Yelp Project

levinemi

14/05/2020

##Where should we take the kids? An analysis of family-friendly businesses in the Yelp dataset The amount of time that parents devote to their children has been increasing steadily since the 1990s. The amount of money parents spend on their children has also increased by more than 50% from 1972 to 2007. So, targeting families could be a good strategy to woo customers. Identifying what makes a business family-friendly may help attract families to a new business or guide improvements to an established business. Yelp makes a dataset of crowd-sourced reviews about businesses available for educational and academic purposes. The dataset includes information about more than 200,000 businesses worldwide, including business type, location and amenities as well as customer reviews. The proposed project is an exploratory analysis of businesses that market themselves as family-friendly. Using machine learning, I aim to identify common features, explore customer sentiment and predict business ratings.

#Load Packages

library(arules) #this will mask recode from dplyr so need to load before dplyr  
library(arulesViz)  
library(tidytext)  
library(tidyr)  
library(widyr)  
library(purrr)  
library(stringr)  
library(ggplot2)  
library(igraph)  
library(ggraph)  
library(topicmodels)  
library(SnowballC)  
library(summarytools)  
library(caret)  
library(gmodels)  
library(FSelector)  
library(leaps)  
library(RColorBrewer)  
library(data.table)  
library(textdata)  
library(textstem)  
library(PMCMR)  
library(sentimentr)  
#library(tm) to be loaded later so as not to interact with arules package.  
library(wordcloud)  
library(reshape2)  
library(mice)  
library(VIM)  
library(corrplot)  
#library(car) to be loaded later so as not to interact with arules package  
library(ROCit)  
library(partykit)  
library(plyr);library(dplyr)

#Load Business Dataset

Set the working directory for the script.

setwd("~/YELP")

The data about Yelp Businesses was originally loaded from a json file and then was exported to an .Rdata format to make future data loads faster.

#Load and flatten 'businesses' collection  
  
#business\_raw <- stream\_in(file("yelp\_academic\_dataset\_business.json"))  
#business\_flat <- flatten(business\_raw, recursive = T)  
  
#export the business\_flat collection to .Rdata  
#save(business\_flat, file = "business.RData")

Load the Yelp business data file from the .Rdata file.

business\_df <- load("Data/business.RData")  
business\_df <- as.data.frame(business\_flat)  
rm(business\_flat)

Create a subset of the data focused on businesses that have the flag “good\_for\_kids”.

#Frenquency table for "good for kids" attributes, which shows all possible values and counts for each value.  
summarytools::freq(business\_df$attributes.GoodForKids)

## Frequencies   
## business\_df$attributes.GoodForKids   
## Type: Character   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- -------- --------- -------------- --------- --------------  
## False 12932 18.87 18.87 6.18 6.18  
## None 76 0.11 18.98 0.04 6.21  
## True 55527 81.02 100.00 26.52 32.73  
## <NA> 140858 67.27 100.00  
## Total 209393 100.00 100.00 100.00 100.00

#Filter the business dataset for businesses with the attribute "good for kids"  
business\_subset <- business\_df %>%   
 filter(attributes.GoodForKids == "True")

# Exploratory Analysis of Businesses that are “good for kids”.

*What are the characteristics of the “good for kids” businesses?* *Location:*

#Number of cities  
business\_subset %>%   
 distinct(city) %>%   
 count()

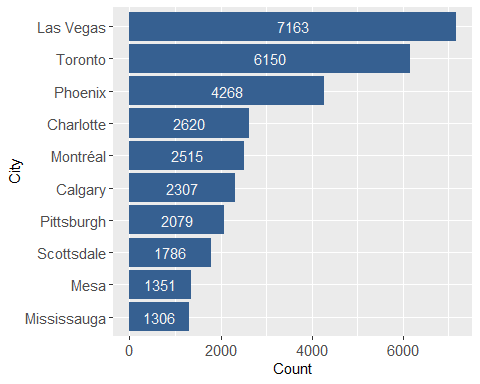
## n  
## 1 765

#Number of states or provinces  
business\_subset %>%   
 distinct(state) %>%   
 count()

## n  
## 1 22

summarytools::freq(business\_subset$state, order = "freq")

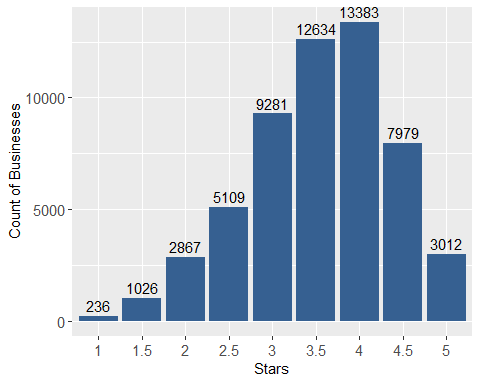
## Frequencies   
## business\_subset$state   
## Type: Character   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- ------- --------- -------------- --------- --------------  
## AZ 12938 23.30 23.30 23.30 23.30  
## ON 12254 22.07 45.37 22.07 45.37  
## NV 8821 15.89 61.25 15.89 61.25  
## OH 4816 8.67 69.93 8.67 69.93  
## NC 4328 7.79 77.72 7.79 77.72  
## QC 3909 7.04 84.76 7.04 84.76  
## PA 3473 6.25 91.02 6.25 91.02  
## AB 2391 4.31 95.32 4.31 95.32  
## WI 1546 2.78 98.11 2.78 98.11  
## IL 608 1.09 99.20 1.09 99.20  
## SC 417 0.75 99.95 0.75 99.95  
## NY 10 0.02 99.97 0.02 99.97  
## CA 4 0.01 99.98 0.01 99.98  
## TX 3 0.01 99.98 0.01 99.98  
## CO 2 0.00 99.99 0.00 99.99  
## AR 1 0.00 99.99 0.00 99.99  
## BC 1 0.00 99.99 0.00 99.99  
## FL 1 0.00 99.99 0.00 99.99  
## HI 1 0.00 99.99 0.00 99.99  
## MB 1 0.00 100.00 0.00 100.00  
## OR 1 0.00 100.00 0.00 100.00  
## VA 1 0.00 100.00 0.00 100.00  
## <NA> 0 0.00 100.00  
## Total 55527 100.00 100.00 100.00 100.00



*Stars:*

#Stars needs to be converted to a factor  
business\_subset$stars <- as.factor(business\_subset$stars)  
  
#Number of businesses with NA for stars  
sum(is.na(business\_subset$stars))

## [1] 0



*Reviews:*

#Number of reviews  
business\_subset %>%   
 count(review\_count) %>%   
 summarise(sum(review\_count\*n))

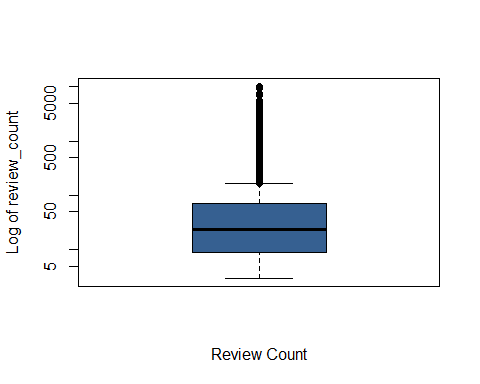
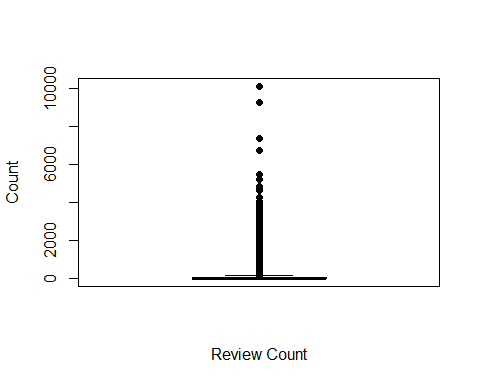
## sum(review\_count \* n)  
## 1 4279745

#average number of reviews per business  
business\_subset %>%   
 summarise(round(mean(review\_count),2))

## round(mean(review\_count), 2)  
## 1 77.08

#distribution of number of reviews  
summary(business\_subset$review\_count)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.00 9.00 24.00 77.08 71.00 10129.00



#number of outliers  
length(boxplot.stats(business\_subset$review\_count)$out)

## [1] 6119

The outliers will be addressed prior to the sentiment analysis

*Open:*

#Is open  
summarytools::freq(business\_subset$is\_open)

## Frequencies   
## business\_subset$is\_open   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- ------- --------- -------------- --------- --------------  
## 0 15212 27.40 27.40 27.40 27.40  
## 1 40315 72.60 100.00 72.60 100.00  
## <NA> 0 0.00 100.00  
## Total 55527 100.00 100.00 100.00 100.00

# Exploratory Analysis of Categories & Names

Using the business categories and business names, *what can we learn about the businesses that are “good for kids”?*

Start by creating separate dataframes for the business names and the business categories. The business\_id (“ID”) is retained in each dataframe as a primary key to link back to the complete business\_subset.

name\_df <- tibble(business\_subset$business\_id, business\_subset$name)  
colnames(name\_df) <- c("ID", "name")  
   
category\_df <- tibble(business\_subset$business\_id, business\_subset$categories)  
colnames(category\_df) <- c("ID", "categories")  
  
#normalize the text to lowercase  
name\_df$Name <- tolower(name\_df$name)  
category\_df$categories <- tolower(category\_df$categories)

The category attribute contains a string with one or more categories. Each category is separated by a comma. To create a tidy dataframe, I split the category attribute at the comma to create a new dataframe with one category and business\_id per row. Businesses with multiple categories appear in multiple rows.

#unnest the category tokens by dividing the list for each business at the comma  
full\_category\_df <- category\_df %>%   
 unnest\_tokens(category, categories, token = "regex",  
 pattern = ", ") %>%   
 ungroup()  
  
full\_category\_df

## # A tibble: 236,726 x 2  
## ID category   
## <chr> <chr>   
## 1 --1UhMGODdWsrMastO9DZw restaurants   
## 2 --1UhMGODdWsrMastO9DZw mexican   
## 3 --6MefnULPED\_I942VcFNA chinese   
## 4 --6MefnULPED\_I942VcFNA restaurants   
## 5 --9QQLMTbFzLJ\_oT-ON3Xw hair salons   
## 6 --9QQLMTbFzLJ\_oT-ON3Xw beauty & spas   
## 7 --cZ6Hhc9F7VkKXxHMVZSQ restaurants   
## 8 --cZ6Hhc9F7VkKXxHMVZSQ spanish   
## 9 --cZ6Hhc9F7VkKXxHMVZSQ peruvian   
## 10 --cZ6Hhc9F7VkKXxHMVZSQ latin american  
## # ... with 236,716 more rows

*How many different categories are there in the list of “good for kids” businesses?* Yelp organizes their categories as heirarchy with categories and subcategories.

#total number of categories and subcategories  
full\_category\_df %>%   
 summarise(n\_distinct(category))

## # A tibble: 1 x 1  
## `n\_distinct(category)`  
## <int>  
## 1 928

#count by category name  
full\_category\_df %>%   
 count(category, sort = T)

## # A tibble: 928 x 2  
## category n  
## <chr> <int>  
## 1 restaurants 43368  
## 2 food 11727  
## 3 fast food 6503  
## 4 sandwiches 6110  
## 5 active life 5580  
## 6 pizza 5284  
## 7 american (traditional) 5283  
## 8 breakfast & brunch 4790  
## 9 burgers 4788  
## 10 nightlife 4595  
## # ... with 918 more rows

The 928 values is the total number of both categories and subcategories. I renamed the category attribute as subcategory as it contains both category and subcategory values. And I created a new category attribute by converting each subcategory value to its parent category.

#import Yelp category list from:https://www.yelp.com/developers/documentation/v3/category\_list  
yelp\_cat <- read.csv("Data/Yelp\_Categories.csv", header = T)  
  
#normalize subcategories/categories to lowercase to match with full\_category\_df  
yelp\_cat$category <- tolower(yelp\_cat$category)  
yelp\_cat$subcategory <- tolower(yelp\_cat$subcategory)  
  
#combine tokenized category list (full\_category\_df) with the Yelp categories   
full\_category\_df <- full\_category\_df %>%   
 left\_join(yelp\_cat, by = c("category"="subcategory")) %>%   
 rename(yelp\_category=category.y) %>%   
 rename(yelp\_subcategory=category)  
  
#create a list of categories  
yelp\_cat\_list <- full\_category\_df %>%   
 distinct(yelp\_category) %>%   
 filter(!is.na(yelp\_category))  
  
yelp\_cat\_list <- as.character(yelp\_cat\_list$yelp\_category)  
  
#The total number of categories  
full\_category\_df %>%   
 summarise(n\_distinct(yelp\_category, na.rm = T))

## # A tibble: 1 x 1  
## `n\_distinct(yelp\_category, na.rm = T)`  
## <int>  
## 1 22

#The total number of subcategories  
full\_category\_df %>%   
 filter(!yelp\_subcategory %in% yelp\_cat\_list) %>%   
 summarise(n\_distinct(yelp\_subcategory, na.rm = T))

## # A tibble: 1 x 1  
## `n\_distinct(yelp\_subcategory, na.rm = T)`  
## <int>  
## 1 906

There are 22 categories and 906 subcategories in the whole dataset.

When I created the category attribute, some records got an NA value. This occurred for records with a subcategory value that are no longer listed on yelp’s business list. The records with NA values for category will be excluded from analysis involving the category attribute.

#what percent of records have NA value for yelp\_category?  
round(sum(is.na(full\_category\_df$yelp\_category))/nrow(full\_category\_df)\*100,1)

## [1] 4.4

#what percent of records have NA value for subcategory?  
round(sum(is.na(full\_category\_df$yelp\_subcategory))/nrow(full\_category\_df)\*100,1)

## [1] 0

*How many categories are listed for each business?*

#the number of distinct yelp (high level) categories per business  
full\_category\_df %>%   
 filter(!is.na(yelp\_category)) %>%  
 group\_by(ID) %>%   
 summarise(n=n\_distinct(yelp\_category)) %>%   
 summarytools::freq(n)

## Frequencies   
## full\_category\_df$n   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- ------- --------- -------------- --------- --------------  
## 1 35481 63.90 63.90 63.90 63.90  
## 2 15851 28.55 92.45 28.55 92.45  
## 3 3263 5.88 98.32 5.88 98.32  
## 4 701 1.26 99.58 1.26 99.58  
## 5 186 0.33 99.92 0.33 99.92  
## 6 31 0.06 99.97 0.06 99.97  
## 7 10 0.02 99.99 0.02 99.99  
## 8 3 0.01 100.00 0.01 100.00  
## 11 1 0.00 100.00 0.00 100.00  
## <NA> 0 0.00 100.00  
## Total 55527 100.00 100.00 100.00 100.00

Most businesses list between 1 and 3 categories and more than 98% of businesses listed between 1 and 3 categories.

*How many businesses per category?*

j <- full\_category\_df %>%   
 filter(!is.na(yelp\_category)) %>%  
 group\_by(yelp\_category) %>%   
 summarise(n=n\_distinct(ID)) %>%   
 summarise(sum(n))#total number of categories selected by businesses  
  
k <- full\_category\_df %>%   
 filter(!is.na(yelp\_category)) %>%  
 group\_by(yelp\_category) %>%   
 summarise(n=n\_distinct(ID)) %>%   
 arrange(desc(n))   
  
k #The number of businesses per category

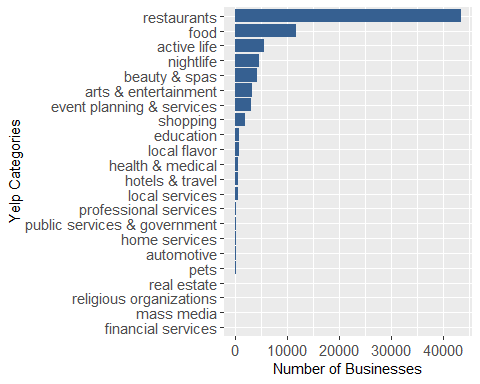
## # A tibble: 22 x 2  
## yelp\_category n  
## <chr> <int>  
## 1 restaurants 43368  
## 2 food 11727  
## 3 active life 5580  
## 4 nightlife 4595  
## 5 beauty & spas 4106  
## 6 arts & entertainment 3209  
## 7 event planning & services 2952  
## 8 shopping 1819  
## 9 education 736  
## 10 local flavor 720  
## # ... with 12 more rows

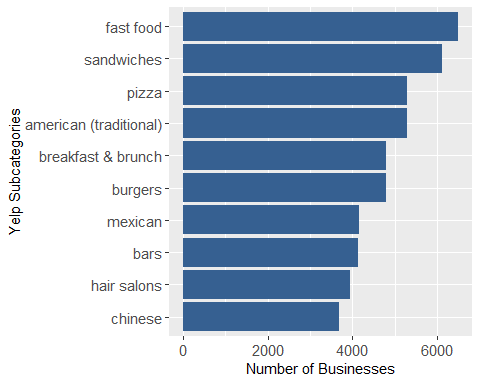
# The percent of category selections that were restaurants  
round((k$n[k$yelp\_category=="restaurants"]/j)\*100,1)

## sum(n)  
## 1 53.5

rm(j)

Note: there is double counting of businesses as they can select more than 1 category.



*What are the top 10 subcategories for good for kid businesses?* 

Note: there are 171 businesses that have only categories and no subcategories. That is why there are fewer total records in the subcategory plot compared to the category plot.

# Refine the question based on findings

The most common categories and subcategories selected in this dataset relate to restaurants. The remaining analsyis will focus on those restaurants that are family friendly (ff = food focused).

