PS4

Yuri Lee, Seunghoon Choi

2021 5 6

1. Clustering and PCA

(1) Data checking

In order to analyze 'color' of wine, data mutation: 'color' of white=0, red=1

Selecting variables: high correlation

quality: volatile.acidity, chlorides, density, alcohol

color: all variables except for citric.acid

So, we will omit 'citric.acid', and use 10 variables

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## [1,]   -0.07674321       -0.2656995  0.08553172    -0.03698048 -0.2006655
##      free.sulfur.dioxide total.sulfur.dioxide    density        pH  sulphates
## [1,]          0.05546306          -0.04138545 -0.3058579 0.0195057 0.03848545
##        alcohol
## [1,] 0.4443185
```

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## [1,]     0.4867398        0.6530356  -0.1873965      -0.348821 0.5126782
##      free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
## [1,]          -0.4716437           -0.7003572 0.3906453 0.3291287  0.487218
##          alcohol
## [1,] -0.03296955
```

(2) Using K-means, K-means++

Run k-means with 3 clusters and 25 starts

Using kmeans++ initialization

Compare the results

Within-cluster of K-means

```
## [1] 39352.84
```

Within-cluster of K-means++

```
## [1] 39352.84
```

Between-cluster of K-means

```
## [1] 25607.16
```

Between-cluster of K-means++

```
## [1] 25607.16
```

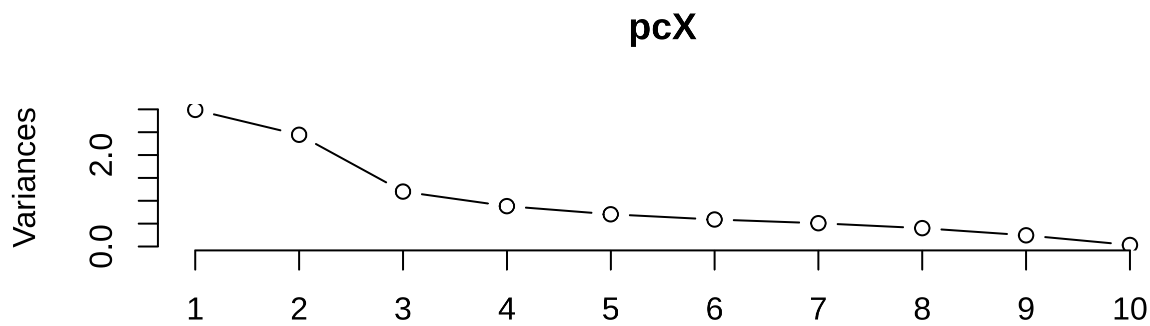
predicted engagement: R-squared too low

```
##
## Call:
## lm(formula = quality ~ z, data = wine_k)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0167 -0.7630 -0.0167  0.4907  3.2370
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.25555    0.03057  171.94   <2e-16 ***
## z            0.25371    0.01294   19.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8486 on 6495 degrees of freedom
## Multiple R-squared:  0.05591,    Adjusted R-squared:  0.05577
## F-statistic: 384.7 on 1 and 6495 DF,  p-value: < 2.2e-16
```

```
##
## Call:
## lm(formula = color ~ z, data = wine_k)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.76491 -0.33911  0.08669  0.08669  1.08669
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.190704   0.009218   129.2   <2e-16 ***
## z           -0.425796   0.003901  -109.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2559 on 6495 degrees of freedom
## Multiple R-squared:  0.6472, Adjusted R-squared:  0.6471
## F-statistic: 1.191e+04 on 1 and 6495 DF,  p-value: < 2.2e-16
```

(3) using PCA

Elbow point = 4




```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.7286 1.5631 1.0961 0.93964 0.83957 0.76865 0.71383
## Proportion of Variance 0.2988 0.2443 0.1201 0.08829 0.07049 0.05908 0.05096
## Cumulative Proportion  0.2988 0.5431 0.6633 0.75156 0.82204 0.88113 0.93208
##                            PC8     PC9    PC10
## Standard deviation     0.63362 0.49486 0.18119
## Proportion of Variance 0.04015 0.02449 0.00328
## Cumulative Proportion  0.97223 0.99672 1.00000
```

Better results: R-squared higher

Especially, predicting 'color' of wine much higher

```
##
## Call:
## lm(formula = quality ~ pc, data = wine_p)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9612 -0.5069 -0.0544  0.5196  4.0816
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.818378   0.009862 589.955  < 2e-16 ***
## pcPC1        0.043629   0.005706   7.646 2.37e-14 ***
## pcPC2       -0.188047   0.006310 -29.802  < 2e-16 ***
## pcPC3       -0.079272   0.008999  -8.809  < 2e-16 ***
## pcPC4        0.188532   0.010497  17.961  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7949 on 6492 degrees of freedom
## Multiple R-squared:  0.1718, Adjusted R-squared:  0.1713
## F-statistic: 336.7 on 4 and 6492 DF,  p-value: < 2.2e-16
```

```
##
## Call:
## lm(formula = color ~ pc, data = wine_p)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.43877 -0.13016 -0.00188  0.11967  1.45638
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.246114   0.002587   95.15  < 2e-16 ***
## pcPC1       -0.208969   0.001497 -139.63  < 2e-16 ***
## pcPC2        0.061496   0.001655   37.16  < 2e-16 ***
## pcPC3        0.042250   0.002360   17.90  < 2e-16 ***
## pcPC4       -0.017344   0.002753   -6.30 3.17e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2085 on 6492 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7657
## F-statistic:  5310 on 4 and 6492 DF,  p-value: < 2.2e-16
```

