STA380 Exercises

Avery Shepherd, Allie Touchstone, Ally McNulty, Chaitra Setty

8/16/2021

Green Buildings

```
## green rent per sqft is 30.02848
```

```
##
## non green rent per sqft is 28.44478
```

The intern says that green buildings cost 27.60 per sqft when the actual cost is 30.03 so there is a 2.43 dollar discrepancy. The intern also says non green buildings cost 25 per sqft when they actually cost 28.58 so there is a 3.58 discrepancy.

```
## the actual difference for the building if it were green is this dollar amount:
```

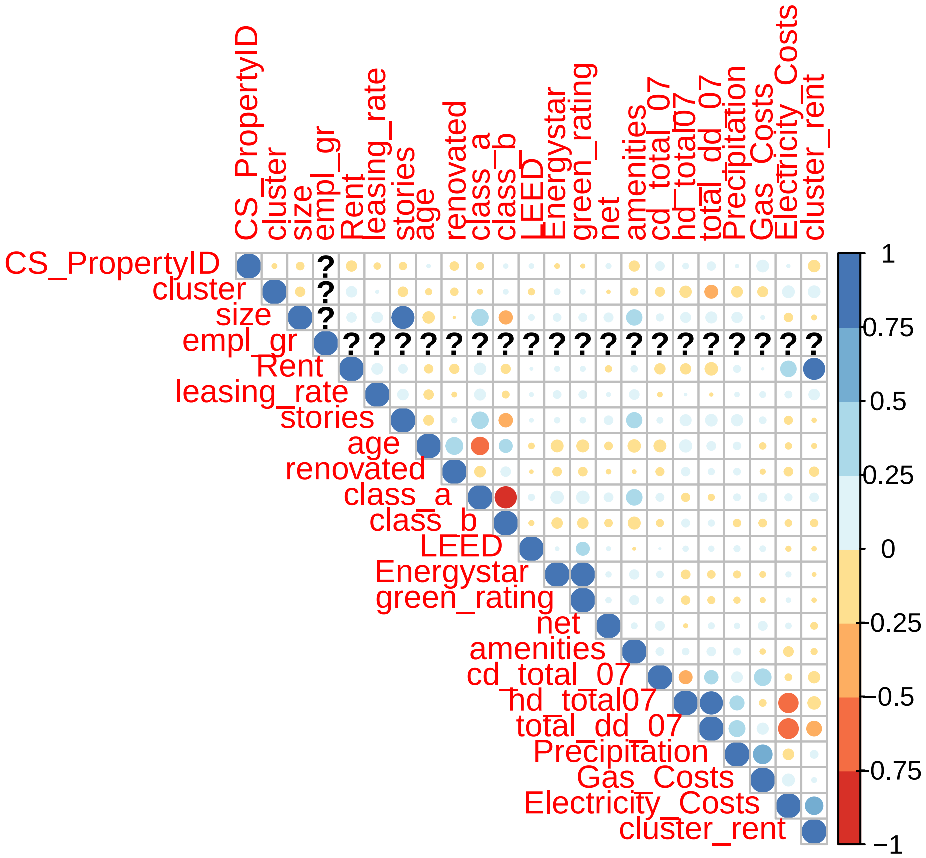
```
## [1] 395925.1
```

That is $288337 less than predicted.

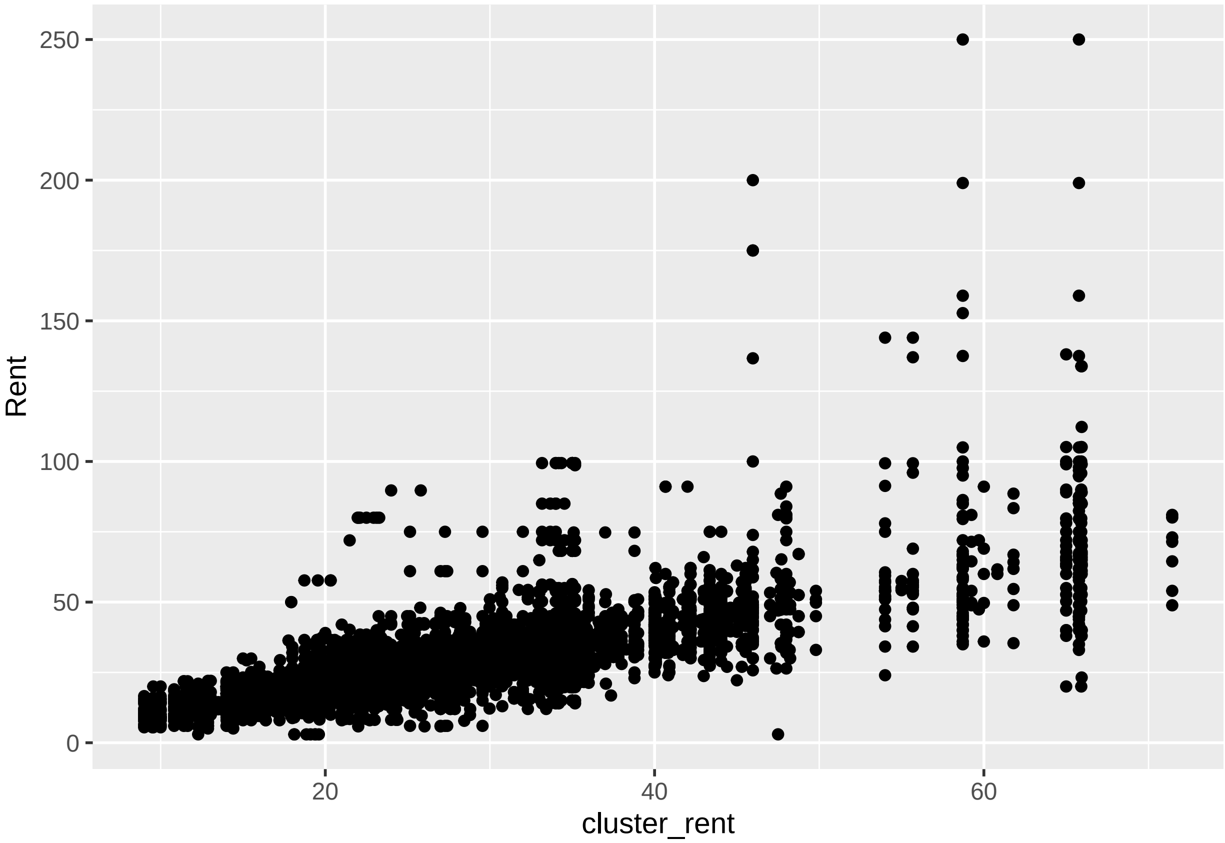
If costs of the building are an extra 5 million to make it green it would actually take this many years to pay off the extra building costs:

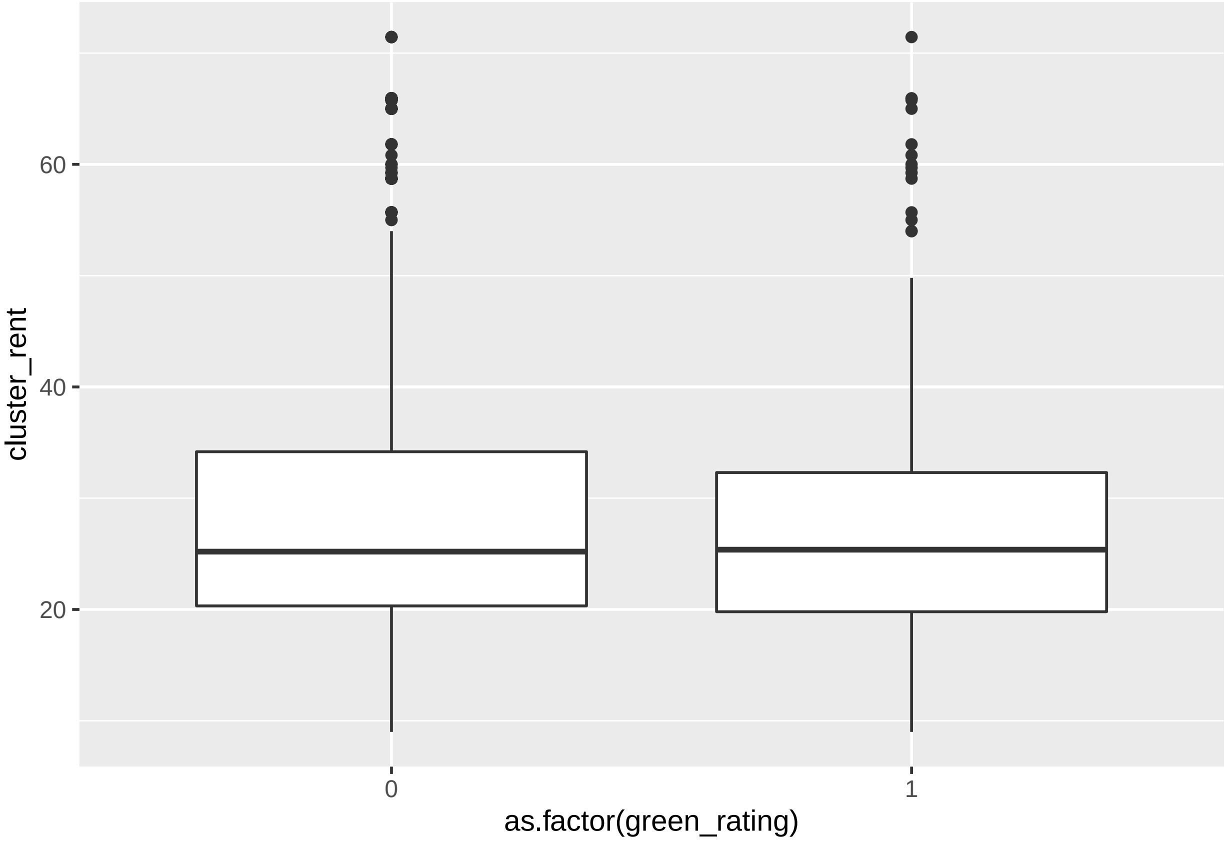
```
## [1] 13.82502
```

Which is over 6 years longer than the analyst projected. After almost 14 years you would then be making $361663 a year extra over having a green building.

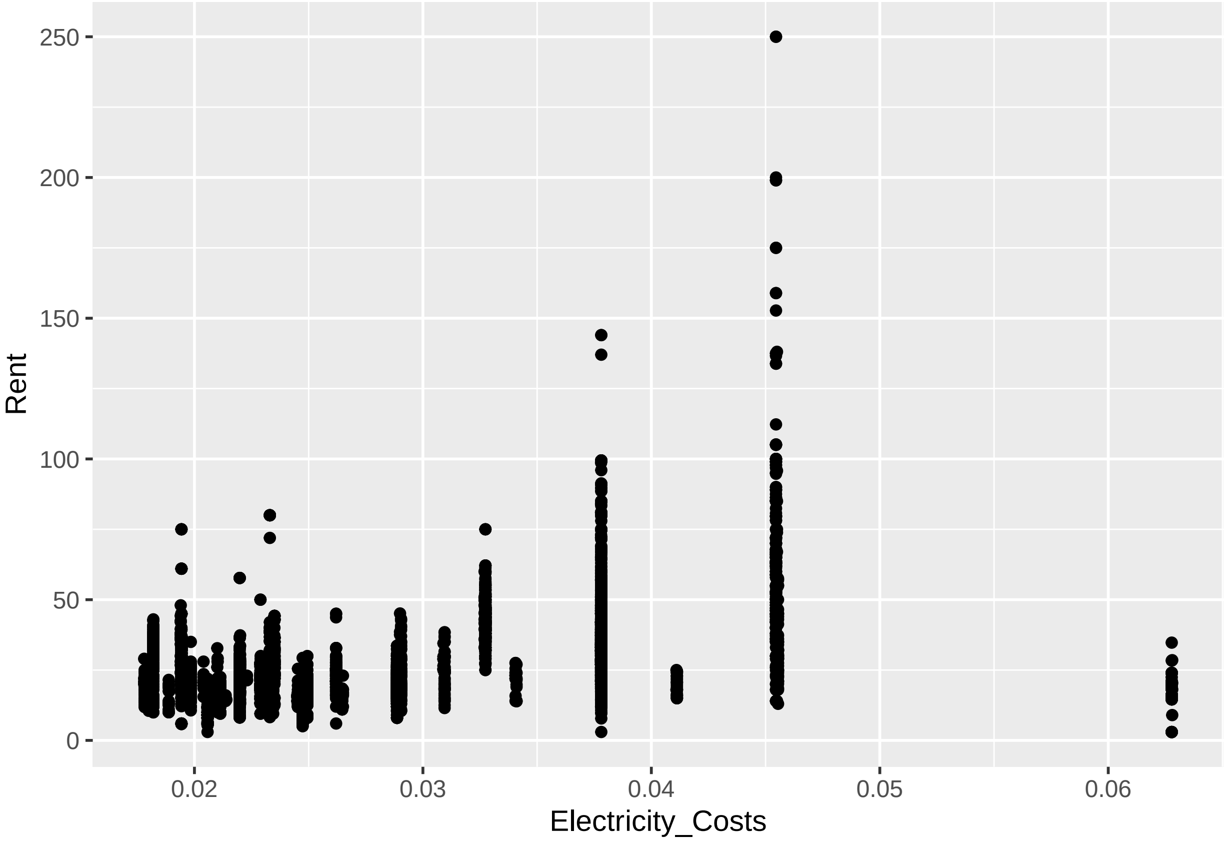


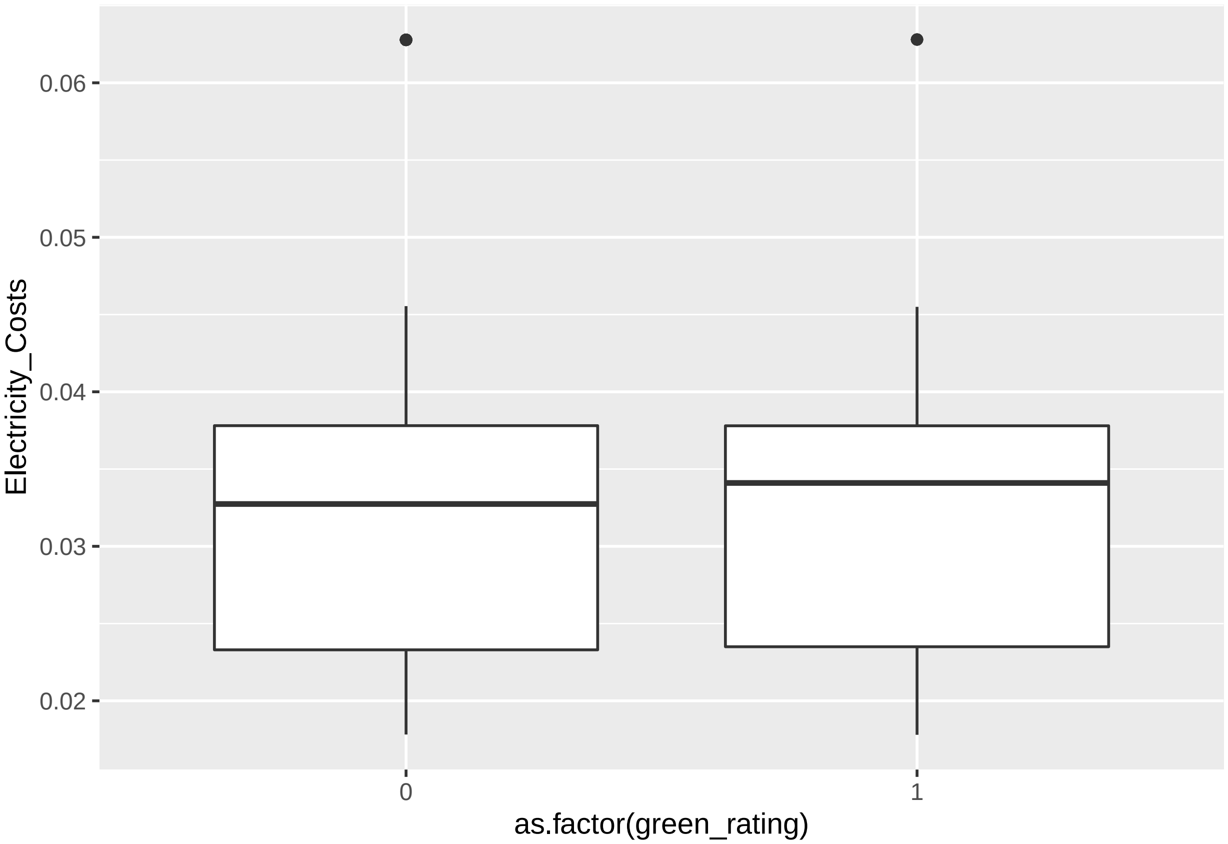
cluster rent (the buildings location) has a high impact on Rent electricity cost has high impact on rent