The revised guiding questions for the project and methods for answering them are: 1) What are the characteristics of family-friendly restaurants? –Explore subcategory frequencies and correlations –Analysis of word-pairs in the titles –Latent Dirichlet Allocation

1. Are there combinations of business features related to good reviews? –Dimentionality Reduction, including a comparison of feature selection techniques –Association rules Analysis

3)What matters most to customers that are satisfied or dissatisfied with their experience? –Sentiment Analysis –Classification model for stars rating using review sentiment and association rules analyses

– # Create list of family-friendly restaurants (ff) and their categories/subcategories

ff\_cat\_df <- full\_category\_df %>%   
 filter(yelp\_category == "restaurants")   
  
head(ff\_cat\_df)

## # A tibble: 6 x 3  
## ID yelp\_subcategory yelp\_category  
## <chr> <chr> <chr>   
## 1 --1UhMGODdWsrMastO9DZw restaurants restaurants   
## 2 --1UhMGODdWsrMastO9DZw mexican restaurants   
## 3 --6MefnULPED\_I942VcFNA chinese restaurants   
## 4 --6MefnULPED\_I942VcFNA restaurants restaurants   
## 5 --cZ6Hhc9F7VkKXxHMVZSQ restaurants restaurants   
## 6 --cZ6Hhc9F7VkKXxHMVZSQ spanish restaurants

ff\_cat\_df %>%   
 summarise(n=n\_distinct(ID)) #total number of food focused businesses

## # A tibble: 1 x 1  
## n  
## <int>  
## 1 43368

Create subset of restaurants including all the characteristics

ff\_restaurants\_df <- full\_category\_df %>%   
 filter(yelp\_category == "restaurants")   
  
ff\_restaurants\_df <- ff\_restaurants\_df %>%   
 select(-c(yelp\_subcategory,yelp\_category)) %>%   
 left\_join(business\_subset, by = c("ID"="business\_id")) %>%   
 distinct(ID, .keep\_all = T)  
  
ff\_restaurants\_df

## # A tibble: 43,368 x 58  
## ID name address city state postal\_code latitude longitude stars  
## <chr> <chr> <chr> <chr> <chr> <chr> <dbl> <dbl> <fct>  
## 1 --1U~ The ~ 821 4 ~ Calg~ AB T2P 0K5 51.0 -114. 4   
## 2 --6M~ John~ 328 Hi~ Rich~ ON L4B 3P7 43.8 -79.4 3   
## 3 --cZ~ Pio ~ 1408 E~ Char~ NC 28203 35.2 -80.8 4   
## 4 --Da~ Sunn~ 1218 S~ Toro~ ON M6E 43.7 -79.4 4   
## 5 --g-~ Kaba~ 3510 E~ Phoe~ AZ 85032 33.6 -112. 4.5   
## 6 --GM~ Mm M~ 407 S ~ Cano~ PA 15317 40.3 -80.2 4   
## 7 --KC~ Hung~ 6426 W~ Char~ NC 28269 35.3 -80.8 3   
## 8 --Ni~ KFC 1245 P~ Brun~ OH 44212 41.2 -81.8 1.5   
## 9 --q7~ Doub~ 9495 L~ Las ~ NV 89123 36.0 -115. 4   
## 10 --S6~ Denn~ 6207 W~ High~ OH 44143 41.5 -81.5 2   
## # ... with 43,358 more rows, and 49 more variables: review\_count <int>,  
## # is\_open <int>, categories <chr>,  
## # attributes.BusinessAcceptsCreditCards <chr>, attributes.BikeParking <chr>,  
## # attributes.GoodForKids <chr>, attributes.BusinessParking <chr>,  
## # attributes.ByAppointmentOnly <chr>,  
## # attributes.RestaurantsPriceRange2 <chr>, attributes.DogsAllowed <chr>,  
## # attributes.WiFi <chr>, attributes.RestaurantsAttire <chr>,  
## # attributes.RestaurantsTakeOut <chr>, attributes.NoiseLevel <chr>,  
## # attributes.RestaurantsReservations <chr>,  
## # attributes.RestaurantsGoodForGroups <chr>, attributes.HasTV <chr>,  
## # attributes.Alcohol <chr>, attributes.RestaurantsDelivery <chr>,  
## # attributes.OutdoorSeating <chr>, attributes.Caters <chr>,  
## # attributes.WheelchairAccessible <chr>, attributes.AcceptsInsurance <chr>,  
## # attributes.RestaurantsTableService <chr>, attributes.Ambience <chr>,  
## # attributes.GoodForMeal <chr>, attributes.HappyHour <chr>,  
## # attributes.BusinessAcceptsBitcoin <chr>, attributes.BYOB <chr>,  
## # attributes.Corkage <chr>, attributes.GoodForDancing <chr>,  
## # attributes.CoatCheck <chr>, attributes.BestNights <chr>,  
## # attributes.Music <chr>, attributes.Smoking <chr>,  
## # attributes.DietaryRestrictions <chr>, attributes.DriveThru <chr>,  
## # attributes.HairSpecializesIn <chr>, attributes.BYOBCorkage <chr>,  
## # attributes.AgesAllowed <chr>, attributes.RestaurantsCounterService <chr>,  
## # attributes.Open24Hours <chr>, hours.Monday <chr>, hours.Tuesday <chr>,  
## # hours.Wednesday <chr>, hours.Thursday <chr>, hours.Friday <chr>,  
## # hours.Saturday <chr>, hours.Sunday <chr>

Exploratory analysis of restaurant characteristics *What are the characteristics of the “good for kids” restaurants?* *Location:*

#Number of cities  
ff\_restaurants\_df %>%   
 distinct(city) %>%   
 count()

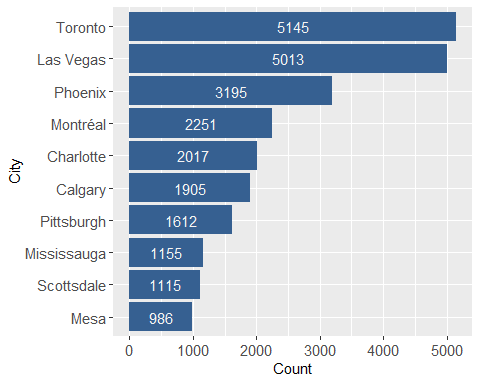
## # A tibble: 1 x 1  
## n  
## <int>  
## 1 684

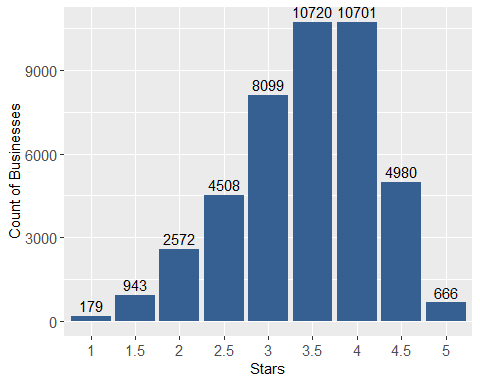
#Number of states or provinces  
ff\_restaurants\_df %>%   
 distinct(state) %>%   
 count()

## # A tibble: 1 x 1  
## n  
## <int>  
## 1 22

summarytools::freq(ff\_restaurants\_df$state, order = "freq")

## Frequencies   
## ff\_restaurants\_df$state   
## Type: Character   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- ------- --------- -------------- --------- --------------  
## ON 10506 24.23 24.23 24.23 24.23  
## AZ 9199 21.21 45.44 21.21 45.44  
## NV 6086 14.03 59.47 14.03 59.47  
## OH 3888 8.97 68.44 8.97 68.44  
## QC 3536 8.15 76.59 8.15 76.59  
## NC 3352 7.73 84.32 7.73 84.32  
## PA 2745 6.33 90.65 6.33 90.65  
## AB 1975 4.55 95.20 4.55 95.20  
## WI 1221 2.82 98.02 2.82 98.02  
## IL 499 1.15 99.17 1.15 99.17  
## SC 338 0.78 99.95 0.78 99.95  
## NY 9 0.02 99.97 0.02 99.97  
## TX 3 0.01 99.97 0.01 99.97  
## CA 2 0.00 99.98 0.00 99.98  
## CO 2 0.00 99.98 0.00 99.98  
## AR 1 0.00 99.99 0.00 99.99  
## BC 1 0.00 99.99 0.00 99.99  
## FL 1 0.00 99.99 0.00 99.99  
## HI 1 0.00 99.99 0.00 99.99  
## MB 1 0.00 100.00 0.00 100.00  
## OR 1 0.00 100.00 0.00 100.00  
## VA 1 0.00 100.00 0.00 100.00  
## <NA> 0 0.00 100.00  
## Total 43368 100.00 100.00 100.00 100.00



*Stars:* 

*Reviews:*

#Number of reviews  
ff\_restaurants\_df %>%   
 count(review\_count) %>%   
 summarise(sum(review\_count\*n))

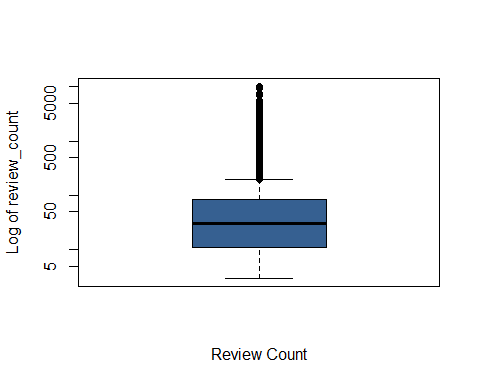
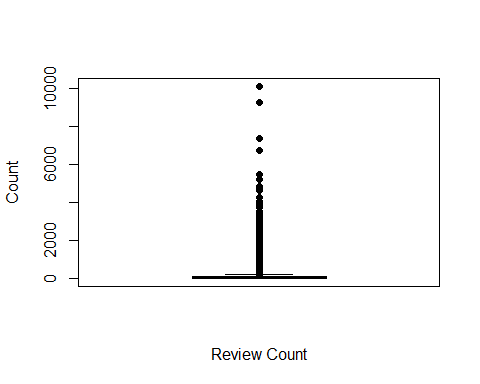
## # A tibble: 1 x 1  
## `sum(review\_count \* n)`  
## <int>  
## 1 3832034

#average number of reviews per business  
ff\_restaurants\_df %>%   
 summarise(round(mean(review\_count),2))

## # A tibble: 1 x 1  
## `round(mean(review\_count), 2)`  
## <dbl>  
## 1 88.4

#distribution of number of reviews  
summary(ff\_restaurants\_df$review\_count)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.00 11.00 31.00 88.36 84.00 10129.00



#number of outliers  
length(boxplot.stats(ff\_restaurants\_df$review\_count)$out)

## [1] 4637

The outliers will be addressed prior to the sentiment analysis

*Open:*

#Is open  
summarytools::freq(ff\_restaurants\_df$is\_open)

## Frequencies   
## ff\_restaurants\_df$is\_open   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- ------- --------- -------------- --------- --------------  
## 0 13390 30.88 30.88 30.88 30.88  
## 1 29978 69.12 100.00 69.12 100.00  
## <NA> 0 0.00 100.00  
## Total 43368 100.00 100.00 100.00 100.00

To understand restaurant types that are family friendly, I looked first at the subcategories which provide details about type of food (e.g., pizza, tacos) and cuisine (e.g., mexican, japanese). I started by exploring *what are the most common subcategories combinations for restaurants?*

ff\_cat\_df %>%   
 filter(!yelp\_subcategory %in% yelp\_cat\_list) %>% #excluding the category values from the subcategory attribute as they are combined  
 group\_by(yelp\_subcategory) %>%   
 summarise(n=n\_distinct(ID)) %>% arrange(desc(n)) %>% top\_n(10,n)

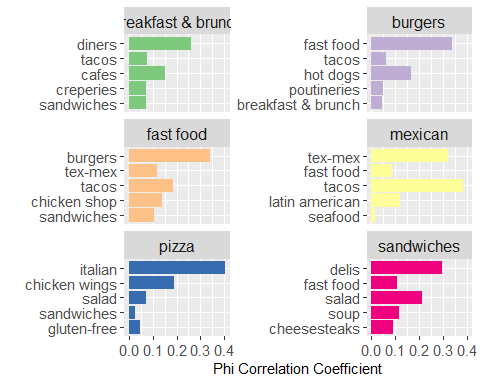
## # A tibble: 10 x 2  
## yelp\_subcategory n  
## <chr> <int>  
## 1 fast food 6503  
## 2 sandwiches 6110  
## 3 pizza 5284  
## 4 breakfast & brunch 4790  
## 5 burgers 4788  
## 6 mexican 4151  
## 7 chinese 3695  
## 8 italian 3470  
## 9 salad 2346  
## 10 cafes 2326

#count the number of times each pair of items appear together within a business.  
ff\_cat\_df %>%   
 filter(!yelp\_subcategory %in% yelp\_cat\_list) %>%   
 pairwise\_count(yelp\_subcategory, ID, sort=T)

## # A tibble: 5,860 x 3  
## item1 item2 n  
## <chr> <chr> <dbl>  
## 1 burgers fast food 2392  
## 2 fast food burgers 2392  
## 3 italian pizza 1986  
## 4 pizza italian 1986  
## 5 fast food sandwiches 1539  
## 6 sandwiches fast food 1539  
## 7 japanese sushi bars 1224  
## 8 sushi bars japanese 1224  
## 9 delis sandwiches 1076  
## 10 sandwiches delis 1076  
## # ... with 5,850 more rows

#phi coefficient showing how often they appear in the same business  
category\_correlations <- ff\_cat\_df %>%   
 filter(!yelp\_subcategory %in% yelp\_cat\_list) %>%   
 group\_by(yelp\_subcategory) %>%  
 filter(n()>=100) %>%   
 pairwise\_cor(yelp\_subcategory, ID, sort = T)  
  
category\_correlations

## # A tibble: 5,112 x 3  
## item1 item2 correlation  
## <chr> <chr> <dbl>  
## 1 japanese sushi bars 0.588  
## 2 sushi bars japanese 0.588  
## 3 pakistani indian 0.506  
## 4 indian pakistani 0.506  
## 5 middle eastern lebanese 0.428  
## 6 lebanese middle eastern 0.428  
## 7 vegetarian vegan 0.414  
## 8 vegan vegetarian 0.414  
## 9 italian pizza 0.402  
## 10 pizza italian 0.402  
## # ... with 5,102 more rows



*Can restaurant names identify popular types of restaurants?* Unlike the categories, restuarant titles are free text fields. The restaurant titles need to be cleaned and normalized by, for example, removing punctuation, stopwords, infrequent- and frequent-words and lemmatizing words.

#Create a dataframe for analysing the title  
business\_title <- left\_join(ff\_cat\_df, name\_df, by = c("ID"="ID"))  
business\_title <- tibble(ID = business\_title$ID, title = business\_title$name)  
  
#Create a dataframe of stop words from english, french and spanish using the snowball lexicon from tidytext (Which is available in multiple langauges)  
custom\_stop\_words <- get\_stopwords(language = "en", source = "snowball")  
custom\_stop\_words <- rbind(custom\_stop\_words, get\_stopwords(language = "fr", source = "snowball"))  
custom\_stop\_words <- rbind(custom\_stop\_words, get\_stopwords(language = "es", source = "snowball"))  
  
#Remove stop words from the business titles  
business\_title <- business\_title %>%   
 unnest\_tokens(word, title) %>% #strips punctuation  
 anti\_join(custom\_stop\_words) %>% #removes stopwords  
 mutate(word = lemmatize\_words(word)) #convert text using lemmatize  
  
#Remove words that are digits  
k <- which(str\_detect(business\_title$word, "\\d+"))  
business\_title <- business\_title[-k,]  
  
#Remove words that are 2 characters or shorter  
k <- which(nchar(business\_title$word)<=2)  
business\_title <- business\_title[-k,]  
  
#What are the 10 most common words in the title   
business\_title %>%   
 count(word, sort = T) %>%   
 top\_n(10)

## # A tibble: 10 x 2  
## word n  
## <chr> <int>  
## 1 restaurant 11052  
## 2 pizza 9063  
## 3 grill 7260  
## 4 cafe 5433  
## 5 bar 4071  
## 6 sushi 3772  
## 7 kitchen 3030  
## 8 house 2793  
## 9 mexican 2678  
## 10 taco 2428

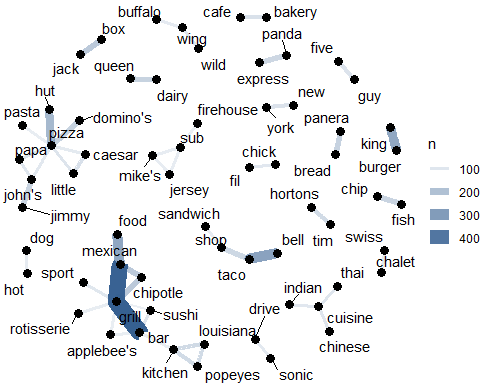
pop\_unpopular <- rbind(business\_title %>% count(word, sort = T) %>% top\_n(1) %>% select(word),   
 business\_title %>% count(word, sort = T) %>% top\_n(-10) %>% select(word)) #least popular words in the title  
  
#remove the word restaurant which is the most popular word and is the category for all records as well as the 630 words that only appear once in the dataset  
business\_title <- business\_title %>%   
 anti\_join(pop\_unpopular)

The most common words in the restaurant titles are similar to the most common YELP subcategories.

*How many times does each pair of words occur in a title*

title\_word\_pairs <- business\_title %>%   
 pairwise\_count(word, ID, sort=T, upper = F) #count the frequency of words pairs  
  
title\_word\_pairs

## # A tibble: 43,880 x 3  
## item1 item2 n  
## <chr> <chr> <dbl>  
## 1 grill bar 460  
## 2 grill mexican 453  
## 3 taco bell 282  
## 4 mexican food 270  
## 5 king burger 267  
## 6 pizza hut 223  
## 7 jack box 178  
## 8 grill chipotle 177  
## 9 mexican chipotle 177  
## 10 panera bread 153  
## # ... with 43,870 more rows

It’s difficult to see patterns in the word pairs from counts alone. So I created a network diagram to show the most common word pairs. Note: pairs must appear at least 100 times in the dataset to be included in the diagram. 

The diagram highlights common words as nodes (e.g., grill, pizza, cuisine).

The diagram also highlights that the most common word pairs are names of fastfood chains (e.g., dairy queen, pizza hut, burger king, panera bread)) or common food pairings (e.g., fish and chip, bakery and cafe).

The terms “grill” and “bar” seems to be popular for may type of food (e.g., rotisserie, sushi, mexican) or ambiance (e.g., sport).  
# Latent Dirichlet Allocation

The next step is to explore the restaurant titles to try and find common topics using Latent Dirichlet Allocation (LDA). LDA is a probabilistic modelling approach, to discover themes in the business names.

