Exercise 2

Clarissa Franklin, Yannick Heard, & Paige Mckenzie

August 13, 2017

Question 1: Flights at ABIA

First, we read in the raw file directly from github and import the ggplot2 library

We can look at the different variables available to us is this data set. Remove cancelled flights. Separate into flights departing from Austin and flights arriving in Austin.

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay"
```

This data set contains flights into and out of Austin. Let’s create a column for Weekday name to make the data more understandable, then We can separate this data into two subsets: flights arriving into Austin, and flights departing from Austin.

We can take a look at the summary statistics to start to understand our data.

```
## Summary statistics: flights departing from Austin
```

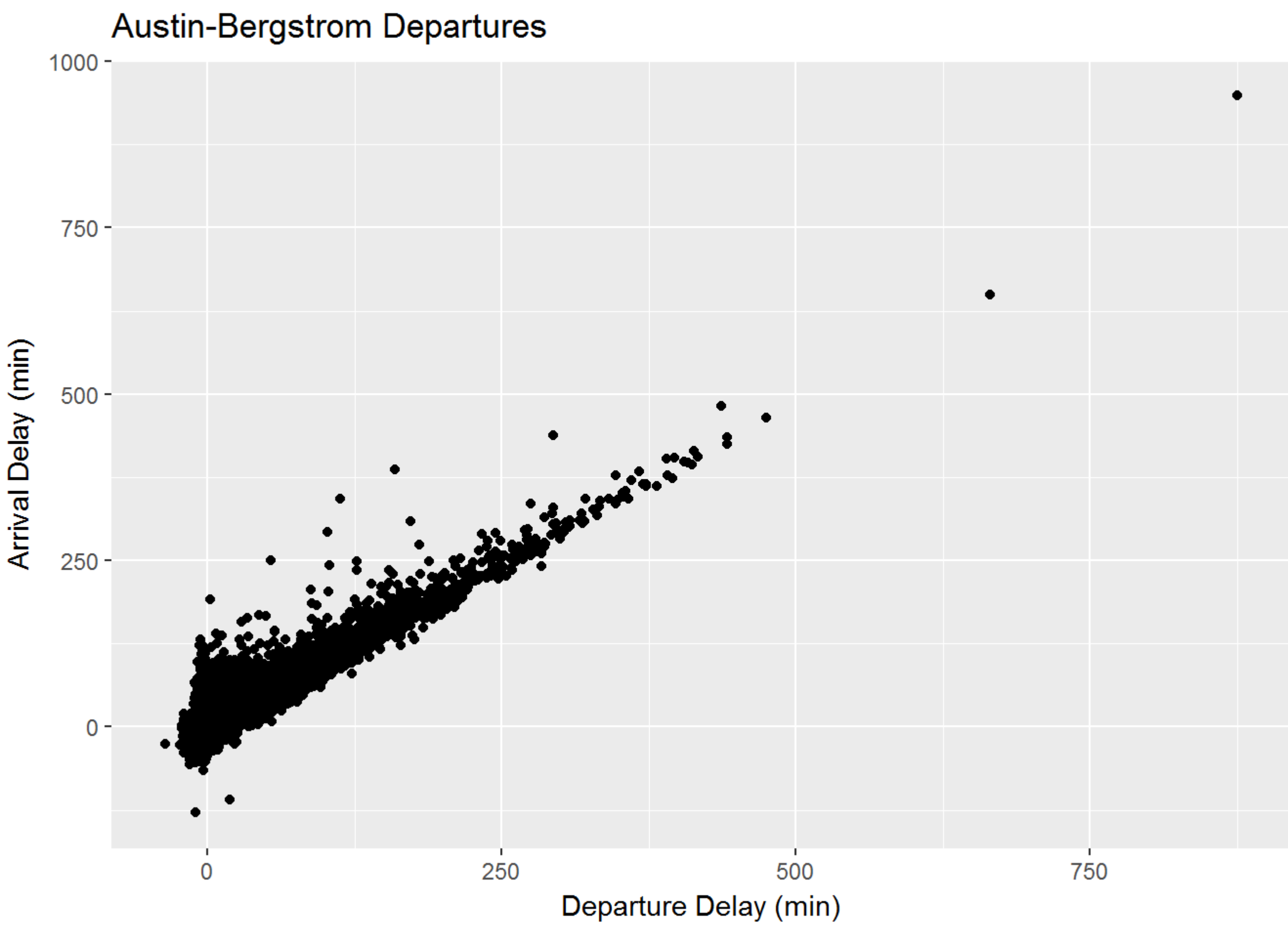
```
##       Year          Month          DayofMonth      DayOfWeek
##  Min.   :2008   Min.   : 1.000   Min.   : 1.00   Min.   :1.000
##  1st Qu.:2008   1st Qu.: 3.000   1st Qu.: 8.00   1st Qu.:2.000
##  Median :2008   Median : 6.000   Median :16.00   Median :4.000
##  Mean   :2008   Mean   : 6.305   Mean   :15.74   Mean   :3.906
##  3rd Qu.:2008   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:6.000
##  Max.   :2008   Max.   :12.000   Max.   :31.00   Max.   :7.000
##
##     DepTime       CRSDepTime      ArrTime       CRSArrTime
##  Min.   :   1   Min.   :  55   Min.   :   1   Min.   : 542
##  1st Qu.: 828   1st Qu.: 825   1st Qu.:1013   1st Qu.:1014
##  Median :1232   Median :1220   Median :1450   Median :1440
##  Mean   :1257   Mean   :1248   Mean   :1430   Mean   :1426
##  3rd Qu.:1641   3rd Qu.:1630   3rd Qu.:1830   3rd Qu.:1820
##  Max.   :2343   Max.   :2200   Max.   :2359   Max.   :2400
##                                NA's   :82
##  UniqueCarrier     FlightNum       TailNum      ActualElapsedTime
##  WN     :17343   Min.   :   1   N678CA :   97   Min.   : 22.0
##  AA     : 9709   1st Qu.: 639   N511SW :   90   1st Qu.: 60.0
##  CO     : 4554   Median :1464   N526SW :   88   Median :127.0
##  YV     : 2455   Mean   :1898   N528SW :   86   Mean   :121.2
##  B6     : 2367   3rd Qu.:2614   N520SW :   84   3rd Qu.:165.0
##  XE     : 2296   Max.   :9741   N501SW :   82   Max.   :427.0
##  (Other):10167                  (Other):48364   NA's   :95
##  CRSElapsedTime     AirTime         ArrDelay           DepDelay
##  Min.   : 37.0   Min.   :  7.0   Min.   :-129.000   Min.   :-36.000
##  1st Qu.: 60.0   1st Qu.: 40.0   1st Qu.:  -9.000   1st Qu.: -5.000
##  Median :130.0   Median :107.0   Median :  -2.000   Median : -1.000
##  Mean   :122.6   Mean   :101.3   Mean   :   6.037   Mean   :  7.423
##  3rd Qu.:165.0   3rd Qu.:143.0   3rd Qu.:   9.000   3rd Qu.:  5.000
##  Max.   :315.0   Max.   :286.0   Max.   : 948.000   Max.   :875.000
##  NA's   :5       NA's   :95      NA's   :95
##      Origin           Dest           Distance        TaxiIn
##  AUS    :48891   DAL    : 5449   Min.   : 140   Min.   :  0.000
##  ABQ    :    0   DFW    : 5350   1st Qu.: 190   1st Qu.:  4.000
##  ATL    :    0   IAH    : 3637   Median : 775   Median :  6.000
##  BHM    :    0   PHX    : 2768   Mean   : 707   Mean   :  7.548
##  BNA    :    0   DEN    : 2659   3rd Qu.:1085   3rd Qu.:  9.000
##  BOS    :    0   ORD    : 2421   Max.   :1770   Max.   :143.000
##  (Other):    0   (Other):26607                  NA's   :82
##     TaxiOut         Cancelled CancellationCode    Diverted
##  Min.   :  1.00   Min.   :0    :48891          Min.   :0.000000
##  1st Qu.:  9.00   1st Qu.:0   A:    0          1st Qu.:0.000000
##  Median : 11.00   Median :0   B:    0          Median :0.000000
##  Mean   : 12.44   Mean   :0   C:    0          Mean   :0.001943
##  3rd Qu.: 14.00   3rd Qu.:0                    3rd Qu.:0.000000
##  Max.   :209.00   Max.   :0                    Max.   :1.000000
##
##   CarrierDelay     WeatherDelay       NASDelay     SecurityDelay
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.0   Min.   :  0.00
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.0   1st Qu.:  0.00
##  Median :  0.00   Median :  0.00   Median :  5.0   Median :  0.00
##  Mean   : 12.13   Mean   :  1.87   Mean   : 16.3   Mean   :  0.04
```

