CS 109B: Midterm Exam 1

Feb 23, 2017

Danqing Wang

The set of questions below address the task of predicting the merit of a restaurant on Yelp. Each restaurant is described by a set of business attributes, and is accompanied by a set of text reviews from customers. For the purpose of the problems below, the average rating (originally on a scale from 0 to 5) was converted into a binary variable depending on whether the average was above 3.5, in which case it is considered "good" (labeled 1), or below 3.5 in which case it is considered "bad" (labeled 0). The overall goal is to predict these binary ratings from the information provided.

The data are split into a training and test set and are in the files `dataset_1_train.txt` and `dataset_1_test.txt` respectively. The first column contains the rating for the restaurant (0 or 1), columns 2-21 contain the business attributes, and columns 22-121 contain text features extracted from the customer reviews for the restaurant. The details about the business attributes are provided in the file `dataset_1_description.txt`.

We use the bag-of-words encoding to generate the text features, where the set of reviews for a restaurant are represented by a vector of word counts. More specifically, we construct a dictionary of 100 frequent words in the customer reviews, and include 100 text features for each restaurant: the i-th feature contains the number of times the dictionary word i occurs in customer reviews for the restaurant. For example, a text feature 'fantastic' with value 18 for a restaurant indicates that the word 'fantastic' was used a total of 18 times in customer reviews for that restaurant.

Problem 1 [20 points]

Does the location of a restaurant relate to its rating? Construct a compelling visualization to address this question, and write a brief (under 300 words) summary that clearly explains your analysis and conclusions to someone without a data science background.

Answer

In this section, we load and explore the data sets before writing our report. Tableau is also used in assisting analysis.

Data exploration

```
# load data
train <- read.csv('./cs109b-midterm1-data/dataset_1_train.txt')
test <- read.csv('./cs109b-midterm1-data/dataset_1_test.txt')

str(train)
```

