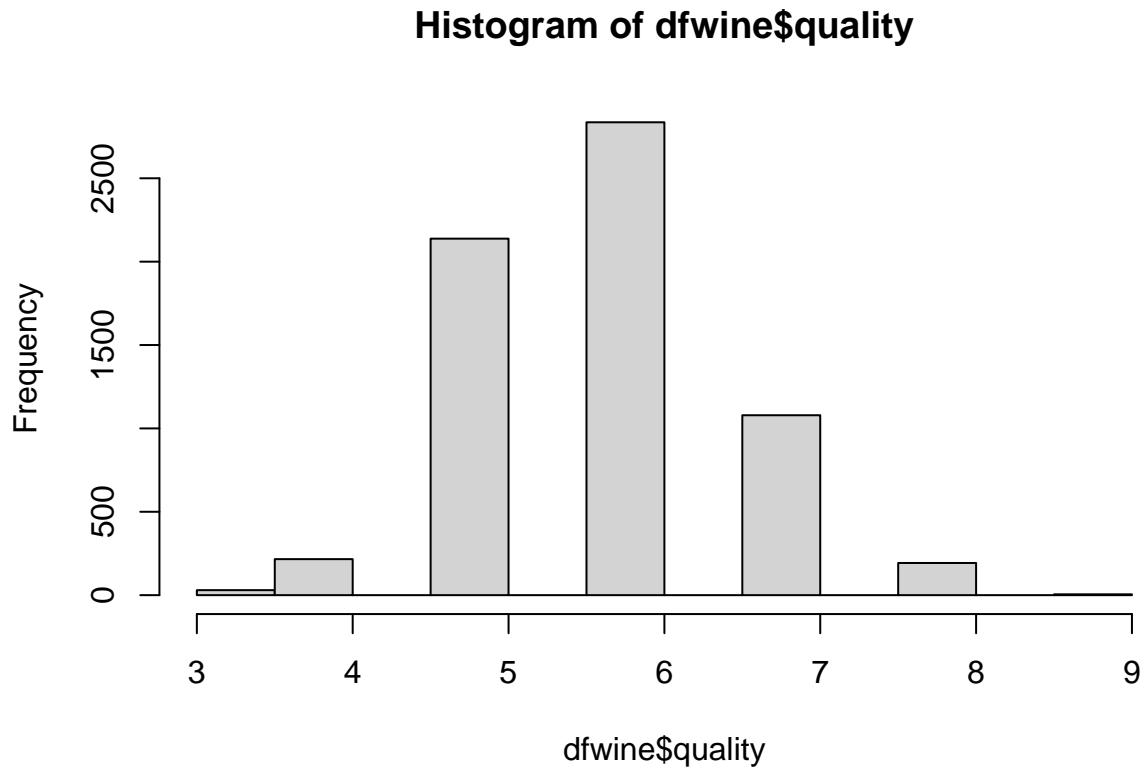
Comments: We chose to have two separate portfolios based off of aggressiveness vs. non-aggressiveness. The aggressive portfolio comprised of: VBK, SCHG, DRN, and SOXX. The safe portfolio comprised of: AOK, JPST, and GOVT.

The aggressive portfolio had a calculated mean profit of \$102,109.40 over the year, with an expectation to gain around 2,109 dollars per day. Using a 5% value at risk means that out of our simulations performed, 95% of them will perform better than a loss of 13,078.49 dollars over a 4-week period.

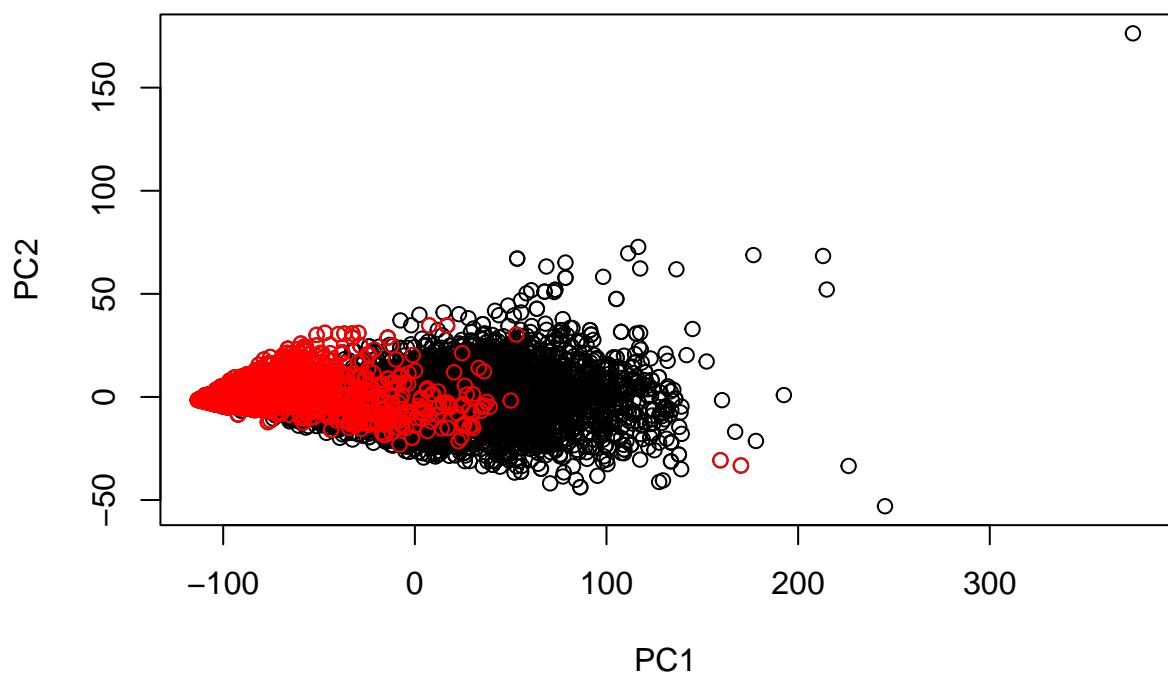
The safe portfolio had a calculated mean profit of \$100,161.60 over the year, with an expectation to gain around 162 dollars per day. Using a 5% value at risk means that out of our simulations, 95% of them will perform better than a loss of 1,351.52 dollars over a 4-week period.

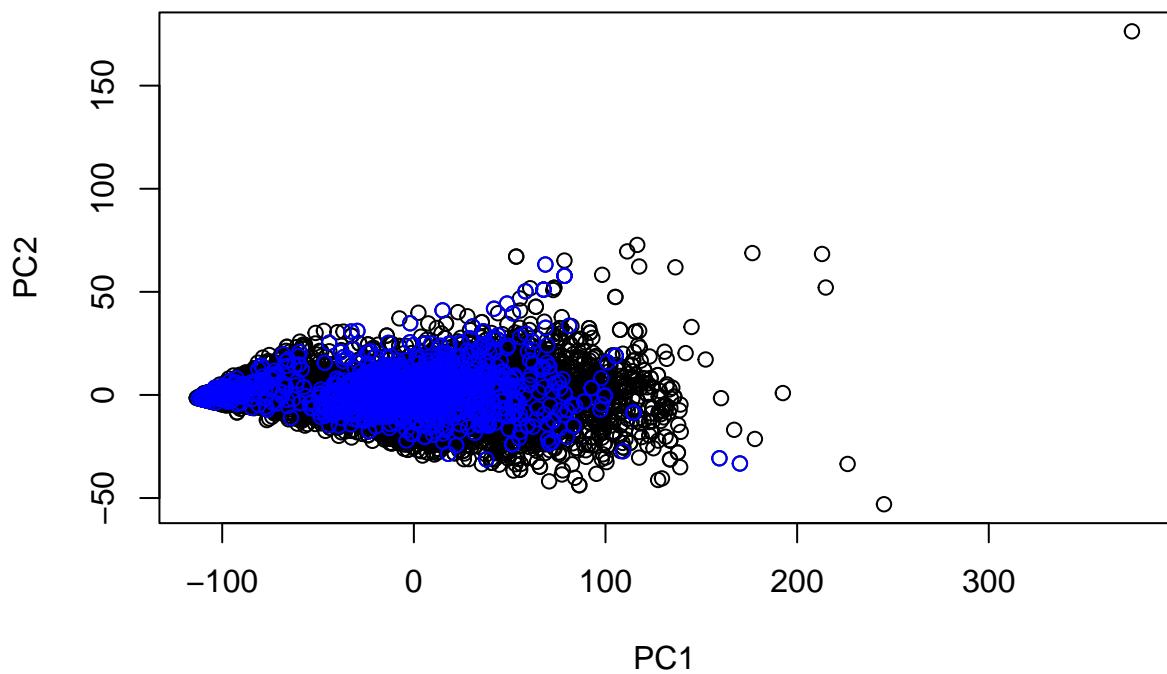
Out of our two portfolios, the aggressive portfolio returned a slightly higher mean profit over the year, although the daily amount gained per day vastly out-performed the safe portfolio (2,109 dollars vs. 162 dollars). Although the aggressive portfolio can incur better gains than the safe portfolio, the potential loss is also much greater than the safe portfolio (5% of simulations have a mean loss of 13,078.49 dollars vs. 1,351.52 dollars).

