Predictive Modeling P2

Eguche Emmanuella & Mervyn Jonathan Giritharn

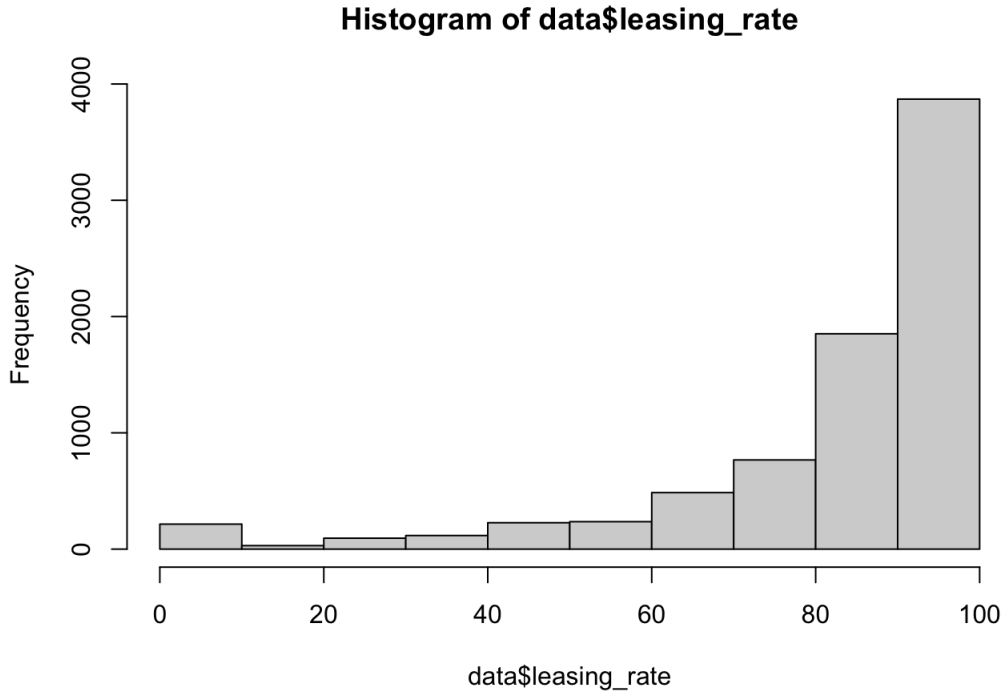
8/18/2020

Green Buildings

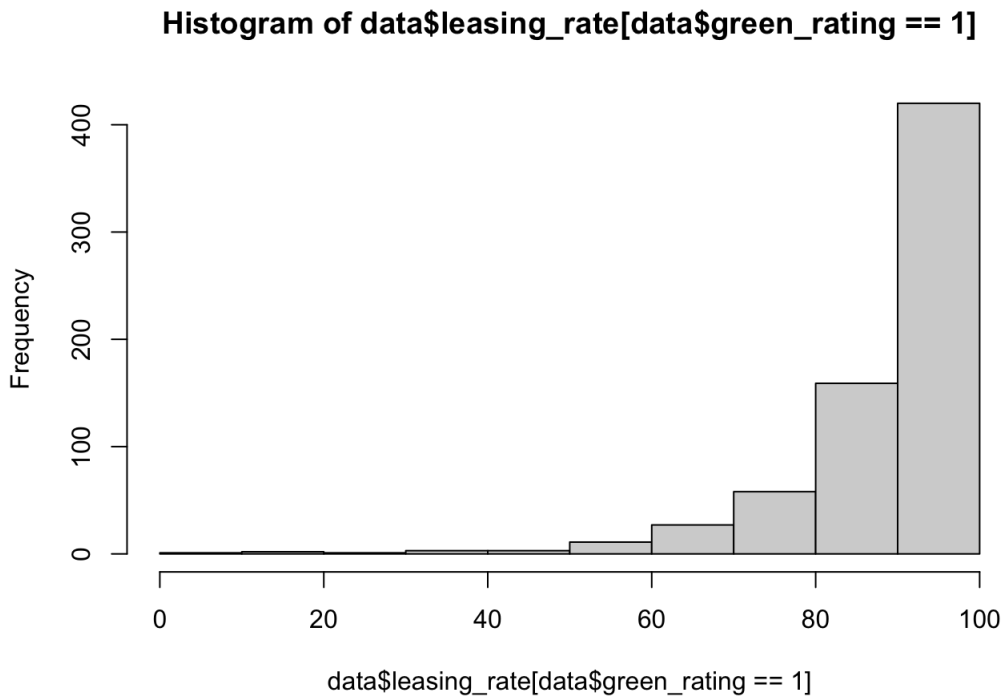
- Figure 1: Summary of Data Set

```
##  CS_PropertyID        cluster            size            empl_gr
##  Min.   :      1   Min.   :   1.0   Min.   :   1624   Min.   :-24.950
##  1st Qu.: 157452   1st Qu.: 272.0   1st Qu.:  50891   1st Qu.:  1.740
##  Median : 313253   Median : 476.0   Median : 128838   Median :  1.970
##  Mean   : 453003   Mean   : 588.6   Mean   : 234638   Mean   :  3.207
##  3rd Qu.: 441188   3rd Qu.:1044.0   3rd Qu.: 294212   3rd Qu.:  2.380
##  Max.   :6208103   Max.   :1230.0   Max.   :3781045   Max.   : 67.780
##                                                       NA's   :74
##       Rent         leasing_rate       stories            age
##  Min.   :  2.98   Min.   :  0.00   Min.   :  1.00   Min.   :  0.00
##  1st Qu.: 19.50   1st Qu.: 77.85   1st Qu.:  4.00   1st Qu.: 23.00
##  Median : 25.16   Median : 89.53   Median : 10.00   Median : 34.00
##  Mean   : 28.42   Mean   : 82.61   Mean   : 13.58   Mean   : 47.24
##  3rd Qu.: 34.18   3rd Qu.: 96.44   3rd Qu.: 19.00   3rd Qu.: 79.00
##  Max.   :250.00   Max.   :100.00   Max.   :110.00   Max.   :187.00
##
##    renovated         class_a          class_b           LEED
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.000000
##  Mean   :0.3795   Mean   :0.3999   Mean   :0.4595   Mean   :0.006841
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.000000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000000
##
##    Energystar       green_rating          net            amenities
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :1.0000
##  Mean   :0.08082   Mean   :0.08677   Mean   :0.03471   Mean   :0.5266
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##
##   cd_total_07     hd_total07    total_dd_07   Precipitation
##  Min.   :  39   Min.   :   0   Min.   :2103   Min.   :10.46
##  1st Qu.: 684   1st Qu.:1419   1st Qu.:2869   1st Qu.:22.71
##  Median : 966   Median :2739   Median :4979   Median :23.16
##  Mean   :1229   Mean   :3432   Mean   :4661   Mean   :31.08
##  3rd Qu.:1620   3rd Qu.:4796   3rd Qu.:6413   3rd Qu.:43.89
##  Max.   :5240   Max.   :7200   Max.   :8244   Max.   :58.02
##
##    Gas_Costs        Electricity_Costs  cluster_rent
##  Min.   :0.009487   Min.   :0.01780   Min.   : 9.00
##  1st Qu.:0.010296   1st Qu.:0.02330   1st Qu.:20.00
##  Median :0.010296   Median :0.03274   Median :25.14
##  Mean   :0.011336   Mean   :0.03096   Mean   :27.50
##  3rd Qu.:0.011816   3rd Qu.:0.03781   3rd Qu.:34.00
##  Max.   :0.028914   Max.   :0.06280   Max.   :71.44
##
```

- Before deleting any data points from the data set, a histogram was made for some variables to see the distribution shape of the data.
- Figure 2: Histogram of Leasing_rate



- The leasing_rate histogram is skewed to the left, and there is a rise in number of leasing rates between 0% and 10%, which is located at the end of the skewed tail.
- Out of 7894 locations, less than 10% of the total number of buildings are green buildings, so a histogram of green buildings' leasing_rates should be considered.
- Figure 3: Histogram of Leasing_rate Conditional on green_rate



- Comparing Figure 2 to Figure 3, the data is still skewed to the left, but there is no rise in the number of leasing rates between 0% and 10% for green buildings.
- Figure 4: Summary Statistics for Green Buildings

```
##  CS_PropertyID        cluster            size            empl_gr
##  Min.   :   2883   Min.   :   1.0   Min.   :  10560   Min.   :-24.950
##  1st Qu.: 246763   1st Qu.: 294.0   1st Qu.: 120000   1st Qu.:  1.770
##  Median : 280498   Median : 489.0   Median : 241150   Median :  2.380
##  Mean   : 399034   Mean   : 622.9   Mean   : 325781   Mean   :  3.506
##  3rd Qu.: 409817   3rd Qu.:1047.0   3rd Qu.: 417446   3rd Qu.:  2.970
##  Max.   :6174162   Max.   :1230.0   Max.   :1721242   Max.   : 67.780
##                                                       NA's   :6
##       Rent         leasing_rate       stories           age
##  Min.   :  8.87   Min.   :  0.00   Min.   : 1.00   Min.   :  0.00
##  1st Qu.: 21.50   1st Qu.: 85.40   1st Qu.: 5.00   1st Qu.: 18.00
##  Median : 27.60   Median : 92.92   Median :11.00   Median : 22.00
##  Mean   : 30.02   Mean   : 89.28   Mean   :15.33   Mean   : 23.85
##  3rd Qu.: 35.50   3rd Qu.: 97.70   3rd Qu.:21.00   3rd Qu.: 26.00
##  Max.   :138.07   Max.   :100.00   Max.   :76.00   Max.   :116.00
##
##    renovated         class_a          class_b            LEED
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.0000   Median :1.0000   Median :0.0000   Median :0.00000
##  Mean   :0.2131   Mean   :0.7971   Mean   :0.1927   Mean   :0.07883
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##
##    Energystar      green_rating      net            amenities
##  Min.   :0.0000   Min.   :1     Min.   :0.00000   Min.   :0.000
##  1st Qu.:1.0000   1st Qu.:1     1st Qu.:0.00000   1st Qu.:0.000
##  Median :1.0000   Median :1     Median :0.00000   Median :1.000
##  Mean   :0.9314   Mean   :1     Mean   :0.05693   Mean   :0.727
##  3rd Qu.:1.0000   3rd Qu.:1     3rd Qu.:0.00000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1     Max.   :1.00000   Max.   :1.000
##
##   cd_total_07     hd_total07    total_dd_07   Precipitation     Gas_Costs
##  Min.   : 130   Min.   :   0   Min.   :2103   Min.   :10.46   Min.   :0.00950
##  1st Qu.: 684   1st Qu.:1419   1st Qu.:2103   1st Qu.:22.71   1st Qu.:0.01030
##  Median : 921   Median :1670   Median :4416   Median :22.71   Median :0.01030
##  Mean   :1426   Mean   :2794   Mean   :4219   Mean   :29.19   Mean   :0.01109
##  3rd Qu.:1813   3rd Qu.:4347   3rd Qu.:5720   3rd Qu.:40.70   3rd Qu.:0.01180
##  Max.   :5240   Max.   :7200   Max.   :8244   Max.   :58.02   Max.   :0.02890
##
##  Electricity_Costs  cluster_rent
##  Min.   :0.01780   Min.   : 9.00
##  1st Qu.:0.02350   1st Qu.:19.80
##  Median :0.03410   Median :25.38
##  Mean   :0.03158   Mean   :26.89
##  3rd Qu.:0.03780   3rd Qu.:32.30
##  Max.   :0.06280   Max.   :71.44
##
```