Start by setting up the Gibbs Sampling

#set parameters for Gibbs sampling  
burnin <-4000  
iter <- 2000  
thin <-500  
seed <- list(6, 8, 110, 3800, 13000)  
nstart <-5  
best <- TRUE

Set the number of topics (k)

#Set the number of topics  
k <- 5

businesstitle\_counts <- business\_title%>%   
 count(ID, word, sort = T) %>%   
 ungroup()  
  
businesstitle\_dtm <- businesstitle\_counts %>%   
 cast\_dtm(ID, word, n) #cast to a document term matrix

#LDA analysis using Gibbs sampling  
businesstitle\_lda <- LDA(businesstitle\_dtm, k = k, method="Gibbs", control = list(nstart=nstart, seed=seed, best=best, burnin = burnin, iter = iter, thin=thin))

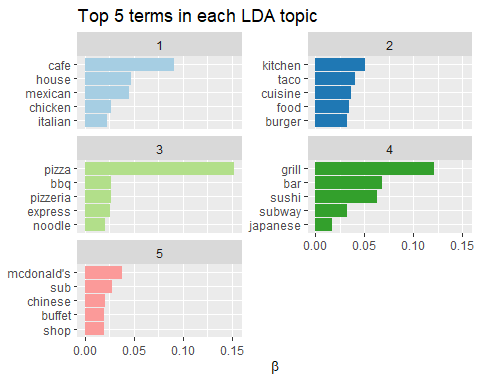
Beta is a metric output from the LDA. It is the probability of each term being generated from each topic

tidy\_beta <-tidy(businesstitle\_lda, matrix = "beta")  
tidy\_beta

## # A tibble: 72,985 x 3  
## topic term beta  
## <int> <chr> <dbl>  
## 1 1 cous 0.000270   
## 2 2 cous 0.00000168  
## 3 3 cous 0.00000168  
## 4 4 cous 0.00000167  
## 5 5 cous 0.00000174  
## 6 1 tin 0.000823   
## 7 2 tin 0.00000168  
## 8 3 tin 0.00000168  
## 9 4 tin 0.00000167  
## 10 5 tin 0.00000174  
## # ... with 72,975 more rows

The visualition below shows the most common words per topic.

top\_terms <- tidy\_beta %>%   
 group\_by(topic) %>%   
 top\_n(5, beta) %>%   
 ungroup() %>%   
 arrange(topic, -beta)  
  
#visualize the terms  
top\_terms %>%  
 mutate(term = reorder\_within(term, beta, topic)) %>%  
 group\_by(topic, term) %>%   
 arrange(desc(beta)) %>%   
 ungroup() %>%  
 ggplot(aes(term, beta, fill = as.factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 scale\_fill\_brewer(type = "qual", palette = 3)+  
 coord\_flip() +  
 scale\_x\_reordered() +  
 labs(title = "Top 5 terms in each LDA topic",  
 x = NULL, y = expression(beta)) +  
 facet\_wrap(~ topic, ncol = 2, scales = "free\_y")



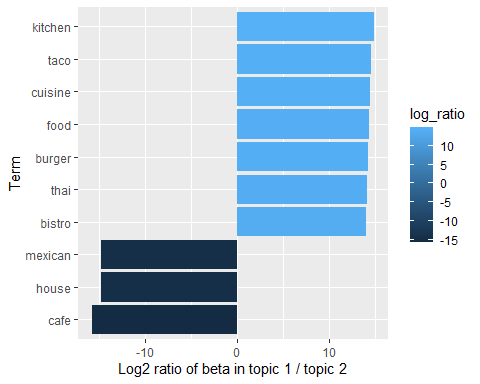
Overall the probabilities of each term per topic are quite low, less than 15%. The LDA reveals some topics that are similar to those identified by the categories. Topic 3 seems to relate to fast food pizza and topic 4 seems to relate to japanese restaurants. But both topics also include words that seem unrelated to the topic (e.g., noodle and subway, respectively). For the other 3 topics, the beta values are all less than 10% and the terms terms don’t seem to hang together well.

An alternative analysis for beta is to look at the spread of terms. That is to see the terms that have the greatest difference between two topics. In the example below, we explore the difference between topic 1 and 2.

beta\_spread <- tidy\_beta %>%   
 mutate(topic = paste0("topic", topic)) %>%   
 spread(topic, beta) %>%   
 filter(topic1 > .001| topic2 > .001) %>%   
 mutate(log\_ratio = log2(topic2 / topic1))  
  
beta\_spread

## # A tibble: 221 x 7  
## term topic1 topic2 topic3 topic4 topic5 log\_ratio  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 al's 0.00118 0.0000856 0.00000168 0.00000167 0.00000174 -3.78  
## 2 aladdin's 0.00000168 0.00126 0.00000168 0.0000184 0.0000191 9.55  
## 3 american 0.00000168 0.00395 0.00000168 0.00000167 0.00000174 11.2   
## 4 arby's 0.00681 0.0000185 0.000204 0.00000167 0.00000174 -8.53  
## 5 asian 0.00000168 0.0161 0.00000168 0.00000167 0.00000174 13.2   
## 6 authentic 0.00000168 0.00180 0.0000354 0.00000167 0.00000174 10.1   
## 7 avenue 0.00000168 0.00168 0.00000168 0.00000167 0.00000174 9.97  
## 8 baja 0.00133 0.0000185 0.00000168 0.00000167 0.00000174 -6.17  
## 9 bakery 0.00000168 0.0163 0.00000168 0.00000167 0.00000174 13.2   
## 10 bamboo 0.00148 0.00000168 0.00000168 0.00000167 0.0000191 -9.78  
## # ... with 211 more rows

#visualise the terms with the biggest spread  
biggest\_spread <- beta\_spread %>%   
 select(term, log\_ratio) %>%   
 top\_n(10,abs(log\_ratio)) %>%  
 arrange(-log\_ratio)  
   
biggest\_spread %>%   
 mutate(term = reorder(term, log\_ratio)) %>%   
 ggplot(aes(term, log\_ratio, fill=log\_ratio))+  
 geom\_bar(stat="identity")+  
 xlab("Term")+  
 ylab("Log2 ratio of beta in topic 1 / topic 2")+  
 coord\_flip()



This analysis doesn’t provide much additional information. This could be because the topics themselves aren’t well defined or it could be because there are few words per restaurant name.

Gamma is another metric output by LDA. It is the per-restaurant-per-topic probabilities. Gamma is useful for seeing how well the LDA identifies a topic for each restaurant.

#create a gamma tibble  
tidy\_gamma <-tidy(businesstitle\_lda, matrix = "gamma")  
tidy\_gamma

## # A tibble: 215,395 x 3  
## document topic gamma  
## <chr> <int> <dbl>  
## 1 kZspuWnM0Y-Losvk2Rl0lA 1 0.415  
## 2 sYXEz38MVmg\_vxe7hYTR6Q 1 0.459  
## 3 4u2ls0mGJNhfyzrqAMXtZg 1 0.161  
## 4 eMz8i2LpafE6D-ZhzDtglw 1 0.382  
## 5 hQ7NikMijOLR2jXpMCTUHg 1 0.235  
## 6 IjsLANGkmAqCsF6-zgIA8w 1 0.147  
## 7 MUGjHMfOhkQ21kVVCpQClQ 1 0.161  
## 8 Oy2WnPyiOlPFvPKMIuOC8w 1 0.355  
## 9 PlYrlj86zAYljZTRCBSZ3A 1 0.152  
## 10 vgc-XO64Uw7otc3sxmns1A 1 0.239  
## # ... with 215,385 more rows

We can explore the relative importance of the assigned topic (e.g., topic with the maximum gamma probability) and the next highest topic probability for each restaurant. If the relative importance of the assigned topic is high, that means that the topics effectively differentiate the restaurants. If the relative importance is close to 1, that means that the probability of a restaurant being assigned to the two topics is similar.

gamma\_spread <- tidy\_gamma %>%   
 group\_by(document) %>%   
 arrange(document, -gamma) %>%   
 mutate(gamma\_rank = rank(-gamma, ties.method = "random")) %>%   
 filter(gamma\_rank == 1|gamma\_rank == 2) %>%   
 arrange(document, gamma\_rank) %>%  
 mutate(topic = rep(c("TopTopic", "NextTopic"))) %>%   
 select(-gamma\_rank) %>%   
 spread(topic, gamma) %>%   
 mutate(relative\_importance = TopTopic/NextTopic)  
  
summarytools::freq(round(gamma\_spread$relative\_importance,1))

## Frequencies   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ----------- ------- --------- -------------- --------- --------------  
## 1 10550 24.49 24.49 24.49 24.49  
## 1.1 3284 7.62 32.11 7.62 32.11  
## 1.2 8224 19.09 51.20 19.09 51.20  
## 1.3 5192 12.05 63.26 12.05 63.26  
## 1.4 4701 10.91 74.17 10.91 74.17  
## 1.5 1061 2.46 76.63 2.46 76.63  
## 1.6 3633 8.43 85.06 8.43 85.06  
## 1.7 285 0.66 85.73 0.66 85.73  
## 1.8 2215 5.14 90.87 5.14 90.87  
## 1.9 945 2.19 93.06 2.19 93.06  
## 2 996 2.31 95.37 2.31 95.37  
## 2.1 23 0.05 95.43 0.05 95.43  
## 2.2 1141 2.65 98.08 2.65 98.08  
## 2.3 30 0.07 98.15 0.07 98.15  
## 2.4 73 0.17 98.31 0.17 98.31  
## 2.5 244 0.57 98.88 0.57 98.88  
## 2.6 172 0.40 99.28 0.40 99.28  
## 2.7 6 0.01 99.29 0.01 99.29  
## 2.8 88 0.20 99.50 0.20 99.50  
## 2.9 1 0.00 99.50 0.00 99.50  
## 3 110 0.26 99.76 0.26 99.76  
## 3.1 33 0.08 99.83 0.08 99.83  
## 3.2 3 0.01 99.84 0.01 99.84  
## 3.4 31 0.07 99.91 0.07 99.91  
## 3.5 18 0.04 99.95 0.04 99.95  
## 3.7 2 0.00 99.96 0.00 99.96  
## 3.8 5 0.01 99.97 0.01 99.97  
## 4 5 0.01 99.98 0.01 99.98  
## 4.2 2 0.00 99.99 0.00 99.99  
## 4.3 1 0.00 99.99 0.00 99.99  
## 4.5 3 0.01 100.00 0.01 100.00  
## 4.6 1 0.00 100.00 0.00 100.00  
## 5.5 1 0.00 100.00 0.00 100.00  
## <NA> 0 0.00 100.00  
## Total 43079 100.00 100.00 100.00 100.00

*Summary of LDA Analysis*

I tested k values between 5 and 20. In all cases, the LDA topics developed from the business names were not clear or useful. The term per topic probabilities (beta) did not create meaningful topics based on the terms. The restaurant per topic probabilities (gamma) also did not show distinct, meaningful topics. Moreover, for all values of k nearly 25% of restaurants were equally likely to be assigned to more than 1 topic.

Topic modelling using LDA does not result in additional information for the exploratory analysis. No clear topics can be identified by business name. The self-identified categories are a more useful way of grouping the than the topics created through LDA.

# Feature selection

The next step in the analysis is to explore common business features/attributes among family-friendly restaurants with high star ratings. Using association analysis, I looked for sets of items that often occur together in highly rated restaurants. In the YELP dataset the business features/attributes are a series of binary attributes where true means present, false means absent and NA indicates the information was not provided. There are 54 total attributes relevant to family-friendly restaurants. Many attributes have high numbers of NA values. I used dimensionality reduction to find the best features to include in the association analysis

*Data Cleaning*

Start by creating a dataframe for the attributes and star ratings for family-friendly restaurants.

ff\_subset <- ff\_cat\_df %>%   
 select(ID) %>%   
 distinct(ID) %>%   
 left\_join(business\_subset, by = c("ID"="business\_id"))  
  
ff\_assoc <- ff\_subset %>%   
 select("ID", "stars", starts\_with("attributes."))

Next, I cleaned the attribute data. For example, BusinessParking, Ambiance and GoodForMeal attributes are arrays. They need to be unnested.

#Create a dataframe for the column that needs to be split + ID (attributes.Ambience)  
k<- ff\_assoc %>%   
 select(ID, attributes.Ambience)   
  
#split the string at the column and seperate each key value pair onto a seperate row with the appropriate business ID  
k<- k %>%   
 mutate(attributes.Ambience = strsplit(attributes.Ambience, ",")) %>%   
 unnest(attributes.Ambience)  
  
#clean up the text by removing extra characters and whitespace  
k$attributes.Ambience<- str\_replace\_all(k$attributes.Ambience,"\\{","")  
k$attributes.Ambience<- str\_replace\_all(k$attributes.Ambience,"\\'","")  
k$attributes.Ambience<- str\_replace\_all(k$attributes.Ambience,"\\}","")  
k$attributes.Ambience<- trimws(k$attributes.Ambience)  
  
# shift the keys to the columns and the values to the cells and rename the attributes  
k <- k %>% separate(attributes.Ambience, c("key1","value"), sep = ": ", extra = "warn", fill = "warn") %>%   
 spread(key1,value) %>%   
 select(-c(V1, `<NA>`, None)) %>%   
 rename\_at(vars(-ID), ~ paste0('Ambience.',.))  
  
#reconnect the dataframe to the ff\_association rules  
ff\_assoc <- left\_join(ff\_assoc, k) %>%   
 select(-c(attributes.Ambience))

Repeat the attribute splitting, cleaning and reconnecting for Business Parking, GoodForMeal, BestNights, Music

##2) Create a dataframe for just the column that needs to be split + ID (attributes.BusinessParking)  
k<- ff\_assoc %>%   
 select(ID, attributes.BusinessParking)   
  
#split the string at the column and seperate each key value pair onto a seperate row with the appropriate business ID  
k<- k %>%   
 mutate(attributes.BusinessParking = strsplit(attributes.BusinessParking, ",")) %>%   
 unnest(attributes.BusinessParking)  
  
#clean up the text by removing extra characters and whitespace  
k$attributes.BusinessParking<- str\_replace\_all(k$attributes.BusinessParking,"\\{","")  
k$attributes.BusinessParking<- str\_replace\_all(k$attributes.BusinessParking,"\\'","")  
k$attributes.BusinessParking<- str\_replace\_all(k$attributes.BusinessParking,"\\}","")  
k$attributes.BusinessParking<- trimws(k$attributes.BusinessParking)  
  
# shift the keys to the columns and the values to the cells and rename the attributes  
k <- k %>% separate(attributes.BusinessParking, c("key1","value"), sep = ": ", extra = "warn", fill = "warn") %>%   
 spread(key1,value) %>%   
 select(-c(V1, `<NA>`, None))%>%  
 rename\_at(vars(-ID), ~ paste0('BusinessParking.',.))  
  
#reconnect the dataframe to the ff\_association rules  
ff\_assoc <- left\_join(ff\_assoc, k) %>%   
 select(-c(attributes.BusinessParking))  
  
##3) Create a dataframe for just the column that needs to be split + ID (attributes.GoodForMeal)  
k<- ff\_assoc %>%   
 select(ID, attributes.GoodForMeal)   
  
#split the string at the column and seperate each key value pair onto a seperate row with the appropriate business ID  
k<- k %>%   
 mutate(attributes.GoodForMeal = strsplit(attributes.GoodForMeal, ",")) %>%   
 unnest(attributes.GoodForMeal)  
  
#clean up the text by removing extra characters and whitespace  
k$attributes.GoodForMeal<- str\_replace\_all(k$attributes.GoodForMeal,"\\{","")  
k$attributes.GoodForMeal<- str\_replace\_all(k$attributes.GoodForMeal,"\\'","")  
k$attributes.GoodForMeal<- str\_replace\_all(k$attributes.GoodForMeal,"\\}","")  
k$attributes.GoodForMeal<- trimws(k$attributes.GoodForMeal)  
  
# shift the keys to the columns and the values to the cells and rename the attributes  
k <- k %>% separate(attributes.GoodForMeal, c("key1","value"), sep = ": ", extra = "warn", fill = "warn") %>%   
 spread(key1,value) %>%   
 select(-c(V1, `<NA>`, None)) %>%   
 rename\_at(vars(-ID), ~ paste0('GoodForMeal.',.))  
  
#reconnect the dataframe to the ff\_association rules  
ff\_assoc <- left\_join(ff\_assoc, k) %>%   
 select(-c(attributes.GoodForMeal))  
  
##4)Create a dataframe for just the column that needs to be split + ID (attributes.BestNights)  
k<- ff\_assoc %>%   
 select(ID, attributes.BestNights)   
  
#split the string at the column and seperate each key value pair onto a seperate row with the appropriate business ID  
k<- k %>%   
 mutate(attributes.BestNights = strsplit(attributes.BestNights, ",")) %>%   
 unnest(attributes.BestNights)  
  
#clean up the text by removing extra characters and whitespace  
k$attributes.BestNights<- str\_replace\_all(k$attributes.BestNights,"\\{","")  
k$attributes.BestNights<- str\_replace\_all(k$attributes.BestNights,"\\'","")  
k$attributes.BestNights<- str\_replace\_all(k$attributes.BestNights,"\\}","")  
k$attributes.BestNights<- trimws(k$attributes.BestNights)  
  
# shift the keys to the columns and the values to the cells and rename the attributes  
k <- k %>% separate(attributes.BestNights, c("key1","value"), sep = ": ", extra = "warn", fill = "warn") %>%   
 spread(key1,value) %>%   
 select(-c(`<NA>`))%>%  
 rename\_at(vars(-ID), ~ paste0('BestNights.',.))  
  
#reconnect the dataframe to the ff\_association rules  
ff\_assoc <- left\_join(ff\_assoc, k) %>%   
 select(-c(attributes.BestNights))  
  
##5) Create a dataframe for just the column that needs to be split + ID (attributes.Music)  
k<- ff\_assoc %>%   
 select(ID, attributes.Music)   
  
#split the string at the column and seperate each key value pair onto a seperate row with the appropriate business ID  
k<- k %>%   
 mutate(attributes.Music = strsplit(attributes.Music, ",")) %>%   
 unnest(attributes.Music)  
  
#clean up the text by removing extra characters and whitespace  
k$attributes.Music<- str\_replace\_all(k$attributes.Music,"\\{","")  
k$attributes.Music<- str\_replace\_all(k$attributes.Music,"\\'","")  
k$attributes.Music<- str\_replace\_all(k$attributes.Music,"\\}","")  
k$attributes.Music<- trimws(k$attributes.Music)  
  
# shift the keys to the columns and the values to the cells and rename the attributes  
k <- k %>% separate(attributes.Music, c("key1","value"), sep = ": ", extra = "warn", fill = "warn") %>%   
 spread(key1,value) %>%   
 select(-c(V1, `<NA>`, None))%>%  
 rename\_at(vars(-ID), ~ paste0('Music.',.))  
  