Clustering and PCA



```
##          PC1        PC2        PC3
## fixed.acidity -7.407964e-03 -5.365624e-03 0.0237980377
## volatile.acidity -1.184329e-03 -7.844986e-04 0.0008841018
## citric.acid    4.868693e-04 -2.479470e-04 0.0019286942
## residual.sugar 4.101972e-02  1.863643e-02 0.9952741053
## chlorides      -1.681987e-04  6.726744e-05 0.0001730199
## free.sulfur.dioxide 2.304818e-01  9.726583e-01 -0.0272149098
## total.sulfur.dioxide 9.721668e-01 -2.314097e-01 -0.0358290013
## density        1.772339e-06  1.329966e-06 0.0004604088
## pH             -6.555205e-04  6.479869e-04 -0.0069116181
## sulphates     -7.043386e-04  3.463575e-04 -0.0019352912
## alcohol       -5.451737e-03  2.850174e-03 -0.0823558184
```





Cluster Dendrogram



```
wine_distance_matrix
hclust (*, "complete")
```

```
##   1   2   3   4   5   6   7
## 2273 2385 1563 101 169 5   1
```

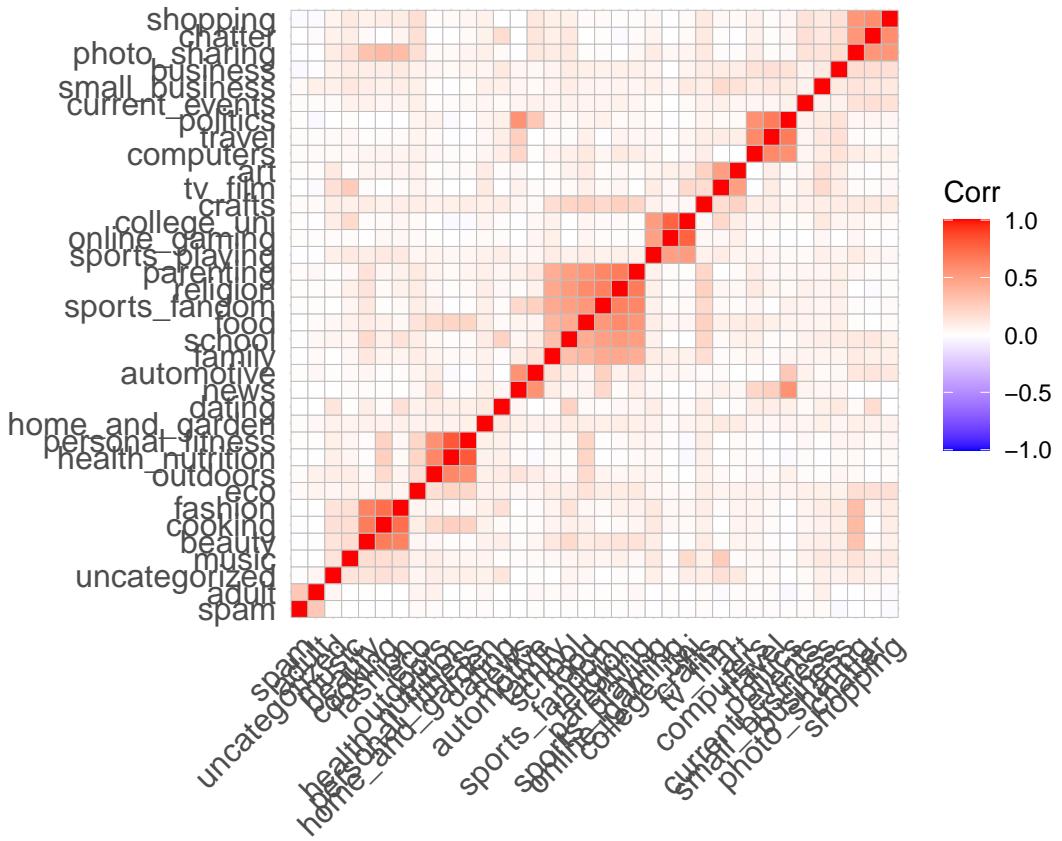
Comments: In our initial analysis, we plotted the histogram of quality values for the wines in our dataset to get an understanding of the distribution of the target variable. We then selected and scaled the 11 predictor variable values. Following that we defined the output variable values based on color (red) and quality (quality > 6 were chosen as the high quality wines).

1] After the preliminary exploratory data analysis, we ran a PCA model with 2 principal components. Observations: 1. In regards to the wine color, the majority of red wines tend to have a PC1 value < 0 whereas most of the white wines had values > 0. This is a good indicator that positive values for PC1 indicate white as the wine color. 2. For the quality of the wine, the distinction was not that clear from the plot. However, we did find that almost all the higher quality wines had PC1 values < 100. 3. Overall, the PC1 proved to be quite significant in segregating wines based on their color and to some extent, the quality as well.

2] After implementing PCA, we tried hierarchical clustering on the dataset. Observations and Key Points: 1. We decided to proceed with 7 clusters with linkage factor as 'complete'. 2. The cluster dendrogram was extremely convoluted with numerous splits suggesting that hierarchical clustering might not be the optimal choice in this test case. 3. The end results from clustering were very sensitive to outliers with only a few observations making up some of the clusters. There were no obvious actionable insights that were drawn from the results.

