HW 4

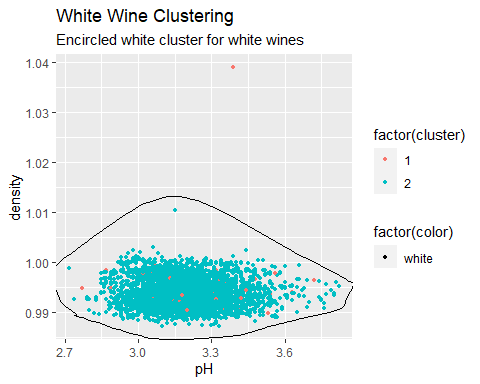
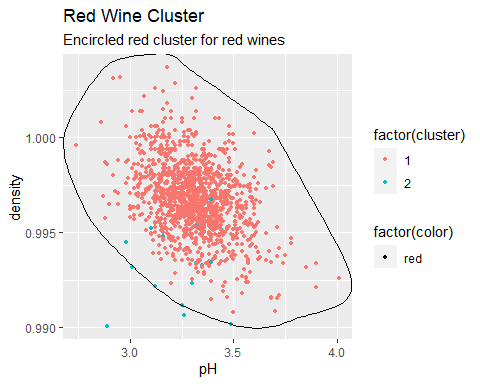
Zayd Abdalla, Areeya Aksornpan

5/6/2021

#Question 1: Clustering and PCA for wine

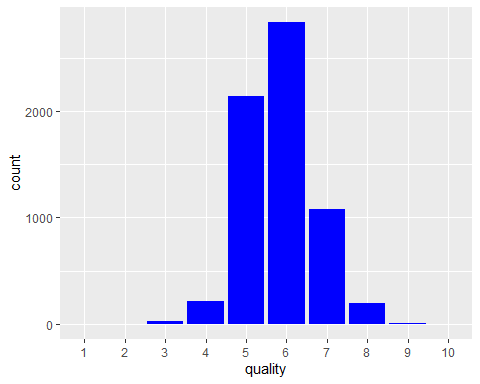
##Cluster We began by clustering to explore if clustering could distinguish between reds and whites along with different levels of quality. We removed the color and quality columns and rescaled other variables for unsupervised analysis.

We utilized 2 centers to test the performance of clustering in distinguishing between wine color.

##Color 

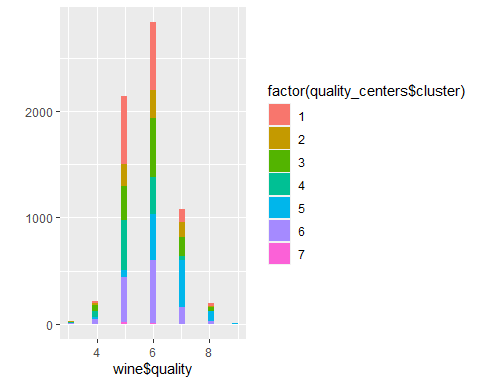
## wine1$color  
## cluster\_2\_centers$cluster red white  
## 1 1575 68  
## 2 24 4830

The plots show us that for each cluster, the points within the clusters overlap well with the wine color. We support this with a confusion matrix, where the accuracy rate is 0.0141604. As a result, our clustering is capable of distinguishing reds from the whites by using 2 centers and pH (chemical properties).

##Quality 

There are no 1,2 and 10 categories, so we opted for 7 clusters corresponding to quality rankings from 3-9.

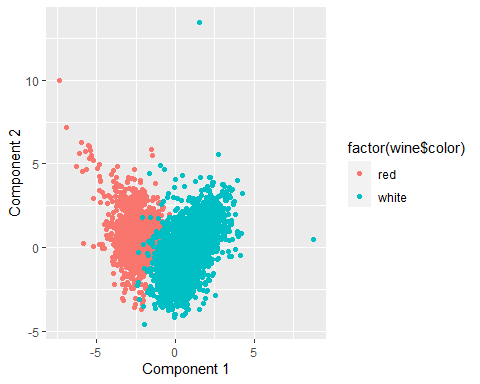
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



## wine$quality  
## quality\_centers$cluster 3 4 5 6 7 8 9  
## 1 7 24 641 637 122 22 1  
## 2 4 15 202 266 139 14 0  
## 3 4 53 320 553 181 36 0  
## 4 7 63 466 344 41 2 0  
## 5 3 14 67 440 438 94 4  
## 6 4 45 422 587 157 25 0  
## 7 1 2 20 9 1 0 0

We created a table that shows us each cluster and the quality levels of wine. Many of the clusters contained mostly wines in the 5-6 or 5-7 range, but in this case, it doesn’t seem our clustering was able to distinguish lower quality from higher quality wines like we did with color.