```
##  3rd Qu.:  8.00   3rd Qu.:  0.00   3rd Qu.: 19.0   3rd Qu.:  0.00
##  Max.   :875.00   Max.   :412.00   Max.   :354.0   Max.   :102.00
##  NA's   :39887    NA's   :39887    NA's   :39887   NA's   :39887
##  LateAircraftDelay   MonthName
##  Min.   :  0.0     June   : 4488
##  1st Qu.:  0.0     May    : 4444
##  Median :  8.0     July   : 4417
##  Mean   : 22.4     March  : 4350
##  3rd Qu.: 29.0     January: 4289
##  Max.   :437.0     August : 4226
##  NA's   :39887     (Other):22677
```

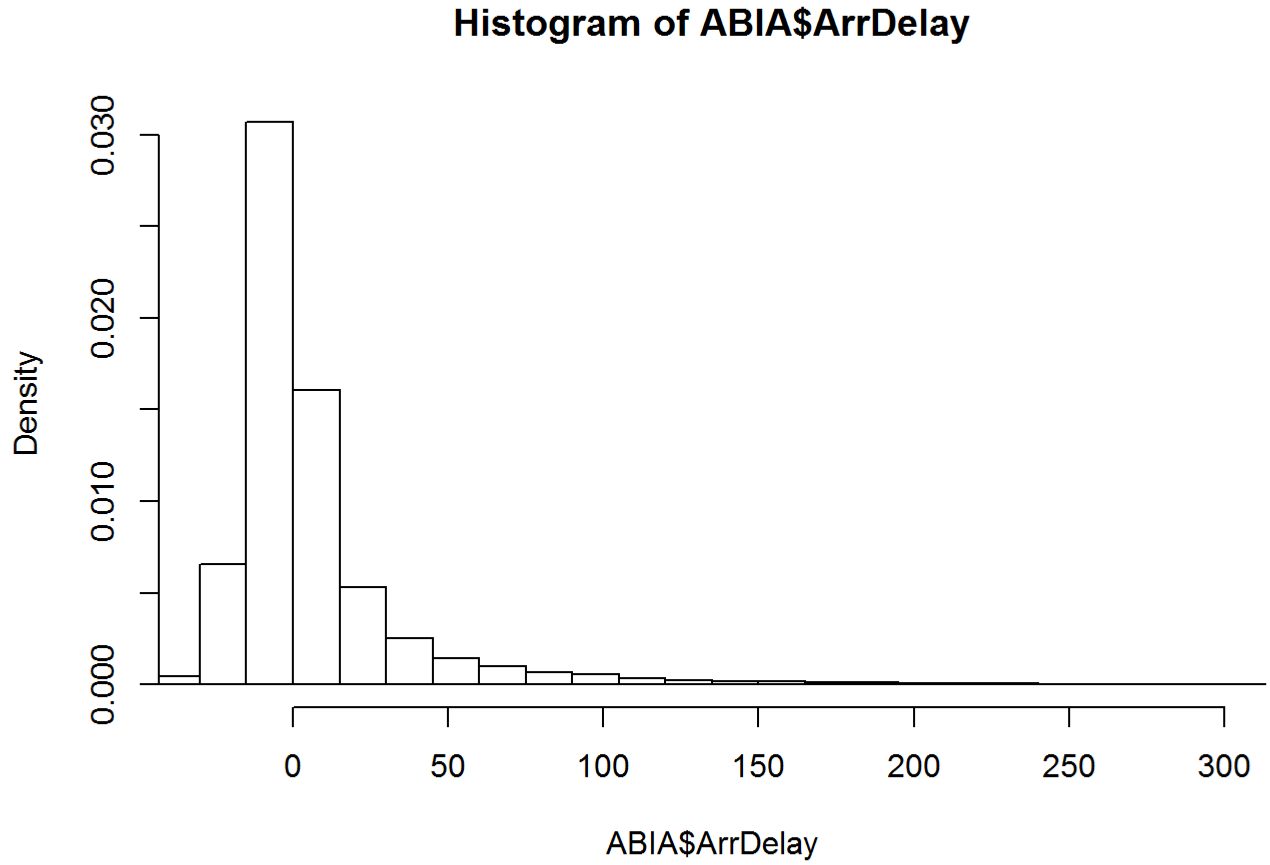
```
##
##
## Summary statistics: flights arriving in Austin
```

```
##       Year          Month          DayofMonth      DayOfWeek
##  Min.   :2008   Min.   : 1.000   Min.   : 1.00   Min.   :1.000
##  1st Qu.:2008   1st Qu.: 3.000   1st Qu.: 8.00   1st Qu.:2.000
##  Median :2008   Median : 6.000   Median :16.00   Median :4.000
##  Mean   :2008   Mean   : 6.304   Mean   :15.74   Mean   :3.904
##  3rd Qu.:2008   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:6.000
##  Max.   :2008   Max.   :12.000   Max.   :31.00   Max.   :7.000
##
##     DepTime       CRSDepTime      ArrTime       CRSArrTime
##  Min.   :   1   Min.   : 545   Min.   :   1   Min.   :   5
##  1st Qu.:1001   1st Qu.:1000   1st Qu.:1153   1st Qu.:1220
##  Median :1404   Median :1355   Median :1601   Median :1615
##  Mean   :1400   Mean   :1390   Mean   :1544   Mean   :1583
##  3rd Qu.:1810   3rd Qu.:1800   3rd Qu.:1949   3rd Qu.:2010
##  Max.   :2400   Max.   :2346   Max.   :2400   Max.   :2359
##                                NA's   :65
##  UniqueCarrier     FlightNum       TailNum      ActualElapsedTime
##  WN     :17350   Min.   :   2   N678CA :   97   Min.   : 33.0
##  AA     : 9718   1st Qu.: 661   N511SW :   90   1st Qu.: 54.0
##  CO     : 4558   Median :1477   N526SW :   87   Median :123.0
##  YV     : 2475   Mean   :1926   N528SW :   86   Mean   :119.1
##  B6     : 2369   3rd Qu.:2653   N520SW :   84   3rd Qu.:163.0
##  XE     : 2293   Max.   :9741   N501SW :   82   Max.   :506.0
##  (Other):10186                  (Other):48423   NA's   :86
##  CRSElapsedTime    AirTime          ArrDelay           DepDelay
##  Min.   : 17    Min.   :  3.00   Min.   :-81.000   Min.   :-42.00
##  1st Qu.: 55    1st Qu.: 34.00   1st Qu.: -9.000   1st Qu.: -3.00
##  Median :127    Median :104.00   Median : -1.000   Median :  0.00
##  Mean   :122    Mean   : 98.36   Mean   :  8.091   Mean   : 10.91
##  3rd Qu.:165    3rd Qu.:140.00   3rd Qu.: 12.000   3rd Qu.: 10.00
##  Max.   :320    Max.   :402.00   Max.   :518.000   Max.   :509.00
##  NA's   :4      NA's   :86       NA's   :86
##      Origin           Dest           Distance          TaxiIn
##  DAL    : 5468   AUS    :48949   Min.   :  66.0   Min.   : 1.00
##  DFW    : 5349   ABQ    :    0   1st Qu.: 190.0   1st Qu.: 4.00
##  IAH    : 3653   ATL    :    0   Median : 775.0   Median : 5.00
##  PHX    : 2779   BNA    :    0   Mean   : 706.3   Mean   : 5.28
##  DEN    : 2712   BOS    :    0   3rd Qu.:1085.0   3rd Qu.: 6.00
##  ORD    : 2425   BWI    :    0   Max.   :1770.0   Max.   :90.00
##  (Other):26563   (Other):    0                    NA's   :65
##     TaxiOut          Cancelled CancellationCode    Diverted
##  Min.   :  1.00   Min.   :0    :48949          Min.   :0.000000
##  1st Qu.:  9.00   1st Qu.:0   A:    0          1st Qu.:0.000000
##  Median : 13.00   Median :0   B:    0          Median :0.000000
##  Mean   : 15.49   Mean   :0   C:    0          Mean   :0.001757
##  3rd Qu.: 18.00   3rd Qu.:0                    3rd Qu.:0.000000
##  Max.   :305.00   Max.   :0                    Max.   :1.000000
##
##   CarrierDelay     WeatherDelay       NASDelay      SecurityDelay
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.0
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.0
##  Median :  5.00   Median :  0.00   Median :  0.00   Median :  0.0
##  Mean   : 18.12   Mean   :  2.55   Mean   :  9.27   Mean   :  0.1
```

```
## 3rd Qu.: 21.00   3rd Qu.:  0.00   3rd Qu.: 13.00   3rd Qu.:  0.0
## Max.   :518.00   Max.   :379.00   Max.   :367.00   Max.   :199.0
## NA's   :38206    NA's   :38206    NA's   :38206    NA's   :38206
## LateAircraftDelay   MonthName
## Min.   :  0.00    June    : 4491
## 1st Qu.:  0.00    May     : 4462
## Median :  5.00    July    : 4424
## Mean   : 23.44    March   : 4349
## 3rd Qu.: 30.00    January: 4299
## Max.   :458.00    August : 4232
## NA's   :38206     (Other):22692
```

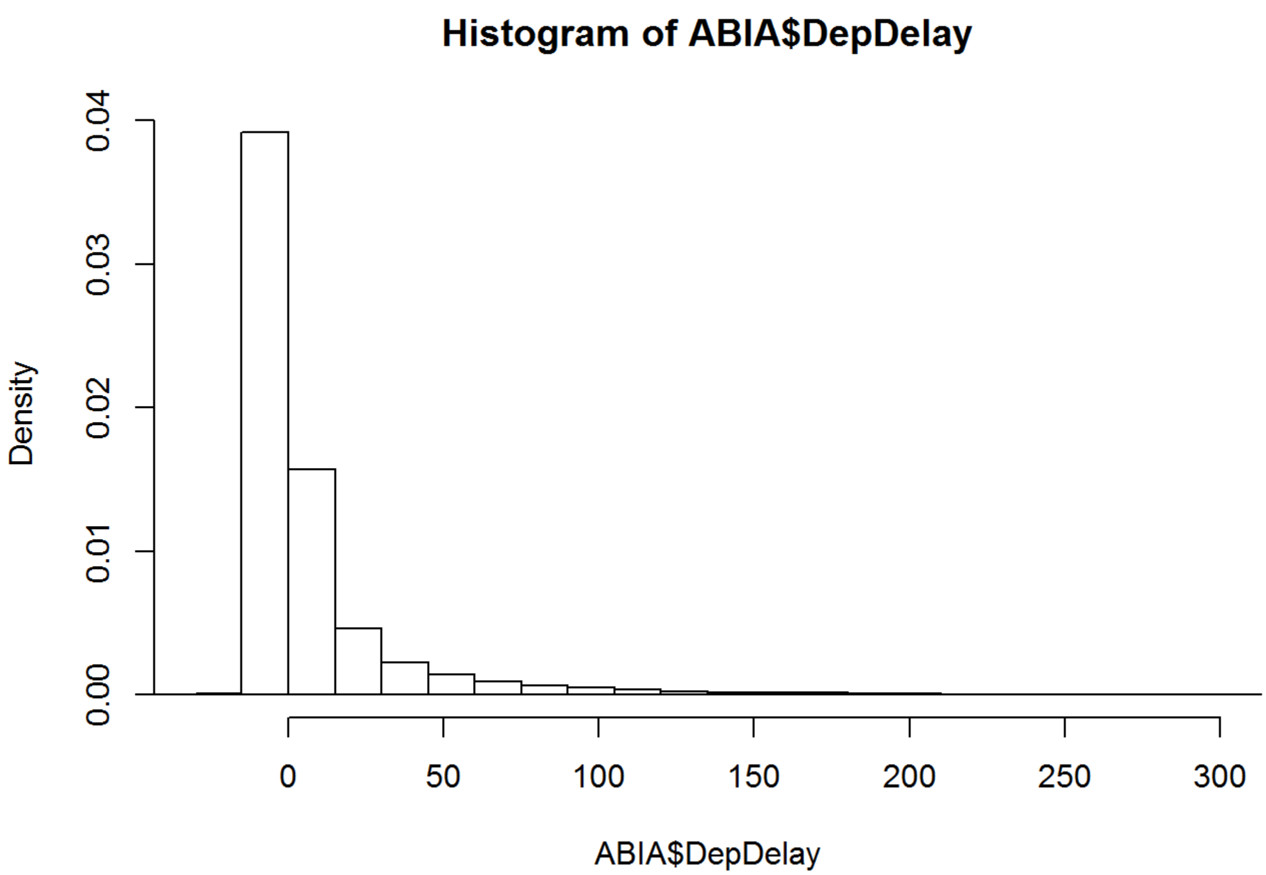


Create a historgram for average arrival and departure delays


Histogram of ABIA$ArrDelay
Density
0.000
0.010
0.020
0.030
0
50
100
150
200
250
300
ABIA$ArrDelay