As you increase cluster, rent increases. cluster does not impact green rating. The location of the apartment (cluster) definitely has an impact on rent price, thus depending on where the apartment is built, rent will change.





Higher electricity costs are generally associated with higher rent. Perhaps electricity is included in places with higher rent and that is why there are higher costs.

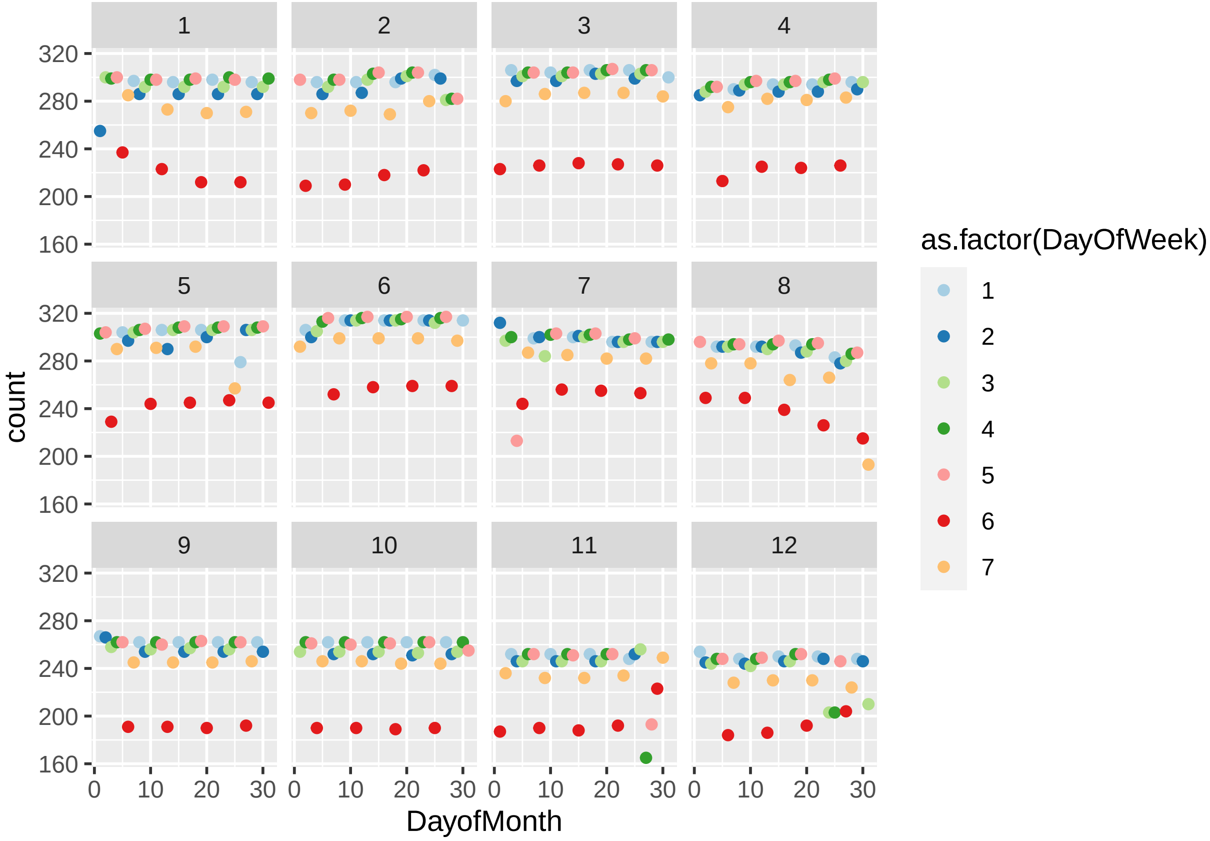
ABIA Visualizations

Our goal for these visualizations was to see when Austinites should to travel to reduce weather delays.