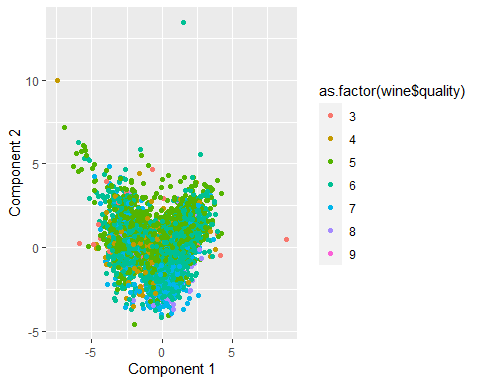
##PCA

 We began by search for the most important components (i.e. have the highest proportion of variance among features) and draw a graph of the PC from highest variance to the lowest variance. We found that PC1 and PC2 are the most important features with variances of 0.2754 and 0.2267, respectively. Once again, we choose K=2 as we did when clustering. From the graph above, we can clearly see PCA does quite well in distinguishing reds from whites. There are two main clusters with very little overlap.

We create another confusion matrix and observe that With 0.9836848 < 0.0141604, we find that eliminating dimensions prior to clustering performed worse than just using K-mean clustering.

Move on to attempt at distinguish wine quality, first we conduct PCA on rescaled wine data.

Similar to simple K-mean clustering, PCA does not perform well in distinguishing wines with different quality levels. The graph is blurry. Different quality levels of wine center in the same area with the similar component 1/component 2 variance.



Next, we apply PCA before trying to conduct a 7 cluster. However, as the graph below represents, it does not help us to distinguish between different quality of wine much better than just PCA.

## wine$quality  
## PCA\_cluster2$cluster 3 4 5 6 7 8 9  
## 1 7 59 365 275 36 2 0  
## 2 0 19 263 248 98 8 0  
## 3 6 19 484 467 94 17 0  
## 4 4 5 90 121 54 6 0  
## 5 5 53 305 566 236 43 0  
## 6 6 39 494 614 128 24 1  
## 7 2 22 137 545 433 93 4

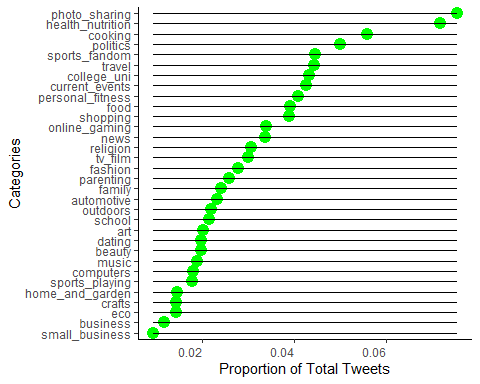
As observed from the table above, the clusters misidentified several observations, rendering it unhelpful in distinguishing wine quality.

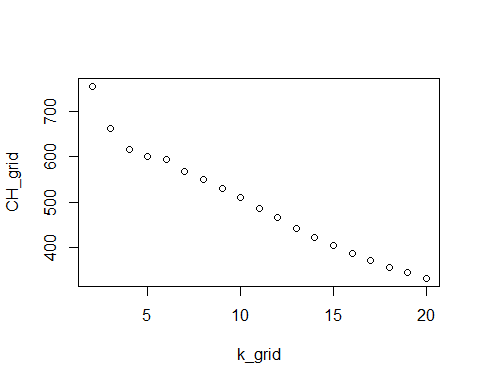
#Question 2: Market segmentation

##Data Cleaning

From the graph below, we see that photo sharing and health nutrition rank as the two most popular categories for tweets. But there is still a wide range of the other tweet categories that may be worth considering in order to broaden our market. Intuitively, we chose to cluster in order to group similar followers and their tweets to better utilize market segments for product promotion.