```
## $breaks
##   [1] -180 -165 -150 -135 -120 -105  -90  -75  -60  -45  -30  -15    0   15
##  [15]   30   45   60   75   90  105  120  135  150  165  180  195  210  225
##  [29]  240  255  270  285  300  315  330  345  360  375  390  405  420  435
##  [43]  450  465  480  495  510  525  540  555  570  585  600  615  630  645
##  [57]  660  675  690  705  720  735  750  765  780  795  810  825  840  855
##  [71]  870  885  900  915  930  945  960  975  990 1005 1020 1035 1050 1065
##  [85] 1080 1095 1110 1125 1140 1155 1170 1185 1200 1215 1230 1245 1260 1275
##  [99] 1290 1305 1320 1335 1350 1365 1380 1395 1410 1425 1440 1455 1470 1485
## [113] 1500 1515 1530 1545 1560 1575 1590 1605 1620 1635 1650 1665 1680 1695
## [127] 1710 1725 1740 1755 1770 1785 1800
## 
## $counts
##   [1]     0     0     0     1     1     0     4    18    77   669  9616
##  [12] 44917 23486  7776  3673  2063  1425   962   793   482   356   278
##  [23]   217   186   156    99    72    70    46    45    31    32    22
##  [34]    16    18     8    11    10     6     3     4     1     2     3
##  [45]     1     0     1     0     0     0     0     0     0     0     0
##  [56]     1     0     0     0     0     0     0     0     0     0     0
##  [67]     0     0     0     0     0     0     0     0     0     1     0
##  [78]     0     0     0     0     0     0     0     0     0     0     0
##  [89]     0     0     0     0     0     0     0     0     0     0     0
## [100]     0     0     0     0     0     0     0     0     0     0     0
## [111]     0     0     0     0     0     0     0     0     0     0     0
## [122]     0     0     0     0     0     0     0     0     0     0     0
## 
## $density
##   [1] 0.000000e+00 0.000000e+00 0.000000e+00 6.826474e-07 6.826474e-07
##   [6] 0.000000e+00 2.730590e-06 1.228765e-05 5.256385e-05 4.566911e-04
##  [11] 6.564338e-03 3.066248e-02 1.603266e-02 5.308267e-03 2.507364e-03
##  [16] 1.408302e-03 9.727726e-04 6.567068e-04 5.413394e-04 3.290361e-04
##  [21] 2.430225e-04 1.897760e-04 1.481345e-04 1.269724e-04 1.064930e-04
##  [26] 6.758210e-05 4.915062e-05 4.778532e-05 3.140178e-05 3.071913e-05
##  [31] 2.116207e-05 2.184472e-05 1.501824e-05 1.092236e-05 1.228765e-05
##  [36] 5.461180e-06 7.509122e-06 6.826474e-06 4.095885e-06 2.047942e-06
##  [41] 2.730590e-06 6.826474e-07 1.365295e-06 2.047942e-06 6.826474e-07
##  [46] 0.000000e+00 6.826474e-07 0.000000e+00 0.000000e+00 0.000000e+00
##  [51] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [56] 6.826474e-07 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [61] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [66] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [71] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [76] 6.826474e-07 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [81] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [86] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [91] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [96] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [101] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [106] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [111] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [116] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [121] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [126] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