Read in ABIA

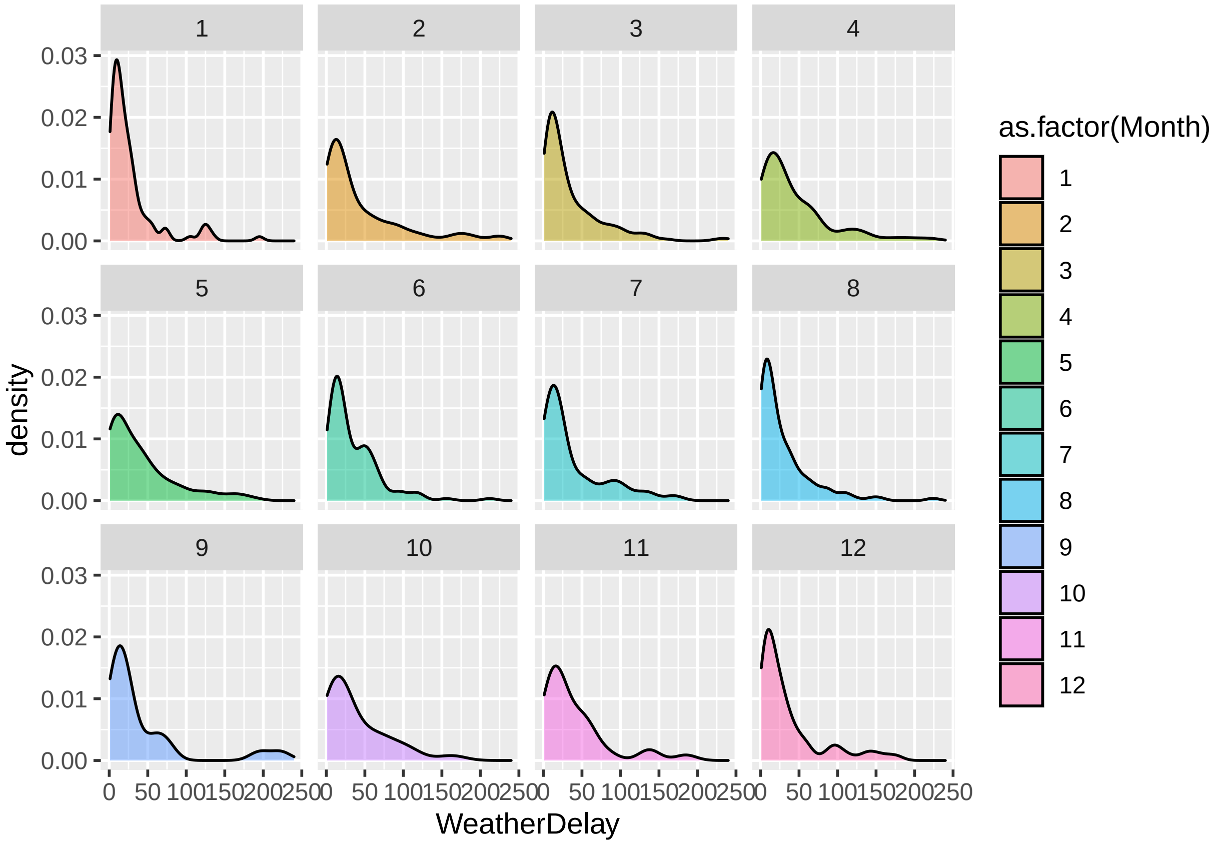
```
## Rows: 99,260
## Columns: 29
## $ Year              <int> 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008~
## $ Month             <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ DayofMonth        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ DayOfWeek         <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ DepTime           <int> 120, 555, 600, 601, 601, 636, 646, 650, 650, 654, 71~
## $ CRSDepTime        <int> 1935, 600, 600, 605, 600, 645, 655, 700, 650, 700, 7~
## $ ArrTime           <int> 309, 826, 728, 727, 654, 934, 735, 841, 1139, 1117, ~
## $ CRSArrTime        <int> 2130, 835, 729, 750, 700, 932, 750, 857, 1145, 1133,~
## $ UniqueCarrier     <chr> "9E", "AA", "YV", "9E", "AA", "NW", "CO", "XE", "AA"~
## $ FlightNum         <int> 5746, 1614, 2883, 5743, 1157, 1674, 340, 541, 1182, ~
## $ TailNum           <chr> "84129E", "N438AA", "N922FJ", "89189E", "N4XAAA", "N~
## $ ActualElapsedTime <int> 109, 151, 148, 86, 53, 178, 49, 111, 169, 203, 53, 7~
## $ CRSElapsedTime    <int> 115, 155, 149, 105, 60, 167, 55, 117, 175, 213, 60, ~
## $ AirTime           <int> 88, 133, 125, 70, 38, 145, 28, 94, 153, 177, 36, 56,~
## $ ArrDelay          <int> 339, -9, -1, -23, -6, 2, -15, -16, -6, -16, 0, -6, -~
## $ DepDelay          <int> 345, -5, 0, -4, 1, -9, -9, -10, 0, -6, 7, 0, -4, -5,~
## $ Origin            <chr> "MEM", "AUS", "AUS", "AUS", "AUS", "AUS", "AUS", "MC~
## $ Dest              <chr> "AUS", "ORD", "PHX", "MEM", "DFW", "MSP", "IAH", "AU~
## $ Distance          <int> 559, 978, 872, 559, 190, 1042, 140, 650, 1242, 1522,~
## $ TaxiIn            <int> 3, 7, 7, 4, 5, 11, 6, 6, 4, 13, 6, 5, 13, 4, 15, 13,~
## $ TaxiOut           <int> 18, 11, 16, 12, 10, 22, 15, 11, 12, 13, 11, 10, 16, ~
## $ Cancelled         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ CancellationCode  <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", ~
## $ Diverted          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ CarrierDelay      <int> 339, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ WeatherDelay      <int> 0, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ NASDelay          <int> 0, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ SecurityDelay     <int> 0, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ LateAircraftDelay <int> 0, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

First we wanted to see the trend for each day of the year so we plotted the count of flights out of AUS, faceted by month and color by Day of Week



Saturdays generally have much lower numbers of flights. Holidays have much lower number of flights as well (Christmas, Fourth of July, Labor Day Weekend, Thanksgiving).

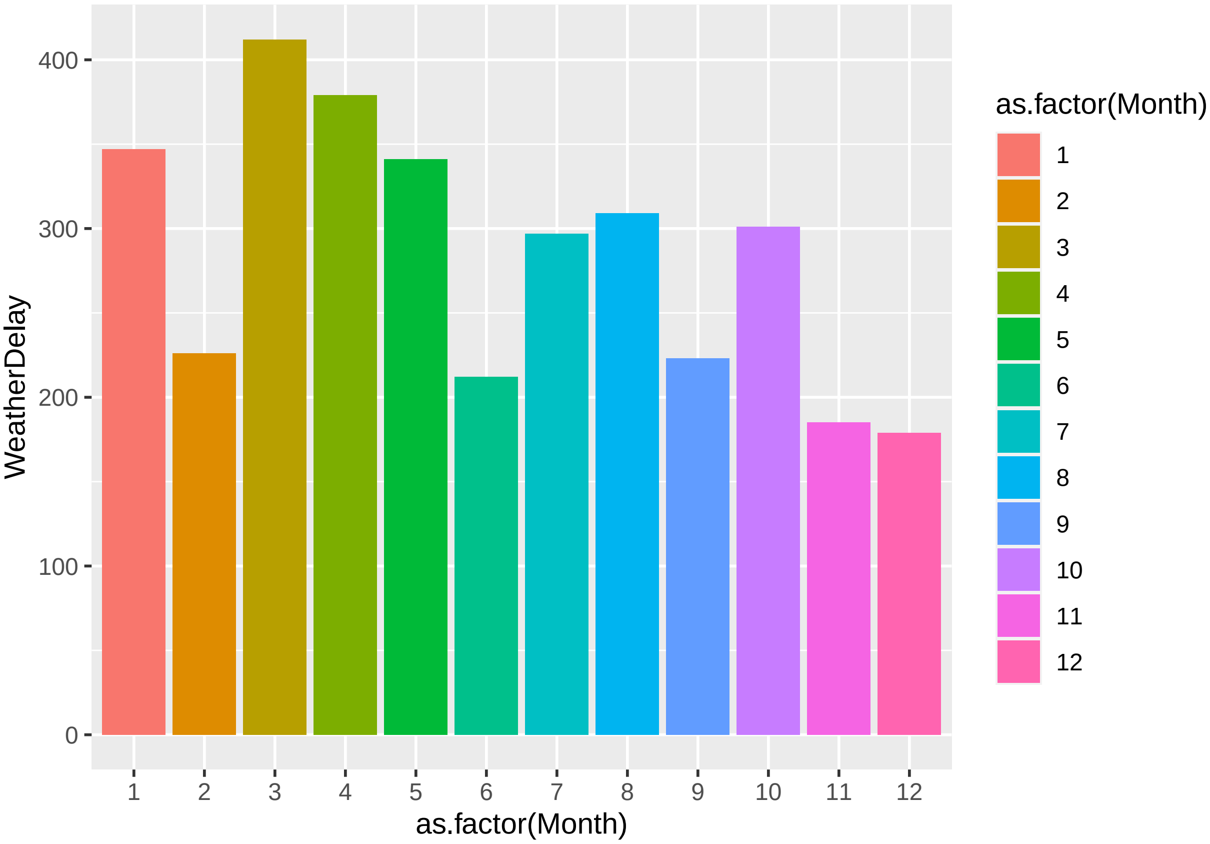
Then we wanted to explore weather delay trends for the year, so we did a density plot of weather delay, faceting



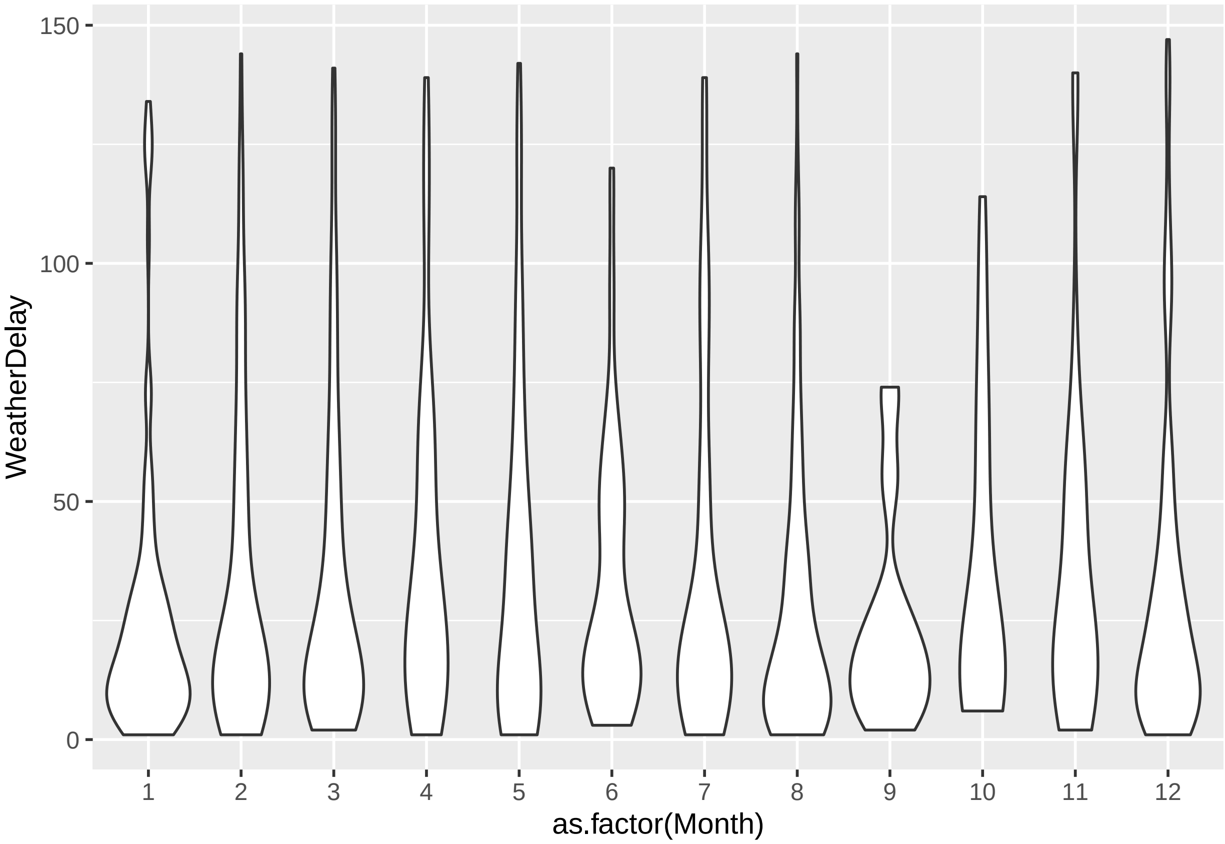
on month

We can see that January is a particularly bad month for weather delays, and that September has quite a few longer weather delays (perhaps specific to that year?).

For a clearer picture, we took the total count of weather delays per month



Here we see that March and April (rainy season) have the highest number of total weather delays. We can see this reflected in the density graph above, with March and April having the largest area under the curve.



This violin plot helps show where the highest number of weather delays are. March and April don't have the widest violins, but the have a large range with lots of delays across the range. January has a large amount of small delays.

If traveling in January, March, or April, plan to have a possible weather delay. If wanting to avoid weather delays, the best time to travel is February, June, September, November, and December.

Portfolio Management

First portfolio - diversified industries including energy, financial, construction, and transportation