- Figure 5: Summary Statistics for Non Green Buildings

```
##  CS_PropertyID        cluster            size            empl_gr
##  Min.   :      1   Min.   :   1.0   Min.   :   1624   Min.   :-24.950
##  1st Qu.: 157087   1st Qu.: 269.0   1st Qu.:  46043   1st Qu.:  1.740
##  Median : 313297   Median : 474.0   Median : 118696   Median :  1.970
##  Mean   : 458131   Mean   : 585.4   Mean   : 225977   Mean   :  3.178
##  3rd Qu.: 455765   3rd Qu.:1044.0   3rd Qu.: 279411   3rd Qu.:  2.380
##  Max.   :6208103   Max.   :1230.0   Max.   :3781045   Max.   : 67.780
##                                                       NA's   :68
##       Rent         leasing_rate       stories            age
##  Min.   :  2.98   Min.   :  0.00   Min.   :  1.00   Min.   :  0.00
##  1st Qu.: 19.18   1st Qu.: 77.05   1st Qu.:  4.00   1st Qu.: 24.00
##  Median : 25.00   Median : 89.17   Median : 10.00   Median : 37.00
##  Mean   : 28.27   Mean   : 81.97   Mean   : 13.42   Mean   : 49.47
##  3rd Qu.: 34.00   3rd Qu.: 96.28   3rd Qu.: 19.00   3rd Qu.: 80.00
##  Max.   :250.00   Max.   :100.00   Max.   :110.00   Max.   :187.00
##
##    renovated         class_a          class_b            LEED     Energystar
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0   Min.   :0
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0   1st Qu.:0
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0   Median :0
##  Mean   :0.3953   Mean   :0.3622   Mean   :0.4848   Mean   :0   Mean   :0
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0   3rd Qu.:0
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :0   Max.   :0
##
##   green_rating      net           amenities       cd_total_07     hd_total07
##  Min.   :0     Min.   :0.0000   Min.   :0.0000   Min.   :  39   Min.   :   0
##  1st Qu.:0     1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 684   1st Qu.:1419
##  Median :0     Median :0.0000   Median :1.0000   Median : 966   Median :2739
##  Mean   :0     Mean   :0.0326   Mean   :0.5076   Mean   :1211   Mean   :3493
##  3rd Qu.:0     3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1620   3rd Qu.:5042
##  Max.   :0     Max.   :1.0000   Max.   :1.0000   Max.   :5240   Max.   :7200
##
##   total_dd_07   Precipitation     Gas_Costs        Electricity_Costs
##  Min.   :2103   Min.   :10.46   Min.   :0.009487   Min.   :0.01782
##  1st Qu.:2869   1st Qu.:22.71   1st Qu.:0.010296   1st Qu.:0.02330
##  Median :4979   Median :23.16   Median :0.010296   Median :0.03274
##  Mean   :4703   Mean   :31.26   Mean   :0.011359   Mean   :0.03090
##  3rd Qu.:6558   3rd Qu.:43.89   3rd Qu.:0.011816   3rd Qu.:0.03781
##  Max.   :8244   Max.   :58.02   Max.   :0.028914   Max.   :0.06278
##
##   cluster_rent
##  Min.   : 9.00
##  1st Qu.:20.17
##  Median :25.13
##  Mean   :27.55
##  3rd Qu.:34.18
##  Max.   :71.44
##
```

- Outliers can make the mean unreliable. However, the mean rent for green buildings is 30.02, and the median rent for green buildings is 27.6. The mean rent for non green buildings is 28.27, and the median rent for non green buildings is 25. Because the mean and median are not the same for green and non green buildings, the mean should be considered in the analysis, especially if the distribution of data is skewed to the left.
- The worker suggested that a 250,000 square foot green building would generate more revenue than a non green building because the median rent is higher. This assumption is problematic. Green buildings have different costs than non green buildings, which would affect profitability.
- The worker assumes rent prices won't change over time. The age and class of a building effects revenue.

Question 2: ABIA

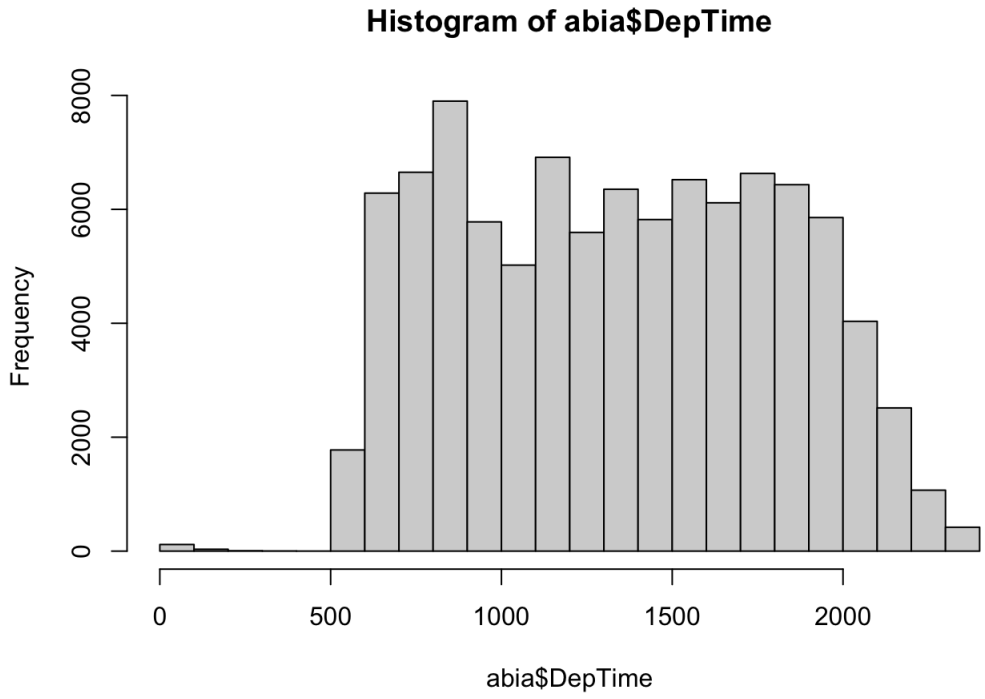
- Figure 1: Summary Statistics for Each Variable

```
##       Year          Month         DayofMonth      DayOfWeek        DepTime
##  Min.   :2008   Min.   : 1.00   Min.   : 1.00   Min.   :1.000   Min.   :   1
##  1st Qu.:2008   1st Qu.: 3.00   1st Qu.: 8.00   1st Qu.:2.000   1st Qu.: 917
##  Median :2008   Median : 6.00   Median :16.00   Median :4.000   Median :1329
##  Mean   :2008   Mean   : 6.29   Mean   :15.73   Mean   :3.902   Mean   :1329
##  3rd Qu.:2008   3rd Qu.: 9.00   3rd Qu.:23.00   3rd Qu.:6.000   3rd Qu.:1728
##  Max.   :2008   Max.   :12.00   Max.   :31.00   Max.   :7.000   Max.   :2400
##                                                                 NA's   :1413
##    CRSDepTime      ArrTime       CRSArrTime   UniqueCarrier        FlightNum
##  Min.   :  55   Min.   :   1   Min.   :   5   Length:99260       Min.   :   1
##  1st Qu.: 915   1st Qu.:1107   1st Qu.:1115   Class :character   1st Qu.: 640
##  Median :1320   Median :1531   Median :1535   Mode  :character   Median :1465
##  Mean   :1320   Mean   :1487   Mean   :1505                      Mean   :1917
##  3rd Qu.:1720   3rd Qu.:1903   3rd Qu.:1902                      3rd Qu.:2653
##  Max.   :2346   Max.   :2400   Max.   :2400                      Max.   :9741
##                 NA's   :1567
##    TailNum          ActualElapsedTime CRSElapsedTime     AirTime
##  Length:99260       Min.   : 22.0     Min.   : 17.0   Min.   :  3.00
##  Class :character   1st Qu.: 57.0     1st Qu.: 58.0   1st Qu.: 38.00
##  Mode  :character   Median :125.0     Median :130.0   Median :105.00
##                     Mean   :120.2     Mean   :122.1   Mean   : 99.81
##                     3rd Qu.:164.0     3rd Qu.:165.0   3rd Qu.:142.00
##                     Max.   :506.0     Max.   :320.0   Max.   :402.00
##                     NA's   :1601      NA's   :11      NA's   :1601
##     ArrDelay           DepDelay          Origin              Dest
##  Min.   :-129.000   Min.   :-42.000   Length:99260       Length:99260
##  1st Qu.:  -9.000   1st Qu.: -4.000   Class :character   Class :character
##  Median :  -2.000   Median :  0.000   Mode  :character   Mode  :character
##  Mean   :   7.065   Mean   :  9.171
##  3rd Qu.:  10.000   3rd Qu.:  8.000
##  Max.   : 948.000   Max.   :875.000
##  NA's   :1601       NA's   :1413
##     Distance        TaxiIn           TaxiOut         Cancelled
##  Min.   :  66   Min.   :  0.000   Min.   :  1.00   Min.   :0.00000
##  1st Qu.: 190   1st Qu.:  4.000   1st Qu.:  9.00   1st Qu.:0.00000
##  Median : 775   Median :  5.000   Median : 12.00   Median :0.00000
##  Mean   : 705   Mean   :  6.413   Mean   : 13.96   Mean   :0.01431
##  3rd Qu.:1085   3rd Qu.:  7.000   3rd Qu.: 16.00   3rd Qu.:0.00000
##  Max.   :1770   Max.   :143.000   Max.   :305.00   Max.   :1.00000
##                 NA's   :1567      NA's   :1419
##  CancellationCode      Diverted         CarrierDelay     WeatherDelay
##  Length:99260       Min.   :0.000000   Min.   :  0.00   Min.   :  0.00
##  Class :character   1st Qu.:0.000000   1st Qu.:  0.00   1st Qu.:  0.00
##  Mode  :character   Median :0.000000   Median :  0.00   Median :  0.00
##                     Mean   :0.001824   Mean   : 15.39   Mean   :  2.24
##                     3rd Qu.:0.000000   3rd Qu.: 16.00   3rd Qu.:  0.00
##                     Max.   :1.000000   Max.   :875.00   Max.   :412.00
##                                        NA's   :79513    NA's   :79513
##     NASDelay      SecurityDelay    LateAircraftDelay
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00
##  Median :  2.00   Median :  0.00   Median :  6.00
##  Mean   : 12.47   Mean   :  0.07   Mean   : 22.97
##  3rd Qu.: 16.00   3rd Qu.:  0.00   3rd Qu.: 30.00
##  Max.   :367.00   Max.   :199.00   Max.   :458.00
##  NA's   :79513    NA's   :79513    NA's   :79513
```