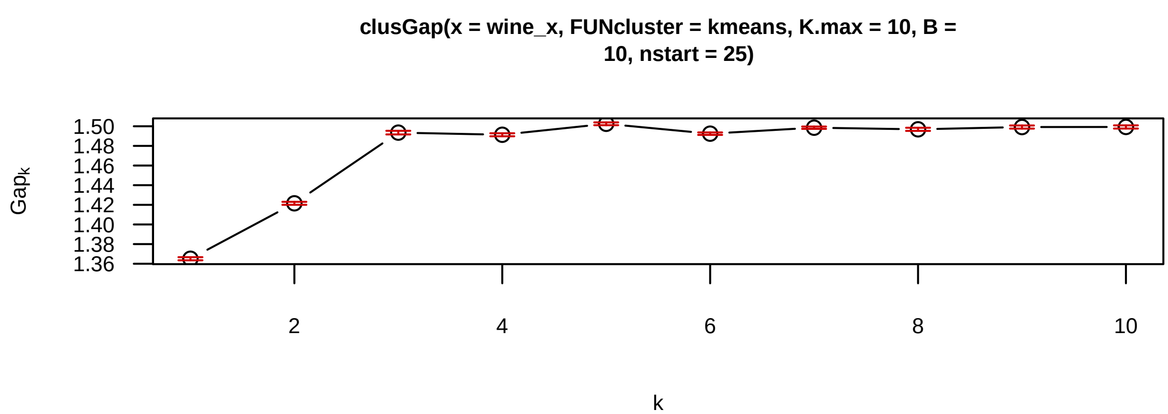
most of wines have levels between 5 and 7

So, the result of PCA is higher than that of Clustering

And because of these traits, Predicting quality is more difficult

```
##
##           3    4    5    6    7    8    9
##   red    10   53  681  638  199   18    0
##   white  20  163 1457 2198  880  175    5
```

(4) Appendix: Using Gap Statistics: K=3



2. Market segmentation

(1) Overview

We analyzed 325,802 tweets from the target company's followers over a seven days in order to find understand its target customers better.

After extracting meaningless or inappropriate tweet categories(chatter, uncategorized, spam and adult), we explored which categories of tweets are most attractive to target customers.

In addition, we searched for relations among categories to look at more clearly which interests can be related to each other through hierarchical clustering.

(2) Analysis

(2-1) Delete meaningless or uninterpretable categories

We deleted chatter, uncategorized, spam and adult

(2-2) Calculate each tweet keyword's proportion of every individual

(2-3) Find the most powerful interests of the potential customers

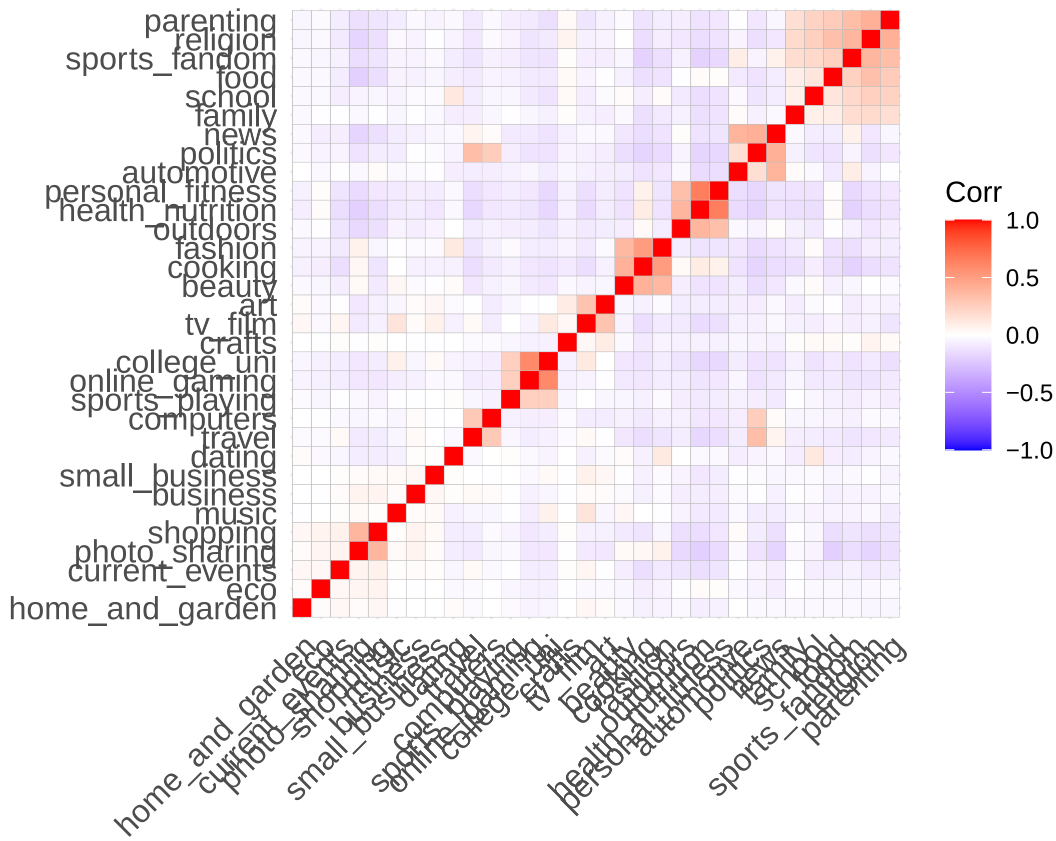
(2-4) Searching for each person's most interested category

(2-5) Frequencies of each category

```
##
##             art       automotive           beauty         business
##             154               55               18                3
##     college_uni        computers          cooking           crafts
##             374               17              612                4
##  current_events           dating              eco           family
##             533              215                7               52
##         fashion             food health_nutrition  home_and_garden
##              56              196             1289               13
##           music             news    online_gaming         outdoors
##              40              281              328                7
##       parenting personal_fitness    photo_sharing         politics
##              73              110             1296              556
##        religion           school         shopping   small_business
##             158               18              235                2
##   sports_fandom  sports_playing           travel          tv_film
##             484                9              427              260
```

This table showed us that most attractive tweet category was 1.photo-sharing(1,296 people), 2. health-nutrition(1,289), 3. cooking(612), 4. politics(556), 5. currenet events(533).