Market segmentation

	PC1	PC2
## chatter	8.377923e-02	6.174357e-01
## current_events	1.674083e-02	5.633466e-02
## travel	7.694647e-03	1.026362e-01
## photo_sharing	1.319231e-01	4.784651e-01
## uncategorized	2.524063e-02	2.637812e-02
## tv_film	5.842740e-03	3.273925e-02
## sports_fandom	2.149987e-02	5.981140e-02
## politics	7.858764e-03	1.707149e-01
## food	9.520239e-02	1.097888e-02
## family	1.999472e-02	4.337977e-02
## home_and_garden	1.529762e-02	1.890571e-02
## music	2.478241e-02	4.490862e-02
## news	1.821436e-02	6.362012e-02
## online_gaming	1.581928e-02	1.210238e-01
## shopping	5.245830e-02	2.497367e-01
## health_nutrition	8.084217e-01	-2.731887e-01
## college_uni	5.810219e-03	1.672739e-01
## sports_playing	1.942244e-02	4.783238e-02
## cooking	3.404963e-01	2.701636e-01
## eco	3.918988e-02	2.448001e-02
## computers	1.599136e-02	6.154333e-02
## business	1.321910e-02	3.705721e-02
## outdoors	1.446150e-01	-3.628227e-02
## crafts	2.138903e-02	2.954968e-02
## automotive	6.331646e-03	7.134999e-02
## art	1.932288e-02	2.483807e-02
## religion	2.785915e-02	3.742165e-02
## beauty	6.000130e-02	1.107477e-01
## parenting	2.929342e-02	4.023737e-02
## dating	3.977599e-02	6.351116e-02
## school	2.176039e-02	5.556398e-02
## personal_fitness	3.873704e-01	-1.057094e-01
## fashion	9.682483e-02	1.739817e-01
## small_business	5.570533e-03	2.839783e-02
## spam	4.113457e-05	1.035245e-05
## adult	-4.331520e-04	2.488791e-03



```
##   1   2   3   4   5   6   7
## 493 3686 1209  607  557  397  933
```

```
## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Warning in mean.default(x, ..., na.rm = na.rm): argument is not numeric or
## logical: returning NA

## Clustering..K.Means.      Max1..K.Means.      Max2..K.Means.      Max3..K.Means.
```

```

## 1          1      chatter_1    photo_sharing_1      travel_1
## 2          2      cooking_1    photo_sharing_1      fashion_1
## 3          3 health_nutrition_1 personal_fitness_1 cooking_1
## 4          4 college_uni_1   online_gaming_1  photo_sharing_1
## 5          5      chatter_1    photo_sharing_1 current_events_1
## 6          6      politics_1    travel_1           news_1
## 7          7 sports_fandom_1 religion_1           food_1
## Original.Groups..Correaltion.Plot.Analysis.
## 1          Group 7
## 2          Group 2
## 3          Group 3
## 4          Group 5
## 5          Group 7
## 6          Group 6
## 7          Group 4

```

Correlation Plot: We began by looking at a simple correlation plot, to see if we could identify any interesting relationships between the different areas of interest. Based on our intuition and understanding of the correlation plot, we concluded there were 7 groups of correlation features that we listed below.

- Group 1 - Bots: adult and spam highly correlated
- Group 2 - fashion, cooking, and beauty
- Group 3 - personal fitness, health nutrition, outdoors
- Group 4 - Parenting, religion, sports_fandom, food, school, family
- Group 5 - college_uni, online_gaming, sports_playing
- Group 6 - politics, travel, computers
- Group 7 - shopping, chatter, photo_sharing

K-Means: Once we established these groups we ran a k-means model to see if we would get clusters that had the same, or similar features to the ones in our initial groups. We average the different amount of tweets for each area of interest in each cluster. We then selected the 3 highest feature averages from each group. We used these three features in each to determine if the clusters determined by kmeans were similarly defined to the groups we had identified using the correlation plot.

Result: As seen in our last data frame (comparison), 6 out of the 7 clusters seemed to have a match with our original groups. The only original group we could not find a match for was Group 1. However, the other 6 clusters/groups are incredibly valuable because they can serve as guides for different market segments that the company can cater to. For example, Group 3 is all about being active and healthy, so that group can now be target marketed to in a way that serves those interests.

The Reuters corpus

```

## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(tolower)):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removeNumbers)): transformation drops documents

## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removePunctuation)): transformation drops documents

```

```

## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(stripWhitespace)): transformation drops documents

## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(removeWords), :
## transformation drops documents

## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(tolower)):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removeNumbers)): transformation drops documents

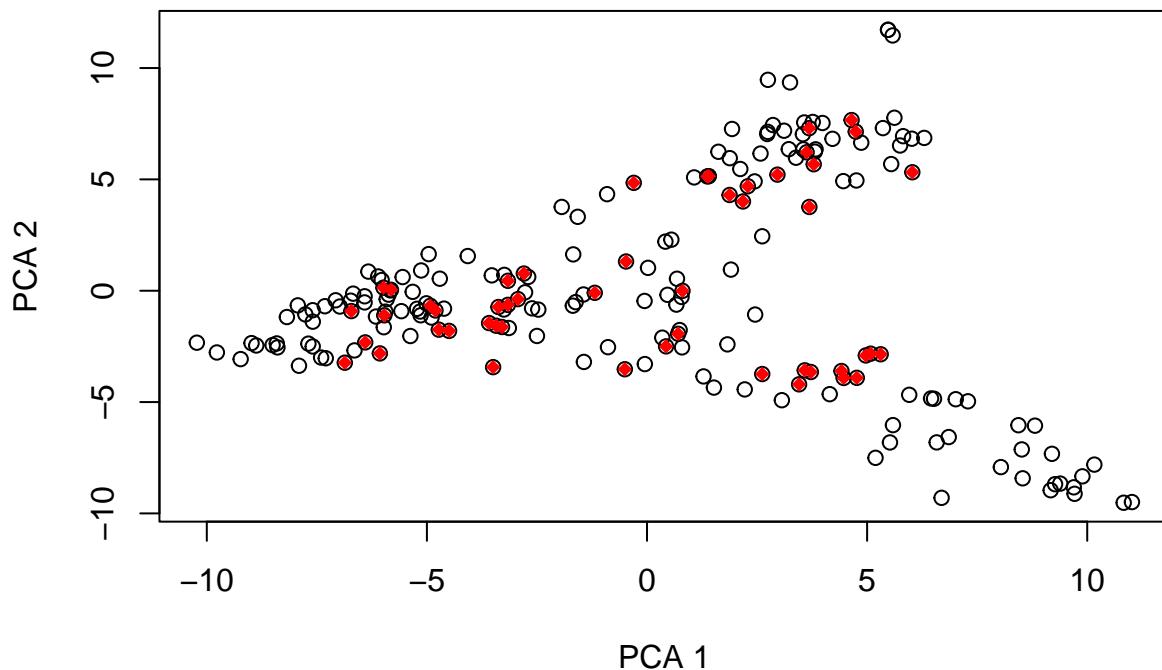
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removePunctuation)): transformation drops documents

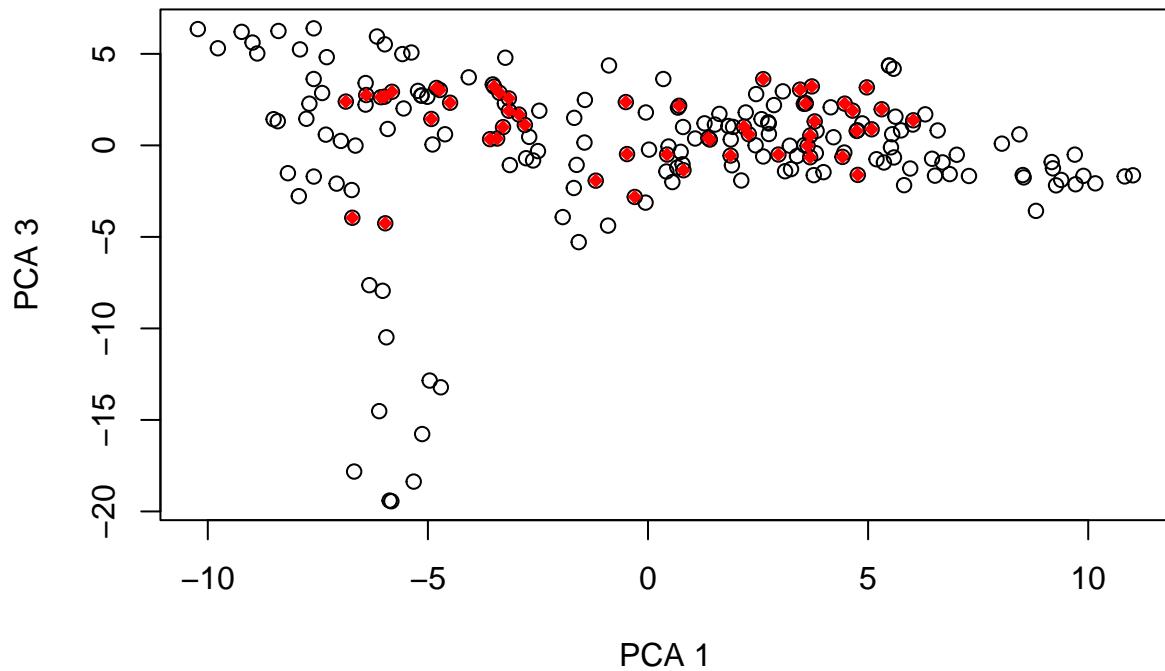
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(stripWhitespace)): transformation drops documents

## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(removeWords), :
## transformation drops documents

##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.
## 0.0000  0.3186  0.4307  0.4983  0.6097  3.1179

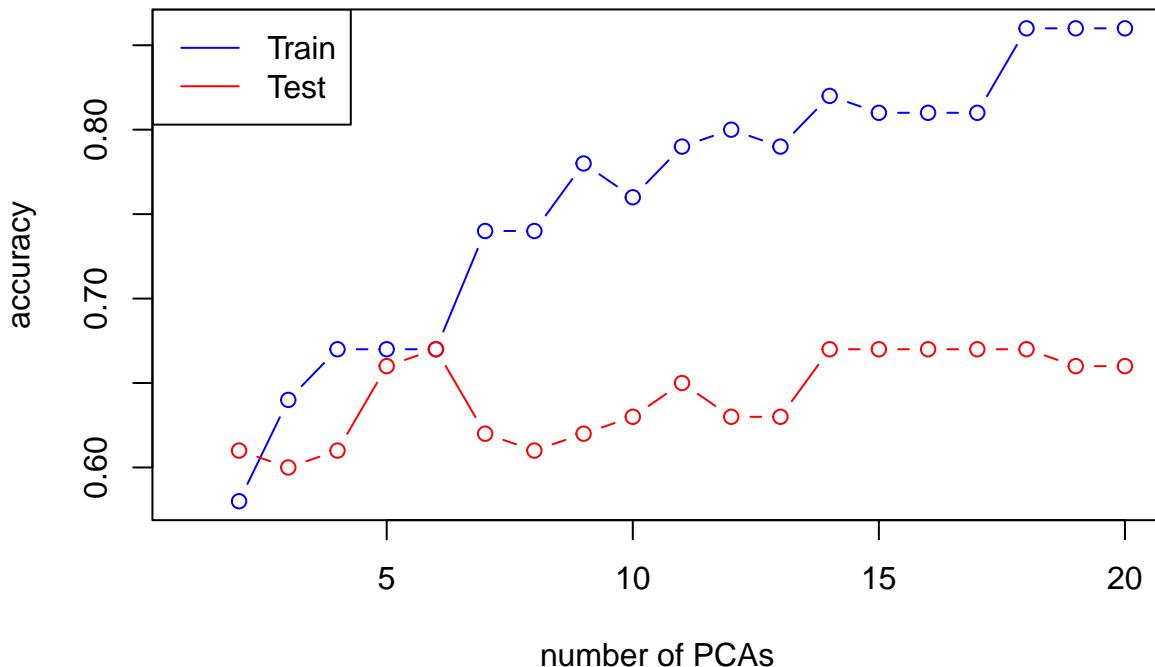
```





```
##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.  
## 0.0000  0.3018  0.4107  0.4836  0.5701  2.7663
```