- The output above shows the minimum & maximum values, the median, and mean for each variable.
- Figure 2: Logistic Regression Dependent-DepTime & Independent-Month, DayOfWeek, ArrTime, & Distance

```
##
## Call:
## glm(formula = abia$DepTime ~ abia$Month + abia$DayOfWeek + abia$ArrTime +
##     abia$Distance)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -550.94  -146.86   -35.55    88.78  2135.22
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    342.091006   4.013364  85.238  < 2e-16 ***
## abia$Month      -1.095293   0.275835  -3.971 7.17e-05 ***
## abia$DayOfWeek   1.203112   0.468969   2.565   0.0103 *
## abia$ArrTime     0.726061   0.001885 385.122  < 2e-16 ***
## abia$Distance   -0.128721   0.001997 -64.457  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 84775.91)
##
##     Null deviance: 2.0941e+10  on 97692  degrees of freedom
## Residual deviance: 8.2816e+09  on 97688  degrees of freedom
##   (1567 observations deleted due to missingness)
## AIC: 1385845
##
## Number of Fisher Scoring iterations: 2
```

- P-values less than .05 are statistically significant to the logistic regresison model. All independent variables are statistically significant to the model.
- Figure 3: Histogram of DepTime



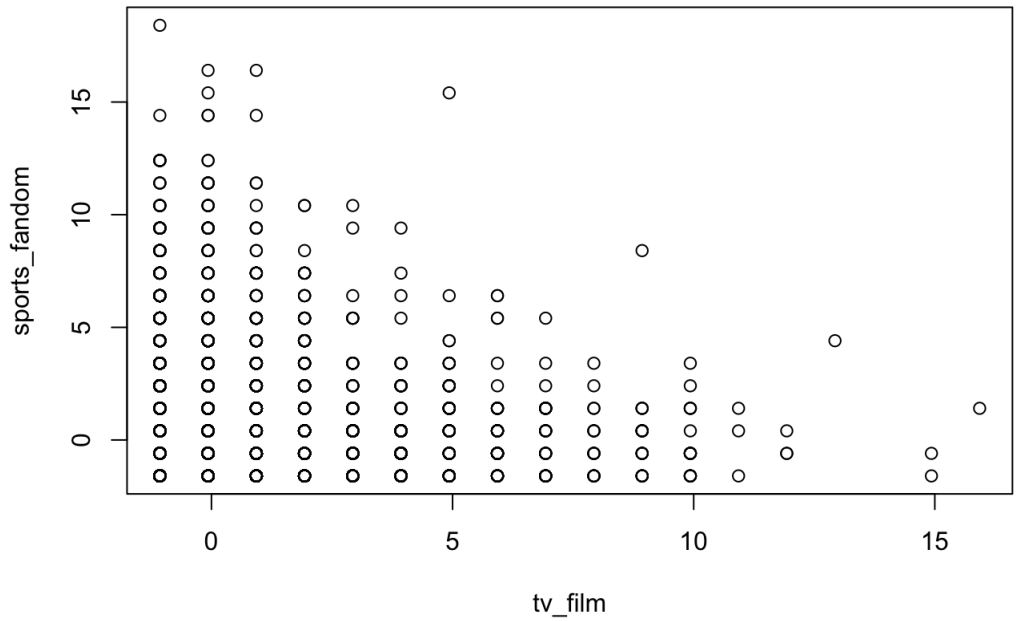
* There aren't very many departures between 0 and 500. It is difficult to assume normality.

##Q4

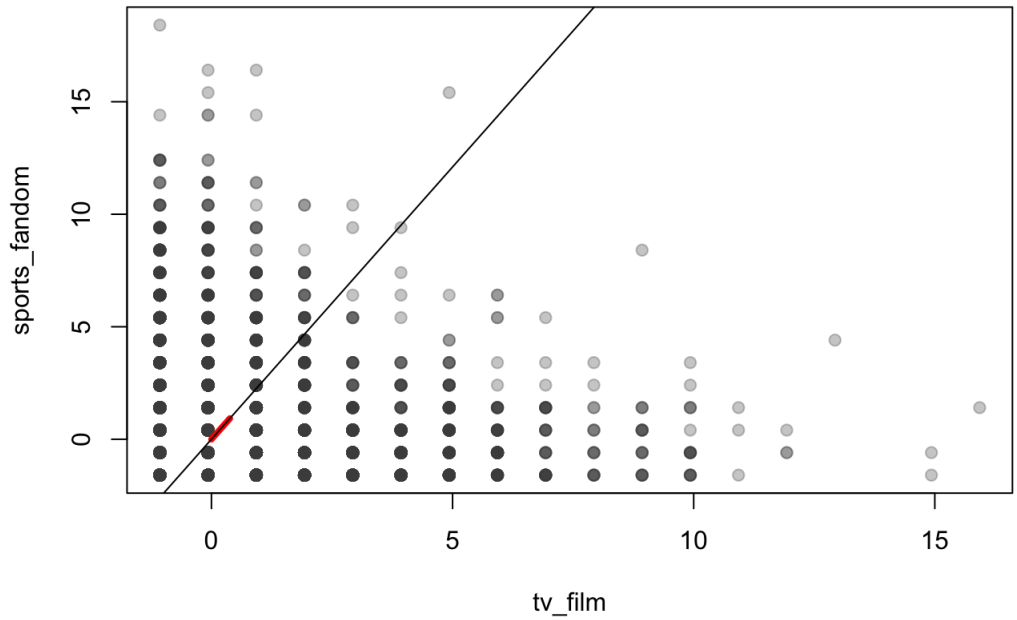
Market Segmentation

I ran a PCA of TV Film and Sports Fandom

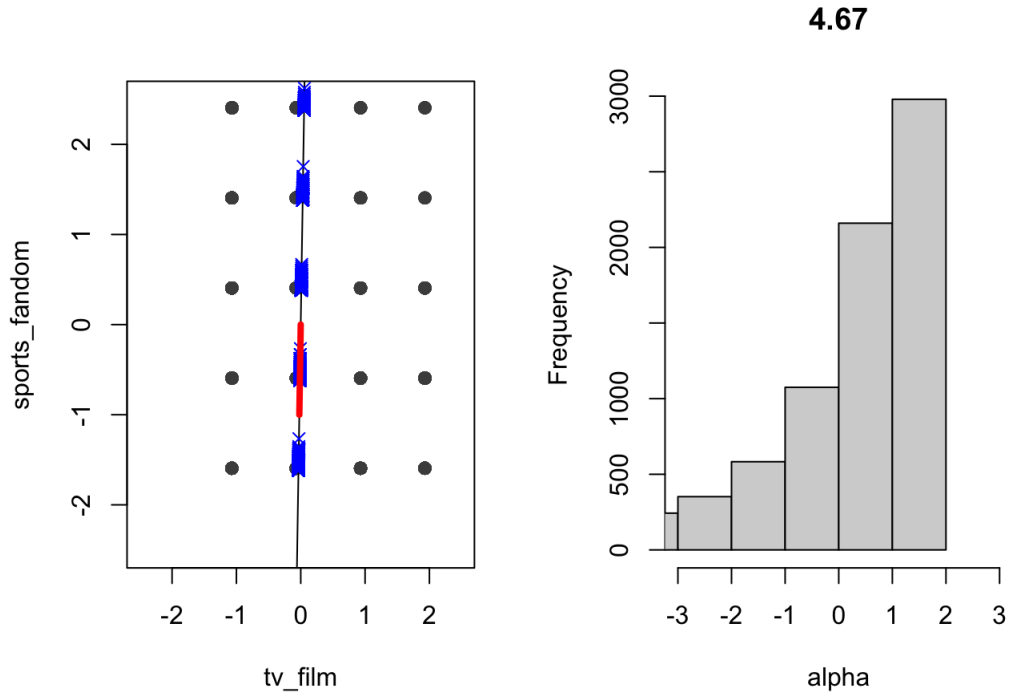
```
sm = read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/social_marketing.csv')

Z = sm[,c(7,8)]

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

plot(Z)
```




```
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))

plot(Z, pch=19, col=rgb(0.3,0.3,0.3,0.3))
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)


slope = v_try[2]/v_try[1]
abline(0, slope)
```



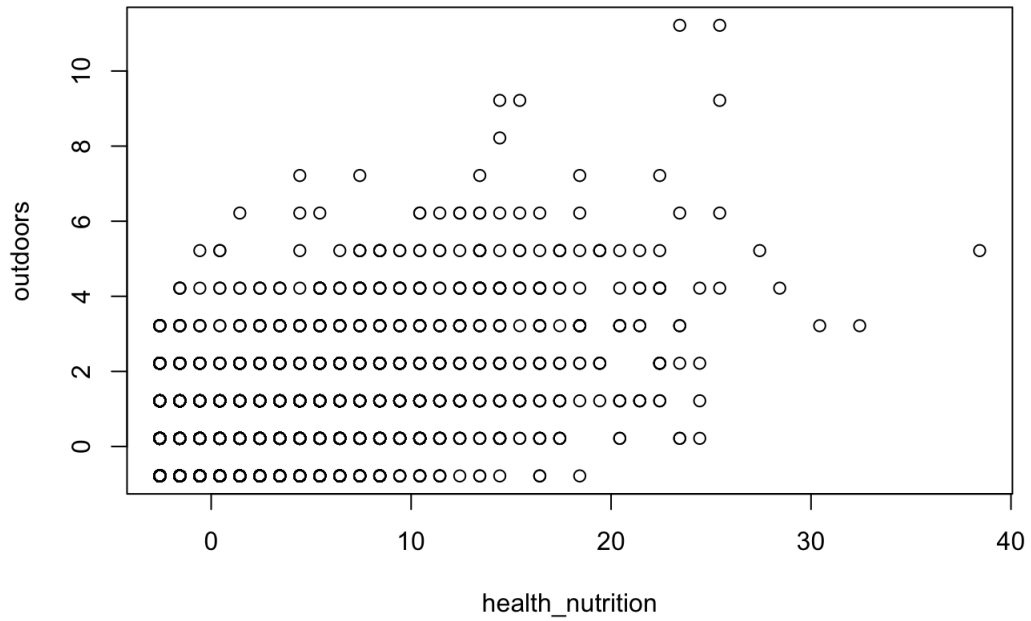

```
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))
par(mfrow=c(1,2))
plot(Z, pch=19, col=rgb(0.3,0.3,0.3,0.3),
     xlim=c(-2.5,2.5), ylim=c(-2.5,2.5))
slope = v_try[2]/v_try[1]
abline(0, slope)


alpha = Z %*% v_try
z_hat = alpha %*% v_try
points(z_hat, col='blue', pch=4)
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)


hist(alpha, 25, xlim=c(-3,3), main=round(var(alpha), 2))
```