(2-6) Analyzing correlations between categories



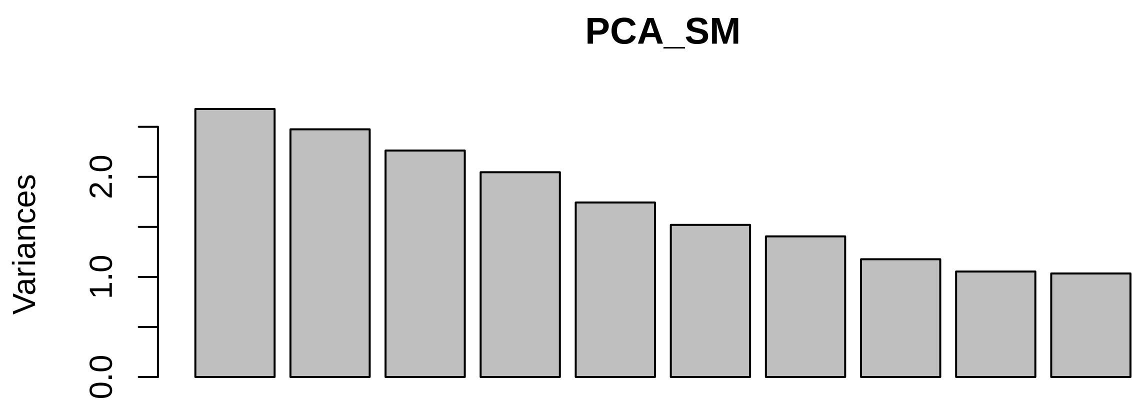
The correlation plot shows there are close correlation among some categories.

We could see most wide correlation among the categories parenting, religion, sports_fandom, food, school and family.

Secondly, personal fitness, health nutrition, and outdoors have high correlation.

News, politics and automotive also represent high correlation.

Analyzing with PCA




```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.6366 1.57326 1.50426 1.43055 1.32050 1.23299 1.18548
## Proportion of Variance 0.0837 0.07735 0.07071 0.06395 0.05449 0.04751 0.04392
## Cumulative Proportion  0.0837 0.16105 0.23176 0.29571 0.35020 0.39771 0.44163
##                            PC8     PC9    PC10    PC11   PC12    PC13    PC14
## Standard deviation     1.08482 1.02673 1.01716 0.98985 0.9880 0.98622 0.97218
## Proportion of Variance 0.03678 0.03294 0.03233 0.03062 0.0305 0.03039 0.02954
## Cumulative Proportion  0.47840 0.51135 0.54368 0.57430 0.6048 0.63520 0.66473
##                           PC15    PC16    PC17   PC18    PC19    PC20   PC21
## Standard deviation     0.94958 0.93770 0.91068 0.8727 0.85452 0.84252 0.8294
## Proportion of Variance 0.02818 0.02748 0.02592 0.0238 0.02282 0.02218 0.0215
## Cumulative Proportion  0.69291 0.72039 0.74631 0.7701 0.79292 0.81511 0.8366
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.80489 0.79903 0.7859 0.77695 0.76964 0.73882 0.68052
## Proportion of Variance 0.02025 0.01995 0.0193 0.01886 0.01851 0.01706 0.01447
## Cumulative Proportion  0.85685 0.87680 0.8961 0.91497 0.93348 0.95054 0.96501
##                           PC29    PC30    PC31     PC32
## Standard deviation     0.65300 0.59676 0.58064 0.009629
## Proportion of Variance 0.01333 0.01113 0.01054 0.000000
## Cumulative Proportion  0.97833 0.98946 1.00000 1.000000
```

```
##              Tweet         PC1
## 1    sports_fandom 0.396532696
## 2         religion 0.389921517
## 3        parenting 0.365393209
## 4             food 0.270397055
## 5           school 0.245230941
## 6           family 0.235341730
## 7       automotive 0.117008882
## 8             news 0.110814602
## 9         politics 0.085007805
## 10          crafts 0.077092198
```

```
## 11         computers  0.041978524
## 12            travel  0.031474172
## 13    current_events  0.006179247
## 14           tv_film  0.006152206
## 15    small_business  0.001588976
## 16               art -0.001939738
## 17   home_and_garden -0.004887476
## 18          business -0.007245329
## 19               eco -0.025556623
## 20            dating -0.028773237
## 21             music -0.035456624
## 22    sports_playing -0.044650031
## 23          shopping -0.061325326
## 24       college_uni -0.066615741
## 25     online_gaming -0.069799023
## 26            beauty -0.106279308
## 27     photo_sharing -0.118053199
## 28          outdoors -0.162930410
## 29           fashion -0.189639988
## 30  personal_fitness -0.251594241
## 31  health_nutrition -0.271567465
## 32           cooking -0.289703428

##               Tweet          PC2
## 1  health_nutrition  0.388407034
## 2  personal_fitness  0.365998334
## 3          outdoors  0.261247293
## 4              food  0.248269512
## 5          religion  0.214003964
## 6         parenting  0.207961664
## 7           cooking  0.183466026
## 8     sports_fandom  0.138067352
## 9            school  0.135500756
## 10           beauty  0.092268055
## 11          fashion  0.077344383
## 12           family  0.053774032
## 13           dating  0.024609445
## 14              eco  0.001971010
## 15           crafts -0.002935894
## 16  home_and_garden -0.064541177
## 17            music -0.074368965
## 18              art -0.080717965
## 19         business -0.094391368
## 20   small_business -0.097324206
## 21   sports_playing -0.111896314
## 22       automotive -0.135842579
## 23        computers -0.146703842
## 24   current_events -0.150155986
## 25             news -0.151001325
## 26    online_gaming -0.157467616
## 27         shopping -0.158230406
## 28    photo_sharing -0.158750815
## 29          tv_film -0.170063004
## 30      college_uni -0.211973696
```

```
## 31           travel -0.237144215
## 32         politics -0.263816889
```