Accuracy in Train/Test by #PCAs



Comments:

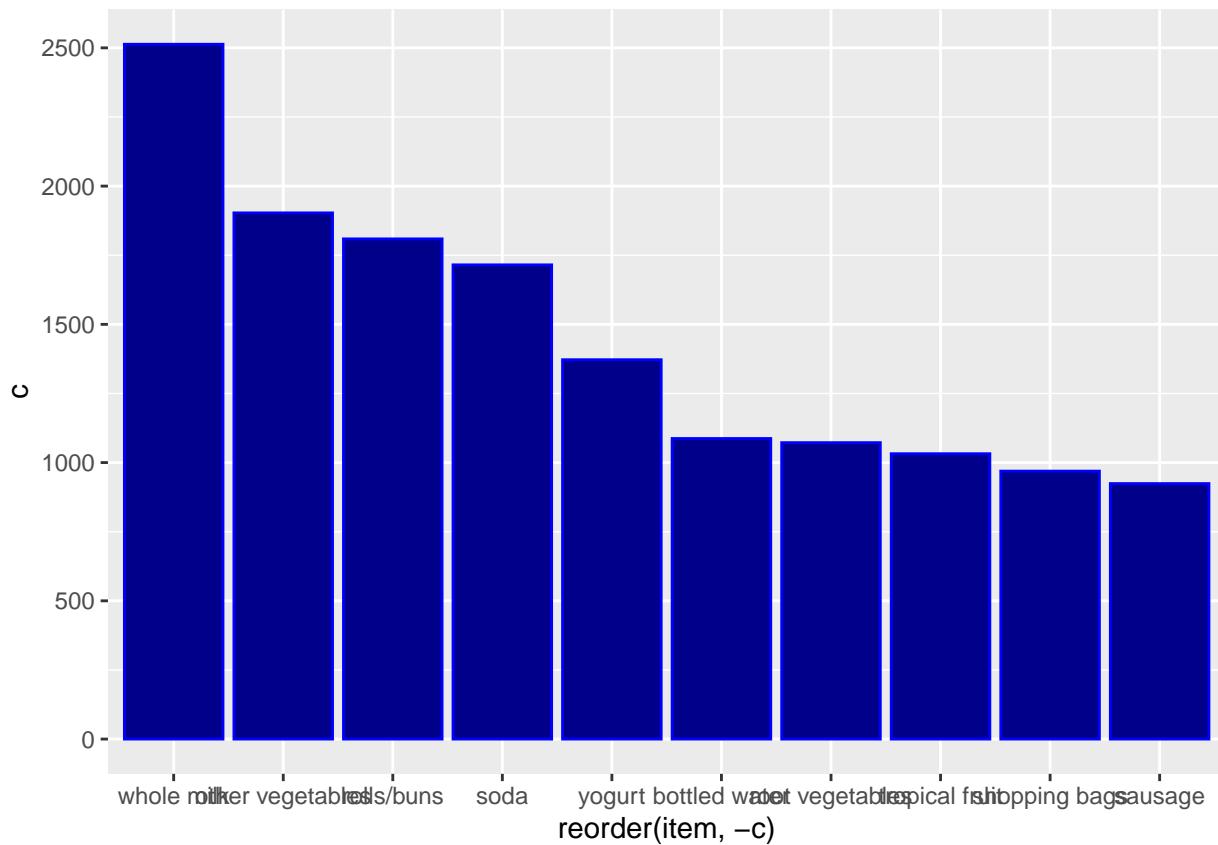
Comments: Question: Are we able to distinguish who wrote a document based on the content and words of that document itself?

Approach: For this problem, we selected documents of four authors: Simon Cowell, Aaron Pressman, Joe Ortiz, and William Kazer, in Train and Test. So, using analytical tools, can we recognize which documents were written by Simon Cowell and which were not? First of all, we began our approach with a Text Analysis. As we learn in class, the first step is converting the unstructured data into a structured one. So, to begin we tokenized the strings and symbols, then we removed stop words and useless characters, and finally we have got DTM matrix and TFIDF weights. Now, we get our data structured.