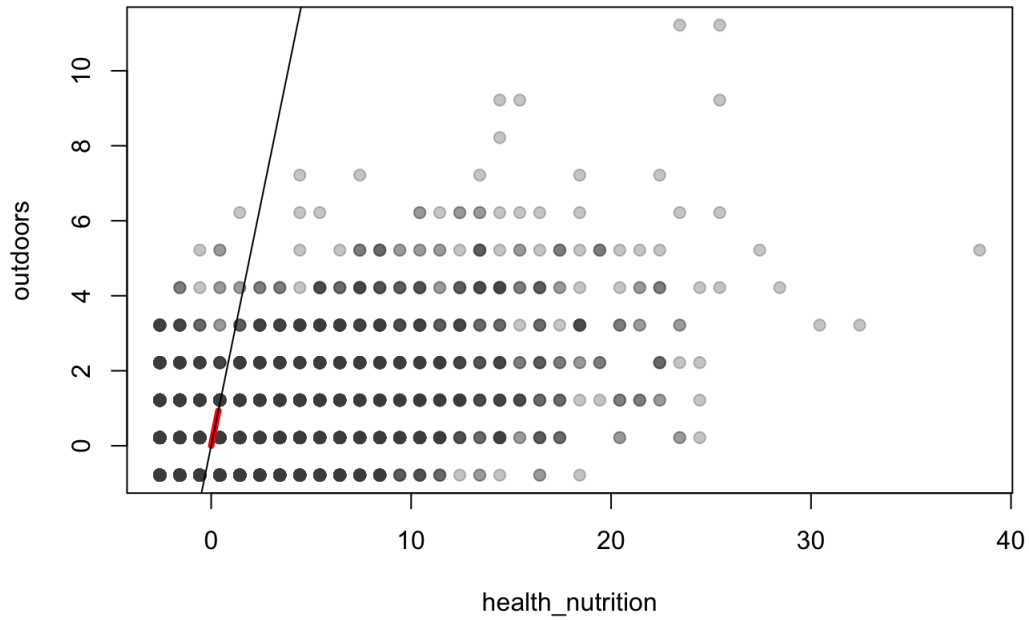
From the obtained data, we can say that around 3.68 people posted about both sports fandom and tv film. Perhaps they posted about their favorite sport movies.

Then I ran a PCA between health nutrition and outdoors.

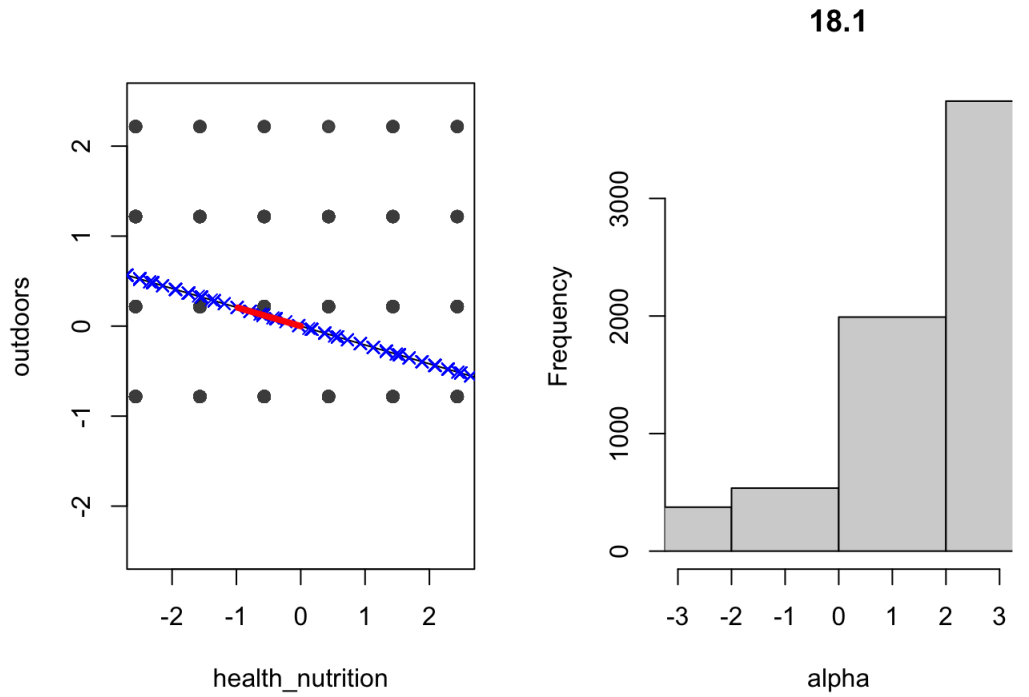
```
sm = read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/social_marketing.csv')

Z = sm[,c(17,24)]

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

plot(Z)
```




```
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))
plot(Z, pch=19, col=rgb(0.3,0.3,0.3,0.3))
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)


slope = v_try[2]/v_try[1]
abline(0, slope)
```



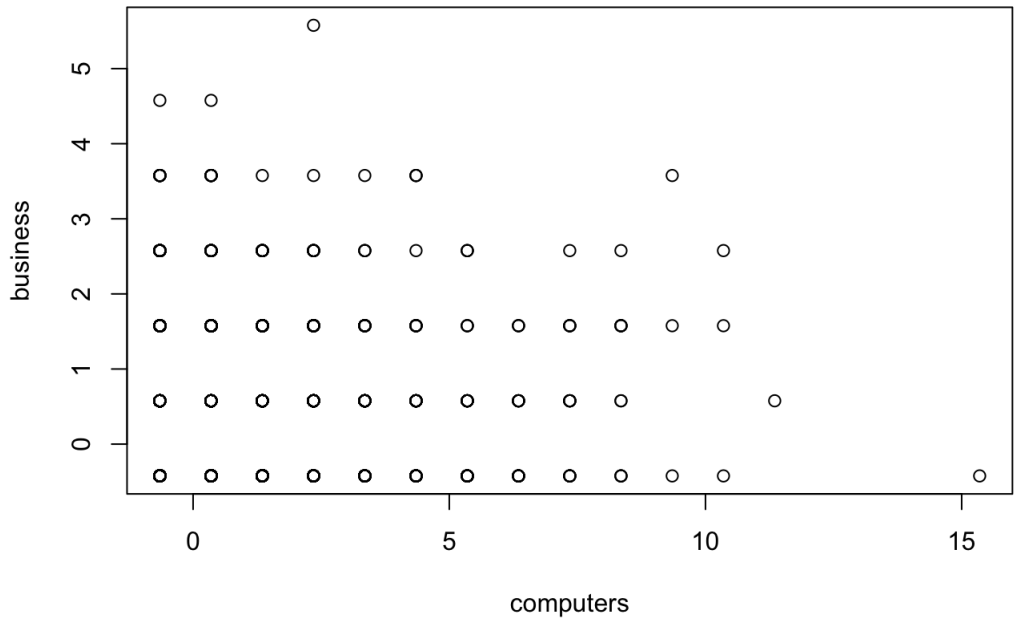

```
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))

par(mfrow=c(1,2))
plot(Z, pch=19, col=rgb(0.3,0.3,0.3,0.3),
     xlim=c(-2.5,2.5), ylim=c(-2.5,2.5))
slope = v_try[2]/v_try[1]
abline(0, slope)


alpha = Z %*% v_try
z_hat = alpha %*% v_try
points(z_hat, col='blue', pch=4)
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)


hist(alpha, 25, xlim=c(-3,3), main=round(var(alpha), 2))
```



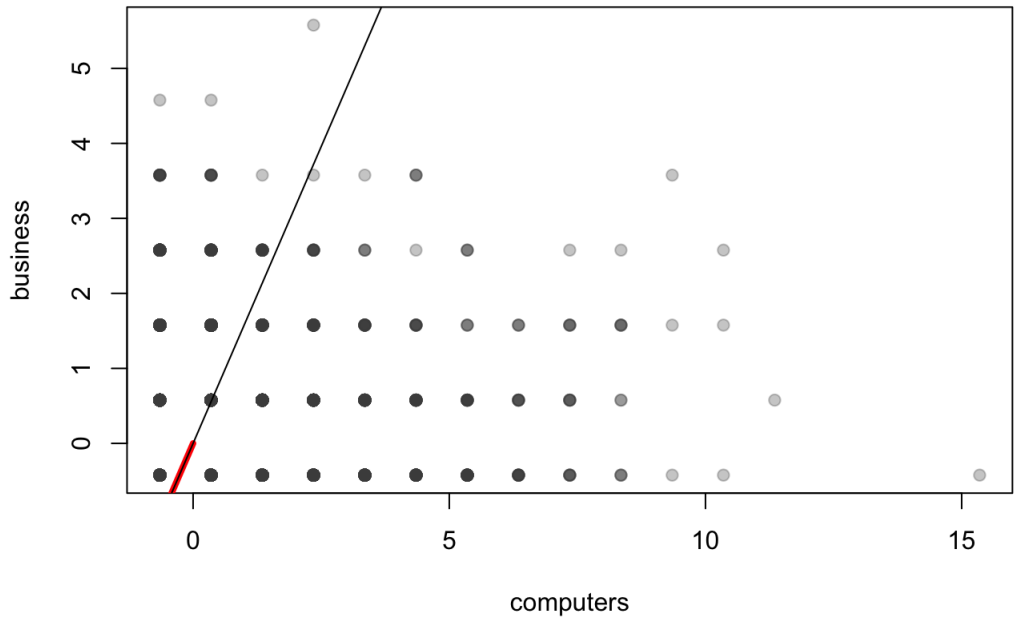
We observed that around 6.53 sent out a tweet relating to both health nutrition and outdoors. This is useful as NutrientH20 can focus on creating products relating to exercise and health. For instance, they could start working on a new energy drink.

Then for my third PCA, I ran it between computer and business.