#reconnect the dataframe to the ff\_association rules  
ff\_assoc <- left\_join(ff\_assoc, k) %>%   
 select(-c(attributes.Music))  
  
rm(k)

Remove “attributes.” from the column names.

ff\_assoc <- ff\_assoc %>%   
 rename\_at(vars(starts\_with("attributes.")), list(~str\_replace(., "attributes.","")))

*Dimensionality Reduction*

Remove business features with more than 30% of the values missing.

#create a function to calculate the % of NA per column  
percent\_na <- function(x){  
 round(sum(is.na(ff\_assoc[,x]))/nrow(ff\_assoc)\*100,1)  
}   
  
col\_num <- c(1:ncol(ff\_assoc))#vector of column numbers  
sum(lapply(col\_num, percent\_na)>30.0) #total number of columns above the threshold of 30% NA

## [1] 42

ff\_assoc\_trim <- ff\_assoc %>% select(-c(which(lapply(col\_num, percent\_na)>30.0)))#create a new df trimmed of attributes with high NA values

The attribute that were removed due to high NA values where:

k<-which((colnames(ff\_assoc) %in% colnames(ff\_assoc\_trim))==FALSE)  
colnames(ff\_assoc)[k]

## [1] "BusinessAcceptsCreditCards" "ByAppointmentOnly"   
## [3] "DogsAllowed" "Caters"   
## [5] "WheelchairAccessible" "AcceptsInsurance"   
## [7] "RestaurantsTableService" "HappyHour"   
## [9] "BusinessAcceptsBitcoin" "BYOB"   
## [11] "Corkage" "GoodForDancing"   
## [13] "CoatCheck" "Smoking"   
## [15] "DietaryRestrictions" "DriveThru"   
## [17] "HairSpecializesIn" "BYOBCorkage"   
## [19] "AgesAllowed" "RestaurantsCounterService"   
## [21] "Open24Hours" "GoodForMeal.breakfast"   
## [23] "GoodForMeal.brunch" "GoodForMeal.dessert"   
## [25] "GoodForMeal.dinner" "GoodForMeal.latenight"   
## [27] "GoodForMeal.lunch" "BestNights.friday"   
## [29] "BestNights.monday" "BestNights.None"   
## [31] "BestNights.saturday" "BestNights.sunday"   
## [33] "BestNights.thursday" "BestNights.tuesday"   
## [35] "BestNights.wednesday" "Music.background\_music"   
## [37] "Music.dj" "Music.jukebox"   
## [39] "Music.karaoke" "Music.live"   
## [41] "Music.no\_music" "Music.video"

Remove “Good for Kids” variables which has low variance (all values are 1).

ff\_assoc\_trim <- ff\_assoc\_trim %>% select(-c("GoodForKids"))

*Additional Data Cleaning and Dummy Variable Preparation*

Binary variables

#replace the values "True" with "1" & "False" and "None" with "0"  
ff\_assoc\_trim <- data.frame(lapply(ff\_assoc\_trim, function(x){  
 gsub("True","1",x)  
}))  
  
ff\_assoc\_trim <- data.frame(lapply(ff\_assoc\_trim, function(x){  
 gsub("False","0",x)  
}))  
  
ff\_assoc\_trim <- data.frame(lapply(ff\_assoc\_trim, function(x){  
 gsub("None","0",x)  
}))

Categorical variables

#Clean the text values as several have extra characters.  
#WiFi  
ff\_assoc\_trim$WiFi <- str\_replace\_all(ff\_assoc\_trim$WiFi,"u\\'","")  
ff\_assoc\_trim$WiFi <- str\_replace\_all(ff\_assoc\_trim$WiFi,"\\'","")  
ff\_assoc\_trim$WiFi <- gsub("0",NA,ff\_assoc\_trim$WiFi)  
  
#RestaurantsAttire  
ff\_assoc\_trim$RestaurantsAttire <- str\_replace\_all(ff\_assoc\_trim$RestaurantsAttire,"u\\'","")  
ff\_assoc\_trim$RestaurantsAttire <- str\_replace\_all(ff\_assoc\_trim$RestaurantsAttire,"\\'","")  
ff\_assoc\_trim$RestaurantsAttire <- gsub("0",NA,ff\_assoc\_trim$RestaurantsAttire)  
  
#NoiseLevel (quiet = 1, average = 2, loud = 3, very loud = 4)  
ff\_assoc\_trim$NoiseLevel <- str\_replace\_all(ff\_assoc\_trim$NoiseLevel,"^u","")  
ff\_assoc\_trim$NoiseLevel <- str\_replace\_all(ff\_assoc\_trim$NoiseLevel,"quiet","1")  
ff\_assoc\_trim$NoiseLevel <- str\_replace\_all(ff\_assoc\_trim$NoiseLevel,"average","2")  
ff\_assoc\_trim$NoiseLevel <- str\_replace\_all(ff\_assoc\_trim$NoiseLevel,"very\_loud","4")  
ff\_assoc\_trim$NoiseLevel <- str\_replace\_all(ff\_assoc\_trim$NoiseLevel,"loud","3")  
ff\_assoc\_trim$NoiseLevel <- gsub("0",NA,ff\_assoc\_trim$NoiseLevel)  
  
#Alcohol  
ff\_assoc\_trim$Alcohol <- str\_replace\_all(ff\_assoc\_trim$Alcohol,"u\\'","")  
ff\_assoc\_trim$Alcohol <- str\_replace\_all(ff\_assoc\_trim$Alcohol,"\\'","")  
ff\_assoc\_trim$Alcohol <- gsub("0",NA,ff\_assoc\_trim$Alcohol)  
  
#RestaurantsPriceRange2  
ff\_assoc\_trim$RestaurantsPriceRange2 <- gsub("0",NA,ff\_assoc\_trim$RestaurantsPriceRange2)

Convert the ordinal and nominal variables variables to rank and dummy variables, respectively.

#WiFi  
dmy <- dummyVars("~ WiFi", data = ff\_assoc\_trim)  
trsf <- data.frame(predict(dmy, ff\_assoc\_trim))  
ff\_assoc\_trim <- cbind(ff\_assoc\_trim, trsf)  
  
#RestaurantsAttire  
dmy <- dummyVars("~ RestaurantsAttire", data = ff\_assoc\_trim)  
trsf <- data.frame(predict(dmy, ff\_assoc\_trim))  
ff\_assoc\_trim <- cbind(ff\_assoc\_trim, trsf)  
  
#Alcohol  
dmy <- dummyVars("~ Alcohol", data = ff\_assoc\_trim)  
trsf <- data.frame(predict(dmy, ff\_assoc\_trim))  
ff\_assoc\_trim <- cbind(ff\_assoc\_trim, trsf)  
  
ff\_assoc\_trim <- ff\_assoc\_trim %>%   
 select(-c("WiFi", "RestaurantsAttire","Alcohol"))  
  
#convert remaining variables to factor  
ff\_assoc\_trim$RestaurantsPriceRange2 <- as.factor(ff\_assoc\_trim$RestaurantsPriceRange2)  
  
ff\_assoc\_trim$NoiseLevel <- as.factor(ff\_assoc\_trim$NoiseLevel)

The star attribute is separated into 9 values (from 1 to 5). For the association rules analysis, I summarized the star attribute into 2 groups “3.5 or less” and “4 or more”. I’m interested in identifying groups of features that are common to restaurants with 4 or more stars.

levels(ff\_assoc\_trim$stars)

## [1] "1" "1.5" "2" "2.5" "3" "3.5" "4" "4.5" "5"

ff\_assoc\_trim$stars <- revalue(ff\_assoc\_trim$stars, c("1"="3.5 orless","1.5"="3.5 orless", "2"="3.5 orless", "2.5"="3.5 orless","3"="3.5 orless", "3.5"="3.5 orless", "4"="4 or more","4.5"= "4 or more", "5"="4 or more"))  
levels(ff\_assoc\_trim$stars)

## [1] "3.5 orless" "4 or more"

table(ff\_assoc\_trim$stars)

##   
## 3.5 orless 4 or more   
## 27021 16347

*Feature selection*

Three different feature selection techniques were used to identify the attributes that are most relevant to my association rules analysis. Again, because I’m looking at features common to restaurants with 4 or more stars, stars is the dependent variable in each of the feature selection analyses.

Information Gain

ff\_assoc\_ig <- information.gain(stars~., data = ff\_assoc\_trim[,-1])  
  
ff\_assoc\_ig %>%   
 arrange(-attr\_importance)

## attr\_importance  
## BusinessParking.street 9.210567e-03  
## NoiseLevel 7.853407e-03  
## Ambience.trendy 6.932003e-03  
## Ambience.classy 6.134387e-03  
## Ambience.hipster 6.128596e-03  
## Ambience.casual 5.845608e-03  
## HasTV 5.217149e-03  
## BikeParking 3.690259e-03  
## WiFifree 3.679061e-03  
## WiFino 3.617589e-03  
## Ambience.intimate 2.921058e-03  
## WiFipaid 2.889986e-03  
## Alcoholfull\_bar 2.334002e-03  
## Ambience.romantic 2.086091e-03  
## Ambience.upscale 1.879282e-03  
## Ambience.touristy 1.783284e-03  
## BusinessParking.lot 1.667545e-03  
## Alcoholbeer\_and\_wine 1.606476e-03  
## OutdoorSeating 1.575104e-03  
## BusinessParking.validated 1.537946e-03  
## BusinessParking.valet 1.412458e-03  
## BusinessParking.garage 1.390468e-03  
## Alcoholnone 7.843223e-04  
## RestaurantsGoodForGroups 6.939159e-04  
## RestaurantsAttirecasual 6.800985e-04  
## RestaurantsAttiredressy 6.800985e-04  
## RestaurantsAttireformal 6.800985e-04  
## RestaurantsPriceRange2 6.325923e-04  
## RestaurantsReservations 4.077658e-04  
## Ambience.divey 2.604864e-04  
## RestaurantsDelivery 6.770611e-05  
## RestaurantsTakeOut 1.834782e-05

Chi-squared feature selection (for categorical variables)

ff\_assoc\_chsq <- chi.squared(stars~., data = ff\_assoc\_trim[,-1])  
  
ff\_assoc\_chsq %>%   
 arrange(-attr\_importance)

## attr\_importance  
## BusinessParking.street 0.134605601  
## NoiseLevel 0.128457657  
## Ambience.trendy 0.112008155  
## HasTV 0.103765903  
## Ambience.classy 0.103020805  
## Ambience.hipster 0.099584540  
## Ambience.casual 0.098185166  
## BikeParking 0.069473067  
## Alcoholfull\_bar 0.060738874  
## Ambience.intimate 0.055085813  
## OutdoorSeating 0.054775830  
## WiFifree 0.045136733  
## Alcoholbeer\_and\_wine 0.044696031  
## WiFino 0.043339760  
## Ambience.romantic 0.032658702  
## RestaurantsReservations 0.029324982  
## RestaurantsGoodForGroups 0.026626024  
## RestaurantsPriceRange2 0.026236435  
## BusinessParking.lot 0.025095195  
## Ambience.upscale 0.020008915  
## BusinessParking.validated 0.018576346  
## Ambience.divey 0.018498083  
## Ambience.touristy 0.017856831  
## RestaurantsDelivery 0.011778936  
## BusinessParking.valet 0.006335083  
## RestaurantsTakeOut 0.002937078  
## BusinessParking.garage 0.002860734  
## WiFipaid 0.000000000  
## RestaurantsAttirecasual 0.000000000  
## RestaurantsAttiredressy 0.000000000  
## RestaurantsAttireformal 0.000000000  
## Alcoholnone 0.000000000

Stepwise regression (suggests that 9 are best)

k <- ff\_assoc\_trim[,-1]  
assoc\_subset <- regsubsets(stars~., data = k)

## Reordering variables and trying again:

assoc\_subset\_sum <- summary(assoc\_subset)  
k<-assoc\_subset\_sum$which[9,]  
k[k==T]

## (Intercept) NoiseLevel'2' NoiseLevel'3'   
## TRUE TRUE TRUE   
## NoiseLevel'4' HasTV1 Ambience.casual1   
## TRUE TRUE TRUE   
## Ambience.classy1 Ambience.trendy1 BusinessParking.street1   
## TRUE TRUE TRUE   
## Alcoholfull\_bar   
## TRUE

When comparing the top 10 features across the three feature selection methods, 6 attributes were identified in all three (BusinessParking.street, NoiseLevel, Ambience.trendy, Ambience.classy, Ambience.casual, HasTV). And 3 attributes appeared in 2 of the 3 models (Ambience.hipster, BikeParking, Alcoholfull\_bar).

The association rules analysis will focus on the attributes that appeared in 2 or 3 of the feature selection methods. Attributes that were rated highly by only 1 model weren’t included.

#create a dataframe with the 9 attributes to be used in the association rules analysis  
ff\_assoc\_fs <- ff\_assoc\_trim[,c(2, 3, 6, 9, 12, 13, 15, 19, 23, 33)]  
  
#remove rows with all attributes, other than stars, having NA or 0  
sum(rowSums(is.na(ff\_assoc\_fs))==9) # number of rows with all NA values

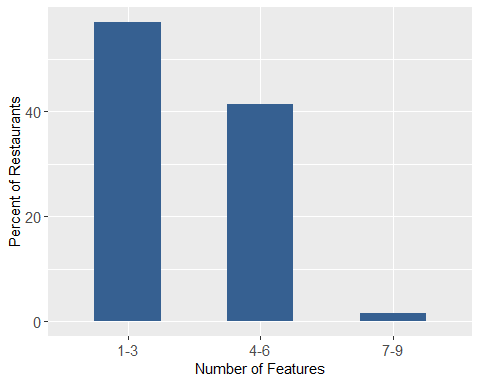
## [1] 1160

k <- which(rowSums(is.na(ff\_assoc\_fs))==9)  
ff\_assoc\_fs <- ff\_assoc\_fs[-k,]  
  
#change all the columns to factors  
ff\_assoc\_fs[1:ncol(ff\_assoc\_fs)] <- lapply(ff\_assoc\_fs[1:ncol(ff\_assoc\_fs)],factor)

#identify the number of features per restaurant  
k <- ff\_assoc\_fs %>%   
 mutate(noise\_present = if\_else(is.na(NoiseLevel),0,1)) %>%   
 select(-c(stars, NoiseLevel)) #creates a dataframe of only the features in binary format (0=absent; 1=present)  
k[1:ncol(k)] <- lapply(k[1:ncol(k)], as.character) #convert factors to characters  
k[1:ncol(k)] <- lapply(k[1:ncol(k)], as.numeric) #then convert to numeric (as you can't convert values directly from factor to numeric)  
  
j <- rowSums(k, na.rm = T) #total the number of present features per row  
table(j) #number of features per restaurant

## j  
## 0 1 2 3 4 5 6 7 8 9   
## 2440 5123 7743 9826 8924 5477 2037 515 100 23

m <- which(j==0) #index restaurants that have no features present (either NA or 0 for all values)  
  
ff\_assoc\_fs <- ff\_assoc\_fs[-m,] #remove rows without any features present  
  
#repeat conversion from factor to numeric without the 2440 rows  
k <- ff\_assoc\_fs %>%   
 mutate(noise\_present = if\_else(is.na(NoiseLevel),0,1)) %>%   
 select(-c(stars, NoiseLevel))  
k[1:ncol(k)] <- lapply(k[1:ncol(k)], as.character)  
k[1:ncol(k)] <- lapply(k[1:ncol(k)], as.numeric)  
  
j<- as.data.frame(rowSums(k, na.rm = T)) %>% rename(n\_attr='rowSums(k, na.rm = T)')   
  
j %>% mutate(num\_features = as.character(if\_else(n\_attr<7,  
 if\_else(n\_attr<4,"1-3","4-6"),"7-9"))) %>%  
 ggplot(aes(x=num\_features))+  
 geom\_bar(aes(y = ((..count..)/sum(..count..)\*100)), fill = "#366091", width = 0.5)+  
 xlab("Number of Features")+  
 ylab("Percent of Restaurants")+  
 theme(axis.text = element\_text(size=11))



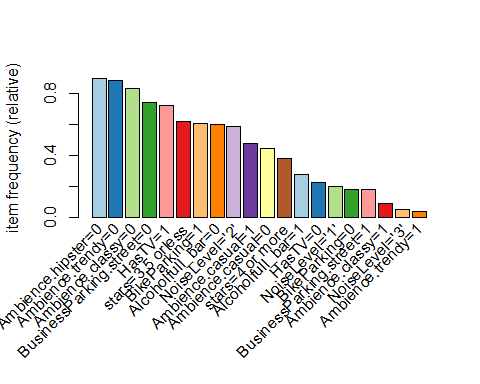
#Association Rule analysis

#convert the dataframe to a transaction object  
ff\_trans <- as(ff\_assoc\_fs, "transactions")  
summary(ff\_trans)

## transactions as itemMatrix in sparse format with  
## 39768 rows (elements/itemsets/transactions) and  
## 22 columns (items) and a density of 0.4128354   
##   
## most frequent items:  
## Ambience.hipster=0 Ambience.trendy=0 Ambience.classy=0   
## 35662 35182 33073   
## BusinessParking.street=0 HasTV=1 (Other)   
## 29457 28695 199119   
##   
## element (itemset/transaction) length distribution:  
## sizes  
## 2 3 4 5 6 7 8 9 10   
## 394 1052 704 652 684 1478 2469 6384 25951   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 9.000 10.000 9.082 10.000 10.000   
##   
## includes extended item information - examples:  
## labels variables levels  
## 1 stars=3.5 orless stars 3.5 orless  
## 2 stars=4 or more stars 4 or more  
## 3 BikeParking=0 BikeParking 0  
##   
## includes extended transaction information - examples:  
## transactionID  
## 1 1  
## 2 2  
## 3 3

The ambience, street parking and “has TV” attributes were most frequent.