Results: Second, to address the question above, we ran a PCA. When we plotted the first and second principal components, we observed that there is no clear pattern to distinguish which documents were written by Simon Cowell, and on the first versus third principal components plot we can't distinguish a clear pattern either. Finally, we ran Logistic regression models using the first principal components as predictors (iterating from 2 to 20 PC). We can observe that as we increase the number of principal components as predictors, the accuracy increases in Train to distinguish written documents by Simon Cowell. However, in the Test set the accuracy become flat after the 6 number of principal components, reaching 64% accuracy on average, which is quite low.

Conclusion: To conclude, after using different analytical tools to distinguish written documents, we can say that it is difficult that task to make document discrimination. However, we can highlight that although the results were not top notch, it is probable that if we extend the analysis more in deep and trying different other predictive methods and different authors, we might get better outcomes.

Association rule mining



```
## transactions as itemMatrix in sparse format with
## 15296 rows (elements/itemsets/transactions) and
## 169 columns (items) and a density of 0.01677625
##
## most frequent items:
##      whole milk other vegetables      rolls/buns      soda
##      2513           1903           1809           1715
##      yogurt          (Other)
##      1372           34055
##
## element (itemset/transaction) length distribution:
## sizes
##   1   2   3   4
## 3485 2630 2102 7079
##
##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.
##      1.000  2.000  3.000  2.835  4.000  4.000
##
## includes extended item information - examples:
##      labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3 baby cosmetics
```

```

##  

## includes extended transaction information - examples:  

##   transactionID  

## 1          1  

## 2          2  

## 3          3  

##  

## 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
## [1]	{}	=> {soda}	0.112120816
## [2]	{}	=> {rolls/buns}	0.118266213
## [3]	{}	=> {other vegetables}	0.124411611
## [4]	{}	=> {whole milk}	0.164291318
## [5]	{butter milk}	=> {whole milk}	0.005033996
## [6]	{onions}	=> {root vegetables}	0.005295502
## [7]	{onions}	=> {other vegetables}	0.007452929
## [8]	{onions}	=> {whole milk}	0.005360879
## [9]	{berries}	=> {other vegetables}	0.005164749
## [10]	{berries}	=> {whole milk}	0.005230126
## [11]	{hamburger meat}	=> {other vegetables}	0.006210774
## [12]	{hamburger meat}	=> {whole milk}	0.005818515
## [13]	{dessert}	=> {whole milk}	0.006603033
## [14]	{cream cheese }	=> {yogurt}	0.005033996
## [15]	{chocolate}	=> {soda}	0.005360879
## [16]	{chicken}	=> {other vegetables}	0.007975941
## [17]	{chicken}	=> {whole milk}	0.006341527
## [18]	{frozen vegetables}	=> {whole milk}	0.005687762
## [19]	{canned beer}	=> {soda}	0.006537657
## [20]	{beef}	=> {citrus fruit}	0.005099372
## [21]	{beef}	=> {root vegetables}	0.008695084
## [22]	{root vegetables}	=> {beef}	0.008695084
## [23]	{beef}	=> {other vegetables}	0.008302824
## [24]	{beef}	=> {whole milk}	0.008172071

```

## [25] {curd}                                     => {yogurt}          0.007649059
## [26] {curd}                                     => {other vegetables} 0.007060669
## [27] {curd}                                     => {whole milk}      0.012617678
## [28] {margarine}                                => {rolls/buns}     0.005491632
## [29] {butter}                                    => {yogurt}          0.006210774
## [30] {butter}                                    => {other vegetables} 0.008237448
## [31] {butter}                                    => {whole milk}      0.014382845
## [32] {pork}                                      => {root vegetables} 0.006733787
## [33] {pork}                                      => {other vegetables} 0.009283473
## [34] {pork}                                      => {whole milk}      0.008695084
## [35] {frankfurter}                               => {sausage}         0.006472280
## [36] {sausage}                                   => {frankfurter}    0.006472280
## [37] {frankfurter}                               => {tropical fruit} 0.005557008
## [38] {frankfurter}                               => {rolls/buns}     0.006210774
## [39] {frankfurter}                               => {other vegetables} 0.007060669
## [40] {frankfurter}                               => {whole milk}      0.008237448
## [41] {bottled beer}                             => {bottled water}   0.006929916
## [42] {bottled beer}                             => {soda}            0.008302824
## [43] {brown bread}                             => {pastry}          0.005033996
## [44] {brown bread}                             => {rolls/buns}     0.006276151
## [45] {brown bread}                             => {whole milk}      0.006733787
## [46] {domestic eggs}                            => {rolls/buns}     0.008172071
## [47] {domestic eggs}                            => {whole milk}      0.008172071
## [48] {fruit/vegetable juice}                   => {bottled water}   0.005753138
## [49] {fruit/vegetable juice}                   => {soda}            0.009348849
## [50] {shopping bags}                           => {soda}            0.006406904
## [51] {whipped/sour cream}                     => {yogurt}          0.009741109
## [52] {yogurt}                                    => {whipped/sour cream} 0.009741109
## [53] {whipped/sour cream}                     => {rolls/buns}     0.005360879
## [54] {whipped/sour cream}                     => {other vegetables} 0.008302824
## [55] {whipped/sour cream}                     => {whole milk}      0.011440900
## [56] {pip fruit}                                => {citrus fruit}   0.008172071
## [57] {citrus fruit}                            => {pip fruit}       0.008172071
## [58] {pip fruit}                                => {sausage}         0.006210774
## [59] {sausage}                                   => {pip fruit}       0.006210774
## [60] {pip fruit}                                => {tropical fruit} 0.012683054
## [61] {tropical fruit}                           => {pip fruit}       0.012683054
## [62] {pip fruit}                                => {root vegetables} 0.008106695
## [63] {root vegetables}                          => {pip fruit}       0.008106695
## [64] {pip fruit}                                => {other vegetables} 0.010917887
## [65] {pip fruit}                                => {whole milk}      0.012552301
## [66] {pastry}                                    => {soda}            0.007256799
## [67] {pastry}                                    => {rolls/buns}     0.010198745
## [68] {pastry}                                    => {whole milk}      0.009414226
## [69] {citrus fruit}                            => {sausage}         0.006929916
## [70] {sausage}                                   => {citrus fruit}   0.006929916
## [71] {citrus fruit}                            => {tropical fruit} 0.012486925
## [72] {tropical fruit}                           => {citrus fruit}   0.012486925
## [73] {citrus fruit}                            => {root vegetables} 0.008695084
## [74] {root vegetables}                          => {citrus fruit}   0.008695084
## [75] {citrus fruit}                            => {yogurt}          0.006733787
## [76] {citrus fruit}                            => {other vegetables} 0.012813808
## [77] {other vegetables}                        => {citrus fruit}   0.012813808
## [78] {citrus fruit}                            => {whole milk}      0.012813808

```