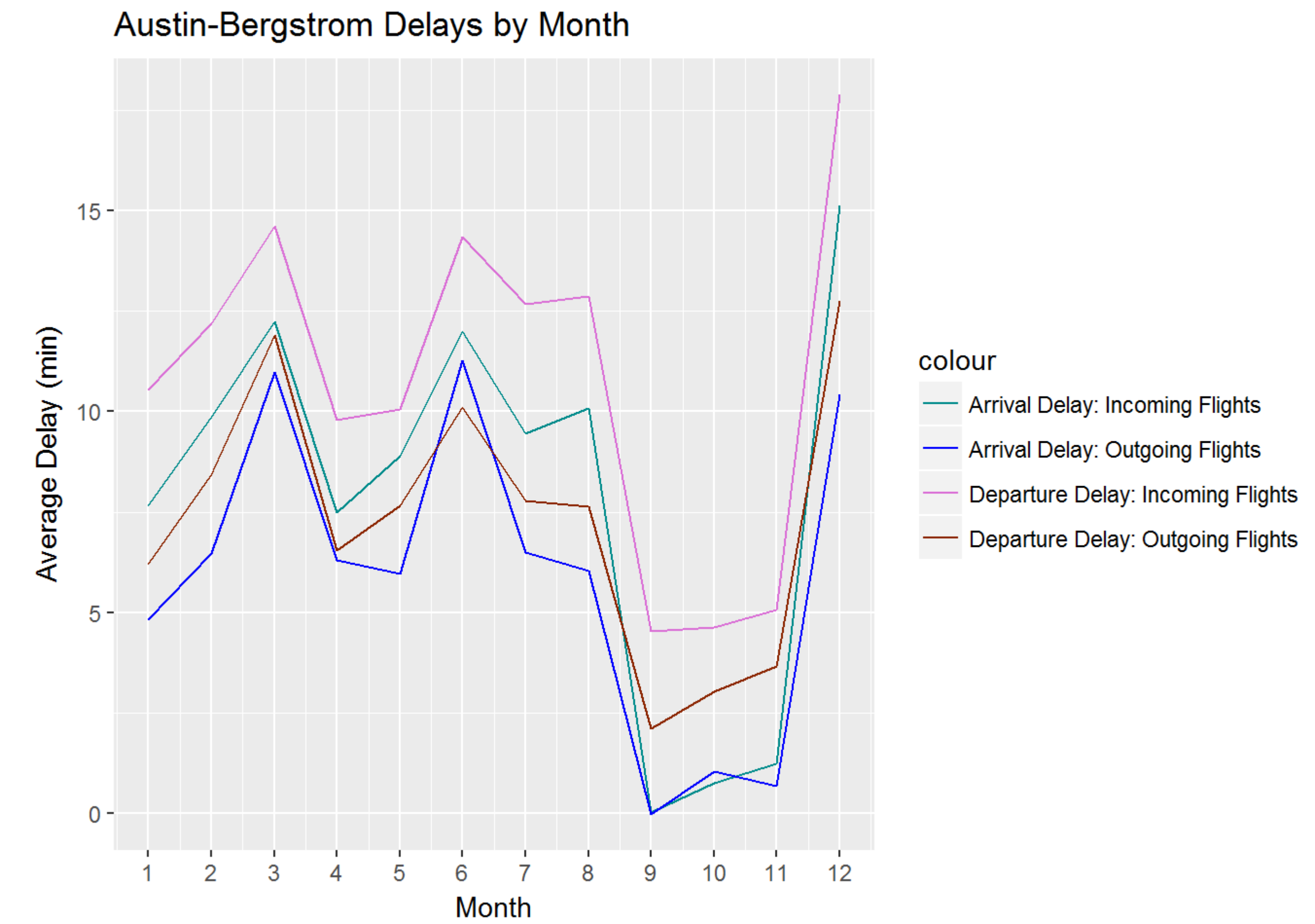
```
## [131] 0.000000e+00 0.000000e+00
##
## $mids
##   [1] -172.5 -157.5 -142.5 -127.5 -112.5  -97.5  -82.5  -67.5  -52.5  -37.5
##  [11]  -22.5   -7.5    7.5   22.5   37.5   52.5   67.5   82.5   97.5  112.5
##  [21]  127.5  142.5  157.5  172.5  187.5  202.5  217.5  232.5  247.5  262.5
##  [31]  277.5  292.5  307.5  322.5  337.5  352.5  367.5  382.5  397.5  412.5
##  [41]  427.5  442.5  457.5  472.5  487.5  502.5  517.5  532.5  547.5  562.5
##  [51]  577.5  592.5  607.5  622.5  637.5  652.5  667.5  682.5  697.5  712.5
##  [61]  727.5  742.5  757.5  772.5  787.5  802.5  817.5  832.5  847.5  862.5
##  [71]  877.5  892.5  907.5  922.5  937.5  952.5  967.5  982.5  997.5 1012.5
##  [81] 1027.5 1042.5 1057.5 1072.5 1087.5 1102.5 1117.5 1132.5 1147.5 1162.5
##  [91] 1177.5 1192.5 1207.5 1222.5 1237.5 1252.5 1267.5 1282.5 1297.5 1312.5
## [101] 1327.5 1342.5 1357.5 1372.5 1387.5 1402.5 1417.5 1432.5 1447.5 1462.5
## [111] 1477.5 1492.5 1507.5 1522.5 1537.5 1552.5 1567.5 1582.5 1597.5 1612.5
## [121] 1627.5 1642.5 1657.5 1672.5 1687.5 1702.5 1717.5 1732.5 1747.5 1762.5
## [131] 1777.5 1792.5
##
## $xname
## [1] "ABIA$ArrDelay"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

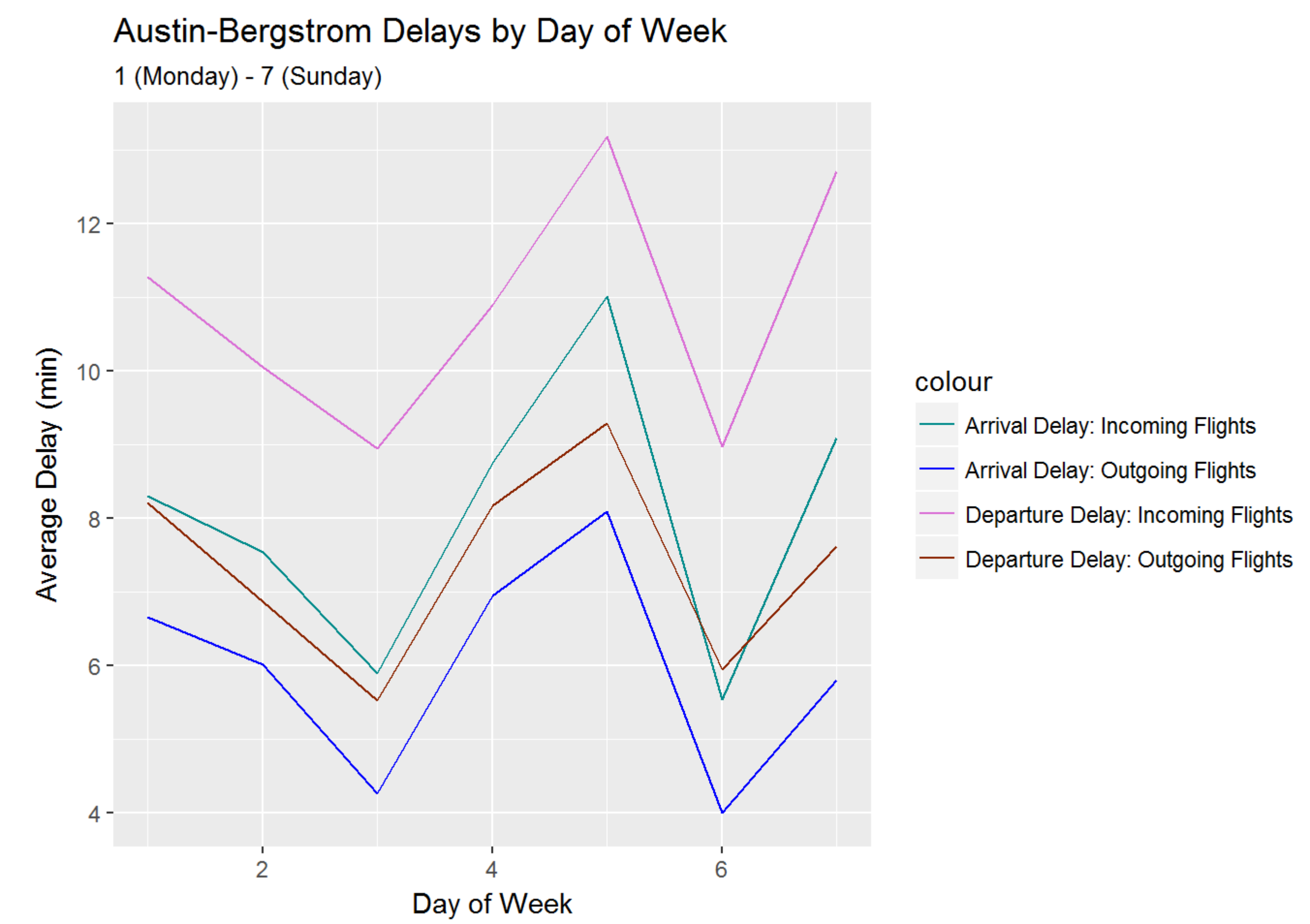


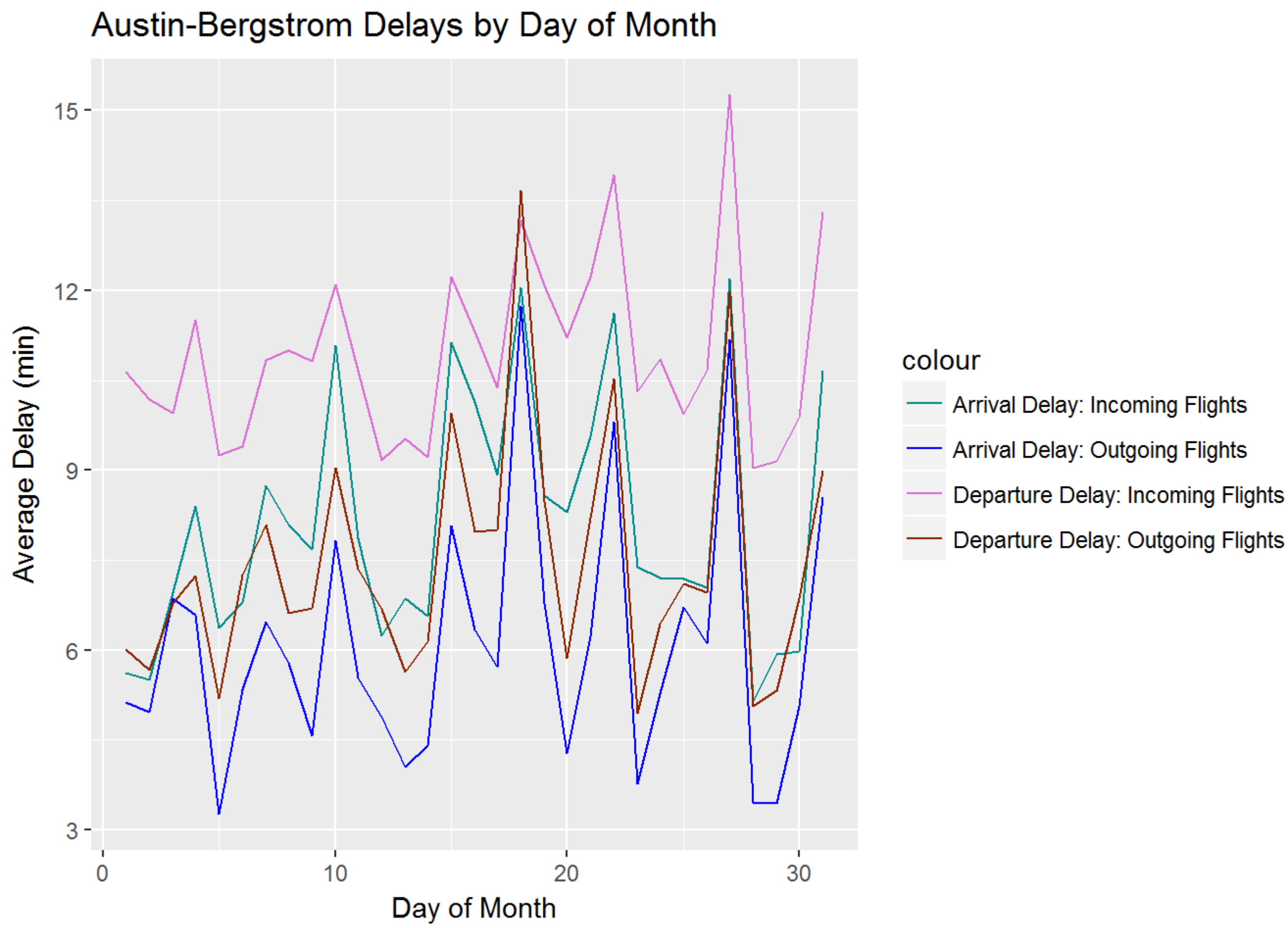
```
## $breaks
##   [1] -180 -165 -150 -135 -120 -105  -90  -75  -60  -45  -30  -15    0   15
##  [15]   30   45   60   75   90  105  120  135  150  165  180  195  210  225
##  [29]  240  255  270  285  300  315  330  345  360  375  390  405  420  435
##  [43]  450  465  480  495  510  525  540  555  570  585  600  615  630  645
##  [57]  660  675  690  705  720  735  750  765  780  795  810  825  840  855
##  [71]  870  885  900  915  930  945  960  975  990 1005 1020 1035 1050 1065
##  [85] 1080 1095 1110 1125 1140 1155 1170 1185 1200 1215 1230 1245 1260 1275
##  [99] 1290 1305 1320 1335 1350 1365 1380 1395 1410 1425 1440 1455 1470 1485
## [113] 1500 1515 1530 1545 1560 1575 1590 1605 1620 1635 1650 1665 1680 1695
## [127] 1710 1725 1740 1755 1770 1785 1800
## 
## $counts
##   [1]     0     0     0     0     0     0     0     0     0     2   125
##  [12] 57467 23008  6802  3288  2064  1357   946   697   487   334   266
##  [23]   210   177   144   100    67    57    51    34    36    29    20
##  [34]    17    12    12    10     7     4     5     1     4     1     1
##  [45]     1     2     0     0     0     0     0     0     0     0     0
##  [56]     0     1     0     0     0     0     0     0     0     0     0
##  [67]     0     0     0     0     1     0     0     0     0     0     0
##  [78]     0     0     0     0     0     0     0     0     0     0     0
##  [89]     0     0     0     0     0     0     0     0     0     0     0
## [100]     0     0     0     0     0     0     0     0     0     0     0
## [111]     0     0     0     0     0     0     0     0     0     0     0
## [122]     0     0     0     0     0     0     0     0     0     0     0
## 
## $density
##   [1] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   [6] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 1.362672e-06
##  [11] 8.516698e-05 3.915433e-02 1.567617e-02 4.634446e-03 2.240232e-03
##  [16] 1.406277e-03 9.245727e-04 6.445437e-04 4.748911e-04 3.318105e-04
##  [21] 2.275662e-04 1.812353e-04 1.430805e-04 1.205964e-04 9.811236e-05
##  [26] 6.813358e-05 4.564950e-05 3.883614e-05 3.474813e-05 2.316542e-05
##  [31] 2.452809e-05 1.975874e-05 1.362672e-05 1.158271e-05 8.176030e-06
##  [36] 8.176030e-06 6.813358e-06 4.769351e-06 2.725343e-06 3.406679e-06
##  [41] 6.813358e-07 2.725343e-06 6.813358e-07 6.813358e-07 6.813358e-07
##  [46] 1.362672e-06 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [51] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [56] 0.000000e+00 6.813358e-07 0.000000e+00 0.000000e+00 0.000000e+00
##  [61] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [66] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [71] 6.813358e-07 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [76] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [81] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [86] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [91] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [96] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [101] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [106] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [111] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [116] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [121] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [126] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

```
## [131] 0.000000e+00 0.000000e+00
##
## $mids
##   [1] -172.5 -157.5 -142.5 -127.5 -112.5  -97.5  -82.5  -67.5  -52.5  -37.5
##  [11]  -22.5   -7.5    7.5   22.5   37.5   52.5   67.5   82.5   97.5  112.5
##  [21]  127.5  142.5  157.5  172.5  187.5  202.5  217.5  232.5  247.5  262.5
##  [31]  277.5  292.5  307.5  322.5  337.5  352.5  367.5  382.5  397.5  412.5
##  [41]  427.5  442.5  457.5  472.5  487.5  502.5  517.5  532.5  547.5  562.5
##  [51]  577.5  592.5  607.5  622.5  637.5  652.5  667.5  682.5  697.5  712.5
##  [61]  727.5  742.5  757.5  772.5  787.5  802.5  817.5  832.5  847.5  862.5
##  [71]  877.5  892.5  907.5  922.5  937.5  952.5  967.5  982.5  997.5 1012.5
##  [81] 1027.5 1042.5 1057.5 1072.5 1087.5 1102.5 1117.5 1132.5 1147.5 1162.5
##  [91] 1177.5 1192.5 1207.5 1222.5 1237.5 1252.5 1267.5 1282.5 1297.5 1312.5
## [101] 1327.5 1342.5 1357.5 1372.5 1387.5 1402.5 1417.5 1432.5 1447.5 1462.5
## [111] 1477.5 1492.5 1507.5 1522.5 1537.5 1552.5 1567.5 1582.5 1597.5 1612.5
## [121] 1627.5 1642.5 1657.5 1672.5 1687.5 1702.5 1717.5 1732.5 1747.5 1762.5
## [131] 1777.5 1792.5
##
## $xname
## [1] "ABIA$DepDelay"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

aggregate (group) the flight data by month and take the mean


Austin-Bergstrom Delays by Month
Average Delay (min)
0
5
10
15
1
2
3
4
5
6
7
8
9
10
11
12
Month
colour
Arrival Delay: Incoming Flights
Arrival Delay: Outgoing Flights
Departure Delay: Incoming Flights
Departure Delay: Outgoing Flights


Austin-Bergstrom Delays by Day of Week
1 (Monday) - 7 (Sunday)
Average Delay (min)
4
6
8
10
12
2
4
6
Day of Week
colour
Arrival Delay: Incoming Flights
Arrival Delay: Outgoing Flights
Departure Delay: Incoming Flights
Departure Delay: Outgoing Flights



Question 2: Author attribution

Using the Reuters 50 articles, we will try to predict the authors of some unattributed articles based on word frequency patterns. To start off, we'll read in the train and test folders, and create corpora for them.