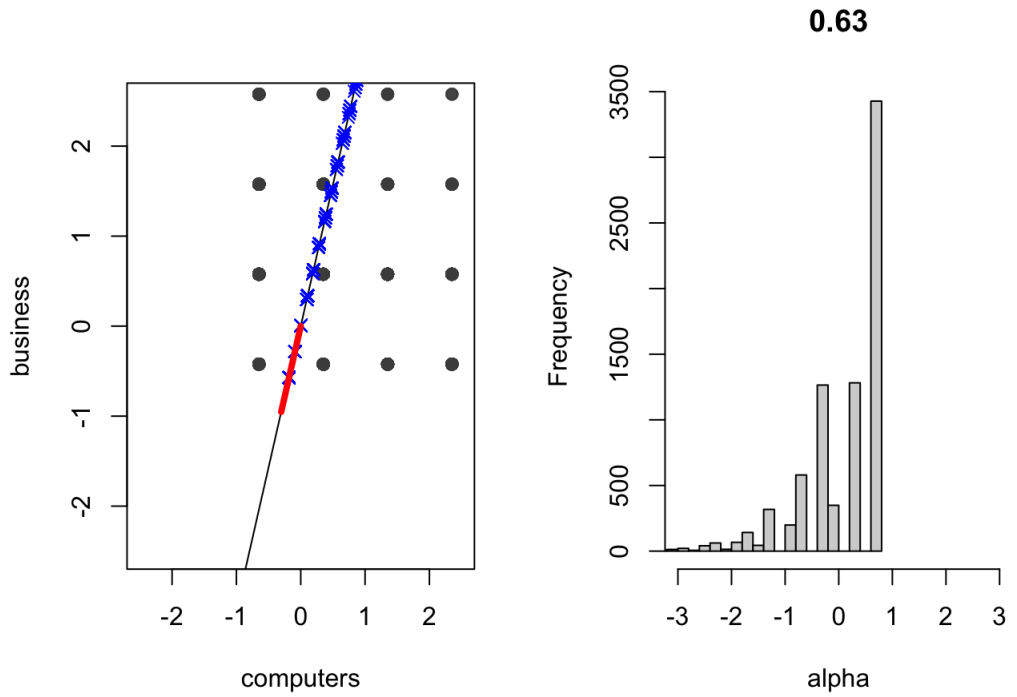
```
sm = read.csv('https://raw.githubusercontent.com/jgscott/STA380/master/data/social_marketing.csv')

Z = sm[,c(22,23)]

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

plot(Z)
```




```
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))
plot(Z, pch=19, col=rgb(0.3,0.3,0.3,0.3))
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)


slope = v_try[2]/v_try[1]
abline(0, slope)
```




```
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))

par(mfrow=c(1,2))
plot(Z, pch=19, col=rgb(0.3,0.3,0.3,0.3),
     xlim=c(-2.5,2.5), ylim=c(-2.5,2.5))
slope = v_try[2]/v_try[1]
abline(0, slope)


alpha = Z %*% v_try
z_hat = alpha %*% v_try
points(z_hat, col='blue', pch=4)
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)


hist(alpha, 25, xlim=c(-3,3), main=round(var(alpha), 2))
```



As we can see above, only about 0.53 people tweeted both about business and computers so its probably best if Nutrient H20 does not create any projects related to the IT industry as there probably won't be a lot of consumers.

##Q5. Loading libraries, reaading in the train and test folders, renaming them and creating corporas.

```
## Loading required package: NLP
```

```
## ── Attaching packages ─────────────────────────────────────────────────────────────────────── tidyver
se 1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.3     ✓ dplyr   1.0.0
## ✓ tidyr   1.1.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────────────────────────────────────── tidyverse_con
flicts() ──
## x ggplot2::annotate() masks NLP::annotate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
file_list_train = Sys.glob('STA380/data/ReutersC50/C50train/*/*.txt')
file_list_test=Sys.glob('STA380/data/ReutersC50/C50test/*/*.txt')

#read them
train=lapply(file_list_train, readerPlain)
test=lapply(file_list_test, readerPlain)
```

```
#rename
names(train)=trainnames
names(test)=testnames

train_raw = Corpus(VectorSource(train))
test_raw = Corpus(VectorSource(test))
```

Filter by making all words lowercase, removing numbers, punction, excess whitespaces and stopwords that end in “en”

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
##  [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
##  [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
## [101] "who's"      "what's"     "here's"     "there's"    "when's"
## [106] "where's"    "why's"      "how's"      "a"          "an"
## [111] "the"        "and"        "but"        "if"         "or"
## [116] "because"    "as"         "until"      "while"      "of"
## [121] "at"         "by"         "for"        "with"       "about"
## [126] "against"    "between"    "into"       "through"    "during"
## [131] "before"     "after"      "above"      "below"      "to"
## [136] "from"       "up"         "down"       "in"         "out"
## [141] "on"         "off"        "over"       "under"      "again"
## [146] "further"    "then"       "once"       "here"       "there"
## [151] "when"       "where"      "why"        "how"        "all"
## [156] "any"        "both"       "each"       "few"        "more"
## [161] "most"       "other"      "some"       "such"       "no"
## [166] "nor"        "not"        "only"       "own"        "same"
## [171] "so"         "than"       "too"        "very"
```

Create a doc-term-matrix from the corpus. Finally, the terms that only occur in one or two documents (noise of the ‘long tail’) which is noise and it doesnt give any useful information. Construct TF IDF weights and change it into matrix format. ``{r, echo = FALSE} DTM_train = DocumentTermMatrix(train_documents) DTM_test = DocumentTermMatrix(test_documents) DTM_train = removeSparseTerms(DTM_train, 0.95) DTM_test = removeSparseTerms(DTM_test, 0.95)

tfidf_train = weightTfIdf(DTM_train) tfidf_test = weightTfIdf(DTM_test)

trainmatrix=as.matrix(tfidf_train)

name_train<-regmatches(trainnames, regexpr(“[[:alpha:]]+”, trainnames)) rownames(trainmatrix)<-name_train

testmatrix=as.matrix(tfidf_test) name_test<-regmatches(testnames, regexpr(“[[:alpha:]]+”, testnames)) rownames(testmatrix)<-name_test

#PCA and Random ForesT

scrub_cols = which(colSums(trainmatrix) == 0) trainmatrix = trainmatrix[,-scrub_cols]

newwords= setdiff(colnames(testmatrix),colnames(trainmatrix)) newmatrix<-matrix(runif(2500*90, 0.0, 0.01), nrow=2500, ncol=length(newwords)) colnames(newmatrix)<-newwords

trainmatrix<-cbind(trainmatrix,newmatrix)

newwords<-setdiff(colnames(trainmatrix), colnames(testmatrix)) newmatrix<-matrix(runif(2500*59, 0.0, 0.01), nrow=2500, ncol=length(newwords)) colnames(newmatrix)<-newwords

testmatrix<-cbind(testmatrix, newmatrix)

PCA analysis

pca_train = prcomp(trainmatrix, scale=TRUE)

pca_train\(rotation[order(abs(pca_train\)rotation[,1]),decreasing=TRUE),1][1:25] pca_train\(rotation[order(abs(pca_train\)rotation[,2]),decreasing=TRUE),2][1:25]

pca_train$x[,1:2]

plot(pca_train\(x[,1:2], xlab="PCA 1 direction", ylab="PCA 2 direction", bty="n", type='n') text(pca_train\)x[,1:2], labels = 1:length(simon), cex=0.7)

apply PCA analysis from train set on the test set

pca_test=predict(pca_train, testmatrix)

train_df = data.frame(pca_train$x,name_train) test_df= data.frame(pca_test, name_test)

var <- paste("PC", 1:70, sep="") fmla <- as.formula(paste("as.factor(name_train) ~ ", paste(var, collapse="+")))

forest_coast = randomForest(fmla, data = train_df, ntree=500) prediction=predict(forest_coast, test_df)

```
Accuracy
``{r}
postResample(prediction, as.factor(test_df$name_test))
```

Association rule mining

Loading in and inspecting the data

```
summary(grocery)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## 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    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
inspect(grocery[1:10])
```