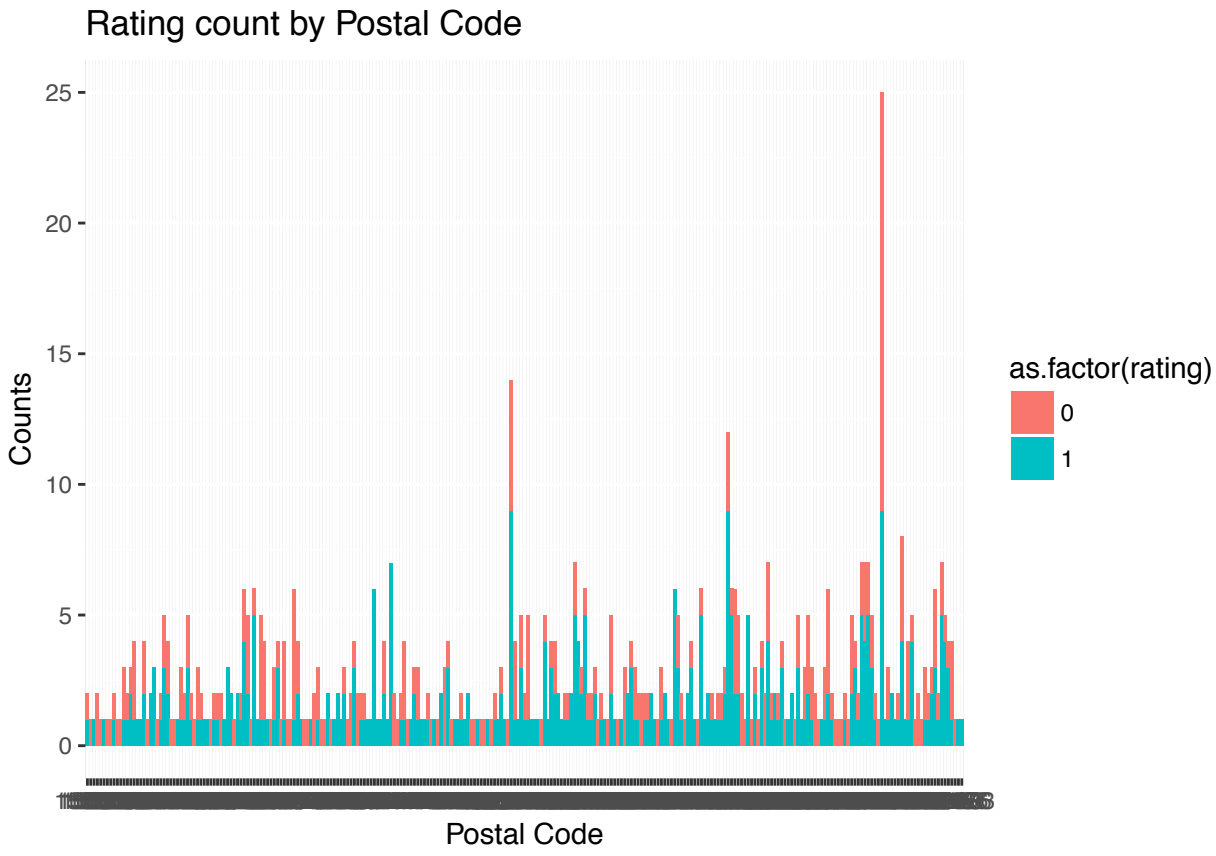
```
## 'data.frame':    670 obs. of  121 variables:
##  $ rating                    : int  0 1 1 0 0 1 0 0 0 0 ...
##  $ name                      : Factor w/ 546 levels "1900 Mexican Grill",..: 34 102 518 233 481 364 1
##  $ postal_code               : int  15017 44077 85281 89014 53715 89146 89128 44125 28209 89052 ...
##  $ city                      : Factor w/ 108 levels "Avon","Avondale",..: 5 77 97 40 54 50 50 31 14 4
##  $ state                     : Factor w/ 8 levels "AZ","IL","NC",..: 6 5 1 4 8 4 4 5 3 4 ...
##  $ latitude                  : num  40.4 41.8 33.4 36.1 43.1 ...
```

```
##  $ longitude                 : num  -80.1 -81.2 -111.9 -115 -89.4 ...
##  $ review_count              : int  11 11 31 36 39 16 21 5 17 111 ...
##  $ cuisine                   : Factor w/ 11 levels "Bakeries","Bars",..: 7 1 3 7 11 10 10 1 6 2 ...
##  $ WheelchairAccessible      : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ WiFi                      : Factor w/ 3 levels "free","no","paid": 1 1 1 2 2 1 1 2 1 2 ...
##  $ BusinessAcceptsCreditCards: Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alcohol                   : Factor w/ 3 levels "beer_and_wine",..: 3 3 3 3 2 3 3 3 3 1 ...
##  $ NoiseLevel                : Factor w/ 4 levels "average","loud",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ RestaurantsPriceRange2    : int  1 1 2 1 2 1 1 1 1 1 ...
##  $ RestaurantsAttire         : Factor w/ 3 levels "casual","dressy",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Smoking                   : Factor w/ 3 levels "no","outdoor",..: 1 2 3 1 1 2 2 2 2 2 ...
##  $ RestaurantsReservations   : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 1 1 1 ...
##  $ OutdoorSeating            : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 1 1 1 2 ...
##  $ GoodForKids               : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 1 2 2 ...
##  $ avg_word_count            : num  69 66.4 82.9 97 116.1 ...
##  $ wine                      : int  0 0 0 0 2 0 0 0 0 0 ...
##  $ chocolate                 : int  0 10 0 0 0 0 0 0 0 0 ...
##  $ sum                       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ dim                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ perfect                   : int  0 0 1 0 3 0 2 1 0 2 ...
##  $ sweet                     : int  0 3 1 1 0 0 0 0 1 0 ...
##  $ chinese                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ bread                     : int  0 0 0 0 1 0 15 0 0 0 ...
##  $ breakfast                 : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ happy                     : int  1 0 2 0 10 0 0 0 0 6 ...
##  $ loved                     : int  0 2 2 0 0 0 1 0 0 2 ...
##  $ selection                 : int  0 0 0 0 6 0 0 0 0 0 ...
##  $ manager                   : int  0 0 0 11 0 1 7 0 1 3 ...
##  $ spot                      : int  1 0 0 0 0 1 0 0 1 3 ...
##  $ corn                      : int  0 0 0 0 0 0 0 0 0 5 ...
##  $ dessert                   : int  0 0 3 0 0 0 0 0 0 2 ...
##  $ fantastic                 : int  1 1 1 0 0 0 0 0 0 1 ...
##  $ sushi                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ super                     : int  0 0 1 1 0 2 1 0 1 7 ...
##  $ bbq                       : int  0 0 0 1 2 0 1 0 0 0 ...
##  $ pastries                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ crust                     : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ brunch                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ taco                      : int  0 0 0 0 0 0 0 3 27 4 ...
##  $ perfectly                 : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ dish                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ asked                     : int  0 0 3 12 3 0 12 0 0 15 ...
##  $ wonderful                 : int  0 0 3 0 3 0 0 0 0 3 ...
##  $ phoenix                   : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ rice                      : int  0 0 0 0 0 0 0 0 0 23 ...
##  $ waitress                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ flavors                   : int  0 2 3 0 1 0 0 0 0 0 ...
##  $ mesa                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ bobby                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ highly                    : int  1 1 4 0 1 2 1 0 0 1 ...
##  $ enjoyed                   : int  0 2 4 0 1 0 0 0 0 0 ...
##  $ pho                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ local                     : int  1 0 0 0 4 0 0 0 0 0 ...
##  $ flay                      : int  0 0 0 0 0 0 0 0 0 0 ...
```