#item frequncy plot of the top 20 items  
itemFrequencyPlot(ff\_trans, topN=20, type = "relative",col=brewer.pal(12,'Paired'))

 *Generating Rules* 38.17% of restaurants in the dataset have 4 or more stars. So I will start the parameters with support (frequency of the occurance of the item in the dataset) at 0.35 and confidence (the percentage of features that appear with 4 or more stars) at 0.50.

summarytools::freq(ff\_assoc\_fs$stars) #Frequency of the star ratings

## Frequencies   
## ff\_assoc\_fs$stars   
## Type: Factor   
##   
## Freq % Valid % Valid Cum. % Total % Total Cum.  
## ---------------- ------- --------- -------------- --------- --------------  
## 3.5 orless 24587 61.83 61.83 61.83 61.83  
## 4 or more 15181 38.17 100.00 38.17 100.00  
## <NA> 0 0.00 100.00  
## Total 39768 100.00 100.00 100.00 100.00

#setting the RHS appearance to "stars >= 4"  
association\_rules <- apriori(ff\_trans, parameter = list(supp=0.05, conf=0.4), appearance = list(default="lhs", rhs="stars=4 or more") )

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.4 0.1 1 none FALSE TRUE 5 0.05 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 1988   
##   
## set item appearances ...[1 item(s)] done [0.00s].  
## set transactions ...[22 item(s), 39768 transaction(s)] done [0.03s].  
## sorting and recoding items ... [19 item(s)] done [0.00s].  
## creating transaction tree ... done [0.02s].  
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.01s].  
## writing ... [295 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

I tested the following combinations for support and confidence. They produced the number of rules listed below. The very low values of both support and confidence required to generate rules, suggest that the rules generated may not be useful or interesting for real-world use.

Support (0.35) & Confidence (0.5) –> 0 Rules; Support (0.25) & Confidence (0.5) –> 0 Rules; Support (0.15) & Confidence (0.5) –> 0 Rules; Support (0.05) & Confidence (0.5) –> 40 Rules; Support (0.05) & Confidence (0.55) –> 5 Rule; Support (0.05) & Confidence (0.6) –> 0 Rules; Support (0.05) & Confidence (0.4) –> 295 Rules;

Inspect Rules

#length(association\_rules)#total number of rules  
present\_rules <- subset(association\_rules, subset=lhs %pin% '=1') #reduces the rules to those that have at least one value present ("=1" on the left-hand-side)  
  
length(present\_rules) #total number of rules that have at least one value present

## [1] 229

inspect(head(present\_rules, 20))

## lhs rhs support confidence coverage lift count  
## [1] {BusinessParking.street=1} => {stars=4 or more} 0.09366828 0.5232476 0.1790133 1.370694 3725  
## [2] {Ambience.casual=1} => {stars=4 or more} 0.20697546 0.4352493 0.4755331 1.140175 8231  
## [3] {BikeParking=1} => {stars=4 or more} 0.25251458 0.4137278 0.6103400 1.083797 10042  
## [4] {Ambience.casual=1,   
## BusinessParking.street=1} => {stars=4 or more} 0.06072722 0.5293731 0.1147153 1.386741 2415  
## [5] {NoiseLevel='2',   
## BusinessParking.street=1} => {stars=4 or more} 0.06123013 0.5218603 0.1173305 1.367060 2435  
## [6] {BusinessParking.street=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.06112955 0.5711936 0.1070207 1.496293 2431  
## [7] {BikeParking=1,   
## BusinessParking.street=1} => {stars=4 or more} 0.07337558 0.5387740 0.1361899 1.411367 2918  
## [8] {HasTV=1,   
## BusinessParking.street=1} => {stars=4 or more} 0.05318346 0.4873272 0.1091330 1.276598 2115  
## [9] {Ambience.classy=0,   
## BusinessParking.street=1} => {stars=4 or more} 0.07380306 0.5059473 0.1458711 1.325374 2935  
## [10] {Ambience.trendy=0,   
## BusinessParking.street=1} => {stars=4 or more} 0.08013981 0.5108190 0.1568849 1.338137 3187  
## [11] {Ambience.hipster=0,   
## BusinessParking.street=1} => {stars=4 or more} 0.08177429 0.5136629 0.1591984 1.345586 3252  
## [12] {BikeParking=1,   
## NoiseLevel='1'} => {stars=4 or more} 0.05484309 0.4807141 0.1140867 1.259274 2181  
## [13] {NoiseLevel='1',   
## HasTV=1} => {stars=4 or more} 0.05338463 0.4111154 0.1298531 1.076954 2123  
## [14] {HasTV=0,   
## Ambience.casual=1} => {stars=4 or more} 0.06495172 0.5136210 0.1264585 1.345477 2583  
## [15] {BikeParking=1,   
## HasTV=0} => {stars=4 or more} 0.07546268 0.5199238 0.1451418 1.361987 3001  
## [16] {NoiseLevel='2',   
## Ambience.casual=1} => {stars=4 or more} 0.15542647 0.4316642 0.3600634 1.130783 6181  
## [17] {Ambience.casual=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.14763126 0.4833292 0.3054466 1.266125 5871  
## [18] {BikeParking=1,   
## Ambience.casual=1} => {stars=4 or more} 0.15947495 0.4532266 0.3518658 1.187268 6342  
## [19] {HasTV=1,   
## Ambience.casual=1} => {stars=4 or more} 0.14159626 0.4068054 0.3480688 1.065663 5631  
## [20] {Ambience.casual=1,   
## BusinessParking.street=0} => {stars=4 or more} 0.14380909 0.4049136 0.3551599 1.060708 5719

rules\_by\_confidence <- sort(present\_rules, by = "confidence")  
inspect(head(rules\_by\_confidence, 5)) #top five rules (by confidence)

## lhs rhs support confidence coverage lift count  
## [1] {BusinessParking.street=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.06112955 0.5711936 0.10702072 1.496293 2431  
## [2] {Ambience.hipster=0,   
## BusinessParking.street=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.05426474 0.5621256 0.09653490 1.472539 2158  
## [3] {Ambience.trendy=0,   
## BusinessParking.street=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.05449105 0.5615444 0.09703782 1.471016 2167  
## [4] {Ambience.classy=0,   
## BusinessParking.street=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.05295715 0.5601064 0.09454838 1.467249 2106  
## [5] {Ambience.hipster=0,   
## Ambience.trendy=0,   
## BusinessParking.street=1,   
## Alcoholfull\_bar=0} => {stars=4 or more} 0.05124723 0.5551621 0.09231040 1.454297 2038

Subset Rules

subset\_rules <- which(colSums(is.subset(present\_rules, present\_rules))>1)  
subset\_present\_rules <- present\_rules[-subset\_rules]  
length(subset\_present\_rules)

## [1] 4

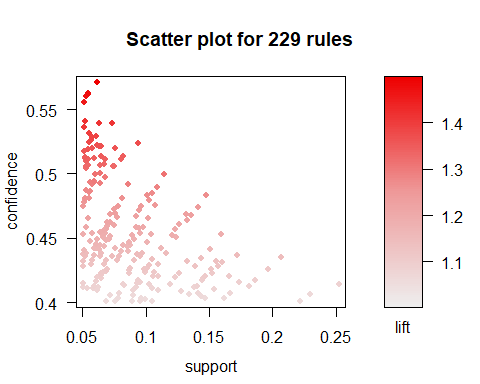
inspect(subset\_present\_rules)

## lhs rhs support confidence  
## [1] {BusinessParking.street=1} => {stars=4 or more} 0.09366828 0.5232476   
## [2] {Ambience.casual=1} => {stars=4 or more} 0.20697546 0.4352493   
## [3] {BikeParking=1} => {stars=4 or more} 0.25251458 0.4137278   
## [4] {NoiseLevel='1',HasTV=1} => {stars=4 or more} 0.05338463 0.4111154   
## coverage lift count  
## [1] 0.1790133 1.370694 3725  
## [2] 0.4755331 1.140175 8231  
## [3] 0.6103400 1.083797 10042  
## [4] 0.1298531 1.076954 2123

The features that are most commonly part of a multi-item set are Street Parking, Casual Ambience, Bike Parking and quiet noise leve & TV.

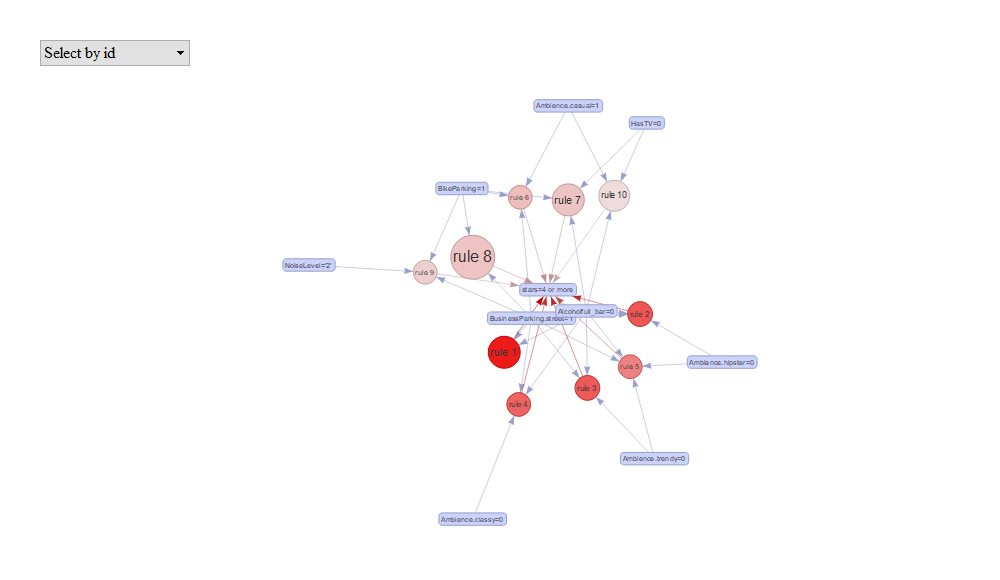
The following plot shows the confidence, support and lift for each association rule. The support is very low, less than 10% for most rules with confidence of between 45 and 55%. The list for a few rules is greater than 1 indicating that for a few rules the combination of feature is more common than expected in restaurants with 4 or more stars.

scatter\_rules <- present\_rules[quality(present\_rules)$confidence>0.3]  
plot(scatter\_rules)



Another way to visualize the rules is through a graph-based visualization where the features and rules are two types of vertices and item-sets as edges. Arrows from an item to a rule represent the LHS of a rule and arrows from a rule to an item represent the RHS of the rule. All the rules point to stars=4 or more, as that is the RHS value for all the rules in the analysis.

set.seed(2020)  
top10\_presrules <- head(present\_rules, n=10, by = "confidence")  
plot(top10\_presrules, method = "graph", engine = "htmlwidget")



#Load Yelp Reviews for Family Friendly Restaurants

The Yelp reviews were converted from JSON to CSV using Apache Spark and Python and provided by the course supervisor. The full review csv was loaded to R, and the subset of reviews for family friendly restaurants was identified by joining the CSV to the family-friendly restauarant subset by business\_ID.

#load a csv file of the Yelp reviews  
#reviews\_df <- fread("dataset\_review.csv")

#Join the ff\_subset of family friendly restaurant2 to the reviews\_df by business\_id  
#ff\_reviews <- left\_join(ff\_subset, reviews\_df[,c(1,7)], by = c("ID"="V1"))

The df that with family friendly restaurant information and their reviews was exported to .Rdata for future data loads.

#remove extra attributes (e.g., address, latitude)  
#ff\_reviews <- ff\_reviews[,c(2,3,5,6,10:60)]  
#save(ff\_reviews, file = "reviews.RData")

Load the combined business/review data file from the .Rdata file.

#ff\_reviews <- load("Data/reviews.RData")

There were more than 4 million reviews available for analysis. A random sample of 25% of the records was selected for the analysis.

#create a sample of restaurants for analysis  
#set.seed(6811)  
#sample\_row <- sample(nrow(ff\_reviews), floor(nrow(ff\_reviews)\*0.25))  
#reviews\_sample <- ff\_reviews[sample\_row,]

The df of the random sample of family friendly restaurants & reviews was exported to .Rdata for future data loads.

#export the sample of ff\_reviews to .Rdata  
#save(reviews\_sample, file = "reviews\_sample.RData")

Load the sample review file from the .Rdata file.

review\_sample\_ff <- load("Data/reviews\_sample.RData")

Select the columns relevant for analysis of reviews and format as tidy text.

reviews\_sample[1:2,]

## ID name city  
## 1072901 CvpXZRF-I7wfm19dW1jOXA Mark Rich's New York Pizza & Pasta Las Vegas  
## 634717 8chcaZIRlP4p5anpzUB3FQ Dirty Fork Las Vegas  
## state stars review\_count is\_open  
## 1072901 NV 3.5 245 1  
## 634717 NV 4.0 562 1  
## categories  
## 1072901 Italian, Pizza, Restaurants  
## 634717 Coffee & Tea, Breakfast & Brunch, Restaurants, Food, Hawaiian  
## attributes.BusinessAcceptsCreditCards attributes.BikeParking  
## 1072901 True True  
## 634717 True True  
## attributes.GoodForKids  
## 1072901 True  
## 634717 True  
## attributes.BusinessParking  
## 1072901 {'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': False}  
## 634717 {'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': False}  
## attributes.ByAppointmentOnly attributes.RestaurantsPriceRange2  
## 1072901 False 2  
## 634717 False 2  
## attributes.DogsAllowed attributes.WiFi attributes.RestaurantsAttire  
## 1072901 <NA> u'free' 'casual'  
## 634717 True 'free' 'casual'  
## attributes.RestaurantsTakeOut attributes.NoiseLevel  
## 1072901 True u'average'  
## 634717 True 'average'  
## attributes.RestaurantsReservations attributes.RestaurantsGoodForGroups  
## 1072901 True True  
## 634717 True True  
## attributes.HasTV attributes.Alcohol attributes.RestaurantsDelivery  
## 1072901 True u'beer\_and\_wine' True  
## 634717 True 'none' True  
## attributes.OutdoorSeating attributes.Caters  
## 1072901 True True  
## 634717 True False  
## attributes.WheelchairAccessible attributes.AcceptsInsurance  
## 1072901 <NA> <NA>  
## 634717 True <NA>  
## attributes.RestaurantsTableService  
## 1072901 <NA>  
## 634717 True  
## attributes.Ambience  
## 1072901 {'romantic': False, 'intimate': False, 'touristy': False, 'hipster': False, 'divey': False, 'classy': False, 'trendy': False, 'upscale': False, 'casual': True}  
## 634717 {'touristy': False, 'hipster': False, 'romantic': False, 'divey': False, 'intimate': False, 'trendy': True, 'upscale': False, 'classy': False, 'casual': True}  
## attributes.GoodForMeal  
## 1072901 {'dessert': None, 'latenight': False, 'lunch': True, 'dinner': True, 'brunch': False, 'breakfast': False}  
## 634717 {'dessert': False, 'latenight': False, 'lunch': True, 'dinner': False, 'brunch': True, 'breakfast': True}  
## attributes.HappyHour attributes.BusinessAcceptsBitcoin attributes.BYOB  
## 1072901 True <NA> <NA>  
## 634717 False <NA> <NA>  
## attributes.Corkage attributes.GoodForDancing attributes.CoatCheck  
## 1072901 <NA> <NA> <NA>  
## 634717 <NA> <NA> <NA>  
## attributes.BestNights attributes.Music attributes.Smoking  
## 1072901 <NA> <NA> <NA>  
## 634717 <NA> <NA> <NA>  
## attributes.DietaryRestrictions attributes.DriveThru  
## 1072901 <NA> <NA>  
## 634717 <NA> <NA>  
## attributes.HairSpecializesIn attributes.BYOBCorkage  
## 1072901 <NA> <NA>  
## 634717 <NA> <NA>  
## attributes.AgesAllowed attributes.RestaurantsCounterService  
## 1072901 <NA> <NA>  
## 634717 <NA> <NA>  
## attributes.Open24Hours hours.Monday hours.Tuesday hours.Wednesday  
## 1072901 <NA> 11:0-21:0 11:0-21:0 11:0-21:0  
## 634717 <NA> 7:0-15:30 7:0-15:30 7:0-15:30  
## hours.Thursday hours.Friday hours.Saturday hours.Sunday  
## 1072901 11:0-21:0 11:0-22:0 11:0-22:0 11:0-21:0  
## 634717 7:0-15:30 7:0-15:30 7:0-15:30 7:0-15:30  
## V7  
## 1072901 If you are in the West Charleston/Downtown Summerlin corridor, stop in Mark Rich's place for a good meal in a local family owned joint. \nConsistently good pizza, fun vibe for a casual date night or family get together at this local neighborhood hotspot. Also try the chicken cacciatore, rice balls and maybe a black & white for dessert. Small selection of beer and wine available. Delivery or takeout options. Not my most favorite pizza place but always good, friendly servers and good atmosphere at a reasonable price.  
## 634717 Came in for brunch today and was totally blown away by everything on the menu. Maggi helped us and was very pleasant. I ordered the Lunch Special and my girlfriend ordered the \\"What the Fork Omelet\\" and was delicious. We are absolutely coming in next time and ordering the blueberry pancakes! Great brunch spot!

tidy\_review <- reviews\_sample %>%   
 select(ID, stars, V7) %>%  
 rename(text=V7) %>%   
 group\_by(ID) %>%   
 mutate(review\_num = row\_number()) %>%   
 ungroup()

*Exploration and preparation of reviews data*

#Number of ff\_restaurants represented in the sample  
reviews\_sample %>%   
 distinct(ID) %>%   
 count()

## n  
## 1 41253

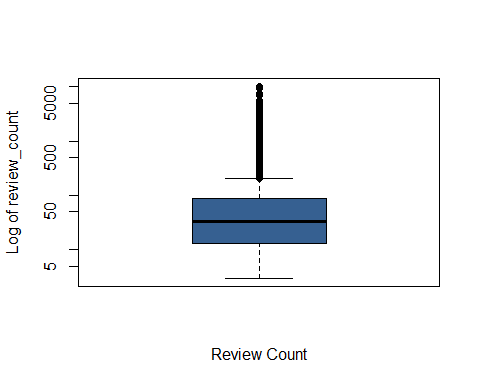
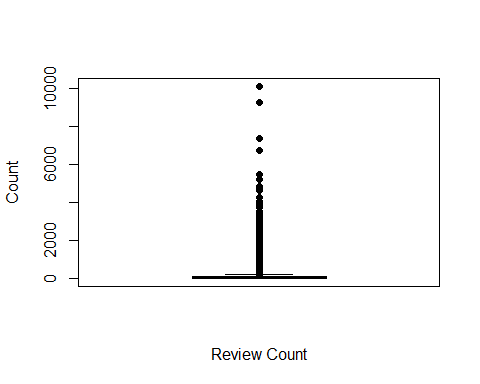
#df of review\_counts  
r\_count <- reviews\_sample %>%   
 select(ID, review\_count) %>%   
 distinct(ID, .keep\_all=TRUE)   
  
#average number of reviews per business in sample  
r\_count %>%   
 summarise(round(mean(review\_count),2))

## round(mean(review\_count), 2)  
## 1 92.63

#distribution of number of reviews  
summary(r\_count$review\_count)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.00 13.00 33.00 92.63 89.00 10129.00

There are 41253 restaurants included in reviews sample, which means 95% of the businesses in the family-friendly restaurant subset are represented in the reviews sample. The average number of reviews per restaurant (92.63) is higher than the median (33.00), suggesting that the sample continues to have outliers.