Top five categories of PC1 were religion, sports_fandom, parenting, food, school, and top five categories of PC2 were politics, travel, news, college_uni and automotive.

This result closely coincides with that of the hierarchical correlation plot.

(3) Suggestion

When we synthesized all analytic results, especially the correlation plot and categorical frequency table, what we would like to suggest to you about your potential customers' top-five interest categories are as below.

1. photo sharing(1,296), shopping(235), total(1,531)

2. health nutrition(1,289), personal fitness(110), outdoor(7), total(1,406)

3. sports fandom(484), food(196), parenting(73), religion(158), family(52), school(18), total(981)

4. politics(556), news(281), automotive(55), total(892)

5. cooking(612), fashion(56), beauty(18), total(686)

If you plan advertise focusing on those categories, I convince, it will work out to your target customers.

3. Association rules for grocery purchases

(1) Data Clearing

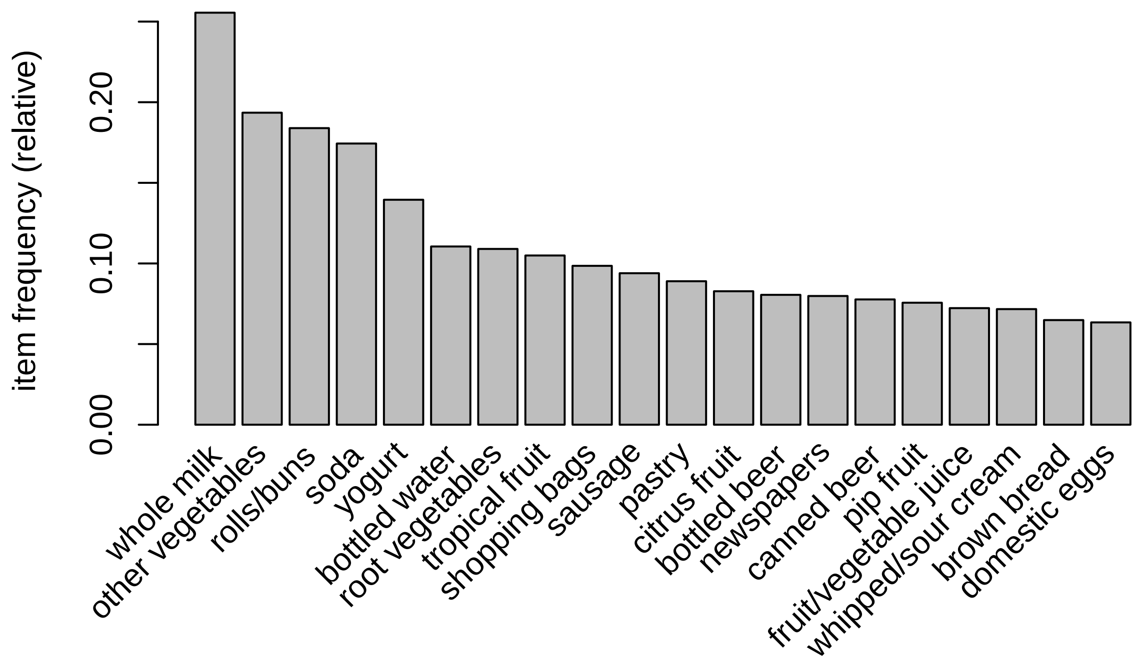
Remove duplicates ("de-dupe")

Cast this resulting list as a special arules "transactions" class

```
## 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
```