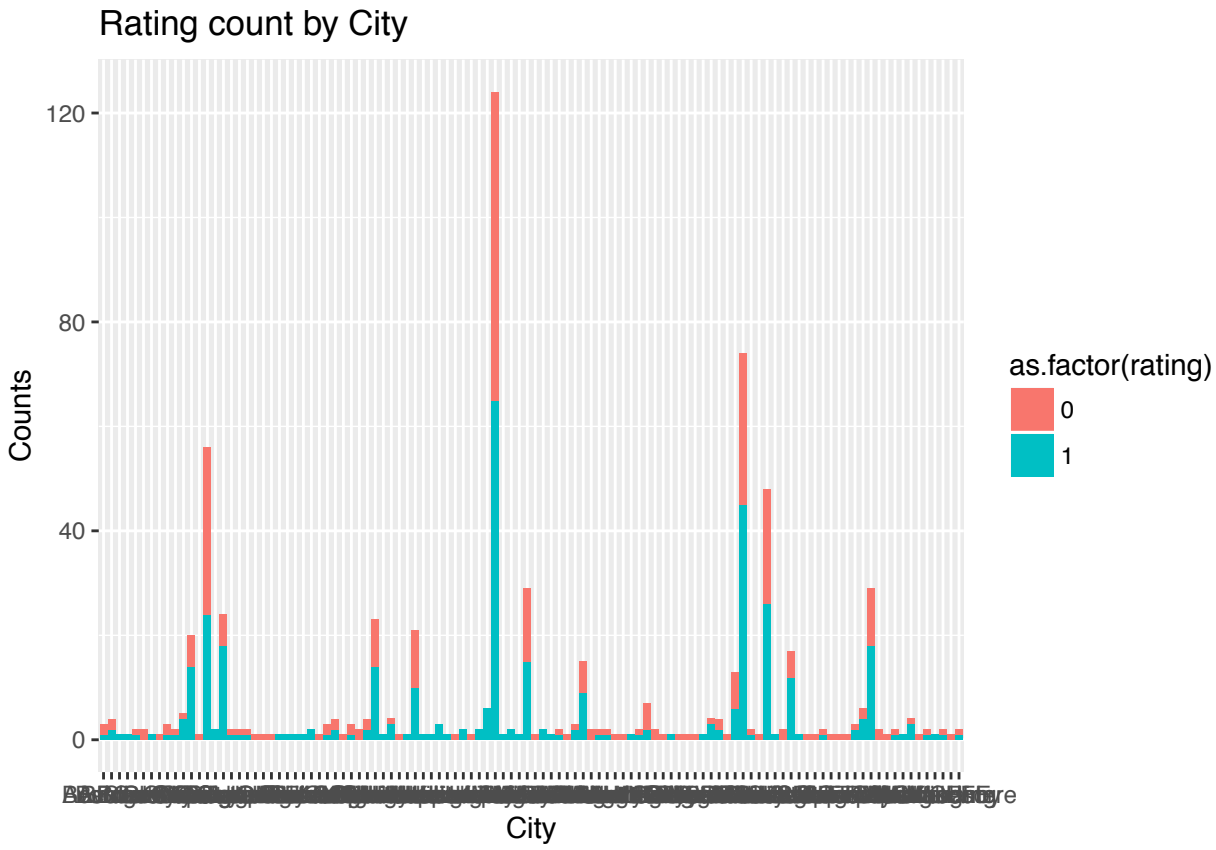
```
##  $ worst                    : int  1 0 0 10 0 0 2 2 1 3 ...
##  $ fried                    : int  0 1 0 6 1 0 0 0 0 0 ...
##  $ chef                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ croissant                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cool                     : int  0 0 2 0 8 2 0 0 0 2 ...
##  $ bakery                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ horrible                 : int  1 0 0 5 0 0 1 0 0 4 ...
##  $ bianco                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ shop                     : int  1 0 0 0 0 0 1 1 0 0 ...
##  $ bacon                    : int  0 0 0 1 4 0 2 0 0 0 ...
##  $ cream                    : int  0 2 0 0 1 0 0 0 1 12 ...
##  $ absolutely               : int  0 0 0 0 0 0 0 0 0 4 ...
##  $ salsa                    : int  0 0 0 0 1 0 0 0 0 11 ...
##  $ spicy                    : int  0 0 0 0 0 0 0 0 1 2 ...
##  $ hour                     : int  0 0 1 3 6 3 0 0 0 1 ...
##  $ yelp                     : int  0 0 5 1 1 0 0 0 3 1 ...
##  $ pittsburgh               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ enjoy                    : int  0 0 2 1 4 1 2 0 2 5 ...
##  $ macarons                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ prices                   : int  1 0 2 0 1 4 0 0 0 3 ...
##  $ italian                  : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ feel                     : int  0 0 2 1 7 0 1 0 0 1 ...
##  $ friends                  : int  0 1 6 0 5 2 0 0 0 1 ...
##  $ list                     : int  0 0 1 0 1 0 0 0 0 1 ...
##  $ primanti                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ingredients              : int  1 0 0 0 0 0 0 0 0 9 ...
##  $ tea                      : int  0 0 3 1 0 0 0 1 1 0 ...
##  $ roasted                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ light                    : int  0 0 0 1 0 0 0 0 0 1 ...
##  $ rude                     : int  0 0 0 1 0 0 2 0 0 0 ...
##  $ pork                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ wings                    : int  2 0 0 1 0 0 0 0 0 2 ...
##  $ lobster                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ribs                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ places                   : int  1 0 0 0 3 0 0 0 0 3 ...
##  $ chips                    : int  0 1 0 0 1 0 2 0 0 24 ...
##  $ roll                     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ terrible                 : int  0 0 0 0 4 0 1 0 0 3 ...
##  $ pasta                    : int  0 0 0 0 0 0 0 0 0 0 ...
##   [list output truncated]
```