#number of outliers  
length(boxplot.stats(r\_count$review\_count)$out)

## [1] 4340

There are 4340 businesses with review counts that are higher than expected in the sample. That is more than 10% of the restaurants in the sample. And those restaurants had a total of 554,103 reviews, which made up more than half the review sample (55.8%). To prevent the reviews for these businesses from skewing the analysis, they will be removed.

reviews\_out <- which(r\_count$review\_count %in% boxplot.stats(r\_count$review\_count)$out)  
reviews\_out <- r\_count$ID[reviews\_out]  
  
reviews\_clean<- tidy\_review %>%   
 filter(!ID %in% reviews\_out)  
  
nrow(reviews\_sample)-nrow(reviews\_clean)#number of reviews in the sample for the outliers

## [1] 554103

round(((nrow(reviews\_sample)-nrow(reviews\_clean))/nrow(reviews\_sample)\*100),2)#percent of reviews in the sample for the outliers

## [1] 55.8

*Identify the reviews that mention children.*

#Tokenize the reviews at the word level   
tidy\_review <- reviews\_clean %>%   
 unnest\_tokens(word, text)

I created a custom lexicon of child related terms (e.g., kid, teen, baby, son, neice) to use to identify the reviews that mention children. Only reviews that mention children are included in this analysis.

#Import custom lexicon with child related terms   
kid\_lexicon <- read.csv("Data/KidLexicon.csv", header = T)  
  
sum(tidy\_review$word %in% kid\_lexicon$Term) #55483 words that match the kid\_lexicon

## [1] 97939

kid\_review\_list <- tidy\_review[which(tidy\_review$word %in% kid\_lexicon$Term),]%>%   
 select(ID, review\_num) %>%   
 group\_by(ID, review\_num) %>%   
 distinct() %>%   
 mutate(uniqueID = paste0(ID,review\_num))%>%  
 ungroup() %>%   
 select(uniqueID)  
   
#subset the data to include only the reviews that mention children.  
reviews\_kids <- reviews\_clean %>%  
 mutate(uniqueID = paste0(ID,review\_num)) %>%  
 filter(uniqueID %in% kid\_review\_list$uniqueID) %>%   
 select(-uniqueID)   
  
nrow(reviews\_kids)#number of reviews that mention kids

## [1] 64806

round((nrow(reviews\_kids)/nrow(reviews\_clean)\*100),2) #percent of the cleaned reviews that mention kids

## [1] 14.76

14.76% of the reviews for the sample of family-friendly restaurants mention kids. These are the reviews that will be the focus of the analysis.

# Sentiment Analysis

#create a tidy tibble with each word in each review on it's own row for bag of words analysis.  
tidy\_reviews\_kids<-reviews\_kids %>%   
 unnest\_tokens(word, text)  
  
#remove words that are numbers  
k <- which(str\_detect(tidy\_reviews\_kids$word, "\\d+"))  
tidy\_reviews\_kids <- tidy\_reviews\_kids[-k,]  
  
#remove words that are "-" or less than 2 charachters long  
tidy\_reviews\_kids <- tidy\_reviews\_kids %>% filter(!str\_detect(word, "\_+"))  
  
#Remove words that are 2 characters or shorter  
k <- which(nchar(tidy\_reviews\_kids$word)<=2)  
tidy\_reviews\_kids <- tidy\_reviews\_kids[-k,]  
  
#remove stop words  
tidy\_reviews\_kids <- tidy\_reviews\_kids %>%   
 anti\_join(custom\_stop\_words)  
  
#lemmatize  
tidy\_reviews\_kids <- tidy\_reviews\_kids %>%   
 mutate(word = lemmatize\_words(word))

#check the frequency of the words  
tidy\_reviews\_kids %>% count(word, sort = T)

## # A tibble: 58,441 x 2  
## word n  
## <chr> <int>  
## 1 good 81571  
## 2 food 65503  
## 3 order 53447  
## 4 get 51803  
## 5 place 51731  
## 6 come 39255  
## 7 like 39244  
## 8 time 35900  
## 9 just 31871  
## 10 one 31812  
## # ... with 58,431 more rows

#top 10 words  
tidy\_reviews\_kids %>% count(word, sort = T) %>% top\_n(10)

## Selecting by n

## # A tibble: 10 x 2  
## word n  
## <chr> <int>  
## 1 good 81571  
## 2 food 65503  
## 3 order 53447  
## 4 get 51803  
## 5 place 51731  
## 6 come 39255  
## 7 like 39244  
## 8 time 35900  
## 9 just 31871  
## 10 one 31812

tidy\_reviews\_kids %>% count(word, sort = T) %>% top\_n(-10)

## Selecting by n

## # A tibble: 27,869 x 2  
## word n  
## <chr> <int>  
## 1 ˆthey 1  
## 2 ˆwe 1  
## 3 µåœ 1  
## 4 14 1  
## 5 a'liege 1  
## 6 a'plenty 1  
## 7 a.f 1  
## 8 a.manager 1  
## 9 a.ok 1  
## 10 a.pizza 1  
## # ... with 27,859 more rows

unpopular <- bind\_rows(tidy\_reviews\_kids %>% count(word, sort = T) %>% top\_n(10) %>% select(word),tidy\_reviews\_kids %>% count(word, sort = T) %>% top\_n(-10) %>% select(word)) #least popular words in the reviews

## Selecting by n  
## Selecting by n

#remove 27,869 words that only appear once in the dataset  
tidy\_reviews\_kids <- tidy\_reviews\_kids %>%   
 anti\_join(unpopular)

## Joining, by = "word"

*Comparing lexicons*

First, I assess sentiment for reviews at the word level. I compared three lexicons: AFINN, NRC and Bing.

*Load the lexicons* AFINN is a general purpose lexicon

afinn\_lexicon <- lexicon\_afinn()  
  
afinn\_lexicon %>% count(value <0) #There are more negative than positive terms

## # A tibble: 2 x 2  
## `value < 0` n  
## <lgl> <int>  
## 1 FALSE 879  
## 2 TRUE 1598

NRC is a general purpose lexicon

nrc\_lexicon <- lexicon\_nrc()  
  
nrc\_lexicon %>% count(sentiment =="negative") #3324 negative

## # A tibble: 2 x 2  
## `sentiment == "negative"` n  
## <lgl> <int>  
## 1 FALSE 10577  
## 2 TRUE 3324

nrc\_lexicon %>% count(sentiment =="positive") #2312 positive (also more negative than positive terms)

## # A tibble: 2 x 2  
## `sentiment == "positive"` n  
## <lgl> <int>  
## 1 FALSE 11589  
## 2 TRUE 2312

Bing is a general purpose lexicon, and some have suggested that it is particularly good for social media (<https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>)

bing\_lexicon <- get\_sentiments("bing")  
  
bing\_lexicon %>% count(sentiment =="negative") #more negative than positive

## # A tibble: 2 x 2  
## `sentiment == "negative"` n  
## <lgl> <int>  
## 1 FALSE 2005  
## 2 TRUE 4781

*Combine lexicons with reviews to calculate sentiment values*

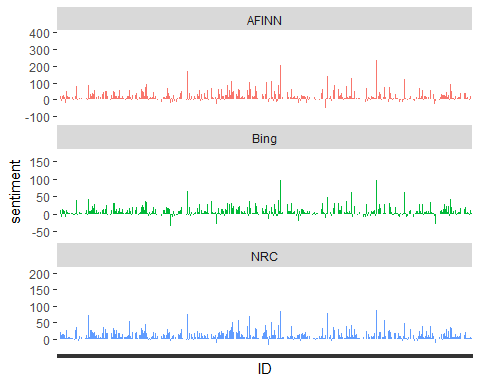
afinn <- tidy\_reviews\_kids %>%   
 inner\_join(afinn\_lexicon) %>%   
 group\_by(ID) %>%   
 summarise(sentiment = sum(value)) %>%   
 mutate(method = "AFINN")

nrc <- tidy\_reviews\_kids %>%   
 inner\_join(nrc\_lexicon) %>%   
 group\_by(ID) %>%   
 count(sentiment)%>%   
 spread(sentiment, n, fill = 0) %>%   
 mutate(sentiment = positive - negative) %>%   
 mutate(method = "NRC") %>%   
 select(ID, negative, positive, sentiment, method)

bing <- tidy\_reviews\_kids %>%   
 inner\_join(bing\_lexicon) %>%   
 group\_by(ID) %>%   
 count(sentiment) %>%   
 spread(sentiment, n, fill = 0) %>%   
 mutate(sentiment = positive - negative) %>%   
 mutate(method = "Bing")

*Combine the three sentiment dataframes into one for comparison.*

bind\_rows(afinn, nrc, bing) %>%   
 ggplot(aes(ID, sentiment, fill = method)) +  
 geom\_col(show.legend = FALSE)+  
 facet\_wrap(~method, ncol=1, scales="free\_y")+  
 theme(axis.text.x = element\_blank())



The Afinn sentiment scores are objectively higher than the Bing and NRC. That’s not unexpected as it was aggregated by adding up individual word sentiment from the likert score for each business. That differs from Bing and NRC which were aggregated by subtracting the total number of positive words from the total number of negative words per business.

While all three lexicons have more negative than positive words, the overall review sentiment for all three is positive.

*What are the most common positive and negative terms for the 3 lexicons?*

#word counts afinn  
afinn\_count <- tidy\_reviews\_kids %>%   
 inner\_join(afinn\_lexicon) %>%   
 count(word, value, sort=T) %>%   
 ungroup  
  
afinn\_count

## # A tibble: 1,146 x 3  
## word value n  
## <chr> <dbl> <int>  
## 1 great 3 29882  
## 2 love 3 19127  
## 3 want 1 16917  
## 4 nice 3 15254  
## 5 friendly 2 12745  
## 6 bad -3 11314  
## 7 fresh 1 11033  
## 8 pretty 1 9710  
## 9 leave -1 9378  
## 10 enjoy 2 8789  
## # ... with 1,136 more rows

#word counts afinn positive  
tidy\_reviews\_kids %>%   
 inner\_join(afinn\_lexicon) %>%   
 count(word, value, sort=T) %>%   
 filter(value > 0) %>%   
 ungroup

## # A tibble: 461 x 3  
## word value n  
## <chr> <dbl> <int>  
## 1 great 3 29882  
## 2 love 3 19127  
## 3 want 1 16917  
## 4 nice 3 15254  
## 5 friendly 2 12745  
## 6 fresh 1 11033  
## 7 pretty 1 9710  
## 8 enjoy 2 8789  
## 9 amaze 2 7859  
## 10 big 1 7593  
## # ... with 451 more rows

#word counts afinn negative  
tidy\_reviews\_kids %>%   
 inner\_join(afinn\_lexicon) %>%   
 count(word, value, sort=T) %>%   
 filter(value<0) %>%   
 ungroup

## # A tibble: 685 x 3  
## word value n  
## <chr> <dbl> <int>  
## 1 bad -3 11314  
## 2 leave -1 9378  
## 3 pay -1 6600  
## 4 disappoint -2 5909  
## 5 stop -1 5581  
## 6 hard -1 3922  
## 7 wrong -2 3741  
## 8 forget -1 2808  
## 9 miss -2 2808  
## 10 cut -1 2495  
## # ... with 675 more rows

#word counts nrc  
nrc\_count <- tidy\_reviews\_kids %>%   
 inner\_join(nrc\_lexicon) %>%  
 filter(sentiment=="negative"|sentiment=="positive") %>%   
 count(word, sentiment, sort=T) %>%   
 ungroup  
  
nrc\_count

## # A tibble: 3,366 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 eat positive 23946  
## 2 love positive 19127  
## 3 wait negative 16981  
## 4 friendly positive 12745  
## 5 delicious positive 12073  
## 6 bad negative 11314  
## 7 bite negative 10681  
## 8 customer positive 10608  
## 9 serve negative 10002  
## 10 pretty positive 9710  
## # ... with 3,356 more rows

#word counts nrc - positive  
tidy\_reviews\_kids %>%   
 inner\_join(nrc\_lexicon) %>%   
 filter(sentiment=="negative"|sentiment=="positive") %>%   
 count(word, sentiment, sort=T) %>%   
 filter(sentiment=="positive") %>%   
 ungroup

## # A tibble: 1,516 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 eat positive 23946  
## 2 love positive 19127  
## 3 friendly positive 12745  
## 4 delicious positive 12073  
## 5 customer positive 10608  
## 6 pretty positive 9710  
## 7 dinner positive 9617  
## 8 visit positive 8855  
## 9 enjoy positive 8789  
## 10 friend positive 8571  
## # ... with 1,506 more rows

#word counts nrc  
tidy\_reviews\_kids %>%   
 inner\_join(nrc\_lexicon) %>%   
 filter(sentiment=="negative"|sentiment=="positive") %>%   
 count(word, sentiment, sort=T) %>%   
 filter(sentiment=="negative") %>%   
 ungroup

## # A tibble: 1,850 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 wait negative 16981  
## 2 bad negative 11314  
## 3 bite negative 10681  
## 4 serve negative 10002  
## 5 small negative 9527  
## 6 leave negative 9378  
## 7 disappoint negative 5909  
## 8 late negative 5281  
## 9 boy negative 4686  
## 10 cold negative 4190  
## # ... with 1,840 more rows

#word counts bing  
bing\_count <- tidy\_reviews\_kids %>%   
 inner\_join(bing\_lexicon) %>%   
 count(word, sentiment, sort=T) %>%   
 ungroup  
  
bing\_count

## # A tibble: 3,077 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 great positive 29882  
## 2 love positive 19127  
## 3 nice positive 15254  
## 4 friendly positive 12745  
## 5 delicious positive 12073  
## 6 bad negative 11314  
## 7 fresh positive 11033  
## 8 work positive 9761  
## 9 pretty positive 9710  
## 10 right positive 9465  
## # ... with 3,067 more rows

#word counts bing - positive  
tidy\_reviews\_kids %>%   
 inner\_join(bing\_lexicon) %>%   
 count(word, sentiment, sort=T) %>%   
 filter(sentiment=="positive") %>%   
 ungroup

## # A tibble: 1,079 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 great positive 29882  
## 2 love positive 19127  
## 3 nice positive 15254  
## 4 friendly positive 12745  
## 5 delicious positive 12073  
## 6 fresh positive 11033  
## 7 work positive 9761  
## 8 pretty positive 9710  
## 9 right positive 9465  
## 10 hot positive 8925  
## # ... with 1,069 more rows

#word counts bing  
tidy\_reviews\_kids %>%   
 inner\_join(bing\_lexicon) %>%   
 count(word, sentiment, sort=T) %>%   
 filter(sentiment=="negative") %>%   
 ungroup

## # A tibble: 1,998 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 bad negative 11314  
## 2 disappoint negative 5909  
## 3 cold negative 4190  
## 4 hard negative 3922  
## 5 wrong negative 3741  
## 6 rude negative 3102  
## 7 miss negative 2808  
## 8 cheap negative 2763  
## 9 slow negative 2538  
## 10 problem negative 2495  
## # ... with 1,988 more rows

*Is the overall review sentiment for businesses consistent across the 3 sentiment lexicons?*

#comparison of the three lexicons  
sentiment\_by\_word <- bind\_rows(nrc, bing) %>%   
 select(-c(negative, positive)) %>%   
 bind\_rows(afinn)  
  
#count of negative scores per method  
sentiment\_by\_word %>% mutate(negative = ifelse(sentiment<0,"True","False")) %>% group\_by(method) %>% count(negative)

## # A tibble: 6 x 3  
## # Groups: method [3]  
## method negative n  
## <chr> <chr> <int>  
## 1 AFINN False 18659  
## 2 AFINN True 3600  
## 3 Bing False 17910  
## 4 Bing True 4409  
## 5 NRC False 19945  
## 6 NRC True 2429

NRC’s lexicon rated the reviews as least negative. Afinn had more negative ratings and Bing had the most negative ratings.

sentiment\_by\_word$method <- as.factor(sentiment\_by\_word$method)  
  
kruskal.test(sentiment~method, data = sentiment\_by\_word)

##   
## Kruskal-Wallis rank sum test  
##   
## data: sentiment by method  
## Kruskal-Wallis chi-squared = 2981.9, df = 2, p-value < 2.2e-16

posthoc.kruskal.nemenyi.test(sentiment~method, data = sentiment\_by\_word)

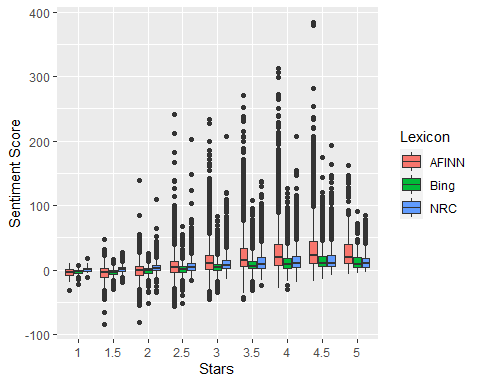
## Warning in posthoc.kruskal.nemenyi.test.default(c(54, 4, 10, -2, 3, 26, : Ties  
## are present, p-values are not corrected.