## Tweet Plot by Category

 ##Rescale data and K grid-CH grid plot

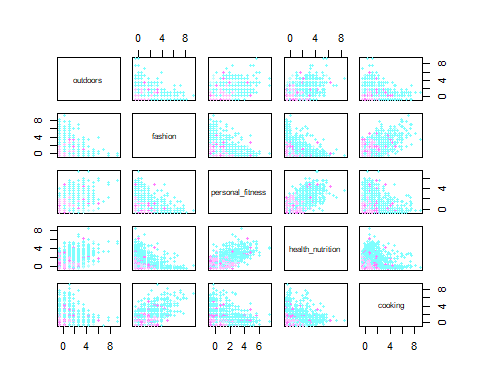
 For our methods, we searched for an optimal K to cluster by. We utilized a CH Index and found that K=2 has a max CH. But it didn’t make sense conceptually to only observe 2 clusters of market segments for NutrientH20 (leaving several markets untapped felt wrong), so we opted for 4 clusters to allow us to account for more market segments we can observe (i.e. more potential profits).

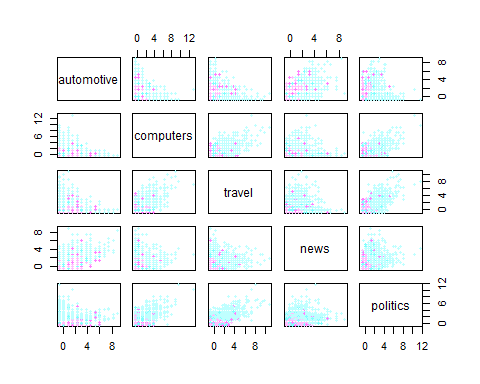
which.max(CH\_grid)

## [1] 1

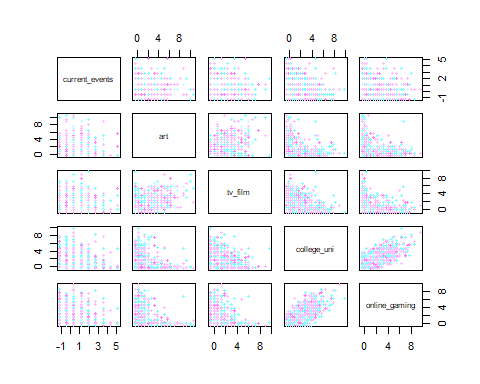
k\_grid[which.max(CH\_grid)]

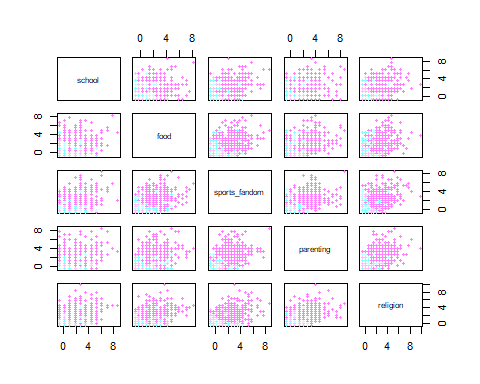
## [1] 2

 This first market segment holds many interests—such as fitness, nutrition, and the outdoors—that pertain to active, healthy adults. We believe that for this market, NutrientH20 ought to market healthy or eco-friendly aspects of their products to attract this segment to their brand. Furthermore, hydration is a key aspect of activity and being in the outdoors, which the company can capitalize on.



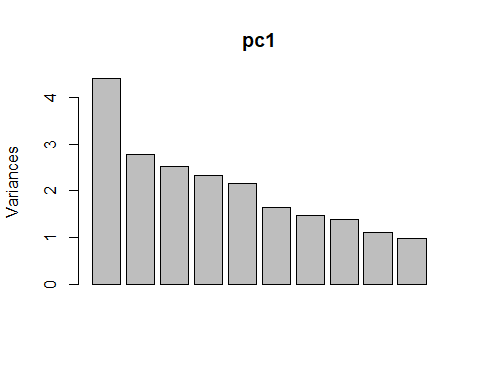
This next market segment contains interests—such as gaming, college, and tv/films—that appeal to college age students who enjoy gaming and media. With this group, advertising their products on twitch, which is a popular gaming-stream service, may help attract this segment to the brand. Additionally, appearing in advertisements on college campuses or media can help bolster the name recognition of this brand with this age group, potentially increasing profits.