```

## [79] {sausage}          => {tropical fruit} 0.008172071
## [80] {tropical fruit} => {sausage}        0.008172071
## [81] {sausage}          => {root vegetables} 0.007322176
## [82] {root vegetables} => {sausage}        0.007322176
## [83] {sausage}          => {rolls/buns}     0.010787134
## [84] {sausage}          => {other vegetables} 0.012617678
## [85] {other vegetables}=> {sausage}        0.012617678
## [86] {sausage}          => {whole milk}     0.012552301
## [87] {bottled water}    => {soda}           0.014644351
## [88] {soda}              => {bottled water} 0.014644351
## [89] {bottled water}    => {rolls/buns}     0.008564331
## [90] {tropical fruit}   => {root vegetables} 0.010983264
## [91] {root vegetables}  => {tropical fruit} 0.010983264
## [92] {tropical fruit}   => {yogurt}          0.008172071
## [93] {tropical fruit}   => {other vegetables} 0.015494247
## [94] {other vegetables}=> {tropical fruit} 0.015494247
## [95] {tropical fruit}   => {whole milk}      0.018305439
## [96] {whole milk}        => {tropical fruit} 0.018305439
## [97] {root vegetables}  => {other vegetables} 0.025366109
## [98] {other vegetables}=> {root vegetables} 0.025366109
## [99] {root vegetables}  => {whole milk}      0.022620293
## [100] {whole milk}       => {root vegetables} 0.022620293
## [101] {yogurt}          => {rolls/buns}     0.011898536
## [102] {rolls/buns}      => {yogurt}          0.011898536
## [103] {yogurt}          => {other vegetables} 0.015886506
## [104] {other vegetables}=> {yogurt}          0.015886506
## [105] {yogurt}          => {whole milk}      0.024254707
## [106] {whole milk}       => {yogurt}          0.024254707
## [107] {soda}             => {rolls/buns}     0.014252092
## [108] {rolls/buns}      => {soda}            0.014252092
## [109] {rolls/buns}      => {whole milk}      0.018305439
## [110] {whole milk}       => {rolls/buns}     0.018305439
## [111] {other vegetables}=> {whole milk}      0.040860356
## [112] {whole milk}       => {other vegetables} 0.040860356
## [113] {other vegetables, root vegetables}=> {whole milk} 0.008172071
## [114] {root vegetables, whole milk}        => {other vegetables} 0.008172071
## [115] {other vegetables, whole milk}        => {root vegetables} 0.008172071
## [116] {other vegetables, yogurt}          => {whole milk}      0.006341527
## [117] {whole milk, yogurt}                => {other vegetables} 0.006341527
## [118] {other vegetables, whole milk}       => {yogurt}          0.006341527
##      confidence coverage lift count
## [1] 0.1121208 1.0000000 1.0000000 1715
## [2] 0.1182662 1.0000000 1.0000000 1809
## [3] 0.1244116 1.0000000 1.0000000 1903
## [4] 0.1642913 1.0000000 1.0000000 2513
## [5] 0.2800000 0.01797856 1.7042897 77
## [6] 0.2655738 0.01993985 3.7893810 81
## [7] 0.3737705 0.01993985 3.0043055 114
## [8] 0.2688525 0.01993985 1.6364374 82
## [9] 0.2415902 0.02137814 1.9418623 79
## [10] 0.2446483 0.02137814 1.4891129 80
## [11] 0.2905199 0.02137814 2.3351508 95
## [12] 0.2721713 0.02137814 1.6566381 89
## [13] 0.2767123 0.02386245 1.6842785 101

```

```

## [14] 0.1974359 0.02549686 2.2011512 77
## [15] 0.1680328 0.03190377 1.4986761 82
## [16] 0.2890995 0.02758891 2.3237343 122
## [17] 0.2298578 0.02758891 1.3990868 97
## [18] 0.1839323 0.03092312 1.1195500 87
## [19] 0.1308901 0.04994770 1.1674019 100
## [20] 0.1511628 0.03373431 2.8405234 78
## [21] 0.2577519 0.03373431 3.6777739 133
## [22] 0.1240672 0.07008368 3.6777739 133
## [23] 0.2461240 0.03373431 1.9783044 127
## [24] 0.2422481 0.03373431 1.4745031 125
## [25] 0.2232824 0.03425732 2.4893063 117
## [26] 0.2061069 0.03425732 1.6566530 108
## [27] 0.3683206 0.03425732 2.2418751 193
## [28] 0.1458333 0.03765690 1.2330938 84
## [29] 0.1743119 0.03563023 1.9433493 95
## [30] 0.2311927 0.03563023 1.8582885 126
## [31] 0.4036697 0.03563023 2.4570363 220
## [32] 0.1816578 0.03706851 2.5920135 103
## [33] 0.2504409 0.03706851 2.0130028 142
## [34] 0.2345679 0.03706851 1.4277559 133
## [35] 0.1706897 0.03791841 2.8256158 99
## [36] 0.1071429 0.06040795 2.8256158 99
## [37] 0.1465517 0.03791841 2.1721465 85
## [38] 0.1637931 0.03791841 1.3849526 95
## [39] 0.1862069 0.03791841 1.4967003 108
## [40] 0.2172414 0.03791841 1.3222937 126
## [41] 0.1338384 0.05177824 1.8833412 106
## [42] 0.1603535 0.05177824 1.4301852 127
## [43] 0.1206897 0.04171025 2.1097931 77
## [44] 0.1504702 0.04171025 1.2723010 96
## [45] 0.1614420 0.04171025 0.9826570 103
## [46] 0.2003205 0.04079498 1.6938102 125
## [47] 0.2003205 0.04079498 1.2193007 125
## [48] 0.1237693 0.04648274 1.7416521 88
## [49] 0.2011252 0.04648274 1.7938255 143
## [50] 0.1011352 0.06334990 0.9020198 98
## [51] 0.2113475 0.04609048 2.3562475 149
## [52] 0.1086006 0.08969665 2.3562475 149
## [53] 0.1163121 0.04609048 0.9834766 82
## [54] 0.1801418 0.04609048 1.4479504 127
## [55] 0.2482270 0.04609048 1.5108951 175
## [56] 0.1680108 0.04864017 3.1571161 125
## [57] 0.1535627 0.05321653 3.1571161 125
## [58] 0.1276882 0.04864017 2.1137644 95
## [59] 0.1028139 0.06040795 2.1137644 95
## [60] 0.2607527 0.04864017 3.8647995 194
## [61] 0.1879845 0.06746862 3.8647995 194
## [62] 0.1666667 0.04864017 2.3781095 124
## [63] 0.1156716 0.07008368 2.3781095 124
## [64] 0.2244624 0.04864017 1.8041915 167
## [65] 0.2580645 0.04864017 1.5707739 192
## [66] 0.1268571 0.05720450 1.1314326 111
## [67] 0.1782857 0.05720450 1.5074949 156

```

```

## [68] 0.1645714 0.05720450 1.0017050 144
## [69] 0.1302211 0.05321653 2.1556952 106
## [70] 0.1147186 0.06040795 2.1556952 106
## [71] 0.2346437 0.05321653 3.4778203 191
## [72] 0.1850775 0.06746862 3.4778203 191
## [73] 0.1633907 0.05321653 2.3313653 133
## [74] 0.1240672 0.07008368 2.3313653 133
## [75] 0.1265356 0.05321653 1.4107062 103
## [76] 0.2407862 0.05321653 1.9354001 196
## [77] 0.1029953 0.12441161 1.9354001 196
## [78] 0.2407862 0.05321653 1.4656054 196
## [79] 0.1352814 0.06040795 2.0051008 125
## [80] 0.1211240 0.06746862 2.0051008 125
## [81] 0.1212121 0.06040795 1.7295341 112
## [82] 0.1044776 0.07008368 1.7295341 112
## [83] 0.1785714 0.06040795 1.5099108 165
## [84] 0.2088745 0.06040795 1.6788984 193
## [85] 0.1014188 0.12441161 1.6788984 193
## [86] 0.2077922 0.06040795 1.2647790 192
## [87] 0.2060718 0.07106433 1.8379438 224
## [88] 0.1306122 0.11212082 1.8379438 224
## [89] 0.1205152 0.07106433 1.0190161 131
## [90] 0.1627907 0.06746862 2.3228046 168
## [91] 0.1567164 0.07008368 2.3228046 168
## [92] 0.1211240 0.06746862 1.3503740 125
## [93] 0.2296512 0.06746862 1.8458982 237
## [94] 0.1245402 0.12441161 1.8458982 237
## [95] 0.2713178 0.06746862 1.6514435 280
## [96] 0.1114206 0.16429132 1.6514435 280
## [97] 0.3619403 0.07008368 2.9092164 388
## [98] 0.2038886 0.12441161 2.9092164 388
## [99] 0.3227612 0.07008368 1.9645663 346
## [100] 0.1376840 0.16429132 1.9645663 346
## [101] 0.1326531 0.08969665 1.1216480 182
## [102] 0.1006081 0.11826621 1.1216480 182
## [103] 0.1771137 0.08969665 1.4236107 243
## [104] 0.1276931 0.12441161 1.4236107 243
## [105] 0.2704082 0.08969665 1.6459066 371
## [106] 0.1476323 0.16429132 1.6459066 371
## [107] 0.1271137 0.11212082 1.0748099 218
## [108] 0.1205086 0.11826621 1.0748099 218
## [109] 0.1547816 0.11826621 0.9421170 280
## [110] 0.1114206 0.16429132 0.9421170 280
## [111] 0.3284288 0.12441161 1.9990636 625
## [112] 0.2487067 0.16429132 1.9990636 625
## [113] 0.3221649 0.02536611 1.9609371 125
## [114] 0.3612717 0.02262029 2.9038421 125
## [115] 0.2000000 0.04086036 2.8537313 125
## [116] 0.3991770 0.01588651 2.4296899 97
## [117] 0.2614555 0.02425471 2.1015364 97
## [118] 0.1552000 0.04086036 1.7302764 97