##   
## Pairwise comparisons using Tukey and Kramer (Nemenyi) test   
## with Tukey-Dist approximation for independent samples   
##   
## data: sentiment by method   
##   
## AFINN Bing   
## Bing <2e-16 -   
## NRC <2e-16 <2e-16  
##   
## P value adjustment method: none

There is a significant difference between the three lexicons in the overall sentiment. Post-hoc tests indicate that each of the lexicons differs significantly from the others.

*Is there a relationship between a restaurant’s star rating and the review sentiment?*

#relationship between stars and word level sentiment (grouped by lexicon)  
sentiment\_by\_word <- left\_join(sentiment\_by\_word, (reviews\_kids %>% select(ID, stars) %>% distinct(ID, .keep\_all = T)) )  
  
sentiment\_by\_word %>%   
 ggplot(aes(x = factor(stars), y = sentiment, fill=factor(method)))+  
 geom\_boxplot()+  
 xlab("Stars")+  
 ylab("Sentiment Score")+  
 guides(fill=guide\_legend(title="Lexicon"))



There is a linear upward trend that as the number of stars increases the sentiment rating is more positive. The relationship is strongest for the Bing lexicon and weakest for the NRC. But for all three methods the strength of association is weak (<0.25).

*What is the relationship between stars and word level sentiment?*

#relationship between stars and word level sentiment (using bing)  
bing\_stars <- sentiment\_by\_word %>% filter(method=="Bing") %>% left\_join(reviews\_sample[,c(1,5)])  
cor.test(bing\_stars$sentiment, bing\_stars$stars, method = "spearman")

## Warning in cor.test.default(bing\_stars$sentiment, bing\_stars$stars, method =  
## "spearman"): Cannot compute exact p-value with ties

##   
## Spearman's rank correlation rho  
##   
## data: bing\_stars$sentiment and bing\_stars$stars  
## S = 4.9617e+15, p-value < 2.2e-16  
## alternative hypothesis: true rho is not equal to 0  
## sample estimates:  
## rho   
## 0.4163207

nrc\_stars <- sentiment\_by\_word %>% filter(method=="NRC") %>% left\_join(reviews\_sample[,c(1,5)])  
cor.test(nrc\_stars$sentiment, nrc\_stars$stars, method = "spearman")

## Warning in cor.test.default(nrc\_stars$sentiment, nrc\_stars$stars, method =  
## "spearman"): Cannot compute exact p-value with ties

##   
## Spearman's rank correlation rho  
##   
## data: nrc\_stars$sentiment and nrc\_stars$stars  
## S = 6.382e+15, p-value < 2.2e-16  
## alternative hypothesis: true rho is not equal to 0  
## sample estimates:  
## rho   
## 0.2517113

afinn\_stars <- sentiment\_by\_word %>% filter(method=="AFINN") %>% left\_join(reviews\_sample[,c(1,5)])  
cor.test(afinn\_stars$sentiment, afinn\_stars$stars, method = "spearman")

## Warning in cor.test.default(afinn\_stars$sentiment, afinn\_stars$stars, method =  
## "spearman"): Cannot compute exact p-value with ties

##   
## Spearman's rank correlation rho  
##   
## data: afinn\_stars$sentiment and afinn\_stars$stars  
## S = 5.2483e+15, p-value < 2.2e-16  
## alternative hypothesis: true rho is not equal to 0  
## sample estimates:  
## rho   
## 0.3795372

The Bing lexicon is used for subsequent analysis as it had the strongest correlation. I looked more closely at the Bing lexicon for variation in sentiment by number of stars.

kruskal.test(sentiment~stars, data = bing\_stars)#testing is sentiment varies by number of stars for Bing

##   
## Kruskal-Wallis rank sum test  
##   
## data: sentiment by stars  
## Kruskal-Wallis chi-squared = 71413, df = 8, p-value < 2.2e-16

posthoc.kruskal.nemenyi.test(sentiment~stars, data = bing\_stars)

## Warning in posthoc.kruskal.nemenyi.test.default(c(30, 30, 30, 30, 30, 30, : Ties  
## are present, p-values are not corrected.

##   
## Pairwise comparisons using Tukey and Kramer (Nemenyi) test   
## with Tukey-Dist approximation for independent samples   
##   
## data: sentiment by stars   
##   
## 1 1.5 2 2.5 3 3.5 4 4.5   
## 1.5 1.000 - - - - - - -   
## 2 0.027 1.0e-13 - - - - - -   
## 2.5 9.1e-14 < 2e-16 < 2e-16 - - - - -   
## 3 < 2e-16 < 2e-16 < 2e-16 < 2e-16 - - - -   
## 3.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 - - -   
## 4 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 - -   
## 4.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -   
## 5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.000  
##   
## P value adjustment method: none

#confirm that these differences also hold for the categories used in association analysis (<=3.5 and >=4)  
bing\_cat <- bing\_stars %>% select(-stars) %>% left\_join(ff\_assoc\_trim[,1:2])  
  
kruskal.test(sentiment~stars, data = bing\_cat)#testing is sentiment varies by number of stars for Bing

##   
## Kruskal-Wallis rank sum test  
##   
## data: sentiment by stars  
## Kruskal-Wallis chi-squared = 38552, df = 1, p-value < 2.2e-16

posthoc.kruskal.nemenyi.test(sentiment~stars, data = bing\_cat)

## Warning in posthoc.kruskal.nemenyi.test.default(c(30, 30, 30, 30, 30, 30, : Ties  
## are present, p-values are not corrected.

##   
## Pairwise comparisons using Tukey and Kramer (Nemenyi) test   
## with Tukey-Dist approximation for independent samples   
##   
## data: sentiment by stars   
##   
## 3.5 orless  
## 4 or more <2e-16   
##   
## P value adjustment method: none

The sentiment rating varies between all star ratings.

library(wordcloud)  
library(reshape2)  
  
n\_star <- c("3.5 orless","4 or more")   
  
lapply(n\_star, function(x){tidy\_reviews\_kids %>%   
 select(-stars) %>%   
 left\_join(bing\_lexicon) %>%   
 left\_join(ff\_assoc\_trim[,1:2]) %>%   
 count(stars, word, sentiment, sort =T) %>%  
 filter(stars==x) %>%   
 filter(!is.na(sentiment)) %>%   
 acast(word~sentiment, value.var="n", fill=0) %>%   
 comparison.cloud(colors=c("dodgerblue4", "orange2"), match.colors = T, title.bg.colors = "snow", max.words=50)})



## [[1]]  
## NULL  
##   
## [[2]]  
## NULL

The type of negative words varies by stars. At the lower star ratings, words like “rude”, “nasty”, and “horrible” appear in the reviews. At higher star ratings, words like “disappoint,”unfortunate" and “miss” appear.

For the positive words, at the lower star rating, words like “like”, “work”, and “smile” appear. At the higher star ratings, “love”, “great” and “friendly” appear.

# Create a restaurant classification model

Create a model to classify restaurants using the attributes identified through the association rules analysis and the aggregated review sentiment ratings.

Create a dataframe with the required attributes –>stars (converted to 4 or more, 3.5 or less) –>parking on the street –>noise = 2 –>ambience.casual –>sentiment

stars\_df <- sentiment\_by\_word %>% filter(method=="Bing") %>% select(-c(stars)) %>%   
 ungroup() %>%   
 left\_join(ff\_assoc\_trim[,c(1,2,3,6,9, 12, 13, 15, 19, 23, 33)], by = "ID") %>%   
 select(-c(ID, method))  
  
stars\_df

## # A tibble: 22,319 x 11  
## sentiment stars BikeParking NoiseLevel HasTV Ambience.casual Ambience.classy  
## <dbl> <fct> <fct> <fct> <fct> <fct> <fct>   
## 1 30 3.5 ~ 0 '2' 1 1 0   
## 2 1 4 or~ 1 '1' 1 1 0   
## 3 24 4 or~ 1 '2' 0 0 0   
## 4 -2 3.5 ~ <NA> '2' <NA> <NA> <NA>   
## 5 2 3.5 ~ 1 '2' 1 0 0   
## 6 21 3.5 ~ 1 '2' 0 1 0   
## 7 13 3.5 ~ 0 '3' 1 1 0   
## 8 -3 3.5 ~ 0 '3' 1 0 0   
## 9 2 3.5 ~ 1 '2' 0 1 0   
## 10 17 3.5 ~ 1 '2' 1 1 0   
## # ... with 22,309 more rows, and 4 more variables: Ambience.hipster <fct>,  
## # Ambience.trendy <fct>, BusinessParking.street <fct>, Alcoholfull\_bar <dbl>

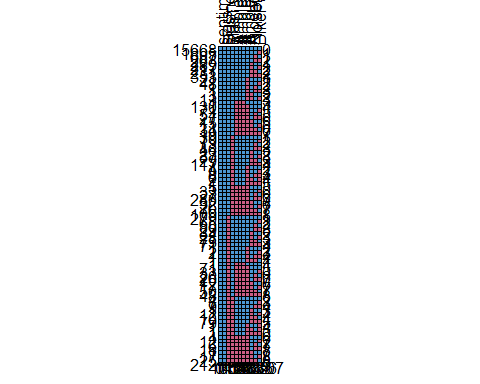
Check the data for completeness

sum(complete.cases(stars\_df))

## [1] 15668

Check the pattern of missing values

md.pattern(stars\_df, rotate.names = T)



## sentiment stars BusinessParking.street HasTV Ambience.casual  
## 15668 1 1 1 1 1  
## 1805 1 1 1 1 1  
## 687 1 1 1 1 1  
## 263 1 1 1 1 1  
## 289 1 1 1 1 1  
## 217 1 1 1 1 1  
## 231 1 1 1 1 1  
## 353 1 1 1 1 1  
## 21 1 1 1 1 1  
## 48 1 1 1 1 1  
## 1 1 1 1 1 1  
## 1 1 1 1 1 1  
## 3 1 1 1 1 1  
## 13 1 1 1 1 1  
## 1 1 1 1 1 0  
## 130 1 1 1 1 0  
## 71 1 1 1 1 0  
## 54 1 1 1 1 0  
## 27 1 1 1 1 0  
## 15 1 1 1 1 0  
## 11 1 1 1 1 0  
## 34 1 1 1 1 0  
## 19 1 1 1 1 0  
## 39 1 1 1 0 1  
## 79 1 1 1 0 1  
## 13 1 1 1 0 1  
## 46 1 1 1 0 1  
## 23 1 1 1 0 1  
## 37 1 1 1 0 1  
## 29 1 1 1 0 1  
## 117 1 1 1 0 1  
## 4 1 1 1 0 1  
## 9 1 1 1 0 1  
## 6 1 1 1 0 1  
## 2 1 1 1 0 1  
## 1 1 1 1 0 1  
## 5 1 1 1 0 0  
## 33 1 1 1 0 0  
## 27 1 1 1 0 0  
## 280 1 1 1 0 0  
## 32 1 1 1 0 0  
## 26 1 1 1 0 0  
## 76 1 1 1 0 0  
## 109 1 1 1 0 0  
## 279 1 1 0 1 1  
## 65 1 1 0 1 1  
## 80 1 1 0 1 1  
## 29 1 1 0 1 1  
## 34 1 1 0 1 1  
## 28 1 1 0 1 1  
## 75 1 1 0 1 1  
## 71 1 1 0 1 1  
## 1 1 1 0 1 1  
## 2 1 1 0 1 1  
## 1 1 1 0 1 1  
## 1 1 1 0 1 1  
## 1 1 1 0 1 0  
## 71 1 1 0 1 0  
## 33 1 1 0 1 0  
## 26 1 1 0 1 0  
## 20 1 1 0 1 0  
## 12 1 1 0 1 0  
## 17 1 1 0 1 0  
## 50 1 1 0 1 0  
## 48 1 1 0 1 0  
## 4 1 1 0 0 1  
## 7 1 1 0 0 1  
## 9 1 1 0 0 1  
## 1 1 1 0 0 1  
## 12 1 1 0 0 1  
## 10 1 1 0 0 1  
## 79 1 1 0 0 1  
## 1 1 1 0 0 1  
## 1 1 1 0 0 1  
## 1 1 1 0 0 1  
## 4 1 1 0 0 0  
## 12 1 1 0 0 0  
## 6 1 1 0 0 0  
## 18 1 1 0 0 0  
## 1 1 1 0 0 0  
## 17 1 1 0 0 0  
## 25 1 1 0 0 0  
## 212 1 1 0 0 0  
## 0 0 1364 1413 1523  
## Ambience.classy Ambience.trendy Ambience.hipster Alcoholfull\_bar  
## 15668 1 1 1 1  
## 1805 1 1 1 1  
## 687 1 1 1 1  
## 263 1 1 1 1  
## 289 1 1 1 0  
## 217 1 1 1 0  
## 231 1 1 1 0  
## 353 1 1 1 0  
## 21 1 1 0 1  
## 48 1 1 0 1  
## 1 1 1 0 1  
## 1 1 1 0 1  
## 3 1 1 0 0  
## 13 1 1 0 0  
## 1 0 0 1 1  
## 130 0 0 0 1  
## 71 0 0 0 1  
## 54 0 0 0 1  
## 27 0 0 0 1  
## 15 0 0 0 0  
## 11 0 0 0 0  
## 34 0 0 0 0  
## 19 0 0 0 0  
## 39 1 1 1 1  
## 79 1 1 1 1  
## 13 1 1 1 1  
## 46 1 1 1 1  
## 23 1 1 1 0  
## 37 1 1 1 0  
## 29 1 1 1 0  
## 117 1 1 1 0  
## 4 1 1 0 1  
## 9 1 1 0 1  
## 6 1 1 0 1  
## 2 1 1 0 0  
## 1 1 1 0 0  
## 5 0 0 0 1  
## 33 0 0 0 1  
## 27 0 0 0 1  
## 280 0 0 0 1  
## 32 0 0 0 0  
## 26 0 0 0 0  
## 76 0 0 0 0  
## 109 0 0 0 0  
## 279 1 1 1 1  
## 65 1 1 1 1  
## 80 1 1 1 1  
## 29 1 1 1 1  
## 34 1 1 1 0  
## 28 1 1 1 0  
## 75 1 1 1 0  
## 71 1 1 1 0  
## 1 1 1 0 1  
## 2 1 1 0 1  
## 1 1 1 0 1  
## 1 1 1 0 0  
## 1 0 0 1 1  
## 71 0 0 0 1  
## 33 0 0 0 1  
## 26 0 0 0 1  
## 20 0 0 0 1  
## 12 0 0 0 0  
## 17 0 0 0 0  
## 50 0 0 0 0  
## 48 0 0 0 0  
## 4 1 1 1 1  
## 7 1 1 1 1  
## 9 1 1 1 1  
## 1 1 1 1 0  
## 12 1 1 1 0  
## 10 1 1 1 0  
## 79 1 1 1 0  
## 1 1 1 0 1  
## 1 1 1 0 0  
## 1 1 1 0 0  
## 4 0 0 0 1  
## 12 0 0 0 1  
## 6 0 0 0 1  
## 18 0 0 0 1  
## 1 0 0 0 0  
## 17 0 0 0 0  
## 25 0 0 0 0  
## 212 0 0 0 0  
## 1523 1523 1638 2332  
## NoiseLevel BikeParking   
## 15668 1 1 0  
## 1805 1 0 1  
## 687 0 1 1  
## 263 0 0 2  
## 289 1 1 1  
## 217 1 0 2  
## 231 0 1 2  
## 353 0 0 3  
## 21 1 1 1  
## 48 1 0 2  
## 1 0 1 2  
## 1 0 0 3  
## 3 1 1 2  
## 13 1 0 3  
## 1 1 1 3  
## 130 1 1 4  
## 71 1 0 5  
## 54 0 1 5  
## 27 0 0 6  
## 15 1 1 5  
## 11 1 0 6  
## 34 0 1 6  
## 19 0 0 7  
## 39 1 1 1  
## 79 1 0 2  
## 13 0 1 2  
## 46 0 0 3  
## 23 1 1 2  
## 37 1 0 3  
## 29 0 1 3  
## 117 0 0 4  
## 4 1 1 2  
## 9 1 0 3  
## 6 0 0 4  
## 2 1 0 4  
## 1 0 0 5  
## 5 1 1 5  
## 33 1 0 6  
## 27 0 1 6  
## 280 0 0 7  
## 32 1 1 6  
## 26 1 0 7  
## 76 0 1 7  
## 109 0 0 8  
## 279 1 1 1  
## 65 1 0 2  
## 80 0 1 2  
## 29 0 0 3  
## 34 1 1 2  
## 28 1 0 3  
## 75 0 1 3  
## 71 0 0 4  
## 1 1 1 2  
## 2 1 0 3  
## 1 0 0 4  
## 1 0 1 4  
## 1 1 1 4  
## 71 1 1 5  
## 33 1 0 6  
## 26 0 1 6  
## 20 0 0 7  
## 12 1 1 6  
## 17 1 0 7  
## 50 0 1 7  
## 48 0 0 8  
## 4 1 1 2  
## 7 1 0 3  
## 9 0 0 4  
## 1 1 1 3  
## 12 1 0 4  
## 10 0 1 4  
## 79 0 0 5  
## 1 1 0 4  
## 1 1 0 5  
## 1 0 0 6  
## 4 1 1 6  
## 12 1 0 7  
## 6 0 1 7  
## 18 0 0 8  
## 1 1 1 7  
## 17 1 0 8  
## 25 0 1 8  
## 212 0 0 9  
## 3135 4256 18707

Impute missing values using k-nearest neighbours

stars\_imputed <- kNN(stars\_df, variable= c("Ambience.casual", "NoiseLevel", "HasTV", "BikeParking", "Ambience.classy", "Ambience.hipster", "Ambience.trendy", "BusinessParking.street", "Alcoholfull\_bar"), k=5)  
  
  
stars\_imputed <- stars\_imputed %>% select(-ends\_with("\_imp"))  
  