```
##      items
## [1]  {citrus fruit,
##       margarine,
##       ready soups,
##       semi-finished bread}
## [2]  {coffee,
##       tropical fruit,
##       yogurt}
## [3]  {whole milk}
## [4]  {cream cheese,
##       meat spreads,
##       pip fruit,
##       yogurt}
## [5]  {condensed milk,
##       long life bakery product,
##       other vegetables,
##       whole milk}
## [6]  {abrasive cleaner,
##       butter,
##       rice,
##       whole milk,
##       yogurt}
## [7]  {rolls/buns}
## [8]  {bottled beer,
##       liquor (appetizer),
##       other vegetables,
##       rolls/buns,
##       UHT-milk}
## [9]  {pot plants}
## [10] {cereals,
##       whole milk}
```

Using arulesViz to examine the different levels of confidence and support determine effectiveness.

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [171 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
##       lhs                            rhs                    support confidence   coverage      lift count
## [1]   {}                          => {whole milk}       0.25551601  0.2555160 1.00000000 1.0000000  2513
## [2]   {hard cheese}               => {whole milk}       0.01006609  0.4107884 0.02450432 1.6076815    99
## [3]   {butter milk}               => {other vegetables} 0.01037112  0.3709091 0.02796136 1.9169159   102
## [4]   {butter milk}               => {whole milk}       0.01159126  0.4145455 0.02796136 1.6223854   114
## [5]   {ham}                       => {whole milk}       0.01148958  0.4414062 0.02602949 1.7275091   113
## [6]   {sliced cheese}             => {whole milk}       0.01077783  0.4398340 0.02450432 1.7213560   106
## [7]   {oil}                       => {whole milk}       0.01128622  0.4021739 0.02806304 1.5739675   111
## [8]   {onions}                    => {other vegetables} 0.01423488  0.4590164 0.03101169 2.3722681   140
## [9]   {onions}                    => {whole milk}       0.01209964  0.3901639 0.03101169 1.5269647   119
## [10]  {berries}                   => {yogurt}           0.01057448  0.3180428 0.03324860 2.2798477   104
## [11]  {berries}                   => {other vegetables} 0.01026945  0.3088685 0.03324860 1.5962805   101
## [12]  {berries}                   => {whole milk}       0.01179461  0.3547401 0.03324860 1.3883281   116
## [13]  {hamburger meat}            => {other vegetables} 0.01382816  0.4159021 0.03324860 2.1494470   136
## [14]  {hamburger meat}            => {whole milk}       0.01474326  0.4434251 0.03324860 1.7354101   145
## [15]  {hygiene articles}          => {whole milk}       0.01281139  0.3888889 0.03294357 1.5219746   126
## [16]  {salty snack}               => {other vegetables} 0.01077783  0.2849462 0.03782410 1.4726465   106
## [17]  {salty snack}               => {whole milk}       0.01118454  0.2956989 0.03782410 1.1572618   110
## [18]  {sugar}                     => {other vegetables} 0.01077783  0.3183183 0.03385867 1.6451186   106
## [19]  {sugar}                     => {whole milk}       0.01504830  0.4444444 0.03385867 1.7393996   148
## [20]  {waffles}                   => {other vegetables} 0.01006609  0.2619048 0.03843416 1.3535645    99
## [21]  {waffles}                   => {whole milk}       0.01270971  0.3306878 0.03843416 1.2941961   125
## [22]  {long life bakery product}  => {other vegetables} 0.01067616  0.2853261 0.03741739 1.4746096   105
## [23]  {long life bakery product}  => {whole milk}       0.01352313  0.3614130 0.03741739 1.4144438   133
## [24]  {dessert}                   => {other vegetables} 0.01159126  0.3123288 0.03711235 1.6141636   114
## [25]  {dessert}                   => {whole milk}       0.01372649  0.3698630 0.03711235 1.4475140   135
## [26]  {cream cheese}              => {yogurt}           0.01240468  0.3128205 0.03965430 2.2424123   122
## [27]  {cream cheese}              => {other vegetables} 0.01372649  0.3461538 0.03965430 1.7889769   135
## [28]  {cream cheese}              => {whole milk}       0.01647178  0.4153846 0.03965430 1.6256696   162
## [29]  {chicken}                   => {root vegetables}  0.01087951  0.2535545 0.04290798 2.3262206   107
## [30]  {chicken}                   => {other vegetables} 0.01789527  0.4170616 0.04290798 2.1554393   176
## [31]  {chicken}                   => {whole milk}       0.01759024  0.4099526 0.04290798 1.6044106   173
## [32]  {white bread}               => {other vegetables} 0.01372649  0.3260870 0.04209456 1.6852681   135
## [33]  {white bread}               => {whole milk}       0.01708185  0.4057971 0.04209456 1.5881474   168
## [34]  {chocolate}                 => {soda}             0.01352313  0.2725410 0.04961871 1.5629391   133
## [35]  {chocolate}                 => {other vegetables} 0.01270971  0.2561475 0.04961871 1.3238103   125
## [36]  {chocolate}                 => {whole milk}       0.01667514  0.3360656 0.04961871 1.3152427   164
## [37]  {coffee}                    => {whole milk}       0.01870869  0.3222417 0.05805796 1.2611408   184
## [38]  {frozen vegetables}         => {yogurt}           0.01240468  0.2579281 0.04809354 1.8489235   122
## [39]  {frozen vegetables}         => {other vegetables} 0.01779359  0.3699789 0.04809354 1.9121083   175
## [40]  {frozen vegetables}         => {whole milk}       0.02043721  0.4249471 0.04809354 1.6630940   201
## [41]  {beef}                      => {root vegetables}  0.01738688  0.3313953 0.05246568 3.0403668   171
## [42]  {beef}                      => {rolls/buns}       0.01362481  0.2596899 0.05246568 1.4118576   134
## [43]  {beef}                      => {other vegetables} 0.01972547  0.3759690 0.05246568 1.9430662   194
## [44]  {beef}                      => {whole milk}       0.02125064  0.4050388 0.05246568 1.5851795   209
## [45]  {curd}                      => {yogurt}           0.01728521  0.3244275 0.05327911 2.3256154   170
## [46]  {curd}                      => {other vegetables} 0.01718353  0.3225191 0.05327911 1.6668288   169
## [47]  {curd}                      => {whole milk}       0.02613116  0.4904580 0.05327911 1.9194805   257
## [48]  {napkins}                   => {other vegetables} 0.01443823  0.2757282 0.05236401 1.4250060   142
## [49]  {napkins}                   => {whole milk}       0.01972547  0.3766990 0.05236401 1.4742678   194
```

```
## [49]  {napkins}                 => {whole milk}       0.01972547 0.3706990 0.05236401 1.4742078   194
## [50]  {pork}                    => {other vegetables} 0.02165735 0.3756614 0.05765125 1.9414764   213
## [51]  {pork}                    => {whole milk}       0.02216573 0.3844797 0.05765125 1.5047187   218
## [52]  {frankfurter}             => {rolls/buns}       0.01921708 0.3258621 0.05897306 1.7716161   189
## [53]  {frankfurter}             => {other vegetables} 0.01647178 0.2793103 0.05897306 1.4435193   162
## [54]  {frankfurter}             => {whole milk}       0.02053889 0.3482759 0.05897306 1.3630295   202
## [55]  {bottled beer}            => {whole milk}       0.02043721 0.2537879 0.08052872 0.9932367   201
## [56]  {brown bread}             => {other vegetables} 0.01870869 0.2884013 0.06487036 1.4905025   184
## [57]  {brown bread}             => {whole milk}       0.02521607 0.3887147 0.06487036 1.5212930   248
## [58]  {margarine}               => {rolls/buns}       0.01474326 0.2517361 0.05856634 1.3686151   145
## [59]  {margarine}               => {other vegetables} 0.01972547 0.3368056 0.05856634 1.7406635   194
## [60]  {margarine}               => {whole milk}       0.02419929 0.4131944 0.05856634 1.6170980   238
## [61]  {butter}                  => {yogurt}           0.01464159 0.2642202 0.05541434 1.8940273   144
## [62]  {butter}                  => {other vegetables} 0.02003050 0.3614679 0.05541434 1.8681223   197
## [63]  {butter}                  => {whole milk}       0.02755465 0.4972477 0.05541434 1.9460530   271
## [64]  {newspapers}              => {whole milk}       0.02735130 0.3426752 0.07981698 1.3411103   269
## [65]  {domestic eggs}           => {other vegetables} 0.02226741 0.3509615 0.06344687 1.8138238   219
## [66]  {domestic eggs}           => {whole milk}       0.02999492 0.4727564 0.06344687 1.8502027   295
## [67]  {fruit/vegetable juice}   => {soda}             0.01840366 0.2545710 0.07229283 1.4598869   181
## [68]  {fruit/vegetable juice}   => {yogurt}           0.01870869 0.2587904 0.07229283 1.8551049   184
## [69]  {fruit/vegetable juice}   => {other vegetables} 0.02104728 0.2911392 0.07229283 1.5046529   207
## [70]  {fruit/vegetable juice}   => {whole milk}       0.02663955 0.3684951 0.07229283 1.4421604   262
## [71]  {whipped/sour cream}      => {yogurt}           0.02074225 0.2893617 0.07168277 2.0742510   204
## [72]  {whipped/sour cream}      => {other vegetables} 0.02887646 0.4028369 0.07168277 2.0819237   284
## [73]  {whipped/sour cream}      => {whole milk}       0.03223183 0.4496454 0.07168277 1.7597542   317
## [74]  {pip fruit}               => {tropical fruit}   0.02043721 0.2701613 0.07564820 2.5746476   201
## [75]  {pip fruit}               => {other vegetables} 0.02613116 0.3454301 0.07564820 1.7852365   257
## [76]  {pip fruit}               => {whole milk}       0.03009659 0.3978495 0.07564820 1.5570432   296
## [77]  {pastry}                  => {other vegetables} 0.02257245 0.2537143 0.08896797 1.3112349   222
## [78]  {pastry}                  => {whole milk}       0.03324860 0.3737143 0.08896797 1.4625865   327
## [79]  {citrus fruit}            => {yogurt}           0.02165735 0.2616708 0.08276563 1.8757521   213
## [80]  {citrus fruit}            => {other vegetables} 0.02887646 0.3488943 0.08276563 1.8031403   284
## [81]  {citrus fruit}            => {whole milk}       0.03050330 0.3685504 0.08276563 1.4423768   300
## [82]  {sausage}                 => {soda}             0.02430097 0.2586580 0.09395018 1.4833245   239
## [83]  {sausage}                 => {rolls/buns}       0.03060498 0.3257576 0.09395018 1.7710480   301
## [84]  {sausage}                 => {other vegetables} 0.02694459 0.2867965 0.09395018 1.4822091   265
## [85]  {sausage}                 => {whole milk}       0.02989324 0.3181818 0.09395018 1.2452520   294
## [86]  {bottled water}           => {soda}             0.02897814 0.2621895 0.11052364 1.5035766   285
## [87]  {bottled water}           => {whole milk}       0.03436706 0.3109476 0.11052364 1.2169396   338
## [88]  {tropical fruit}          => {yogurt}           0.02928317 0.2790698 0.10493137 2.0004746   288
## [89]  {tropical fruit}          => {other vegetables} 0.03589222 0.3420543 0.10493137 1.7677896   353
## [90]  {tropical fruit}          => {whole milk}       0.04229792 0.4031008 0.10493137 1.5775950   416
## [91]  {root vegetables}         => {other vegetables} 0.04738180 0.4347015 0.10899847 2.2466049   466
## [92]  {root vegetables}         => {whole milk}       0.04890696 0.4486940 0.10899847 1.7560310   481
## [93]  {yogurt}                  => {other vegetables} 0.04341637 0.3112245 0.13950178 1.6084566   427
## [94]  {yogurt}                  => {whole milk}       0.05602440 0.4016035 0.13950178 1.5717351   551
## [95]  {rolls/buns}              => {whole milk}       0.05663447 0.3079049 0.18393493 1.2050318   557
## [96]  {other vegetables}        => {whole milk}       0.07483477 0.3867578 0.19349263 1.5136341   736
## [97]  {whole milk}              => {other vegetables} 0.07483477 0.2928770 0.25551601 1.5136341   736
## [98]  {curd,
##        yogurt}                  => {whole milk}       0.01006609 0.5823529 0.01728521 2.2791250    99
## [99]  {curd,
##        whole milk}              => {yogurt}           0.01006609 0.3852140 0.02613116 2.7613555    99
## [100] {other vegetables,
##        pork}                    => {whole milk}       0.01016777 0.4694836 0.02165735 1.8373939   100
## [101] {pork,
##        whole milk}              => {other vegetables} 0.01016777 0.4587156 0.02216573 2.3707136   100
## [102] {butter,
##        other vegetables}        => {whole milk}       0.01148958 0.5736041 0.02003050 2.2448850   113
## [103] {butter,
##        whole milk}              => {other vegetables} 0.01148958 0.4169742 0.02755465 2.1549874   113
## [104] {domestic eggs,
##        other vegetables}        => {whole milk}       0.01230300 0.5525114 0.02226741 2.1623358   121
## [105] {domestic eggs,
##        whole milk}              => {other vegetables} 0.01230300 0.4101695 0.02999492 2.1198197   121
## [106] {fruit/vegetable juice,
##        other vegetables}        => {whole milk}       0.01047280 0.4975845 0.02104728 1.9473713   103
## [107] {fruit/vegetable juice,
##        whole milk}              => {other vegetables} 0.01047280 0.3931298 0.02663955 2.0317558   103
## [108] {whipped/sour cream,
##        yogurt}                  => {other vegetables} 0.01016777 0.4901961 0.02074225 2.5334096   100
## [109] {other vegetables,
##        whipped/sour cream}      => {yogurt}           0.01016777 0.3521127 0.02887646 2.5240730   100
```

```
## [110] {whipped/sour cream,
##        yogurt}                 => {whole milk}       0.01087951  0.5245098 0.02074225 2.0527473   107
## [111] {whipped/sour cream,
##        whole milk}             => {yogurt}           0.01087951  0.3375394 0.03223183 2.4196066   107
## [112] {other vegetables,
##        whipped/sour cream}     => {whole milk}       0.01464159  0.5070423 0.02887646 1.9843854   144
## [113] {whipped/sour cream,
##        whole milk}             => {other vegetables} 0.01464159  0.4542587 0.03223183 2.3476795   144
## [114] {other vegetables,
##        pip fruit}              => {whole milk}       0.01352313  0.5175097 0.02613116 2.0253514   133
## [115] {pip fruit,
##        whole milk}             => {other vegetables} 0.01352313  0.4493243 0.03009659 2.3221780   133
## [116] {other vegetables,
##        pastry}                 => {whole milk}       0.01057448  0.4684685 0.02257245 1.8334212   104
## [117] {pastry,
##        whole milk}             => {other vegetables} 0.01057448  0.3180428 0.03324860 1.6436947   104
## [118] {citrus fruit,
##        root vegetables}        => {other vegetables} 0.01037112  0.5862069 0.01769192 3.0296084   102
## [119] {citrus fruit,
##        other vegetables}       => {root vegetables}  0.01037112  0.3591549 0.02887646 3.2950455   102
## [120] {citrus fruit,
##        yogurt}                 => {whole milk}       0.01026945  0.4741784 0.02165735 1.8557678   101
## [121] {citrus fruit,
##        whole milk}             => {yogurt}           0.01026945  0.3366667 0.03050330 2.4133503   101
## [122] {citrus fruit,
##        other vegetables}       => {whole milk}       0.01301474  0.4507042 0.02887646 1.7638982   128
## [123] {citrus fruit,
##        whole milk}             => {other vegetables} 0.01301474  0.4266667 0.03050330 2.2050797   128
## [124] {other vegetables,
##        sausage}                => {whole milk}       0.01016777  0.3773585 0.02694459 1.4768487   100
## [125] {sausage,
##        whole milk}             => {other vegetables} 0.01016777  0.3401361 0.02989324 1.7578760   100
## [126] {bottled water,
##        other vegetables}       => {whole milk}       0.01077783  0.4344262 0.02480935 1.7001918   106
## [127] {bottled water,
##        whole milk}             => {other vegetables} 0.01077783  0.3136095 0.03436706 1.6207825   106
## [128] {root vegetables,
##        tropical fruit}         => {other vegetables} 0.01230300  0.5845411 0.02104728 3.0209991   121
## [129] {other vegetables,
##        tropical fruit}         => {root vegetables}  0.01230300  0.3427762 0.03589222 3.1447798   121
## [130] {other vegetables,
##        root vegetables}        => {tropical fruit}   0.01230300  0.2596567 0.04738180 2.4745380   121
## [131] {root vegetables,
##        tropical fruit}         => {whole milk}       0.01199797  0.5700483 0.02104728 2.2309690   118
## [132] {tropical fruit,
##        whole milk}             => {root vegetables}  0.01199797  0.2836538 0.04229792 2.6023653   118
## [133] {tropical fruit,
##        yogurt}                 => {other vegetables} 0.01230300  0.4201389 0.02928317 2.1713431   121
## [134] {other vegetables,
##        tropical fruit}         => {yogurt}           0.01230300  0.3427762 0.03589222 2.4571457   121
## [135] {other vegetables,
##        yogurt}                 => {tropical fruit}   0.01230300  0.2833724 0.04341637 2.7005496   121
## [136] {tropical fruit,
##        yogurt}                 => {whole milk}       0.01514997  0.5173611 0.02928317 2.0247698   149
## [137] {tropical fruit,
##        whole milk}             => {yogurt}           0.01514997  0.3581731 0.04229792 2.5675162   149
## [138] {whole milk,
##        yogurt}                 => {tropical fruit}   0.01514997  0.2704174 0.05602440 2.5770885   149
## [139] {rolls/buns,
##        tropical fruit}         => {whole milk}       0.01098119  0.4462810 0.02460600 1.7465872   108
## [140] {tropical fruit,
##        whole milk}             => {rolls/buns}       0.01098119  0.2596154 0.04229792 1.4114524   108
## [141] {other vegetables,
##        tropical fruit}         => {whole milk}       0.01708185  0.4759207 0.03589222 1.8625865   168
## [142] {tropical fruit,
##        whole milk}             => {other vegetables} 0.01708185  0.4038462 0.04229792 2.0871397   168
## [143] {root vegetables,
##        yogurt}                 => {other vegetables} 0.01291307  0.5000000 0.02582613 2.5840778   127
## [144] {other vegetables,
##        root vegetables}        => {yogurt}           0.01291307  0.2725322 0.04738180 1.9536108   127
## [145] {other vegetables,
##        yogurt}                 => {root vegetables}  0.01291307  0.2974239 0.04341637 2.7286977   127
## [146] {root vegetables,
```

```
##        yogurt}                   => {whole milk}       0.01453991  0.5629921 0.02582613 2.2033536   143
## [147] {root vegetables,
##        whole milk}               => {yogurt}           0.01453991  0.2972973 0.04890696 2.1311362   143
## [148] {whole milk,
##        yogurt}                   => {root vegetables}  0.01453991  0.2595281 0.05602440 2.3810253   143
## [149] {rolls/buns,
##        root vegetables}          => {other vegetables} 0.01220132  0.5020921 0.02430097 2.5948898   120
## [150] {other vegetables,
##        root vegetables}          => {rolls/buns}       0.01220132  0.2575107 0.04738180 1.4000100   120
## [151] {other vegetables,
##        rolls/buns}               => {root vegetables}  0.01220132  0.2863962 0.04260295 2.6275247   120
## [152] {rolls/buns,
##        root vegetables}          => {whole milk}       0.01270971  0.5230126 0.02430097 2.0468876   125
## [153] {root vegetables,
##        whole milk}               => {rolls/buns}       0.01270971  0.2598753 0.04890696 1.4128652   125
## [154] {other vegetables,
##        root vegetables}          => {whole milk}       0.02318251  0.4892704 0.04738180 1.9148326   228
## [155] {root vegetables,
##        whole milk}               => {other vegetables} 0.02318251  0.4740125 0.04890696 2.4497702   228
## [156] {other vegetables,
##        whole milk}               => {root vegetables}  0.02318251  0.3097826 0.07483477 2.8420820   228
## [157] {soda,
##        yogurt}                   => {whole milk}       0.01047280  0.3828996 0.02735130 1.4985348   103
## [158] {soda,
##        whole milk}               => {yogurt}           0.01047280  0.2614213 0.04006101 1.8739641   103
## [159] {other vegetables,
##        soda}                     => {whole milk}       0.01392984  0.4254658 0.03274021 1.6651240   137
## [160] {soda,
##        whole milk}               => {other vegetables} 0.01392984  0.3477157 0.04006101 1.7970490   137
## [161] {rolls/buns,
##        yogurt}                   => {other vegetables} 0.01148958  0.3343195 0.03436706 1.7278153   113
## [162] {other vegetables,
##        yogurt}                   => {rolls/buns}       0.01148958  0.2646370 0.04341637 1.4387534   113
## [163] {other vegetables,
##        rolls/buns}               => {yogurt}           0.01148958  0.2696897 0.04260295 1.9332351   113
## [164] {rolls/buns,
##        yogurt}                   => {whole milk}       0.01555669  0.4526627 0.03436706 1.7715630   153
## [165] {whole milk,
##        yogurt}                   => {rolls/buns}       0.01555669  0.2776770 0.05602440 1.5096478   153
## [166] {rolls/buns,
##        whole milk}               => {yogurt}           0.01555669  0.2746858 0.05663447 1.9690488   153
## [167] {other vegetables,
##        yogurt}                   => {whole milk}       0.02226741  0.5128806 0.04341637 2.0072345   219
## [168] {whole milk,
##        yogurt}                   => {other vegetables} 0.02226741  0.3974592 0.05602440 2.0541308   219
## [169] {other vegetables,
##        whole milk}               => {yogurt}           0.02226741  0.2975543 0.07483477 2.1329789   219
## [170] {other vegetables,
##        rolls/buns}               => {whole milk}       0.01789527  0.4200477 0.04260295 1.6439194   176
## [171] {rolls/buns,
##        whole milk}               => {other vegetables} 0.01789527  0.3159785 0.05663447 1.6330258   176
```