Plot top 20 of list

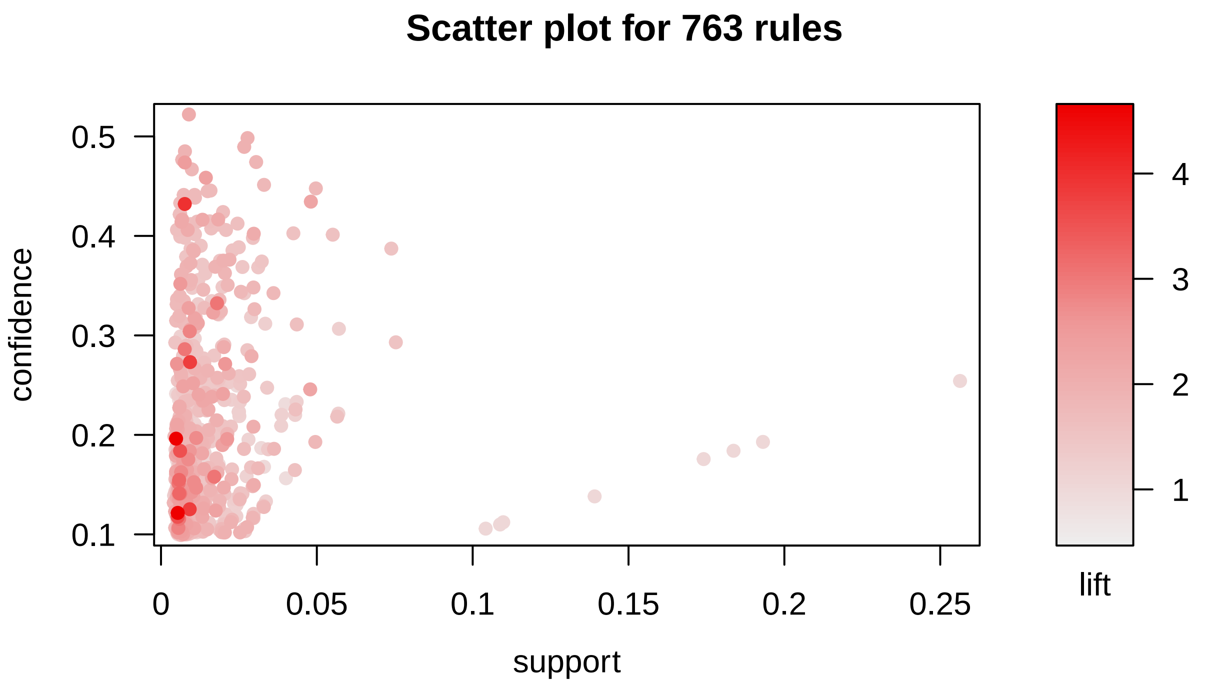


(2) Analysis

Now run the 'apriori' algorithm(support=.005, confidence=.1, maxlen=2)