```
## [1] "ITB" "OIH" "IYT" "KRE"
```

Adjust for splits and dividends

Portfolio 1 - Randomly selected 5 ETFs in Asia Pacific

```
##               ClCl.ITBa   ClCl.OIHa    ClCl.IYTa     ClCl.KREa
## 2016-01-04           NA          NA           NA            NA
## 2016-01-05  0.002662609 -0.01722204  0.001212398 -0.0009782588
## 2016-01-06 -0.023141160 -0.04533331 -0.019906071 -0.0122399027
## 2016-01-07 -0.041553398 -0.02913010 -0.030813228 -0.0302353671
## 2016-01-08 -0.022285292 -0.02137278 -0.007330661 -0.0204446455
## 2016-01-11  0.005801948 -0.02057963 -0.003692398 -0.0020871901
```

```
## [1] 114049.6
```

Second portfolio - Health & Biotech ETFs with highest percentage increase based on YTD. This is considered an aggressive portfolio

```
## [1] "BBH"  "HELX" "CHNA" "IDNA" "RYH"
```

Adjust for splits and dividends

All returns as a matrix

total wealth

```
## [1] 113273.5
```

Third portfolio - Mix of mid and large cap growth ETFs

```
##  [1] "XMMO" "SCHM" "SFYX" "VTI"  "ITOT" "SCHG" "GSLC" "MDYG" "IVOG" "DIA"
```

Adjust for splits and dividends

total wealth

```
## [1] 97285.58
```

```
## Portfolio 1 value at risk 5%:  13169.24
```

```
## Portfolio 2 value at risk 5%:  9396.382
```

```
## Portfolio 3 value at risk 5%:  10097.12
```

I tried to pick a few different types of portfolios to see the change in VaR. The first portfolio includes ETfs from 4 different industries including transportation, construction, financial, and energy. This diverse portfolio returned the highest 5% value at risk with an expected loss of ~14%. Portfolio 2 was the most aggressive portfolio with Health & Biotech ETFs that had the highest percentage increase YTD. This portfolio had the lowest 5% VaR with an expected loss of about ~9.5%. The third portfolio included a mix of large and mid cap growth ETFs and returned an expected loss of ~10.5% for the 5% VaR.

Based on these VaR results, we can assume having a very diverse portfolio leads to more risk and having a portfolio with high growth ETFs will minimize risk. Of course, we would have to try out many other portfolios to fully understand high and low risk portfolios.

Author attribution

In order to predict the author of an article on the basis of the article's textual content, we took several steps to clean the data and split it into a training and test set. After all the data cleaning was complete, we created a random forest model. Our data cleaning steps and final conclusions are outlined below.

Data cleaning Outline:

1. Read the file information for the training and test set using a readerplain function that allowed us to read the plain text documents in english.
2. Created for loops for both the train and test data that separated the strings at "/" and created lists for the author names and file names for both the training and the test set.
3. Used a name function to add the names of the authors to the author list and add ".txt" back to each file name in the file name list.
4. Created a corpus for both the training and the test set and
5. Transformed both corpuses to make everything lowercase, remove numbers, remove punctuation, excess white space, and stopwords.
6. Converted the transformed data into a sparse matrix and removed sparse words.
7. Converted both the training matrix and test matrix into two new data frames and removed columns that were not common to both the test and the training set.This ensured that all the words in test set also appear in the training set.

Random Forest Modeling:

After the data cleaning and pre-processing were completed, our team had a usable data frame for both the training and test data. We ran a random forest model with 50 trees to predict the accuracy of predicting the author of an article based on the article's textual context. Our final prediction accuracy was 57.52%.

Accuracy Result