I used 'sort' so I could plot only the top 10 results based on their confidence or lift.

```
plotly_arules(itemset)
```

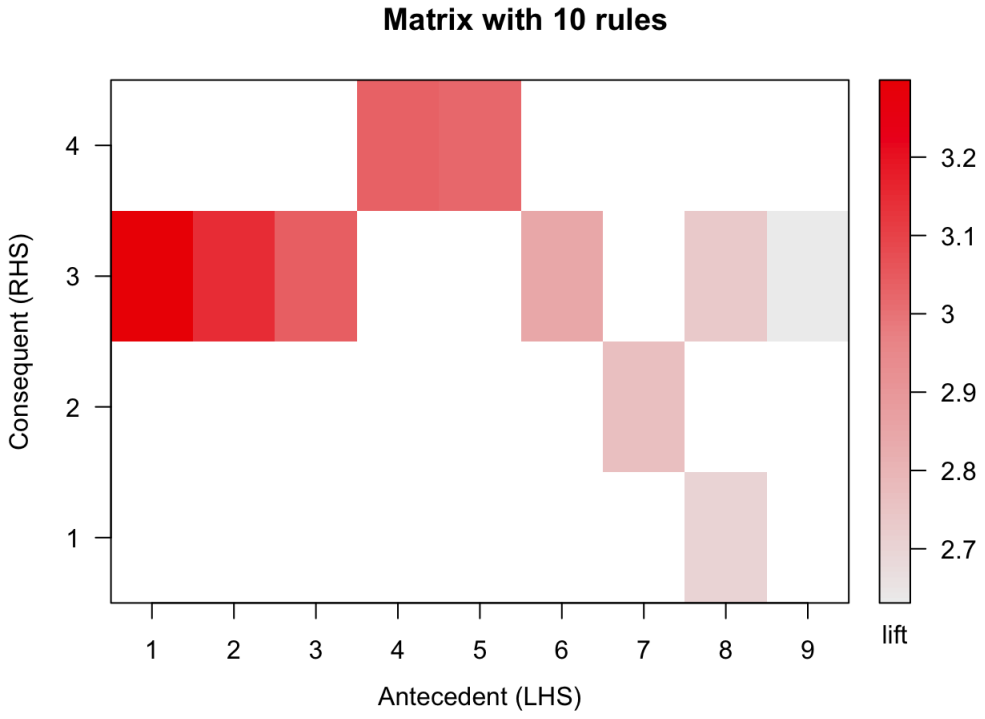
```
## Warning: 'plotly_arules' is deprecated.
## Use 'plot' instead.
## See help("Deprecated")
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

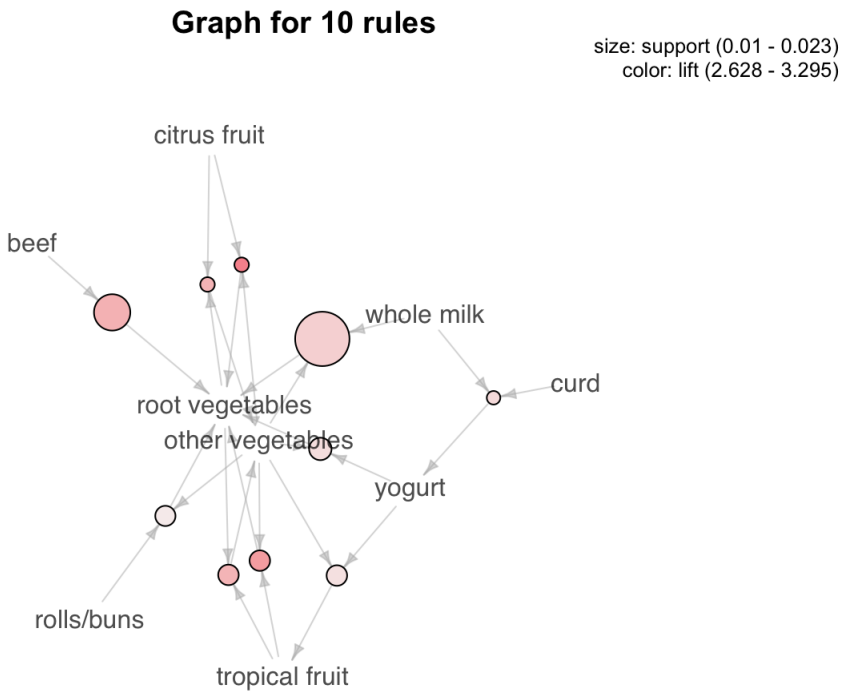
```
## Warning: `arrange_()` is deprecated as of dplyr 0.7.0.
## Please use `arrange()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
plot(itemset_II, method='matrix')
```

```
## Itemsets in Antecedent (LHS)
## [1] "{citrus fruit,other vegetables}"   "{other vegetables,tropical fruit}"
## [3] "{beef}"                            "{citrus fruit,root vegetables}"
## [5] "{root vegetables,tropical fruit}"  "{other vegetables,whole milk}"
## [7] "{curd,whole milk}"                 "{other vegetables,yogurt}"
## [9] "{other vegetables,rolls/buns}"
## Itemsets in Consequent (RHS)
## [1] "{tropical fruit}"   "{yogurt}"           "{root vegetables}"
## [4] "{other vegetables}"
```




```
plot(itemset_II, method='graph')
```



The plot below lets see the itemsets with high degrees of confidence and lift values.