 This cluster contains interests—such as parenting, religion, sports, and food—which all appeal to parents and more traditional/conservative Americans. The importance of this market segment is that marketing to one twitter user is potentially marketing to that user’s family due to the nature of proximity. Furthermore, appealing to sports fans can lead to potential collaborations with sports leagues that these fans watch, which can improve brand appeal and consequently profits.

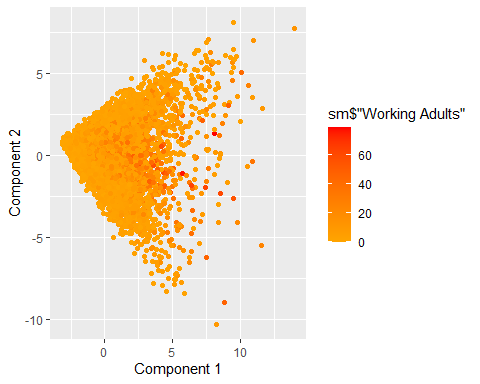
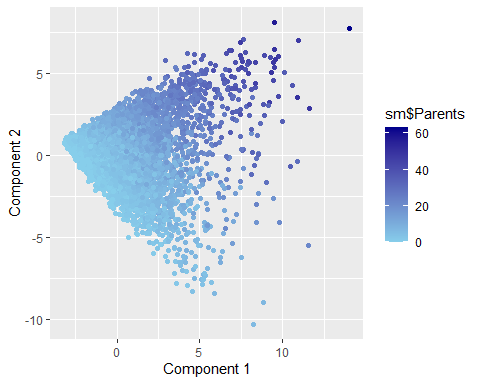
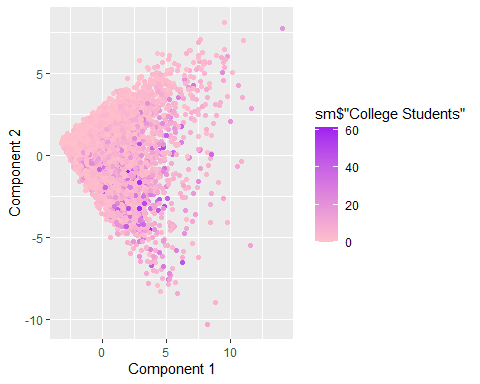
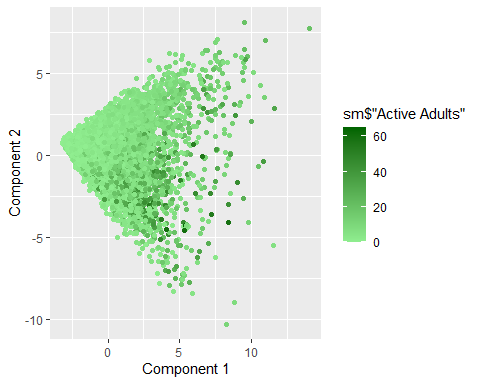


This market segment contains topics—such as automotive, travel, news, and politics—that would appeal to older working adults. This demographic is (generally) actively earning higher incomes and are concerned with the latest information on new, politics, and cars. Tapping into this market successfully could be very rewarding due to the nature of the income for this segment compared to younger ones.

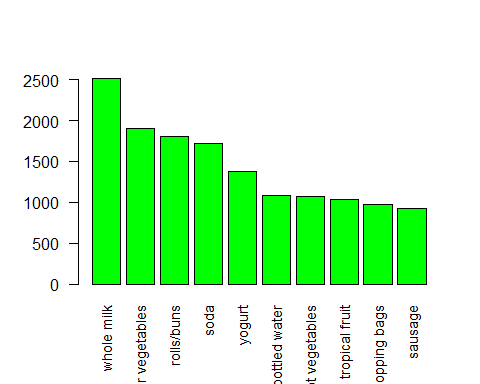
Young college students that game or are into media seem to make up the majority of the tweets from these groups, which may be the main group of interest for NutrientH20 to focus on marketing to, but not the only one.