```
## [1] 0.5752
```

We have a 57.53% accuracy using random forests.

Association Rule Mining

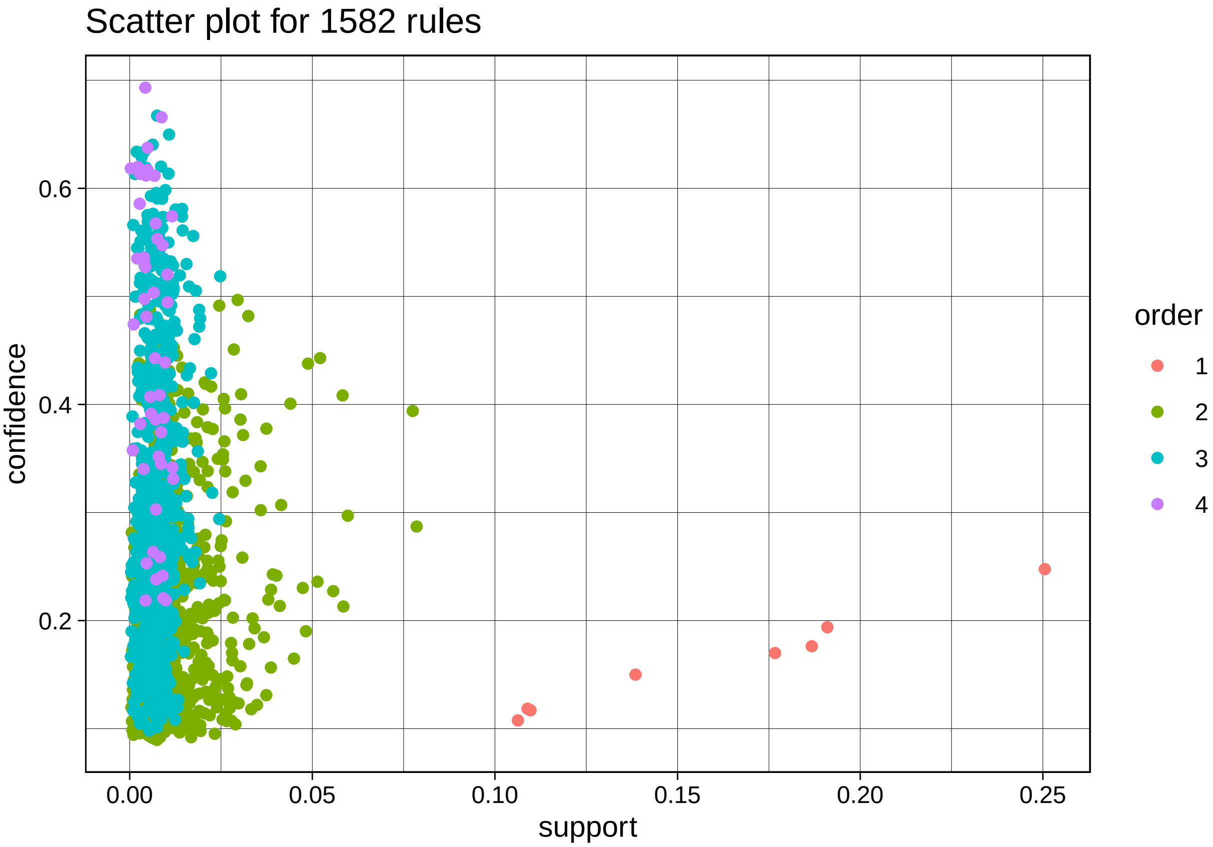
First we separated the data out into a format that the code could easily read. Then we put parameters on the data and are only looking at the data that has at more than a confidence of 0.005 and a support of 0.1.

The pertinent information from the summery of this subset from the groceries.txt file is that in all there are 1582 rules.

```
## 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
##       5  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 ... [1582 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
## set of 1582 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3   4
##   8 755 771  48
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.543   3.000   4.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.005084   Min.   :0.1000   Min.   :0.008134   Min.   :0.4457
##  1st Qu.:0.005897   1st Qu.:0.1454   1st Qu.:0.022979   1st Qu.:1.4732
##  Median :0.007321   Median :0.2189   Median :0.037112   Median :1.8174
##  Mean   :0.010537   Mean   :0.2557   Mean   :0.053306   Mean   :1.9028
##  3rd Qu.:0.010371   3rd Qu.:0.3315   3rd Qu.:0.058058   3rd Qu.:2.2444
##  Max.   :0.255516   Max.   :0.7000   Max.   :1.000000   Max.   :4.6399
##      count
##  Min.   :  50.0
##  1st Qu.:  58.0
##  Median :  72.0
##  Mean   : 103.6
##  3rd Qu.: 102.0
##  Max.   :2513.0
##
## mining info:
##       data ntransactions support confidence
##  groceries          9835   0.005        0.1
```

This is a plots of the rules where the groceries data is grouped off into 4 orders.



These are the rules where the support is larger than 0.05.

```
##      lhs              rhs                support    confidence coverage
## [1]  {}            => {bottled water}    0.11052364 0.1105236  1.0000000
## [2]  {}            => {tropical fruit}   0.10493137 0.1049314  1.0000000
## [3]  {}            => {root vegetables}  0.10899847 0.1089985  1.0000000
## [4]  {}            => {soda}             0.17437722 0.1743772  1.0000000
## [5]  {}            => {yogurt}           0.13950178 0.1395018  1.0000000
## [6]  {}            => {rolls/buns}       0.18393493 0.1839349  1.0000000
## [7]  {}            => {other vegetables} 0.19349263 0.1934926  1.0000000
## [8]  {}            => {whole milk}       0.25551601 0.2555160  1.0000000
## [9]  {yogurt}      => {whole milk}       0.05602440 0.4016035  0.1395018
## [10] {whole milk}  => {yogurt}           0.05602440 0.2192598  0.2555160
## [11] {rolls/buns}  => {whole milk}       0.05663447 0.3079049  0.1839349
## [12] {whole milk}  => {rolls/buns}       0.05663447 0.2216474  0.2555160
## [13] {other vegetables} => {whole milk}       0.07483477 0.3867578  0.1934926
## [14] {whole milk}       => {other vegetables} 0.07483477 0.2928770  0.2555160
##      lift     count
## [1]  1.000000 1087
## [2]  1.000000 1032
## [3]  1.000000 1072
## [4]  1.000000 1715
## [5]  1.000000 1372
## [6]  1.000000 1809
## [7]  1.000000 1903
## [8]  1.000000 2513
## [9]  1.571735  551
## [10] 1.571735  551
## [11] 1.205032  557
## [12] 1.205032  557
## [13] 1.513634  736
## [14] 1.513634  736
```

You can see from these rules which items are the top 8 grocery items bought on there own. Each item bought at least 1000 times a piece. (There are 9835 entrys in the data).

These are the rules where the confidence is larger than 0.6. While it might look overwhelming due to formatting issues, the reason this is being included is to point out the rhs column.

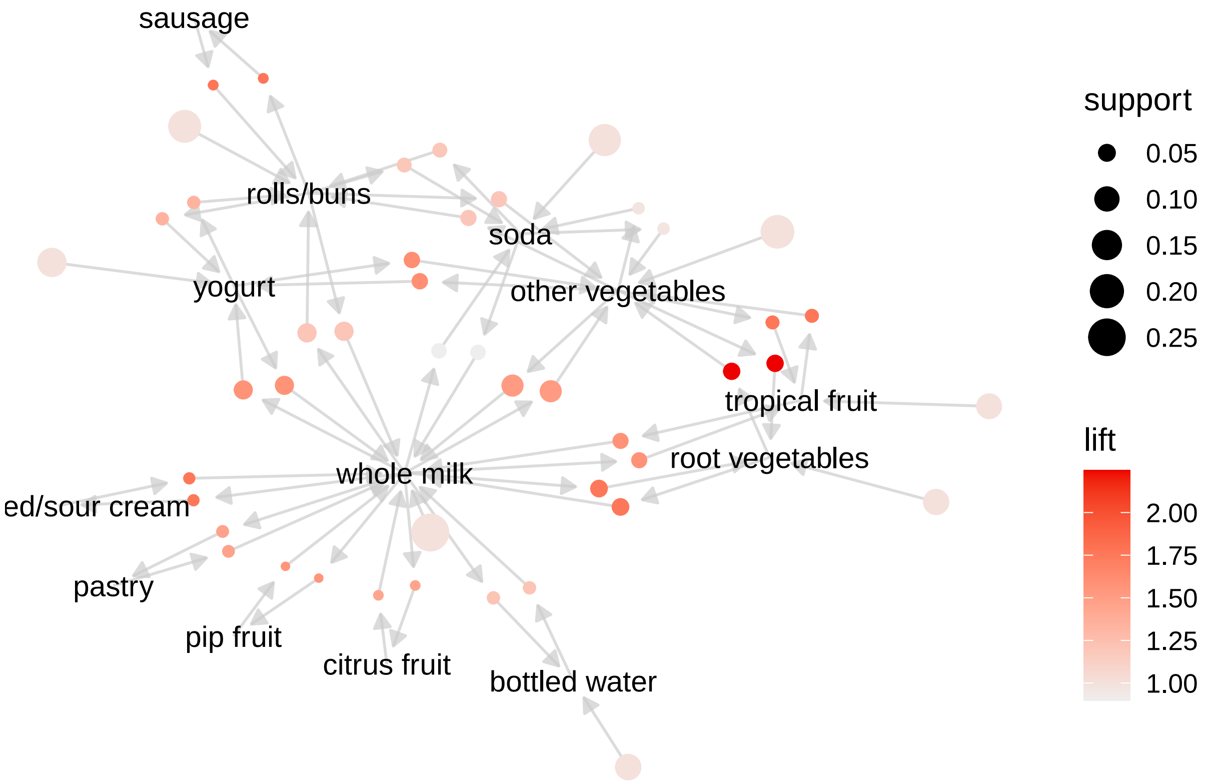
```
##      lhs                           rhs                     support confidence    coverage     lift count
## [1]  {onions,
##       root vegetables}       => {other vegetables} 0.005693950  0.6021505 0.009456024 3.112008    56
## [2]  {curd,
##       tropical fruit}        => {whole milk}       0.006507372  0.6336634 0.010269446 2.479936    64
## [3]  {domestic eggs,
##       margarine}             => {whole milk}       0.005185562  0.6219512 0.008337570 2.434099    51
## [4]  {butter,
##       domestic eggs}         => {whole milk}       0.005998983  0.6210526 0.009659380 2.430582    59
## [5]  {butter,
##       whipped/sour cream}    => {whole milk}       0.006710727  0.6600000 0.010167768 2.583008    66
## [6]  {bottled water,
##       butter}                => {whole milk}       0.005388917  0.6022727 0.008947636 2.357084    53
## [7]  {butter,
##       tropical fruit}        => {whole milk}       0.006202339  0.6224490 0.009964413 2.436047    61
## [8]  {butter,
##       root vegetables}       => {whole milk}       0.008235892  0.6377953 0.012913066 2.496107    81
## [9]  {butter,
##       yogurt}                => {whole milk}       0.009354347  0.6388889 0.014641586 2.500387    92
## [10] {domestic eggs,
##       pip fruit}             => {whole milk}       0.005388917  0.6235294 0.008642603 2.440275    53
## [11] {domestic eggs,
##       tropical fruit}        => {whole milk}       0.006914082  0.6071429 0.011387900 2.376144    68
## [12] {pip fruit,
##       whipped/sour cream}    => {other vegetables} 0.005592272  0.6043956 0.009252669 3.123610    55
## [13] {pip fruit,
##       whipped/sour cream}    => {whole milk}       0.005998983  0.6483516 0.009252669 2.537421    59
## [14] {fruit/vegetable juice,
##       other vegetables,
##       yogurt}                => {whole milk}       0.005083884  0.6172840 0.008235892 2.415833    50
## [15] {other vegetables,
##       root vegetables,
##       whipped/sour cream}    => {whole milk}       0.005185562  0.6071429 0.008540925 2.376144    51
## [16] {other vegetables,
##       pip fruit,
##       root vegetables}       => {whole milk}       0.005490595  0.6750000 0.008134215 2.641713    54
## [17] {pip fruit,
##       root vegetables,
##       whole milk}            => {other vegetables} 0.005490595  0.6136364 0.008947636 3.171368    54
## [18] {other vegetables,
##       pip fruit,
##       yogurt}                => {whole milk}       0.005083884  0.6250000 0.008134215 2.446031    50
## [19] {citrus fruit,
##       root vegetables,
##       whole milk}            => {other vegetables} 0.005795628  0.6333333 0.009150991 3.273165    57
## [20] {root vegetables,
##       tropical fruit,
##       yogurt}                => {whole milk}       0.005693950  0.7000000 0.008134215 2.739554    56
## [21] {other vegetables,
##       tropical fruit,
##       yogurt}                => {whole milk}       0.007625826  0.6198347 0.012302999 2.425816    75
## [22] {other vegetables,
##       root vegetables,
##       yogurt}                => {whole milk}       0.007829181  0.6062992 0.012913066 2.372842    77
```

These rules show how whole milk is bought with just about everything, as well other vegetables are commonly bought with a wide variety of other items.

This first plot is considering all rules where the confidence and the support are greater than 0.03.