##      lhs                  rhs          support    confidence coverage
## [1]  {onions}        => {root vegetables} 0.005295502 0.2655738  0.01993985

```

```

## [2] {onions}          => {other vegetables} 0.007452929 0.3737705 0.01993985
## [3] {beef}            => {root vegetables} 0.008695084 0.2577519 0.03373431
## [4] {root vegetables} => {beef}           0.008695084 0.1240672 0.07008368
## [5] {pip fruit}       => {citrus fruit}   0.008172071 0.1680108 0.04864017
## [6] {citrus fruit}    => {pip fruit}     0.008172071 0.1535627 0.05321653
## [7] {pip fruit}       => {tropical fruit} 0.012683054 0.2607527 0.04864017
## [8] {tropical fruit} => {pip fruit}     0.012683054 0.1879845 0.06746862
## [9] {citrus fruit}    => {tropical fruit} 0.012486925 0.2346437 0.05321653
## [10] {tropical fruit}=> {citrus fruit}   0.012486925 0.1850775 0.06746862
##      lift   count
## [1] 3.789381 81
## [2] 3.004306 114
## [3] 3.677774 133
## [4] 3.677774 133
## [5] 3.157116 125
## [6] 3.157116 125
## [7] 3.864800 194
## [8] 3.864800 194
## [9] 3.477820 191
## [10] 3.477820 191

##      lhs                      rhs          support
## [1] {onions}          => {other vegetables} 0.007452929
## [2] {curd}            => {whole milk}      0.012617678
## [3] {butter}          => {whole milk}      0.014382845
## [4] {root vegetables} => {other vegetables} 0.025366109
## [5] {root vegetables} => {whole milk}      0.022620293
## [6] {other vegetables}=> {whole milk}      0.040860356
## [7] {other vegetables, root vegetables}=> {whole milk} 0.008172071
## [8] {root vegetables, whole milk}        => {other vegetables} 0.008172071
## [9] {other vegetables, yogurt}         => {whole milk}      0.006341527
##      confidence coverage   lift   count
## [1] 0.3737705 0.01993985 3.004306 114
## [2] 0.3683206 0.03425732 2.241875 193
## [3] 0.4036697 0.03563023 2.457036 220
## [4] 0.3619403 0.07008368 2.909216 388
## [5] 0.3227612 0.07008368 1.964566 346
## [6] 0.3284288 0.12441161 1.999064 625
## [7] 0.3221649 0.02536611 1.960937 125
## [8] 0.3612717 0.02262029 2.903842 125
## [9] 0.3991770 0.01588651 2.429690 97