 ## Fun, Informative Visuals on Market Segments

For the 4 graphs below, we utilize the two most important components and they each represent a twitter user of a certain demographic (market segment). Visually, the darker the color, the more a user’s tweets relate to the topics attached to each demographic.



#Question 3: Association rules for grocery purchases

##Observe distribution of data 

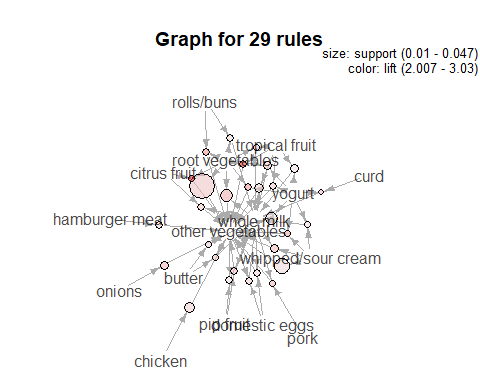
## sup con avg\_inspection  
## 31 0.009 0.50 2.225524  
## 27 0.019 0.45 2.014596  
## 32 0.019 0.50 2.007235  
## 22 0.019 0.40 1.863106  
## 28 0.029 0.45 1.850203  
## 23 0.029 0.40 1.793654  
## 24 0.039 0.40 1.787991  
## 17 0.019 0.35 1.767332  
## 19 0.039 0.35 1.733120  
## 12 0.019 0.30 1.730407

From the table above, the findings that are best for support = 0.009 and confidence = 0.5 with a max average lift of 2.225524. Increasing the value of support is associated with higher sales including items of interest. However, we observe that there is a cost, which is the decrease in lift values. A slightly larger support value, however, would allow for more transactions and rules, but with a smaller effect on lift.