#compare the attributes before and after imputation  
summary(stars\_df)

## sentiment stars BikeParking NoiseLevel HasTV   
## Min. :-51.000 3.5 orless:14126 0 : 4166 '1' : 4237 0 : 5040   
## 1st Qu.: 1.000 4 or more : 8193 1 :13897 '2' :13338 1 :15866   
## Median : 5.000 NA's: 4256 '3' : 1202 NA's: 1413   
## Mean : 8.468 '4' : 407   
## 3rd Qu.: 13.000 NA's: 3135   
## Max. :175.000   
##   
## Ambience.casual Ambience.classy Ambience.hipster Ambience.trendy  
## 0 : 8737 0 :19139 0 :20351 0 :20091   
## 1 :12059 1 : 1657 1 : 330 1 : 705   
## NA's: 1523 NA's: 1523 NA's: 1638 NA's: 1523   
##   
##   
##   
##   
## BusinessParking.street Alcoholfull\_bar   
## 0 :16914 Min. :0.0000   
## 1 : 4041 1st Qu.:0.0000   
## NA's: 1364 Median :0.0000   
## Mean :0.3098   
## 3rd Qu.:1.0000   
## Max. :1.0000   
## NA's :2332

summary(stars\_imputed)

## sentiment stars BikeParking NoiseLevel HasTV   
## Min. :-51.000 3.5 orless:14126 0: 5022 '1': 4969 0: 5278   
## 1st Qu.: 1.000 4 or more : 8193 1:17297 '2':15679 1:17041   
## Median : 5.000 '3': 1253   
## Mean : 8.468 '4': 418   
## 3rd Qu.: 13.000   
## Max. :175.000   
## Ambience.casual Ambience.classy Ambience.hipster Ambience.trendy  
## 0:10168 0:20652 0:21989 0:21614   
## 1:12151 1: 1667 1: 330 1: 705   
##   
##   
##   
##   
## BusinessParking.street Alcoholfull\_bar   
## 0:18202 Min. :0.0000   
## 1: 4117 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.2927   
## 3rd Qu.:1.0000   
## Max. :1.0000

*Create a Logistic Regression model to classify restaurants* assumptions - 1) linearity - check the sentiment variable as it is continuous

library(car)

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

## The following object is masked from 'package:ROCit':  
##   
## logit

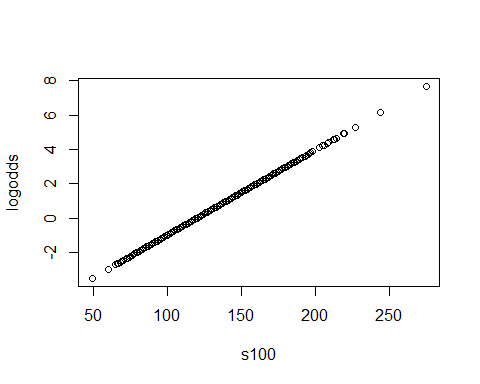
## The following object is masked from 'package:purrr':  
##   
## some

## The following object is masked from 'package:arules':  
##   
## recode

s100 <- stars\_imputed$sentiment+100  
  
lreg <- glm(stars\_imputed$stars~s100, family=binomial(link = "logit"))  
logodds <- lreg$linear.predictors  
boxTidwell(logodds~s100, data = stars\_imputed)

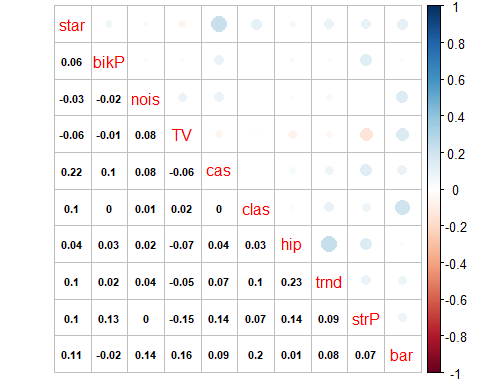
## MLE of lambda Score Statistic (z) Pr(>|z|)  
## 1 0.2354 0.8139  
##   
## iterations = 0

plot(logodds~s100)

 There is a linear relationship between the logit of the dependent variable and the continuous, independent variable (sentiment). Note: sentiment was transformed by addion 100 to all values for this test, as the box tidewell test must be done with positive numbers.

1. multicolinearity - check with a correlation matrix

S <- stars\_imputed[,-c(1,2)]   
#features in binary format (0=absent; 1=present)  
S[1:ncol(S)] <- lapply(S[1:ncol(S)], as.character) #convert factors to characters  
S$NoiseLevel <- gsub("'","",S$NoiseLevel)  
S[1:ncol(S)] <- lapply(S[1:ncol(S)], as.numeric) #then convert to numeric (as you can't convert values directly from factor to numeric)  
S <-cbind(stars\_imputed[,1],S)  
  
names(S) <- c("star", "bikP","nois","TV","cas","clas","hip","trnd","strP","bar")  
  
M <- cor(S, method = "spearman", use = "pairwise.complete.obs")  
corrplot.mixed(M, lower.col = "black", number.cex = .7)

 None of the independent variables are highly correlated with one another.

Partition the data

set.seed(3939)  
train\_index <- createDataPartition(stars\_imputed$stars, p=0.75, list=F)  
stars\_train <- stars\_imputed[train\_index,]  
stars\_test <- stars\_imputed[-train\_index,]

Train a logistic regression model

#train the model using 10-fold cross validation  
stars\_lr\_model <- train(form = stars~., data = stars\_train, trControl = trainControl(method = "cv", number=10), method = "glm", family= "binomial")  
  
stars\_lr\_model #view the model and check performance of cross-validation accuracy

## Generalized Linear Model   
##   
## 16740 samples  
## 10 predictor  
## 2 classes: '3.5 orless', '4 or more'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 15065, 15066, 15067, 15066, 15066, 15067, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.6909804 0.2656956

stars\_lr\_model$finalModel #view the final model weights, etc.

##   
## Call: NULL  
##   
## Coefficients:  
## (Intercept) sentiment BikeParking1   
## -0.58086 0.05328 0.17291   
## `NoiseLevel'2'` `NoiseLevel'3'` `NoiseLevel'4'`   
## -0.40917 -1.23501 -1.56094   
## HasTV1 Ambience.casual1 Ambience.classy1   
## -0.20061 -0.06868 0.21752   
## Ambience.hipster1 Ambience.trendy1 BusinessParking.street1   
## 0.98046 0.34038 0.60748   
## Alcoholfull\_bar   
## -0.63491   
##   
## Degrees of Freedom: 16739 Total (i.e. Null); 16727 Residual  
## Null Deviance: 22010   
## Residual Deviance: 19520 AIC: 19540

summary(stars\_lr\_model)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.1445 -0.8930 -0.6549 1.1348 2.5904   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.580861 0.057955 -10.023 < 2e-16 \*\*\*  
## sentiment 0.053276 0.001591 33.497 < 2e-16 \*\*\*  
## BikeParking1 0.172913 0.043081 4.014 5.98e-05 \*\*\*  
## `NoiseLevel'2'` -0.409167 0.041160 -9.941 < 2e-16 \*\*\*  
## `NoiseLevel'3'` -1.235009 0.097798 -12.628 < 2e-16 \*\*\*  
## `NoiseLevel'4'` -1.560944 0.180693 -8.639 < 2e-16 \*\*\*  
## HasTV1 -0.200611 0.041021 -4.890 1.01e-06 \*\*\*  
## Ambience.casual1 -0.068682 0.036629 -1.875 0.06078 .   
## Ambience.classy1 0.217523 0.067870 3.205 0.00135 \*\*   
## Ambience.hipster1 0.980463 0.152005 6.450 1.12e-10 \*\*\*  
## Ambience.trendy1 0.340377 0.099942 3.406 0.00066 \*\*\*  
## BusinessParking.street1 0.607478 0.044887 13.534 < 2e-16 \*\*\*  
## Alcoholfull\_bar -0.634911 0.042258 -15.025 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 22009 on 16739 degrees of freedom  
## Residual deviance: 19517 on 16727 degrees of freedom  
## AIC: 19543  
##   
## Number of Fisher Scoring iterations: 4

exp(stars\_lr\_model$finalModel$coefficients)

## (Intercept) sentiment BikeParking1   
## 0.5594162 1.0547212 1.1887627   
## `NoiseLevel'2'` `NoiseLevel'3'` `NoiseLevel'4'`   
## 0.6642030 0.2908321 0.2099378   
## HasTV1 Ambience.casual1 Ambience.classy1   
## 0.8182306 0.9336233 1.2429946   
## Ambience.hipster1 Ambience.trendy1 BusinessParking.street1   
## 2.6656910 1.4054774 1.8357951   
## Alcoholfull\_bar   
## 0.5299826

Make predictions

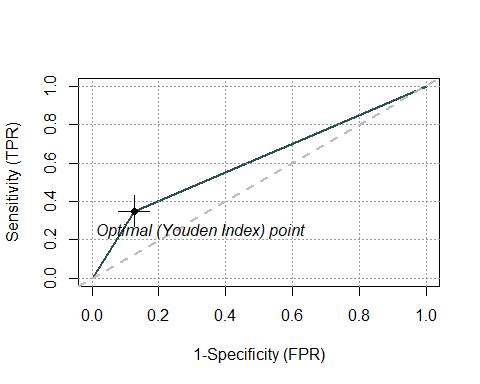
stars\_lr\_predict <- predict(stars\_lr\_model, newdata = stars\_test) #use the model to predict star ratings for test data  
  
confusionMatrix(stars\_lr\_predict, stars\_test$stars, positive = "4 or more") #create a confusion matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 3.5 orless 4 or more  
## 3.5 orless 3087 1330  
## 4 or more 444 718  
##   
## Accuracy : 0.682   
## 95% CI : (0.6696, 0.6942)  
## No Information Rate : 0.6329   
## P-Value [Acc > NIR] : 8.272e-15   
##   
## Kappa : 0.2473   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.3506   
## Specificity : 0.8743   
## Pos Pred Value : 0.6179   
## Neg Pred Value : 0.6989   
## Prevalence : 0.3671   
## Detection Rate : 0.1287   
## Detection Prevalence : 0.2083   
## Balanced Accuracy : 0.6124   
##   
## 'Positive' Class : 4 or more   
##

exp(summary(stars\_lr\_model)$coefficients[,1])#exponentiate the coefficients and interpret as odds ratios

## (Intercept) sentiment BikeParking1   
## 0.5594162 1.0547212 1.1887627   
## `NoiseLevel'2'` `NoiseLevel'3'` `NoiseLevel'4'`   
## 0.6642030 0.2908321 0.2099378   
## HasTV1 Ambience.casual1 Ambience.classy1   
## 0.8182306 0.9336233 1.2429946   
## Ambience.hipster1 Ambience.trendy1 BusinessParking.street1   
## 2.6656910 1.4054774 1.8357951   
## Alcoholfull\_bar   
## 0.5299826

lr\_ROC <- rocit(score = as.numeric(stars\_lr\_predict), class=as.numeric(stars\_test$stars))  
  
plot(lr\_ROC, legend=F)



*Create a C5.0 Decision Tree model* Train the model

stars\_tree\_model <- train(stars~., data = stars\_train, method = "C5.0",trControl = trainControl(method = 'cv', number=10), tuneGrid=expand.grid(trials = c(1, 2, 3),   
model = c("tree"), winnow = c(FALSE, FALSE, FALSE)))  
  
stars\_tree\_model

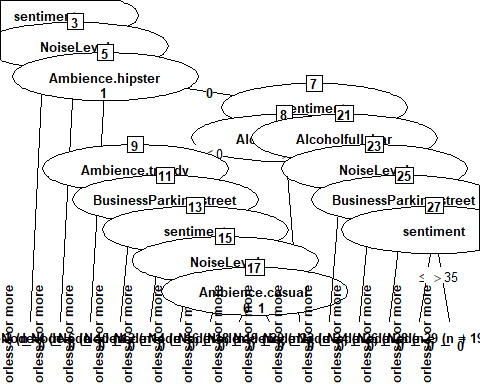
## C5.0   
##   
## 16740 samples  
## 10 predictor  
## 2 classes: '3.5 orless', '4 or more'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 15066, 15066, 15065, 15066, 15066, 15066, ...   
## Resampling results across tuning parameters:  
##   
## trials Accuracy Kappa   
## 1 0.6913377 0.2866165  
## 2 0.6938481 0.2922892  
## 3 0.6921739 0.2852753  
##   
## Tuning parameter 'model' was held constant at a value of tree  
## Tuning  
## parameter 'winnow' was held constant at a value of FALSE  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were trials = 2, model = tree and winnow  
## = FALSE.

Make Predictions

stars\_tree\_predict <- predict(stars\_tree\_model, newdata = stars\_test) #use the model to predict star ratings for test data  
  
confusionMatrix(stars\_tree\_predict, stars\_test$stars, positive = "4 or more") #create a confusion matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 3.5 orless 4 or more  
## 3.5 orless 3058 1243  
## 4 or more 473 805  
##   
## Accuracy : 0.6924   
## 95% CI : (0.6801, 0.7045)  
## No Information Rate : 0.6329   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.2813   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.3931   
## Specificity : 0.8660   
## Pos Pred Value : 0.6299   
## Neg Pred Value : 0.7110   
## Prevalence : 0.3671   
## Detection Rate : 0.1443   
## Detection Prevalence : 0.2291   
## Balanced Accuracy : 0.6296   
##   
## 'Positive' Class : 4 or more   
##

library(C50) #use the C50 package to produce a decision tree visual  
c5\_model <- C50::C5.0.default(x=stars\_train[,-2], y=stars\_train$stars, trials = 1, rules = F, control =C5.0Control(winnow=stars\_tree\_model$bestTune$winnow, minCases = 100))  
  
plot(c5\_model, gp = gpar(fontsize=10, fontface = "bold"))

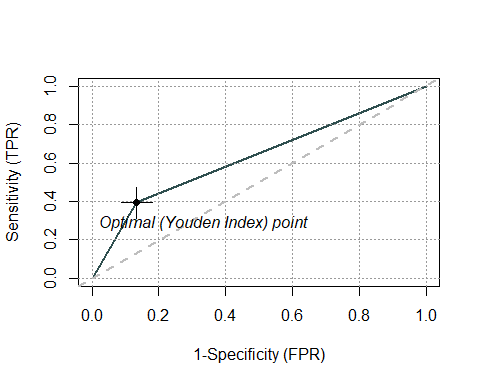


Make Predictions - version2

c5\_predict <- predict(c5\_model, newdata = stars\_test) #use the model to predict star ratings for test data  
  
confusionMatrix(c5\_predict, stars\_test$stars, positive = "4 or more") #create a confusion matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 3.5 orless 4 or more  
## 3.5 orless 2983 1169  
## 4 or more 548 879  
##   
## Accuracy : 0.6922   
## 95% CI : (0.6799, 0.7043)  
## No Information Rate : 0.6329   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.2926   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.4292   
## Specificity : 0.8448   
## Pos Pred Value : 0.6160   
## Neg Pred Value : 0.7184   
## Prevalence : 0.3671   
## Detection Rate : 0.1576   
## Detection Prevalence : 0.2558   
## Balanced Accuracy : 0.6370   
##   
## 'Positive' Class : 4 or more   
##

dt\_ROC <- rocit(score = as.numeric(stars\_tree\_predict), class=as.numeric(stars\_test$stars))  
  
plot(dt\_ROC, legend=F)



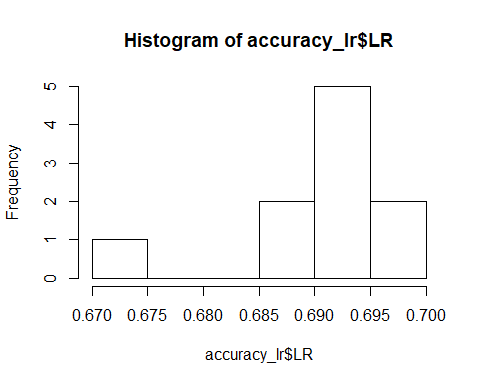
Compare the two models

accuracy\_lr <- stars\_lr\_model$resample %>% arrange(Resample) %>% rename(LR = Accuracy) %>% select(c(1)) #accuracy of the 10 cross folds for LR  
accuracy\_dt <- stars\_tree\_model$resample %>% arrange(Resample) %>% rename(DT = Accuracy) %>% select(c(1))#accuracy of the 10 cross folds for DR  
accuracy\_df <- cbind(accuracy\_lr,accuracy\_dt) #dataframe combining the accuracy of training values of both models  
accuracy\_df

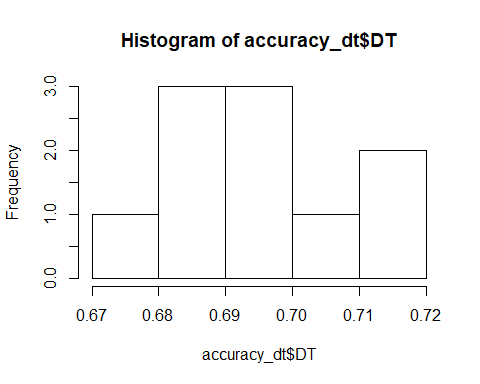
## LR DT  
## 1 0.6871642 0.6917563  
## 2 0.6947431 0.6959379  
## 3 0.6909743 0.6752239  
## 4 0.6935484 0.7108722  
## 5 0.6971326 0.6821983  
## 6 0.6969516 0.7162485  
## 7 0.6883582 0.6887694  
## 8 0.6732378 0.6927675  
## 9 0.6929510 0.7019116  
## 10 0.6947431 0.6827957

Comparison of the two models using wilcoxon rank sum test. A non-parametric test was used because the accuracy of the folds for both models was right skewed or flat.

hist(accuracy\_lr$LR)



hist(accuracy\_dt$DT)



wilcox.test(accuracy\_df$LR, accuracy\_df$DT, paired = F)

## Warning in wilcox.test.default(accuracy\_df$LR, accuracy\_df$DT, paired = F):  
## cannot compute exact p-value with ties

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: accuracy\_df$LR and accuracy\_df$DT  
## W = 48, p-value = 0.9097  
## alternative hypothesis: true location shift is not equal to 0