Check the output and plot all the rules

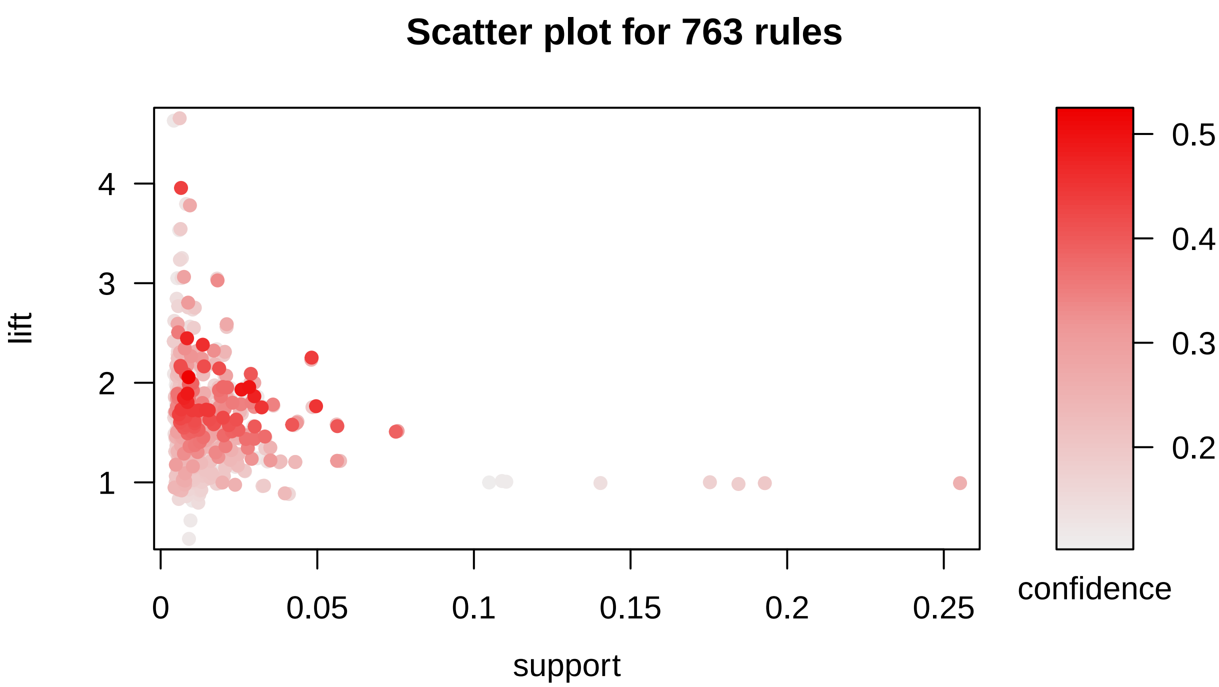
```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



can swap the axes and color scales

Pick the thresholds for lift and confidence: lift > 2, confidence > 0.2

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

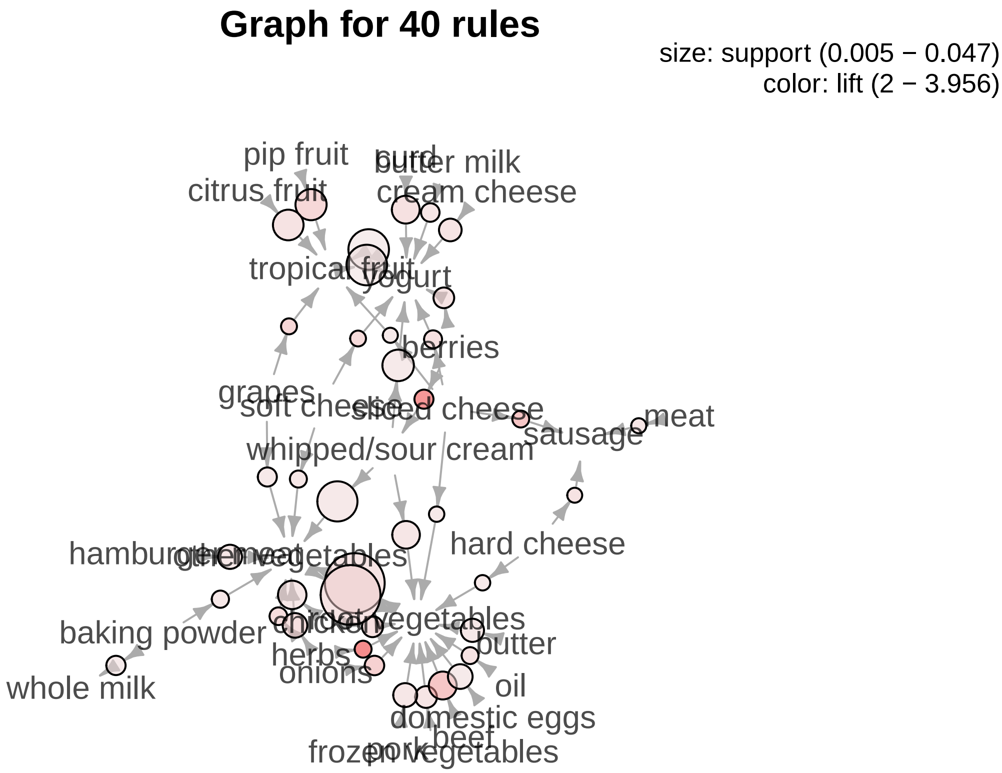


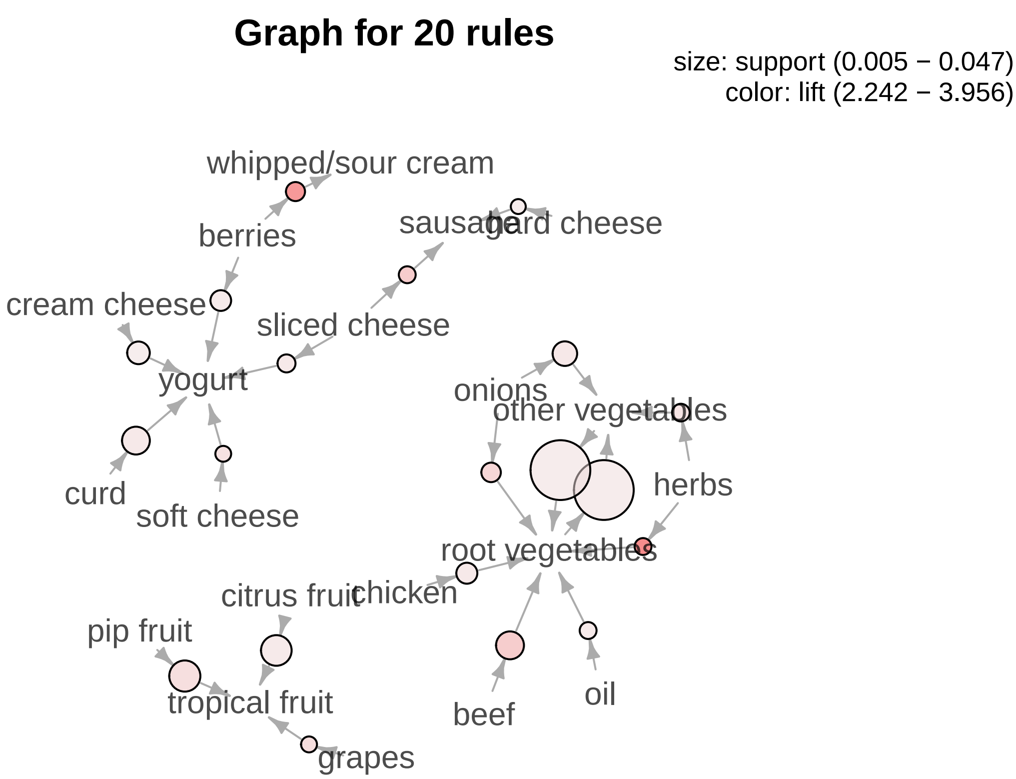
can now look at subsets driven by the plot