##      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

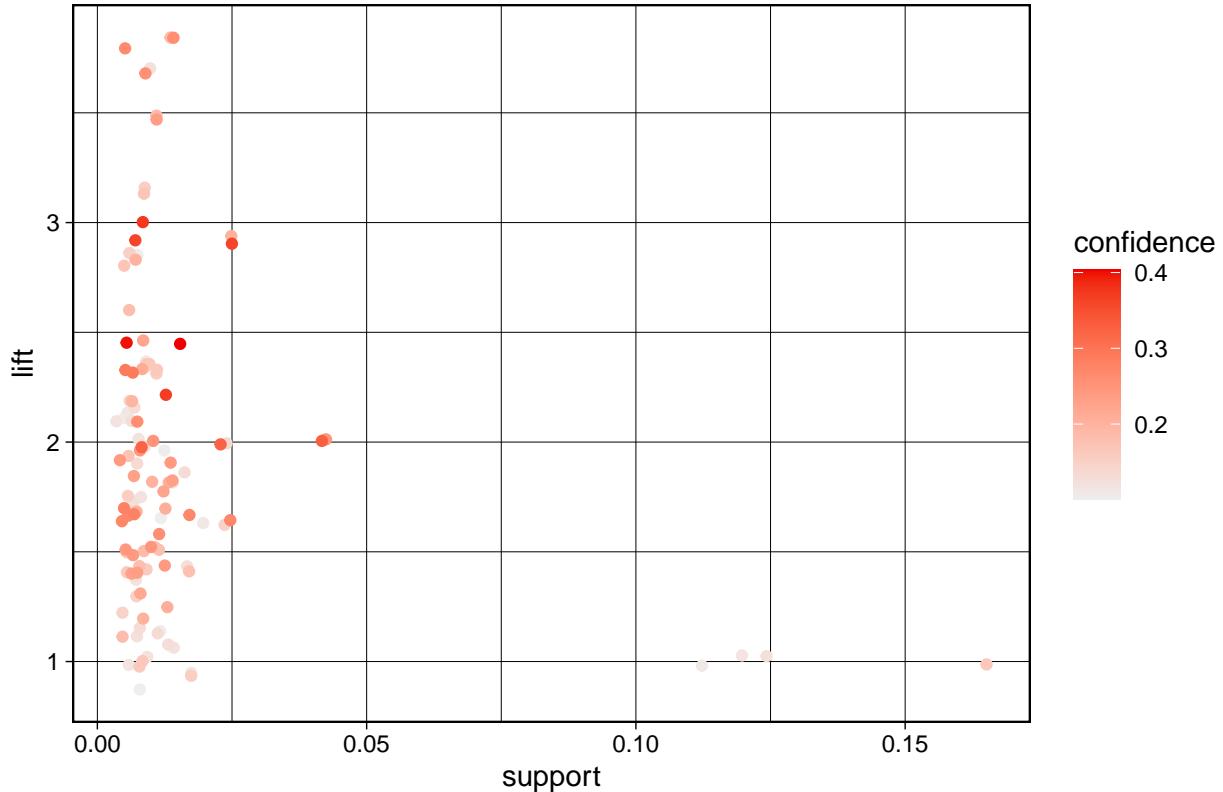
```

```

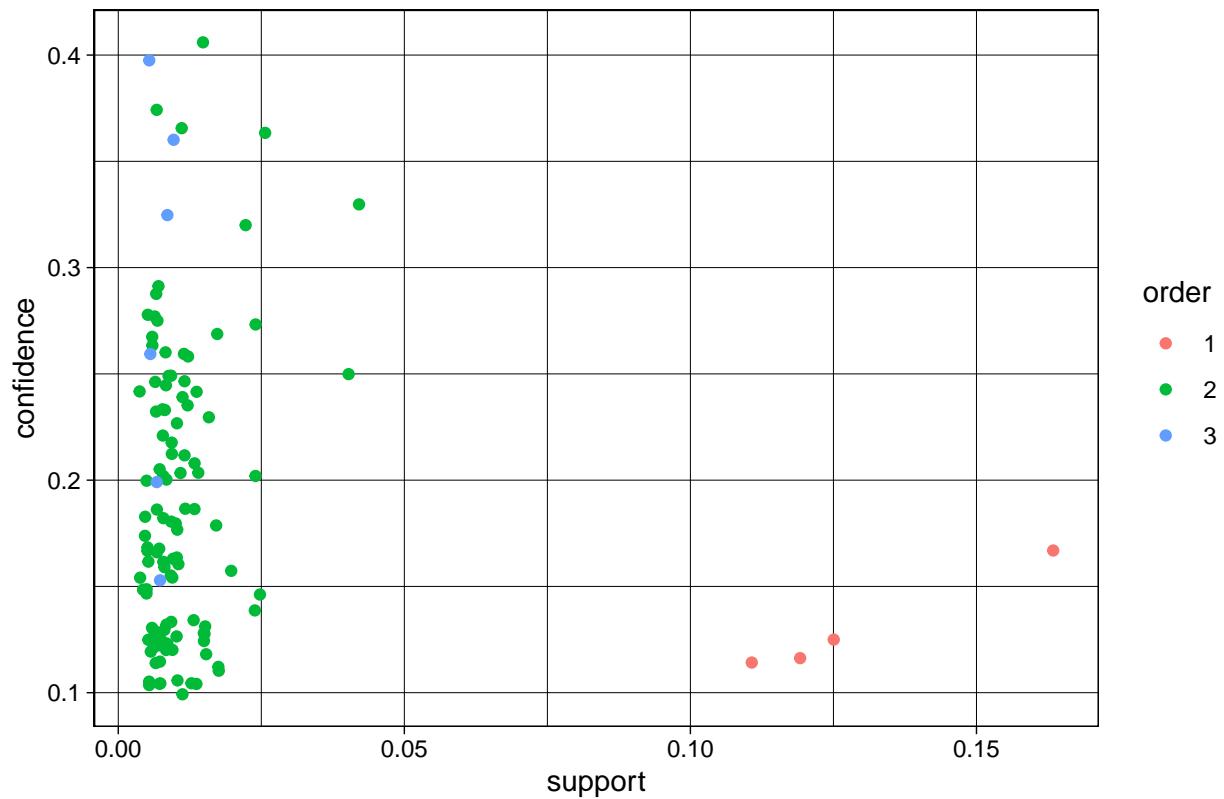
## [12] {citrus fruit}          => {tropical fruit}  0.012486925 0.2346437
## [13] {root vegetables}      => {other vegetables} 0.025366109 0.3619403
## [14] {other vegetables}     => {root vegetables}  0.025366109 0.2038886
## [15] {root vegetables, whole milk} => {other vegetables} 0.008172071 0.3612717
## [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

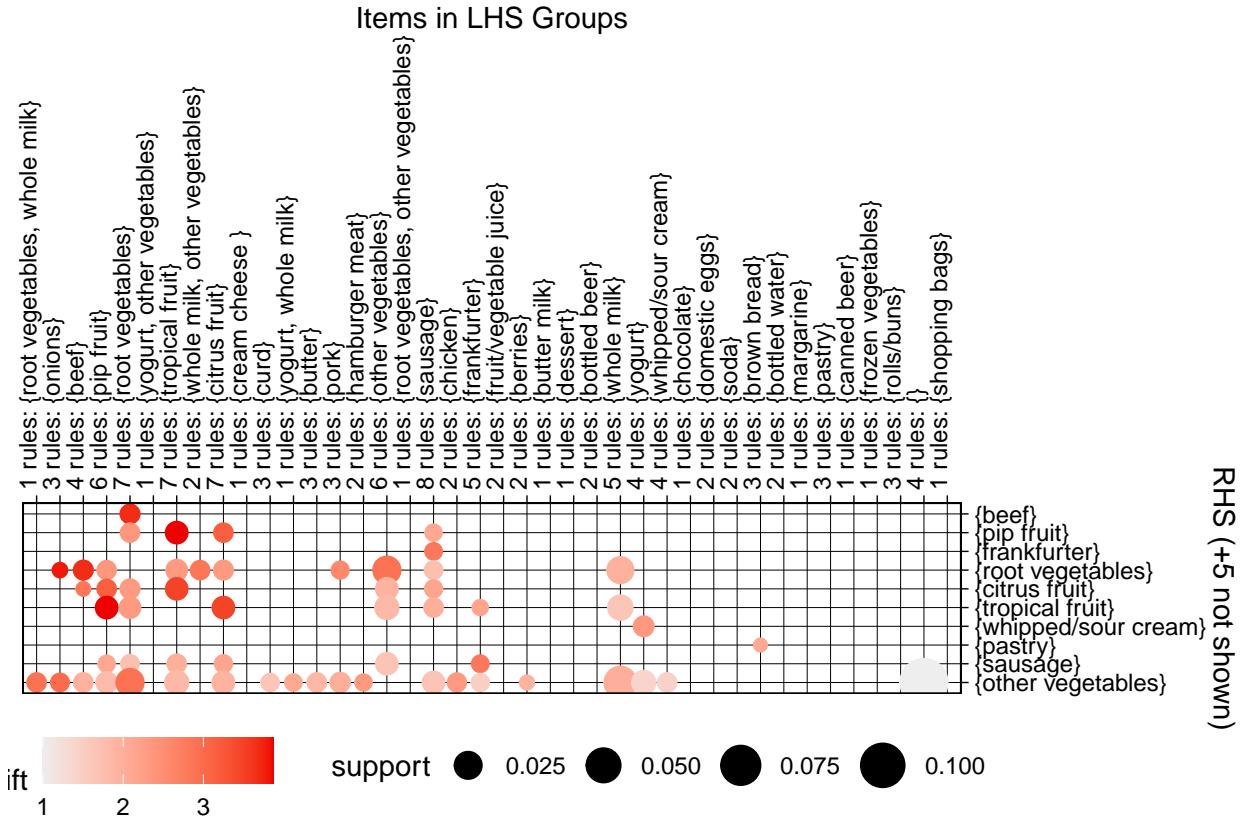
```

Scatter plot for 118 rules



Scatter plot for 118 rules





Comments:

First, we had to do some data-wrangling in order to have the data in the correct format to be read. We then plotted the top 10 items in the dataset. After noticing we had a sizable number of each item, we proceeded to set some thresholds for our association rules.

Support: We picked a support of 0.005, because we only wanted to consider itemsets that occurred at least 50 times out of a total of 10,000 transactions in order to be confident that we have enough information to draw a conclusion from our rules.

Confidence: We chose to evaluate items with a confidence level threshold of 0.3. Although this is not extremely high, since confidence can be misleading due to association between items, we chose to have a high lift to make up for this.

Lift: We chose to have a lift threshold of 3, since a high lift means that the rise in probability of having item "Y" in the cart with the knowledge of item "X" being present over the probability of having item "Y" on the cart with no knowledge of item "X" being in the cart. Thus, having a lift of 3 guarantees that there is a strong chance that an individual will purchase item Y if they are already purchasing item X."

Conclusions: Our association rules make sense, as those who buy meat are likely trying to cook a meal, which requires produce. In addition, people who buy drinks are more likely to purchase other types of drinks available in a grocery store. The importance of each association rule is determined by as high of a lift (the darker the red) and the size of each bubble (the support) shown above. Thus, based on our association rules, we believe that organizing a grocery store would have the best layout by organizing the whole milk next to the vegetables, all non-perishable drinks, such as soda, beer, and water, next to each other (in the same aisle), and with all meat products close to the produce, particularly beef next to citrus fruit.