```
inspect(head(itemset_cfd))
```

```
##     lhs                                rhs                support
## [1] {citrus fruit,root vegetables}   => {other vegetables} 0.01037112
## [2] {root vegetables,tropical fruit} => {other vegetables} 0.01230300
## [3] {curd,yogurt}                    => {whole milk}       0.01006609
## [4] {butter,other vegetables}        => {whole milk}       0.01148958
## [5] {root vegetables,tropical fruit} => {whole milk}       0.01199797
## [6] {root vegetables,yogurt}         => {whole milk}       0.01453991
##     confidence coverage   lift     count
## [1] 0.5862069  0.01769192 3.029608 102
## [2] 0.5845411  0.02104728 3.020999 121
## [3] 0.5823529  0.01728521 2.279125  99
## [4] 0.5736041  0.02003050 2.244885 113
## [5] 0.5700483  0.02104728 2.230969 118
## [6] 0.5629921  0.02582613 2.203354 143
```

```
inspect(head(itemset_lift))
```

```
##     lhs                                  rhs                support
## [1] {citrus fruit,other vegetables}   => {root vegetables}  0.01037112
## [2] {other vegetables,tropical fruit} => {root vegetables}  0.01230300
## [3] {beef}                            => {root vegetables}  0.01738688
## [4] {citrus fruit,root vegetables}    => {other vegetables} 0.01037112
## [5] {root vegetables,tropical fruit}  => {other vegetables} 0.01230300
## [6] {other vegetables,whole milk}     => {root vegetables}  0.02318251
##     confidence coverage   lift     count
## [1] 0.3591549  0.02887646 3.295045 102
## [2] 0.3427762  0.03589222 3.144780 121
## [3] 0.3313953  0.05246568 3.040367 171
## [4] 0.5862069  0.01769192 3.029608 102
## [5] 0.5845411  0.02104728 3.020999 121
## [6] 0.3097826  0.07483477 2.842082 228
```

From the inspection below, I can see a decent amount of different options that indicated that ‘citrus fruit’ should be included in the basket.

```
itemset = apriori(data=grocery, parameter=list(supp=0.001, conf=0.09), appearance = list(default = 'lhs', rh
s = 'citrus fruit'), control=list(verbose=F))
itemset = sort(itemset, decreasing=TRUE, by='confidence')
```

```
inspect(itemset[1:10])
```

```
##      lhs                         rhs                support confidence    coverage     lift count
## [1]  {fruit/vegetable juice,
##       grapes,
##       tropical fruit}        => {citrus fruit} 0.001118454  0.6875000 0.001626843 8.306588    11
## [2]  {pastry,
##       rolls/buns,
##       whipped/sour cream,
##       whole milk}            => {citrus fruit} 0.001016777  0.6666667 0.001525165 8.054873    10
## [3]  {herbs,
##       other vegetables,
##       tropical fruit}        => {citrus fruit} 0.001016777  0.6250000 0.001626843 7.551443    10
## [4]  {cream cheese,
##       domestic eggs,
##       other vegetables,
##       whole milk}            => {citrus fruit} 0.001118454  0.6111111 0.001830198 7.383634    11
## [5]  {bottled water,
##       herbs,
##       whole milk}            => {citrus fruit} 0.001016777  0.5882353 0.001728521 7.107241    10
## [6]  {fruit/vegetable juice,
##       oil,
##       root vegetables}       => {citrus fruit} 0.001016777  0.5555556 0.001830198 6.712394    10
## [7]  {fruit/vegetable juice,
##       root vegetables,
##       tropical fruit,
##       whole milk}            => {citrus fruit} 0.001016777  0.5555556 0.001830198 6.712394    10
## [8]  {fruit/vegetable juice,
##       other vegetables,
##       root vegetables,
##       soda}                  => {citrus fruit} 0.001016777  0.5263158 0.001931876 6.359110    10
## [9]  {fruit/vegetable juice,
##       oil,
##       other vegetables}      => {citrus fruit} 0.001118454  0.5238095 0.002135231 6.328829    11
## [10] {chicken,
##       domestic eggs,
##       yogurt}                => {citrus fruit} 0.001016777  0.5000000 0.002033554 6.041155    10
```

Setting lhs as 'citrus fruit' showed a decent connection which means it might not have given us any new information as it showed up in 13 out of 40 baskets.

```
itemset2 <- apriori(data=grocery, parameter=list(supp=0.01, conf=0.05), appearance = list(default = 'rhs', l
hs = 'citrus fruit'), control=list(verbose=F))
itemset2 <- sort(itemset2, by='confidence', decreasing=TRUE)
```

```
inspect(itemset2)
```

```
##      lhs               rhs                    support    confidence coverage
## [1]  {citrus fruit} => {whole milk}            0.03050330 0.36855037 0.08276563
## [2]  {citrus fruit} => {other vegetables}      0.02887646 0.34889435 0.08276563
## [3]  {citrus fruit} => {yogurt}                0.02165735 0.26167076 0.08276563
## [4]  {}             => {whole milk}            0.25551601 0.25551601 1.00000000
## [5]  {citrus fruit} => {tropical fruit}        0.01992883 0.24078624 0.08276563
## [6]  {citrus fruit} => {root vegetables}       0.01769192 0.21375921 0.08276563
## [7]  {citrus fruit} => {rolls/buns}            0.01677682 0.20270270 0.08276563
## [8]  {}             => {other vegetables}      0.19349263 0.19349263 1.00000000
## [9]  {}             => {rolls/buns}            0.18393493 0.18393493 1.00000000
## [10] {}             => {soda}                  0.17437722 0.17437722 1.00000000
## [11] {citrus fruit} => {pip fruit}             0.01382816 0.16707617 0.08276563
## [12] {citrus fruit} => {bottled water}         0.01352313 0.16339066 0.08276563
## [13] {citrus fruit} => {soda}                  0.01281139 0.15479115 0.08276563
## [14] {}             => {yogurt}                0.13950178 0.13950178 1.00000000
## [15] {citrus fruit} => {sausage}               0.01128622 0.13636364 0.08276563
## [16] {citrus fruit} => {whipped/sour cream}    0.01087951 0.13144963 0.08276563
## [17] {citrus fruit} => {domestic eggs}         0.01037112 0.12530713 0.08276563
## [18] {citrus fruit} => {fruit/vegetable juice} 0.01037112 0.12530713 0.08276563
## [19] {}             => {bottled water}         0.11052364 0.11052364 1.00000000
## [20] {}             => {root vegetables}       0.10899847 0.10899847 1.00000000
## [21] {}             => {tropical fruit}        0.10493137 0.10493137 1.00000000
## [22] {}             => {shopping bags}         0.09852567 0.09852567 1.00000000
```

```
## [23] {}             => {sausage}               0.09395018 0.09395018 1.00000000
## [24] {}             => {pastry}                0.08896797 0.08896797 1.00000000
## [25] {}             => {bottled beer}          0.08052872 0.08052872 1.00000000
## [26] {}             => {newspapers}            0.07981698 0.07981698 1.00000000
## [27] {}             => {canned beer}           0.07768175 0.07768175 1.00000000
## [28] {}             => {pip fruit}             0.07564820 0.07564820 1.00000000
## [29] {}             => {fruit/vegetable juice} 0.07229283 0.07229283 1.00000000
## [30] {}             => {whipped/sour cream}    0.07168277 0.07168277 1.00000000
## [31] {}             => {brown bread}           0.06487036 0.06487036 1.00000000
## [32] {}             => {domestic eggs}         0.06344687 0.06344687 1.00000000
## [33] {}             => {frankfurter}           0.05897306 0.05897306 1.00000000
## [34] {}             => {margarine}             0.05856634 0.05856634 1.00000000
## [35] {}             => {coffee}                0.05805796 0.05805796 1.00000000
## [36] {}             => {pork}                  0.05765125 0.05765125 1.00000000
## [37] {}             => {butter}                0.05541434 0.05541434 1.00000000
## [38] {}             => {curd}                  0.05327911 0.05327911 1.00000000
## [39] {}             => {beef}                  0.05246568 0.05246568 1.00000000
## [40] {}             => {napkins}               0.05236401 0.05236401 1.00000000
##      lift      count
## [1]  1.4423768  300
## [2]  1.8031403  284
## [3]  1.8757521  213
## [4]  1.0000000 2513
## [5]  2.2947022  196
## [6]  1.9611211  174
## [7]  1.1020349  165
## [8]  1.0000000 1903
## [9]  1.0000000 1809
## [10] 1.0000000 1715
## [11] 2.2085942  136
## [12] 1.4783323  133
## [13] 0.8876799  126
## [14] 1.0000000 1372
## [15] 1.4514463  111
## [16] 1.8337690  107
## [17] 1.9749929  102
## [18] 1.7333271  102
## [19] 1.0000000 1087
## [20] 1.0000000 1072
## [21] 1.0000000 1032
## [22] 1.0000000  969
## [23] 1.0000000  924
## [24] 1.0000000  875
## [25] 1.0000000  792
## [26] 1.0000000  785
## [27] 1.0000000  764
## [28] 1.0000000  744
## [29] 1.0000000  711
## [30] 1.0000000  705
## [31] 1.0000000  638
## [32] 1.0000000  624
## [33] 1.0000000  580
## [34] 1.0000000  576
## [35] 1.0000000  571
## [36] 1.0000000  567
## [37] 1.0000000  545
## [38] 1.0000000  524
## [39] 1.0000000  516
## [40] 1.0000000  515
```

For this analysis, I conducted tests for different values and combinations for support and confidence, and decided on these parameters. I chose these parameters because I wanted to capture that the quatiles looked like at 25% and 75%. To be time efficient, I set the support at 0.01 as it predicted the items that occurred the most and tested a support for 0.005 with a higher confidence of 0.75 to pick up items I may have left out.

I found that the discovered item sets make sense because they have a connection and are repeated often. An association analysis was run at different of confidence and support levels, as citrus fruit had high rhs at all levels when sorted by confidence and lift but as the only item as lhs, it doesn't tell us much.

In summary, Similar and simpler items should be placed together or close to each other because this makes it easier for the customer and could possibly increase revenue because they would have an incentive to return.