```
##      lhs                      rhs                  support     confidence
## [1]  {herbs}               => {root vegetables}    0.007015760 0.4312500
## [2]  {herbs}               => {other vegetables}   0.007727504 0.4750000
## [3]  {baking powder}       => {other vegetables}   0.007320793 0.4137931
## [4]  {baking powder}       => {whole milk}         0.009252669 0.5229885
## [5]  {soft cheese}         => {yogurt}             0.005998983 0.3511905
## [6]  {soft cheese}         => {other vegetables}   0.007117438 0.4166667
## [7]  {grapes}              => {tropical fruit}     0.006100661 0.2727273
## [8]  {grapes}              => {other vegetables}   0.009049314 0.4045455
## [9]  {meat}                => {sausage}            0.005287239 0.2047244
## [10] {hard cheese}         => {sausage}            0.005185562 0.2116183
## [11] {hard cheese}         => {root vegetables}    0.005592272 0.2282158
## [12] {butter milk}         => {yogurt}             0.008540925 0.3054545
## [13] {sliced cheese}       => {sausage}            0.007015760 0.2863071
## [14] {sliced cheese}       => {tropical fruit}     0.005287239 0.2157676
## [15] {sliced cheese}       => {root vegetables}    0.005592272 0.2282158
## [16] {sliced cheese}       => {yogurt}             0.008032537 0.3278008
## [17] {oil}                 => {root vegetables}    0.007015760 0.2500000
## [18] {onions}              => {root vegetables}    0.009456024 0.3049180
## [19] {onions}              => {other vegetables}   0.014234875 0.4590164
## [20] {berries}             => {whipped/sour cream} 0.009049314 0.2721713
## [21] {berries}             => {yogurt}             0.010574479 0.3180428
## [22] {hamburger meat}      => {other vegetables}   0.013828165 0.4159021
## [23] {cream cheese }       => {yogurt}             0.012404677 0.3128205
## [24] {chicken}             => {root vegetables}    0.010879512 0.2535545
## [25] {chicken}             => {other vegetables}   0.017895272 0.4170616
## [26] {frozen vegetables}   => {root vegetables}    0.011591256 0.2410148
## [27] {beef}                => {root vegetables}    0.017386884 0.3313953
## [28] {curd}                => {yogurt}             0.017285206 0.3244275
## [29] {pork}                => {root vegetables}    0.013624809 0.2363316
## [30] {butter}              => {root vegetables}    0.012913066 0.2330275
## [31] {domestic eggs}       => {root vegetables}    0.014336553 0.2259615
## [32] {whipped/sour cream} => {root vegetables}    0.017081851 0.2382979
## [33] {whipped/sour cream} => {yogurt}             0.020742247 0.2893617
## [34] {whipped/sour cream} => {other vegetables}   0.028876462 0.4028369
## [35] {pip fruit}           => {tropical fruit}     0.020437214 0.2701613
## [36] {citrus fruit}        => {tropical fruit}     0.019928826 0.2407862
## [37] {tropical fruit}      => {yogurt}             0.029283172 0.2790698
## [38] {yogurt}              => {tropical fruit}     0.029283172 0.2099125
## [39] {root vegetables}     => {other vegetables}   0.047381800 0.4347015
## [40] {other vegetables}    => {root vegetables}    0.047381800 0.2448765
##      coverage   lift     count
## [1]  0.01626843 3.956477  69
## [2]  0.01626843 2.454874  76
## [3]  0.01769192 2.138547  72
## [4]  0.01769192 2.046793  91
## [5]  0.01708185 2.517462  59
## [6]  0.01708185 2.153398  70
## [7]  0.02236909 2.599101  60
## [8]  0.02236909 2.090754  89
## [9]  0.02582613 2.179074  52
## [10] 0.02450432 2.252452  51
```

```
## [11] 0.02450432 2.093752  55
## [12] 0.02796136 2.189610  84
## [13] 0.02450432 3.047435  69
## [14] 0.02450432 2.056274  52
## [15] 0.02450432 2.093752  55
## [16] 0.02450432 2.349797  79
## [17] 0.02806304 2.293610  69
## [18] 0.03101169 2.797452  93
## [19] 0.03101169 2.372268 140
## [20] 0.03324860 3.796886  89
## [21] 0.03324860 2.279848 104
## [22] 0.03324860 2.149447 136
## [23] 0.03965430 2.242412 122
## [24] 0.04290798 2.326221 107
## [25] 0.04290798 2.155439 176
## [26] 0.04809354 2.211176 114
## [27] 0.05246568 3.040367 171
## [28] 0.05327911 2.325615 170
## [29] 0.05765125 2.168210 134
## [30] 0.05541434 2.137897 127
## [31] 0.06344687 2.073071 141
## [32] 0.07168277 2.186250 168
## [33] 0.07168277 2.074251 204
## [34] 0.07168277 2.081924 284
## [35] 0.07564820 2.574648 201
## [36] 0.08276563 2.294702 196
## [37] 0.10493137 2.000475 288
## [38] 0.13950178 2.000475 288
## [39] 0.10899847 2.246605 466
## [40] 0.19349263 2.246605 466
```

graph-based visualization





(3) Conclusion

interesting rules here: {herbs} => {root vegetables}, {sliced cheese} => {sausage}, {berries} => {whipped/sour cream}, {beef} => {root vegetables} and so on

4. Author attribution

(1) Importing data files in ReutersC50 folder

Initially we browsed one of the reader functions in tm library, and imported 2,500 text files from the ReutersC50 file.

After making the labels name concise, we created a text mining corpus with training data set.

By the pre-processing(tokenixation), we got rid of capital letters, numbers, punctuation, excess white-space, and stop words. In addition, sparse terms which have count 0 in more than 95% of documents also were removed. Then, we created a doc-term-matrix for a training set.

Next, we stepped the same pre-process for the test set. Especially, test-set vocabulary was restricted to the terms in DTM_train to enhance the prediction's accuracy.

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 2500
```

(2-1) Set operations with testing set

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 2500
```

(2-2) Create training and testing feature matrices and restrict test-set vocabulary to the terms in DTM_train

```
## <<DocumentTermMatrix (documents: 2500, terms: 31423)>>
## Non-/sparse entries: 425955/78131545
## Sparsity           : 99%
## Maximal term length: 36
## Weighting          : term frequency (tf)
```

```
## <<DocumentTermMatrix (documents: 6, terms: 641)>>
## Non-/sparse entries: 246/3600
## Sparsity           : 94%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
## <<DocumentTermMatrix (documents: 6, terms: 641)>>
## Non-/sparse entries: 382/3464
## Sparsity           : 90%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

(3) Fit prediction model

Before fitting prediction models, we set the outcome vector for all authors. We could represents all the authors as the number 1 to 50 through this process.