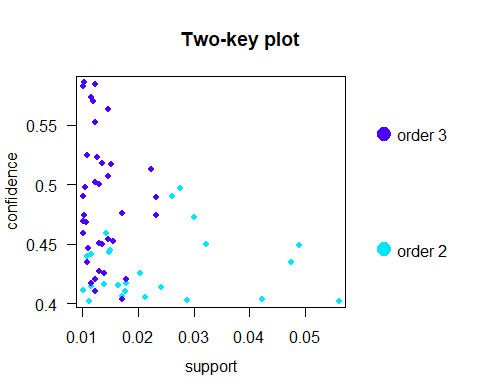
## lhs rhs support   
## [1] {onions} => {other vegetables} 0.01423488  
## [2] {hamburger meat} => {other vegetables} 0.01382816  
## [3] {chicken} => {other vegetables} 0.01789527  
## [4] {whipped/sour cream} => {other vegetables} 0.02887646  
## [5] {root vegetables} => {other vegetables} 0.04738180  
## [6] {curd,yogurt} => {whole milk} 0.01006609  
## [7] {pork,whole milk} => {other vegetables} 0.01016777  
## [8] {butter,other vegetables} => {whole milk} 0.01148958  
## [9] {butter,whole milk} => {other vegetables} 0.01148958  
## [10] {domestic eggs,other vegetables} => {whole milk} 0.01230300  
## [11] {domestic eggs,whole milk} => {other vegetables} 0.01230300  
## [12] {whipped/sour cream,yogurt} => {other vegetables} 0.01016777  
## [13] {whipped/sour cream,yogurt} => {whole milk} 0.01087951  
## [14] {whipped/sour cream,whole milk} => {other vegetables} 0.01464159  
## [15] {other vegetables,pip fruit} => {whole milk} 0.01352313  
## [16] {pip fruit,whole milk} => {other vegetables} 0.01352313  
## [17] {citrus fruit,root vegetables} => {other vegetables} 0.01037112  
## [18] {citrus fruit,whole milk} => {other vegetables} 0.01301474  
## [19] {root vegetables,tropical fruit} => {other vegetables} 0.01230300  
## [20] {root vegetables,tropical fruit} => {whole milk} 0.01199797  
## [21] {tropical fruit,yogurt} => {other vegetables} 0.01230300  
## [22] {tropical fruit,yogurt} => {whole milk} 0.01514997  
## [23] {tropical fruit,whole milk} => {other vegetables} 0.01708185  
## [24] {root vegetables,yogurt} => {other vegetables} 0.01291307  
## [25] {root vegetables,yogurt} => {whole milk} 0.01453991  
## [26] {rolls/buns,root vegetables} => {other vegetables} 0.01220132  
## [27] {rolls/buns,root vegetables} => {whole milk} 0.01270971  
## [28] {root vegetables,whole milk} => {other vegetables} 0.02318251  
## [29] {other vegetables,yogurt} => {whole milk} 0.02226741  
## confidence coverage lift count  
## [1] 0.4590164 0.03101169 2.372268 140   
## [2] 0.4159021 0.03324860 2.149447 136   
## [3] 0.4170616 0.04290798 2.155439 176   
## [4] 0.4028369 0.07168277 2.081924 284   
## [5] 0.4347015 0.10899847 2.246605 466   
## [6] 0.5823529 0.01728521 2.279125 99   
## [7] 0.4587156 0.02216573 2.370714 100   
## [8] 0.5736041 0.02003050 2.244885 113   
## [9] 0.4169742 0.02755465 2.154987 113   
## [10] 0.5525114 0.02226741 2.162336 121   
## [11] 0.4101695 0.02999492 2.119820 121   
## [12] 0.4901961 0.02074225 2.533410 100   
## [13] 0.5245098 0.02074225 2.052747 107   
## [14] 0.4542587 0.03223183 2.347679 144   
## [15] 0.5175097 0.02613116 2.025351 133   
## [16] 0.4493243 0.03009659 2.322178 133   
## [17] 0.5862069 0.01769192 3.029608 102   
## [18] 0.4266667 0.03050330 2.205080 128   
## [19] 0.5845411 0.02104728 3.020999 121   
## [20] 0.5700483 0.02104728 2.230969 118   
## [21] 0.4201389 0.02928317 2.171343 121   
## [22] 0.5173611 0.02928317 2.024770 149   
## [23] 0.4038462 0.04229792 2.087140 168   
## [24] 0.5000000 0.02582613 2.584078 127   
## [25] 0.5629921 0.02582613 2.203354 143   
## [26] 0.5020921 0.02430097 2.594890 120   
## [27] 0.5230126 0.02430097 2.046888 125   
## [28] 0.4740125 0.04890696 2.449770 228   
## [29] 0.5128806 0.04341637 2.007235 219

Subset for rules with lifts values greater than 2 because the mean is roughly that value and eliminate weakly associated rules. There remaains 29 strongly associated rules. From the sample, whole milk appears the most followed by other vegetables.

## Available control parameters (with default values):  
## main = Graph for 29 rules  
## max = 100  
## nodeCol = c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF", "#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF", "#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF", "#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF", "#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF", "#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF", "#EE7878FF", "#EE7B7BFF", "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DFF", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBCBCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC6C6FF", "#EEC8C8FF", "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#EED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")  
## itemnodeCol = #66CC66FF  
## edgeCol = #ABABABFF  
## labelCol = #000000B3  
## itemLabels = TRUE  
## measureLabels = FALSE  
## precision = 3  
## arrowSize = 0.5  
## alpha = 0.5  
## cex = 1  
## layout = NULL  
## layoutParams = list()  
## engine = igraph  
## plot = TRUE  
## plot\_options = list()  
## verbose = FALSE



The visualization above illustrates importance of basket items, with Whole milk and other vegetables ranking as the most common items.

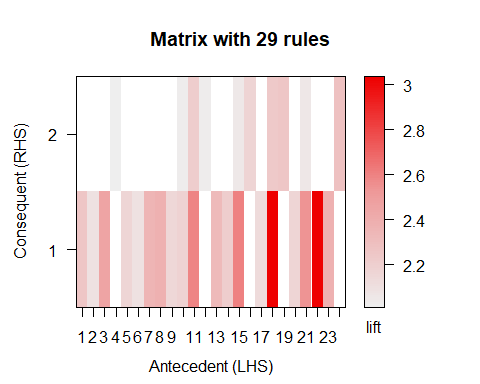


The graph above is a two-key plot for all values as a function of support and confidence below.