```
library(dplyr)
```

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

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

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

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

```
library(tm)
```

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

```
##
## Attaching package: 'NLP'
```

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

```
library(caret)
```

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

```
#Get train and test data
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }
setwd("~/MSBA/Scott/STA380")

file_list_train = Sys.glob('data/ReutersC50/C50train/*/*.txt')
file_list_test = Sys.glob('data/ReutersC50/C50test/*/*.txt')
all_train = lapply(file_list_train, readerPlain)
all_test = lapply(file_list_test, readerPlain)

# Some more concise document names via basic string manipulation
names(all_train) = file_list_train
names(all_train) = substring(names(all_train),first=26)
names(all_train) = t(data.frame(strsplit(names(all_train),'/')))[,1]

names(all_test) = file_list_test
names(all_test) = substring(names(all_test),first=25)
names(all_test) = t(data.frame(strsplit(names(all_test),'/')))[,1]

## once you have documents in a vector, you
## create a text mining 'corpus' with:
corpus_train = Corpus(VectorSource(all_train))
corpus_test = Corpus(VectorSource(all_test))
```

Next, we'll filter the corpora by making all words lowercase, removing numbers and punction, and removing excess whitespace. We'll also remove the stopwords that appear in the "en" set.

```
## Some pre-processing/tokenization steps to corpus_train
corpus_train = tm_map(corpus_train, content_transformer(tolower)) # make everything lowercase
corpus_train = tm_map(corpus_train, content_transformer(removeNumbers)) # remove numbers
corpus_train = tm_map(corpus_train, content_transformer(removePunctuation)) # remove punctuation
corpus_train = tm_map(corpus_train, content_transformer(stripWhitespace)) ## remove excess white
-space

corpus_train = tm_map(corpus_train, content_transformer(removeWords), stopwords("en"))

#Do it again to corpus_test
corpus_test = tm_map(corpus_test, content_transformer(tolower)) # make everything lowercase
corpus_test = tm_map(corpus_test, content_transformer(removeNumbers)) # remove numbers
corpus_test  = tm_map(corpus_test, content_transformer(removePunctuation)) # remove punctuation
corpus_test  = tm_map(corpus_test, content_transformer(stripWhitespace)) ## remove excess white-
space

corpus_test  = tm_map(corpus_test , content_transformer(removeWords), stopwords("en"))
```

Finally, our text is processed enough to make the document term matrices. We'll also weight the words using the TF-IDF, or term frequency - inverse document frequency method. We'll also remove 'sparse' terms, using the 7.5% cut off.

Most importantly, we took the intersection of words that appear in the training articles and testing articles, and will ONLY use those words. This way, words that only appear in the testing set will not throw errors, and words that appear only in the training set won't be used at all and won't waste memory or processing time.

With all this processing done, we'll format the data into X_train and X_test, ready to fit and predict our models.

```
DTM_train = DocumentTermMatrix(corpus_train, control = list(weighting = weightTfIdf))
DTM_train = removeSparseTerms(DTM_train, 0.925)

DTM_test = DocumentTermMatrix(corpus_test, control = list(weighting = weightTfIdf))
DTM_test = removeSparseTerms(DTM_test, 0.925)

#convert both to dataframe
DF_train = as.data.frame(as.matrix(DTM_train))
names(DF_train) = paste(names(DF_train),'.w',sep='')
list_authors_train = factor(names(all_train))

DF_test = as.data.frame(as.matrix(DTM_test))
names(DF_test) = paste(names(DF_test),'.w',sep='')
list_authors_test = factor(names(all_test))

#take intersection of words
intersection = intersect(names(DF_train),names(DF_test))
DF_train = DF_train[,intersection]
DF_test = DF_test[,intersection]

#split into appropriate form for model fitting
X_train = DF_train
X_train$author = list_authors_train
X_test = DF_test
X_test$author = list_authors_test
```

Model 1: Naive Bayes

```
library(naivebayes)
naive_bayes_model = naive_bayes(author ~ ., data = X_train)
naive_bayes_pred = data.frame(predict(naive_bayes_model, X_test))

conf_mat_nb = confusionMatrix(table(unlist(naive_bayes_pred),X_test$author))

#Print out result (number of correct/total number of predictions)
cat("Percent correct out-of-sample for Naive Bayes:", conf_mat_nb$overall[1])
```

```
## Percent correct out-of-sample for Naive Bayes: 0.4424
```

```
sensitivity_df_nb = as.data.frame(conf_mat_nb$byClass)
as.data.frame(sensitivity_df_nb)[order(-sensitivity_df_nb$Sensitivity),1:2]
```