Location-related attributes:

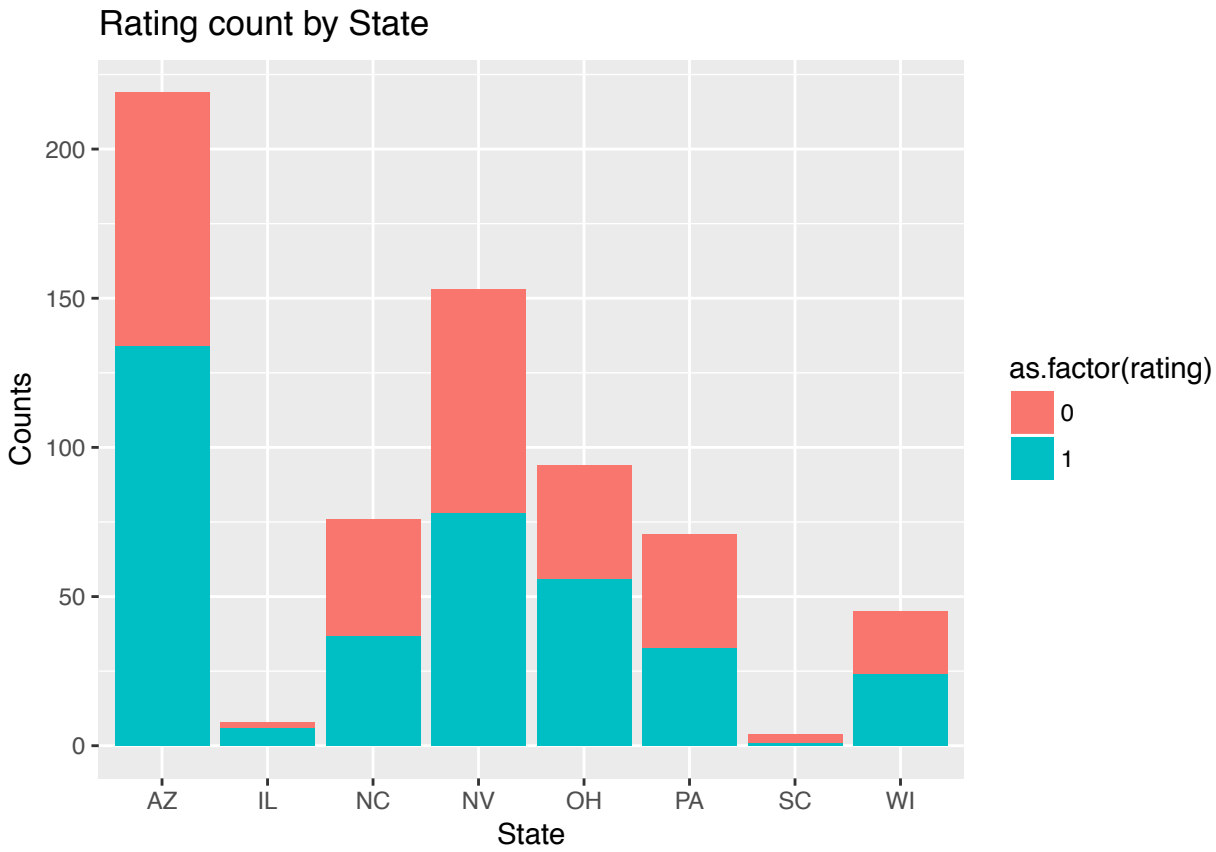
```
library(ggplot2)

ggplot(train, aes(x = as.factor(postal_code), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'Postal Code', y = 'Counts', title = 'Rating count by Postal Code')
```




```
ggplot(train, aes(x = as.factor(city), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'City', y = 'Counts', title = 'Rating count by City')
```




```
ggplot(train, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Counts', title = 'Rating count by State')
```




```
# ggsave(filename = 'state_bar.png', plot = last_plot(),
#   scale = 1, width = par("din")[1], height = par("din")[2], units = c("in",
#           "cm", "mm"), dpi = 300)
```