```
## set of 46 rules
##
## rule length distribution (lhs + rhs):sizes
##  1  2
##  8 38
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   1.826   2.000   2.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.03010   Min.   :0.1049   Min.   :0.07168   Min.   :0.8991
##  1st Qu.:0.03353   1st Qu.:0.1671   1st Qu.:0.13950   1st Qu.:1.0488
##  Median :0.04230   Median :0.2200   Median :0.19349   Median :1.4424
##  Mean   :0.06175   Mean   :0.2420   Mean   :0.32140   Mean   :1.3938
##  3rd Qu.:0.05648   3rd Qu.:0.3112   3rd Qu.:0.25552   3rd Qu.:1.6007
##  Max.   :0.25552   Max.   :0.4496   Max.   :1.00000   Max.   :2.2466
##      count
##  Min.   : 296.0
##  1st Qu.: 329.8
##  Median : 416.0
##  Mean   : 607.3
##  3rd Qu.: 555.5
##  Max.   :2513.0
##
## mining info:
```

```
##       data ntransactions support confidence
##  groceries           9835   0.005        0.1
```



Following it up we have this graph used in gephi to break it up into 7 orders (each shown as a different color) and how the grocery items are connected.

This second group of plots is when there is a more stict set of rules on the data. Here the confidence has tp be larger than 0.3 and the support has to be larger than 0.03.