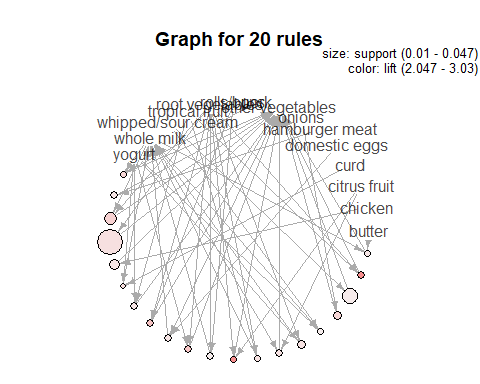
Below we have a matrix representations of the association rules, allowing us to match our matrix to lift values above and obtain the grocery items.

subrules <- sample(subset\_groc, 20)  
plot(subset\_groc, method="matrix", measure="lift", control=list(reorder='support/confidence'))

## Itemsets in Antecedent (LHS)  
## [1] "{root vegetables}" "{whipped/sour cream}"   
## [3] "{root vegetables,whole milk}" "{other vegetables,yogurt}"   
## [5] "{chicken}" "{tropical fruit,whole milk}"   
## [7] "{whipped/sour cream,whole milk}" "{onions}"   
## [9] "{hamburger meat}" "{tropical fruit,yogurt}"   
## [11] "{root vegetables,yogurt}" "{other vegetables,pip fruit}"   
## [13] "{pip fruit,whole milk}" "{citrus fruit,whole milk}"   
## [15] "{rolls/buns,root vegetables}" "{domestic eggs,other vegetables}"  
## [17] "{domestic eggs,whole milk}" "{root vegetables,tropical fruit}"  
## [19] "{butter,other vegetables}" "{butter,whole milk}"   
## [21] "{whipped/sour cream,yogurt}" "{citrus fruit,root vegetables}"   
## [23] "{pork,whole milk}" "{curd,yogurt}"   
## Itemsets in Consequent (RHS)  
## [1] "{other vegetables}" "{whole milk}"



plot(subrules, method="graph", control=list(layout=igraph::in\_circle()))



#Question 4.Author Attribution

## Author Attribution

## Collect data  
  
#Training data  
train\_data <- readtext(Sys.glob('~/Documents/GitHub/ECO395M/data/ReutersC50/C50train/\*'))  
# head(train\_data$text, n = 1)  
  
#Testing data  
test\_data <- readtext(Sys.glob('~/Documents/GitHub/ECO395M/data/ReutersC50/C50test/\*'))

#Author names  
author\_names <- as.data.frame(rep(basename(list.dirs('~/Documents/GitHub/ECO395M/data/ReutersC50/C50train')), each = 50))  
author\_names <- author\_names[-(1:50),]  
  
#Assign author name to Text  
test\_data$author <- author\_names  
train\_data$author <- author\_names  
  
#Dropping ID column  
test\_data <- test\_data[-1]  
train\_data <- train\_data[-1]  
  
#Converting author column to factor  
test\_data$author <- as.factor(test\_data$author)  
train\_data$author <- as.factor(train\_data$author)  
  
table(train\_data$author) %>% kbl("pipe")