```
##                          Sensitivity Specificity
## Class: FumikoFujisaki           0.88   0.9971429
## Class: LynnleyBrowning          0.78   0.9963265
## Class: LynneO'Donnell           0.74   0.9959184
## Class: MatthewBunce             0.72   0.9971429
## Class: RobinSidel               0.72   0.9955102
## Class: KarlPenhaul              0.70   0.9938776
## Class: BradDorfman              0.68   0.9734694
## Class: KeithWeir                0.64   0.9848980
## Class: LydiaZajc                0.62   0.9991837
## Class: NickLouth                0.62   0.9795918
## Class: AaronPressman            0.58   0.9951020
## Class: BernardHickey            0.58   0.9914286
## Class: GrahamEarnshaw           0.58   0.9922449
## Class: JimGilchrist             0.56   0.9975510
## Class: PeterHumphrey            0.56   0.9816327
## Class: RogerFillion             0.52   0.9971429
## Class: TheresePoletti           0.52   0.9873469
## Class: KevinMorrison            0.50   0.9877551
## Class: KirstinRidley            0.50   0.9763265
## Class: SimonCowell              0.50   0.9832653
## Class: KouroshKarimkhany        0.48   0.9971429
## Class: SamuelPerry              0.48   0.9832653
## Class: SarahDavison             0.48   0.9783673
## Class: JonathanBirt             0.44   0.9857143
## Class: JoWinterbottom           0.44   0.9942857
## Class: MichaelConnor            0.44   0.9910204
## Class: PatriciaCommins          0.44   0.9865306
## Class: EricAuchard              0.40   0.9820408
## Class: TanEeLyn                 0.40   0.9763265
## Class: ToddNissen               0.38   0.9946939
## Class: AlexanderSmith           0.36   0.9783673
## Class: JoeOrtiz                 0.36   0.9763265
## Class: MarcelMichelson          0.36   0.9914286
## Class: MartinWolk               0.36   0.9865306
## Class: TimFarrand               0.36   0.9902041
## Class: MureDickie               0.34   0.9857143
## Class: AlanCrosby               0.32   0.9987755
## Class: WilliamKazer             0.32   0.9795918
## Class: HeatherScoffield         0.30   0.9959184
## Class: KevinDrawbaugh           0.30   0.9885714
## Class: EdnaFernandes            0.26   0.9918367
## Class: JanLopatka               0.26   0.9914286
## Class: PierreTran               0.22   0.9906122
## Class: BenjaminKangLim          0.20   0.9873469
## Class: ScottHillis              0.20   0.9718367
## Class: JaneMacartney            0.18   0.9800000
## Class: JohnMastrini             0.18   0.9922449
## Class: MarkBendeich             0.18   0.9893878
## Class: DavidLawder              0.10   0.9971429
## Class: DarrenSchuettler         0.08   0.9955102
```

Naive Bayes only accurately predicts 44% of the testing articles. The table above dispays the sensitivity and specificity for each author, once again demonstrating that Naive Bayes is not particularly accurate for the majority of the authors in the testing data. This may be because Naive Bayes assumes each word's frequency is independent, when in fact word choice may be highly correlated depending on the author's topic of interest.

Model 2: Random Forest with 350 trees

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## 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
```

```
random_forest_model = randomForest(author ~ ., data = X_train,
                          distribution = 'multinomial',
                          n.trees=350)
random_forest_pred = data.frame(predict(random_forest_model,newdata = X_test))

conf_mat_rf = confusionMatrix(table(unlist(random_forest_pred),X_test$author))

#Print out result (number of correct/total number of predictions)
cat("Percent correct out-of-sample for Random Forest:", conf_mat_rf$overall[1])
```

```
## Percent correct out-of-sample for Random Forest: 0.5636
```

```
sensitivity_df_rf = as.data.frame(conf_mat_rf$byClass)
as.data.frame(sensitivity_df_rf)[order(-sensitivity_df_rf$Sensitivity),1:2]
```

```
##                          Sensitivity Specificity
## Class: FumikoFujisaki           0.98   0.9971429
## Class: LynnleyBrowning          0.96   0.9902041
## Class: JimGilchrist             0.94   0.9942857
## Class: KarlPenhaul              0.92   0.9840816
## Class: PeterHumphrey            0.90   0.9812245
## Class: AaronPressman            0.86   0.9893878
## Class: GrahamEarnshaw           0.84   0.9967347
## Class: KouroshKarimkhany        0.84   0.9906122
## Class: LynneO'Donnell           0.84   0.9967347
## Class: RobinSidel               0.84   0.9910204
## Class: JoWinterbottom           0.82   0.9942857
## Class: RogerFillion             0.82   0.9946939
## Class: MatthewBunce             0.80   0.9934694
## Class: KeithWeir                0.76   0.9959184
## Class: PatriciaCommins          0.76   0.9938776
## Class: NickLouth                0.74   0.9910204
## Class: MichaelConnor            0.70   0.9902041
## Class: BradDorfman              0.64   0.9816327
## Class: LydiaZajc                0.64   0.9983673
## Class: SimonCowell              0.64   0.9922449
## Class: TimFarrand               0.64   0.9869388
## Class: KirstinRidley            0.62   0.9987755
## Class: MarcelMichelson          0.58   0.9889796
## Class: MureDickie               0.58   0.9881633
## Class: AlexanderSmith           0.56   0.9914286
## Class: KevinMorrison            0.56   0.9926531
## Class: SarahDavison             0.56   0.9906122
## Class: BernardHickey            0.52   0.9959184
## Class: JanLopatka               0.52   0.9865306
## Class: TheresePoletti           0.52   0.9836735
## Class: JaneMacartney            0.48   0.9751020
## Class: JonathanBirt             0.48   0.9885714
## Class: KevinDrawbaugh           0.46   0.9881633
## Class: MarkBendeich             0.46   0.9951020
## Class: JohnMastrini             0.44   0.9853061
## Class: JoeOrtiz                 0.42   0.9897959
## Class: SamuelPerry              0.40   0.9942857
## Class: TanEeLyn                 0.40   0.9906122
## Class: HeatherScoffield         0.38   0.9795918
## Class: AlanCrosby               0.36   0.9963265
## Class: ToddNissen               0.28   0.9889796
## Class: PierreTran               0.26   0.9926531
## Class: BenjaminKangLim          0.24   0.9869388
## Class: MartinWolk               0.24   0.9938776
## Class: DavidLawder              0.22   0.9926531
## Class: WilliamKazer             0.22   0.9951020
## Class: EricAuchard              0.18   0.9955102
## Class: EdnaFernandes            0.16   0.9946939
## Class: DarrenSchuettler         0.12   0.9979592
## Class: ScottHillis              0.08   0.9926531
```

Clearly, the Random Forest mode is much more accurate than the Naive Bayes, though a 55% accuracy is nothing to write home about. Once again, the table above dispays the sensitivity and specificity for each author, demonstrating that Random Forest is a much better model than Naive Bayes.

Question 3: Association rule mining

Using arules package for association mining

```
## 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
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   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
```

```
##      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}
```

Loading in arulesViz in order to help us look at the different levels of confidence and support along with their results to determine effectiveness

```
library(arulesViz)
```

```
## Loading required package: grid
```

```
rules <- apriori(grocery, parameter=list(support=0.01, confidence=0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## 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 [0.00s].
## 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 ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules_2 <- apriori(grocery, parameter=list(support=0.001, confidence=0.9))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [129 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspectDT(rules)
```

Show 10 **entries** **Search:**

	LHS	RHS	support	confidence	lift
	All	All	All	All	All
[1]	{curd,yogurt}	{whole milk}	0.010	0.582	2.279
[2]	{butter,other vegetables}	{whole milk}	0.011	0.574	2.245
[3]	{domestic eggs,other vegetables}	{whole milk}	0.012	0.553	2.162
[4]	{whipped/sour cream,yogurt}	{whole milk}	0.011	0.525	2.053
[5]	{other vegetables,whipped/sour cream}	{whole milk}	0.015	0.507	1.984
[6]	{other vegetables,pip fruit}	{whole milk}	0.014	0.518	2.025
[7]	{citrus fruit,root vegetables}	{other vegetables}	0.010	0.586	3.030
[8]	{root vegetables,tropical fruit}	{other vegetables}	0.012	0.585	3.021
[9]	{root vegetables,tropical fruit}	{whole milk}	0.012	0.570	2.231
[10]	{tropical fruit,yogurt}	{whole milk}	0.015	0.517	2.025

Showing 1 to 10 of 15 entries

Previous 1 2 Next

```
inspectDT(rules_2)
```

Show 10 ▼ **entries**

Search:

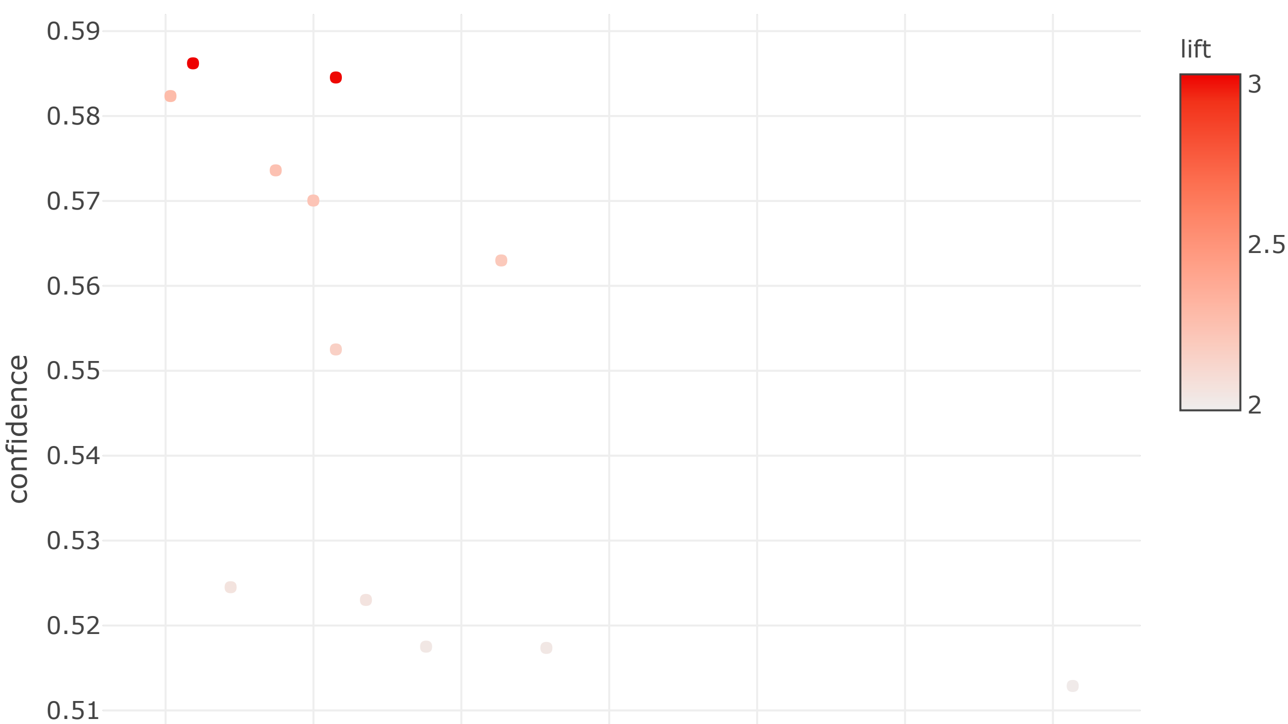
	LHS	RHS	support	confidence	lift
	All	All	All	All	All
[1]	{liquor,red/blush wine}	{bottled beer}	0.002	0.905	11.235
[2]	{cereals,curd}	{whole milk}	0.001	0.909	3.558
[3]	{bottled beer,soups}	{whole milk}	0.001	0.917	3.588

	LHS	RHS	support	confidence	lift
[4]	{house keeping products,whipped/sour cream}	{whole milk}	0.001	0.923	3.613
[5]	{pastry,sweet spreads}	{whole milk}	0.001	0.909	3.558
[6]	{rice,sugar}	{whole milk}	0.001	1.000	3.914
[7]	{bottled water,rice}	{whole milk}	0.001	0.923	3.613
[8]	{canned fish,hygiene articles}	{whole milk}	0.001	1.000	3.914
[9]	{grapes,onions}	{other vegetables}	0.001	0.917	4.737
[10]	{hard cheese,oil}	{other vegetables}	0.001	0.917	4.737

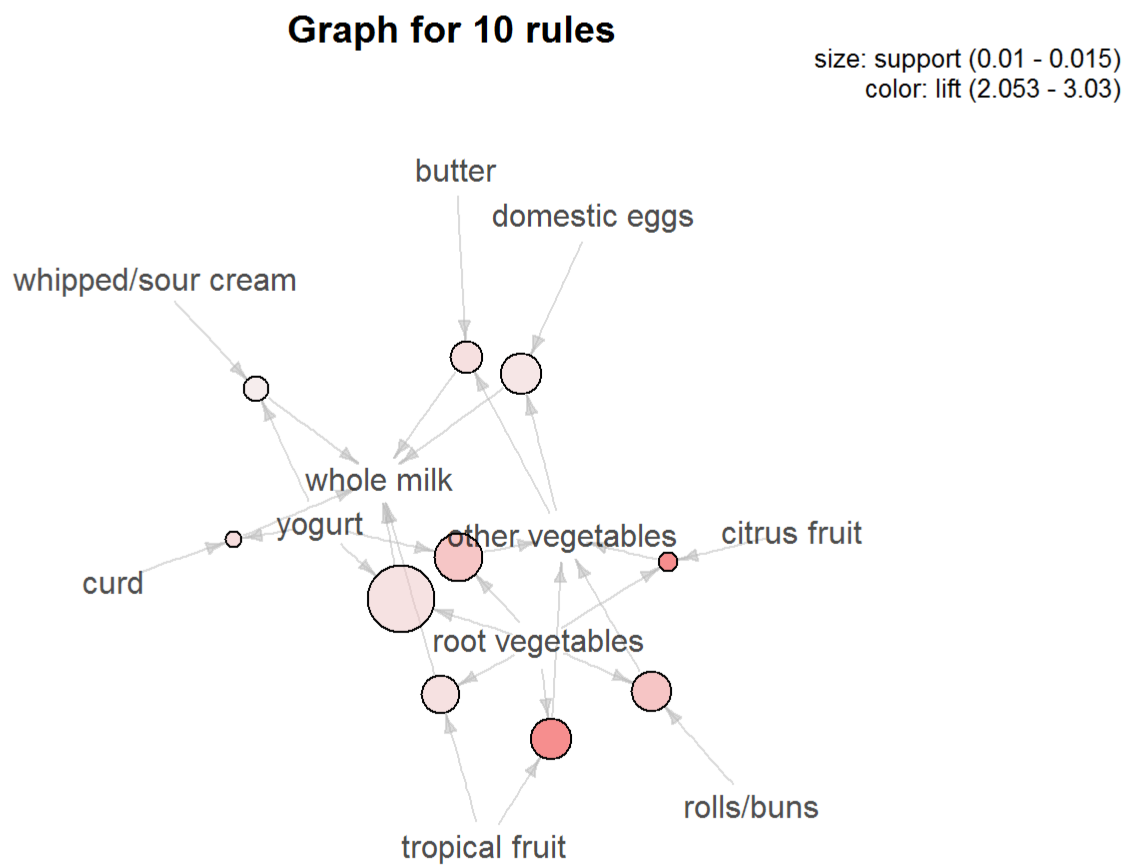
Showing 1 to 10 of 129 entries Previous 1 2 3 4 5 … 13 Next

Plotting results and using sorted options to allow us to only plot the top 10 rules based on lift or confidence.

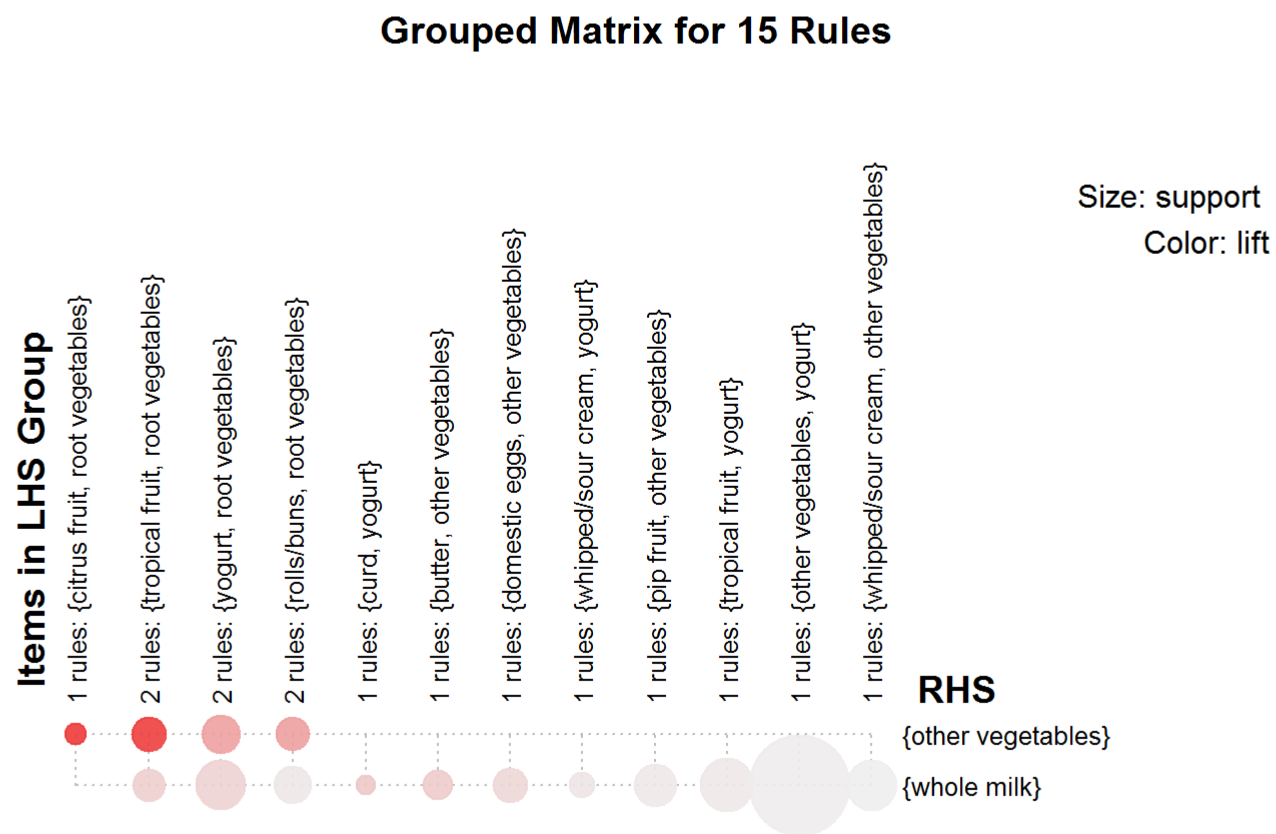
```
rules_sorted <- sort(rules, by='confidence', decreasing=TRUE)
plotly_arules(rules)
```