```
## set of 14 rules
##
## rule length distribution (lhs + rhs):sizes
##  2
## 14
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       2       2       2       2       2       2
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.03010   Min.   :0.3079   Min.   :0.07168   Min.   :1.205
##  1st Qu.:0.03249   1st Qu.:0.3298   1st Qu.:0.09021   1st Qu.:1.475
##  Median :0.03910   Median :0.3802   Median :0.10696   Median :1.575
##  Mean   :0.04260   Mean   :0.3759   Mean   :0.11484   Mean   :1.604
##  3rd Qu.:0.04853   3rd Qu.:0.4027   3rd Qu.:0.13226   3rd Qu.:1.759
##  Max.   :0.07483   Max.   :0.4496   Max.   :0.19349   Max.   :2.247
##      count
##  Min.   :296.0
##  1st Qu.:319.5
##  Median :384.5
##  Mean   :419.0
```

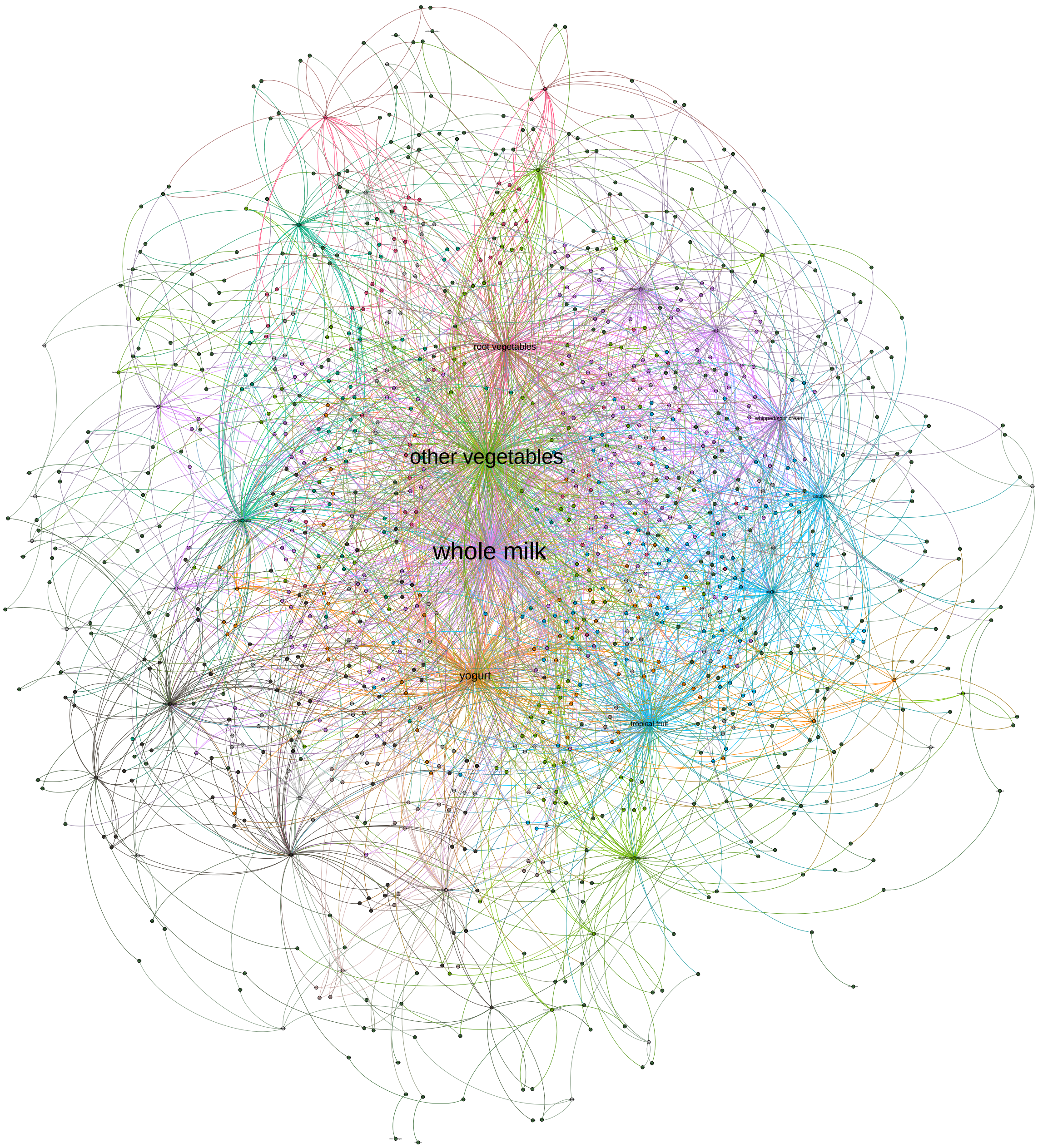
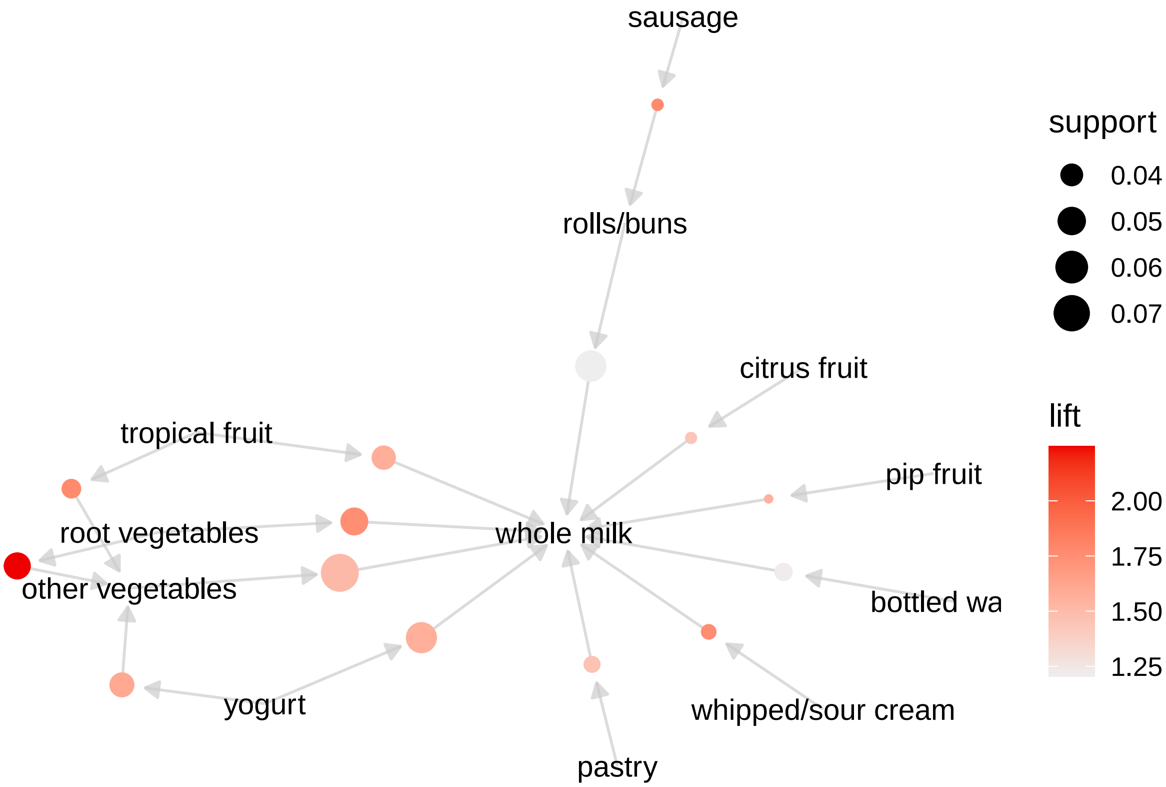


Figure 1: Support > 0.03, Confidence > 0.03

```
##  3rd Qu.:477.2
##  Max.   :736.0
##
## mining info:
##       data ntransactions support confidence
##  groceries          9835   0.005        0.1
```



Once again, the following graph created in gephi breaks up the grocery items into 7 orders (each shown as a different color) and how they items are connected to each other.

As we can see this is a much simpler visual of the data and both versions of the plot can help us easily determine certain things. Such as the first shows some of the other items commonly bought with groceries such as pasties, fruit, and bottle water. While in the second plot it is easier to what items are more connected to items in other groups, and how connected things like vegtables and whole milk are to the rest of the information.

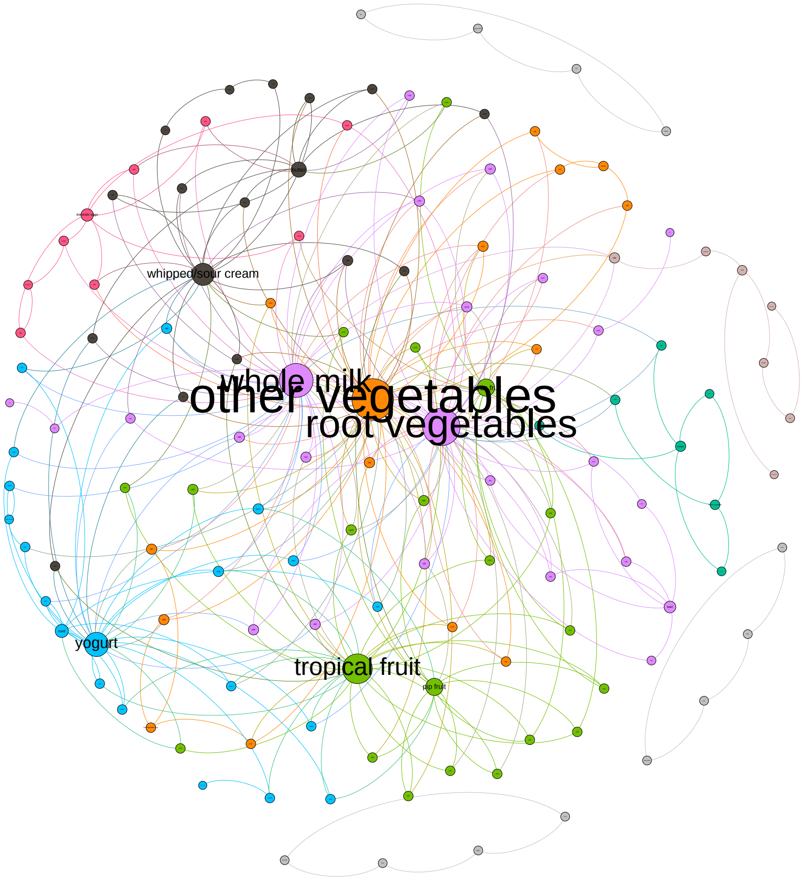


Figure 2: Support > 0.03, Confidence > 0.3