|  |  |
| --- | --- |
| Var1 | Freq |
| AaronPressman | 50 |
| AlanCrosby | 50 |
| AlexanderSmith | 50 |
| BenjaminKangLim | 50 |
| BernardHickey | 50 |
| BradDorfman | 50 |
| DarrenSchuettler | 50 |
| DavidLawder | 50 |
| EdnaFernandes | 50 |
| EricAuchard | 50 |
| FumikoFujisaki | 50 |
| GrahamEarnshaw | 50 |
| HeatherScoffield | 50 |
| JaneMacartney | 50 |
| JanLopatka | 50 |
| JimGilchrist | 50 |
| JoeOrtiz | 50 |
| JohnMastrini | 50 |
| JonathanBirt | 50 |
| JoWinterbottom | 50 |
| KarlPenhaul | 50 |
| KeithWeir | 50 |
| KevinDrawbaugh | 50 |
| KevinMorrison | 50 |
| KirstinRidley | 50 |
| KouroshKarimkhany | 50 |
| LydiaZajc | 50 |
| LynneO’Donnell | 50 |
| LynnleyBrowning | 50 |
| MarcelMichelson | 50 |
| MarkBendeich | 50 |
| MartinWolk | 50 |
| MatthewBunce | 50 |
| MichaelConnor | 50 |
| MureDickie | 50 |
| NickLouth | 50 |
| PatriciaCommins | 50 |
| PeterHumphrey | 50 |
| PierreTran | 50 |
| RobinSidel | 50 |
| RogerFillion | 50 |
| SamuelPerry | 50 |
| SarahDavison | 50 |
| ScottHillis | 50 |
| SimonCowell | 50 |
| TanEeLyn | 50 |
| TheresePoletti | 50 |
| TimFarrand | 50 |
| ToddNissen | 50 |
| WilliamKazer | 50 |
|  |  |

#Create corpus  
test\_cp <- Corpus(VectorSource(test\_data$text))  
train\_cp <- Corpus(VectorSource(train\_data$text))  
  
#Clean corpus  
test\_cp <-  
 test\_cp %>%  
 tm\_map(., content\_transformer(tolower)) %>%  
 tm\_map(., content\_transformer(removeNumbers)) %>%  
 tm\_map(., content\_transformer(removePunctuation)) %>%  
 tm\_map(., content\_transformer(stripWhitespace)) %>%  
 tm\_map(., content\_transformer(removeWords), stopwords("SMART"))  
  
#inspect(test\_cp[1])  
wordcloud(test\_cp, min.freq = 40, random.order = FALSE)



train\_cp <-  
 train\_cp %>%  
 tm\_map(., content\_transformer(tolower)) %>%  
 tm\_map(., content\_transformer(removeNumbers)) %>%  
 tm\_map(., content\_transformer(removePunctuation)) %>%  
 tm\_map(., content\_transformer(stripWhitespace)) %>%  
 tm\_map(., content\_transformer(removeWords), stopwords("SMART"))

#Document term matrix (sparse matrices)  
test\_dtm <- DocumentTermMatrix(test\_cp)  
train\_dtm <- DocumentTermMatrix(train\_cp)

#Document term matrix (sparse matrices)  
test\_dtm <- DocumentTermMatrix(test\_cp)  
train\_dtm <- DocumentTermMatrix(train\_cp)  
  
#inspect(train\_dtm)

## Naive Bayes Classification

freq\_words <- findFreqTerms(train\_dtm, 5)

# saving List using Dictionary() Function

Dictionary <- function(x) {

if (is.character(x)) {

return(x)

}

stop('x is not a character vector')

}

data\_dict <- Dictionary(findFreqTerms(train\_dtm, 5))

# appending Document Term Matrix to Train and Test Dataset

data\_train <- DocumentTermMatrix(train\_corpus, list(data\_dict))

data\_test <- DocumentTermMatrix(test\_corpus, list(data\_dict))

# converting the frequency of word to count

convert\_counts <- function(x) {

x <- ifelse(x > 0, 1, 0)

x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))

return(x)

}

# appending count function to Train and Test Dataset

data\_train <- apply(data\_train, MARGIN = 2, convert\_counts)

data\_test <- apply(data\_test, MARGIN = 2, convert\_counts)

# train model

data\_classifier <- naiveBayes(data\_train, Data\_train$author)

data\_test\_pred <- predict(data\_classifier, data\_test)

final\_df <-

tibble(

"predicted" = data\_test\_pred,

"actual" = Data\_test$author

)

num\_correct <-

final\_df %>%

mutate(correct = if\_else(predicted == actual, 1, 0)) %>%

pull(correct) %>%

sum()

num\_rows <- final\_df %>% nrow()

num\_correct / num\_rows

## [1] 0.6724

From a sample of 50 authors, our model predicted correctly with 67.24% accuracy.