(3-1) Outcome vector for 50 authors

```
y_train = 0 + {labels_train=='AaronPressman'} + 2*{labels_train=='AlanCrosby'} + 3*{labels_train=='Alexa
  4*{labels_train=='BenjaminKangLim'} + 5*{labels_train=='BernardHickey'} + 6*{labels_train=='BradDorfma
  7*{labels_train=='DarrenSchuettler'} + 8*{labels_train=='DavidLawder'} + 9*{labels_train=='EdnaFernand
  10*{labels_train=='EricAuchard'} + 11*{labels_train=='FumikoFujisaki'} + 12*{labels_train=='GrahamEarr
  13*{labels_train=='HeatherScoffield'} + 14*{labels_train=='JaneMacartney'} + 15*{labels_train=='JanLop
  16*{labels_train=='JimGilchrist'} + 17*{labels_train=='JoeOrtiz'} + 18*{labels_train=='JohnMastrini'}
  19*{labels_train=='JonathanBirt'} + 20*{labels_train=='JoWinterbottom'} + 21*{labels_train=='KarlPenha
  22*{labels_train=='KeithWeir'} + 23*{labels_train=='KevinDrawbaugh'} + 24*{labels_train=='KevinMorriso
  25*{labels_train=='KristinRidley'} + 26*{labels_train=='KouroshKarimkhany'} + 27*{labels_train=='Lydia
  28*{labels_train=='LynneODonell'} + 29*{labels_train=='LynnleyBrowning'} + 30*{labels_train=='MarcelMi
  31*{labels_train=='MarkBendeich'} + 32*{labels_train=='MartinWolk'} + 33*{labels_train=='MattewBunce'}
  34*{labels_train=='MichaelConnor'} + 35*{labels_train=='MureDickie'} + 36*{labels_train=='NickLouth'}
  37*{labels_train=='PatriciaCommins'} + 38*{labels_train=='PeterHumphrey'} + 39*{labels_train=='Pierre
  40*{labels_train=='RobinSidel'} + 41*{labels_train=='RogerFillion'} + 42*{labels_train=='SamuelPerry
  43*{labels_train=='SarahDavison'} + 44*{labels_train=='ScottHillis'} + 45*{labels_train=='SimonCowell
  46*{labels_train=='TanEeLyn'} + 47*{labels_train=='TheresePoletti'} + 48*{labels_train=='TimFarrand'}
  49*{labels_train=='ToddNissen'} + 50*{labels_train=='WilliamKazer'}

y_test = 0 + {labels_test=='AaronPressman'} + 2*{labels_test=='AlanCrosby'} + 3*{labels_test=='Alexande
  4*{labels_test=='BenjaminKangLim'} + 5*{labels_test=='BernardHickey'} + 6*{labels_test=='BradDorfman'}
  7*{labels_test=='DarrenSchuettler'} + 8*{labels_test=='DavidLawder'} + 9*{labels_test=='EdnaFernandes
  10*{labels_test=='EricAuchard'} + 11*{labels_test=='FumikoFujisaki'} + 12*{labels_test=='GrahamEarnsha
  13*{labels_test=='HeatherScoffield'} + 14*{labels_test=='JaneMacartney'} + 15*{labels_test=='JanLopat
  16*{labels_test=='JimGilchrist'} + 17*{labels_test=='JoeOrtiz'} + 18*{labels_test=='JohnMastrini'} +
  19*{labels_test=='JonathanBirt'} + 20*{labels_test=='JoWinterbottom'} + 21*{labels_test=='KarlPenhaul
  22*{labels_test=='KeithWeir'} + 23*{labels_test=='KevinDrawbaugh'} + 24*{labels_test=='KevinMorrison'}
  25*{labels_test=='KristinRidley'} + 26*{labels_test=='KouroshKarimkhany'} + 27*{labels_test=='LydiaZa
  28*{labels_test=='LynneODonell'} + 29*{labels_test=='LynnleyBrowning'} + 30*{labels_test=='MarcelMiche
  31*{labels_test=='MarkBendeich'} + 32*{labels_test=='MartinWolk'} + 33*{labels_test=='MattewBunce'} +
  34*{labels_test=='MichaelConnor'} + 35*{labels_test=='MureDickie'} + 36*{labels_test=='NickLouth'} +
  37*{labels_test=='PatriciaCommins'} + 38*{labels_test=='PeterHumphrey'} + 39*{labels_test=='PierreTra
  40*{labels_test=='RobinSidel'} + 41*{labels_test=='RogerFillion'} + 42*{labels_test=='SamuelPerry'} +
  43*{labels_test=='SarahDavison'} + 44*{labels_test=='ScottHillis'} + 45*{labels_test=='SimonCowell'} +
  46*{labels_test=='TanEeLyn'} + 47*{labels_test=='TheresePoletti'} + 48*{labels_test=='TimFarrand'} +
  49*{labels_test=='ToddNissen'} + 50*{labels_test=='WilliamKazer'}
```

(3-2) Lasso logistic regression for document classification

Next, we fitted logit prediction models. The first model(logit1) had the default condition and the second one(logit2) had gaussian.

We predicted each authors of every documents with these two models.

y_test has a form of "list", while yhat-test_final1 has vector, so we converted form of y_test from list to matrix form to compare the prediction results and original authors.

Convert form of y_test list-> df->matrix

convert form of yhat_test from vector to matrix

(3-3) Comparison of test set and prediction

The two models' accuracy from the confusion matrix were around 2.12%.

logit1

logit2