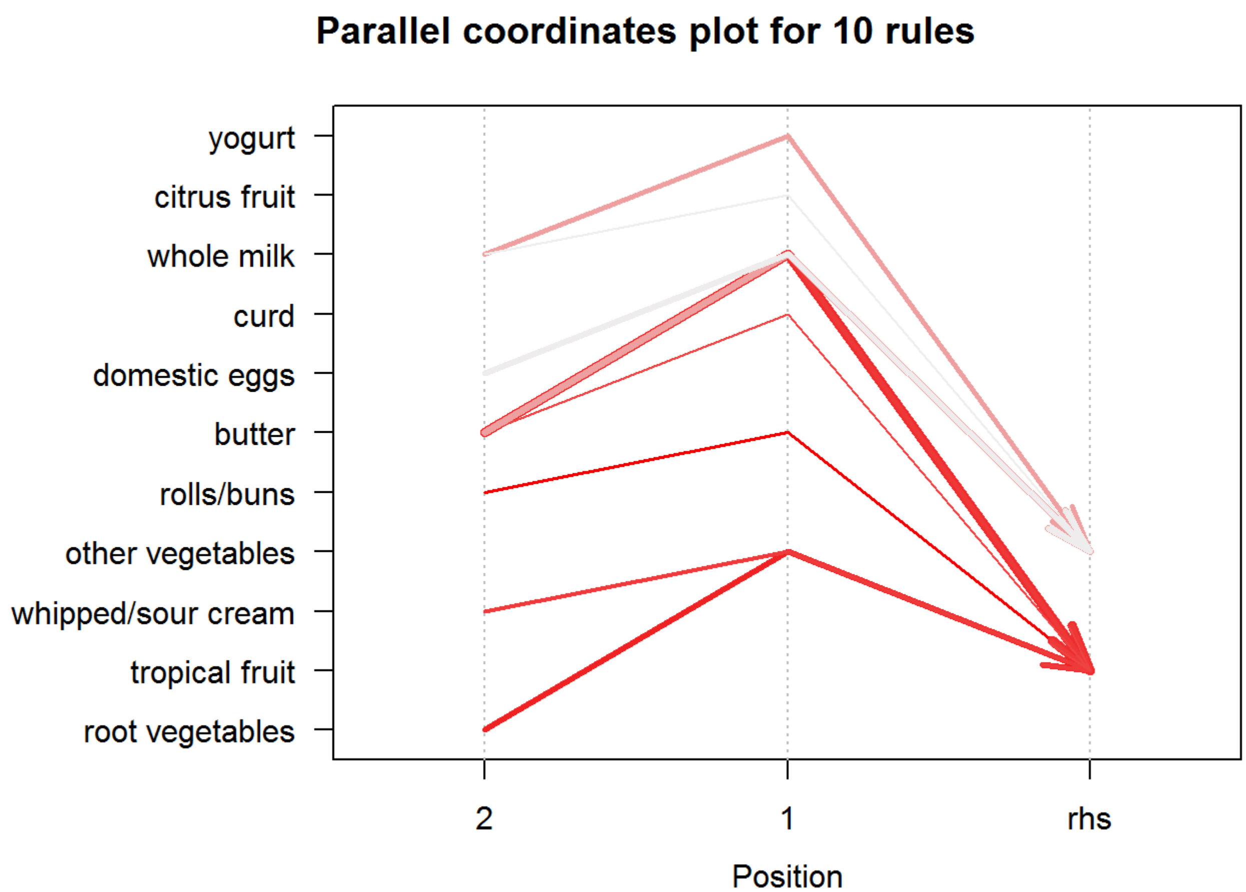

```
subrules <- head(sort(rules, by='lift'),10) #Graph 10 rules by 10 highest lifts
plot(subrules, method='graph')
```




```
plot(rules, method='grouped') #Grouped Matrix  to show LHS and RHS
```




```
plot(subrules,method='paracoord', control=list(reorder=TRUE))
```




```
#Parallel Coordinates plot for 10 rules
```

Allows us to look at the rules with high degrees of confidence and rules with high lift values

```
rules_conf <- sort(rules, by='confidence', decreasing=TRUE)
inspect(head(rules_conf)) #High-confidence rules
```

```
##     lhs                     rhs                   support confidence     lift
## [1] {citrus fruit,
##      root vegetables}  => {other vegetables} 0.01037112  0.5862069 3.029608
## [2] {root vegetables,
##      tropical fruit}   => {other vegetables} 0.01230300  0.5845411 3.020999
## [3] {curd,
##      yogurt}           => {whole milk}       0.01006609  0.5823529 2.279125
## [4] {butter,
##      other vegetables} => {whole milk}       0.01148958  0.5736041 2.244885
## [5] {root vegetables,
##      tropical fruit}   => {whole milk}       0.01199797  0.5700483 2.230969
## [6] {root vegetables,
##      yogurt}           => {whole milk}       0.01453991  0.5629921 2.203354
```

```
rules_lift <- sort(rules, by='lift', decreasing=TRUE)
inspect(head(rules_lift)) #High lift rules
```

```
##     lhs                     rhs                   support confidence     lift
## [1] {citrus fruit,
##      root vegetables}  => {other vegetables} 0.01037112  0.5862069 3.029608
## [2] {root vegetables,
##      tropical fruit}   => {other vegetables} 0.01230300  0.5845411 3.020999
## [3] {rolls/buns,
##      root vegetables}  => {other vegetables} 0.01220132  0.5020921 2.594890
## [4] {root vegetables,
##      yogurt}           => {other vegetables} 0.01291307  0.5000000 2.584078
## [5] {curd,
##      yogurt}           => {whole milk}       0.01006609  0.5823529 2.279125
## [6] {butter,
##      other vegetables} => {whole milk}       0.01148958  0.5736041 2.244885
```

This allowed us to see a lot of different basket options that indicated margarine should be included in the basket.

```
rules <- apriori(data=grocery, parameter=list(supp=0.001, conf=0.08), appearance = list(default
= 'lhs', rhs = 'margarine'), control=list(verbose=F))
rules <- sort(rules, decreasing=TRUE, by='confidence')
inspect(rules[1:5])
```

```
##     lhs                     rhs             support confidence     lift
## [1] {bottled water,
##      domestic eggs,
##      tropical fruit}   => {margarine} 0.001016777  0.4545455 7.761206
## [2] {flour,
##      tropical fruit}   => {margarine} 0.001423488  0.4375000 7.470161
## [3] {flour,
##      whole milk,
##      yogurt}           => {margarine} 0.001016777  0.4000000 6.829861
## [4] {bottled water,
##      flour}            => {margarine} 0.001016777  0.3703704 6.323945
## [5] {flour,
##      other vegetables,
##      yogurt}           => {margarine} 0.001016777  0.3703704 6.323945
```

Using lhs as margarine we wanted to see if it provided any knowledge, but appearing in such connected area meant it didnt have any useful insights.

```
rules2 <- apriori(data=grocery, parameter=list(supp=0.01, conf=0.1), appearance = list(default =
 'rhs', lhs = 'margarine'), control=list(verbose=F))
rules2 <- sort(rules2, by='confidence', decreasing=TRUE)
inspect(rules2)
```

```
##      lhs            rhs                support    confidence lift
## [1]  {margarine} => {whole milk}       0.02419929 0.4131944  1.6170980
## [2]  {margarine} => {other vegetables} 0.01972547 0.3368056  1.7406635
## [3]  {}          => {whole milk}       0.25551601 0.2555160  1.0000000
## [4]  {margarine} => {rolls/buns}       0.01474326 0.2517361  1.3686151
## [5]  {margarine} => {yogurt}           0.01423488 0.2430556  1.7423115
## [6]  {}          => {other vegetables} 0.19349263 0.1934926  1.0000000
## [7]  {margarine} => {root vegetables}  0.01108287 0.1892361  1.7361354
## [8]  {}          => {rolls/buns}       0.18393493 0.1839349  1.0000000
## [9]  {margarine} => {bottled water}    0.01026945 0.1753472  1.5865133
## [10] {}          => {soda}             0.17437722 0.1743772  1.0000000
## [11] {margarine} => {soda}             0.01016777 0.1736111  0.9956066
## [12] {}          => {yogurt}           0.13950178 0.1395018  1.0000000
## [13] {}          => {bottled water}    0.11052364 0.1105236  1.0000000
## [14] {}          => {root vegetables}  0.10899847 0.1089985  1.0000000
## [15] {}          => {tropical fruit}   0.10493137 0.1049314  1.0000000
```

We tested a few different values and combinations for support and confidence, and eventually decided to use two different levels in order to look at slightly different things. We decide on this as it made sure from a confidence level that we were making sure that there was actually a degree of consistency for that rule of above 50%. With support we kept it at 0.01 so that it would predict only options that occurred slightly more frequently so as not to waste time and effort on minor occurrences. We also tested a version with a support of 0.001 so it would pick up many different options and a confidence of 0.9, allowing us to have some knowledge about options that occur less frequently, but are far more likely. These were also both selected to prevent us having far too long of a list to work with.

The discovered item sets make sense as they are typically related food items, and they primarily cover groceries that are consistent commodities. When placed into a connection map it shows that margarine is the most connected grocery, and has the greatest degree of between-ness. This agrees with association analysis that was run after at varying levels of confidence and support, as margarine was the highest rhs at all levels when sorted by confidence and lift. However having margarine as the sole item in lhs, as we screened for after, does not provide much information other than showing that you should be buying other commodities in general. Other items that had a high degree of association between them were items that were clearly related to baking, and therefore when someone was purchasing one of these items they were far more likely to be purchasing other baking items. For the low support and high confidence interval we found pieces of info that would impact the placement of single items near each other. This includes making sure all the alchohol is in the same section as buying wine was highly indicative of also purchasing beer. Others include cereal and milk, which could be included in the commodities section discussed below. ## Key Grocery takeaways The key takeaways were that simple commodities should be placed in one area as these are often spread across stores and by providing a grouping of them you can simplify the shopping experience for people only coming for simple items. This would also hopefully help them remember all the commodities that they needed and hopefully increase revenue of the store.