Plot of rating counts by postal code and city are not very telling, as there are too many postal codes and cities. Also, it is difficult to tell the proximity of different cities from one another from the plot above. Even though there are some postal codes / cities with only good or bad ratings, the number of data points are too few to make any conclusive statement.

A map analysis from Tableau shows us that most data points in each state are concentrated around one major city, such as Las Vegas in Nevada, Pheonix in Arizona, etc, and there is only one cluster in each state. Hence, it makes sense to examine if there is any relation between a restaurant's rating and which state it is located in.

We examine the percentage of good and bad ratings in each state:

```
# percentage good/bad rating in each state
states <- levels(train$state)

# initialization
percentage.good <- list()
percentage.bad <- list()
ratio.good.bad <- list() # ratio of number of good ratings / bad ratings

# calculate percentages
for(i in states){
  data <- train[train$state == i,] # extract entries from the ith state
  percentage.good[[i]] <- mean(data$rating == 1)
  percentage.bad[[i]] <- mean(data$rating == 0)
```

```
  ratio.good.bad[[i]] <- percentage.good[[i]] / percentage.bad[[i]]
}

cat('Percentage of restaurants rated good and bad according to state')
```

```
## Percentage of restaurants rated good and bad according to state
```

```
data.frame(state = states,
           percentage.good = as.numeric(percentage.good),
           percentage.bad = as.numeric(percentage.bad),
           ratio.good.bad = as.numeric(ratio.good.bad))
```

```
##   state percentage.good percentage.bad ratio.good.bad
## 1    AZ       0.6118721      0.3881279      1.5764706
## 2    IL       0.7500000      0.2500000      3.0000000
## 3    NC       0.4868421      0.5131579      0.9487179
## 4    NV       0.5098039      0.4901961      1.0400000
## 5    OH       0.5957447      0.4042553      1.4736842
## 6    PA       0.4647887      0.5352113      0.8684211
## 7    SC       0.2500000      0.7500000      0.3333333
## 8    WI       0.5333333      0.4666667      1.1428571
```

- In Arizona and Ohio, the percentage of restaurants rated good are about 1.5 times the percentage of those rated bad.
- In North Carolina, Nevada, Pennsylvania, and Wisconsin, the percentage of restaurants rated good and bad are about the same.
- The ratings in Illinois and South Carolina are skewed to mostly good and mostly bad, respectively. However, there are too few restaurant data points in either state to make any conclusion.

It would be helpful if we have additional infomation about the proximity of restaurants to places of interest, significant landmarks, town centers, highway, train or bus stations, etc, and this would give us a better idea of whether a restaurant's rating is related to its location other than the state it is in. However, since such information is not availabe, we base our analysis on states only.

Let's take a closer look at the different business attributes:

```
summary(train[1:21])
```

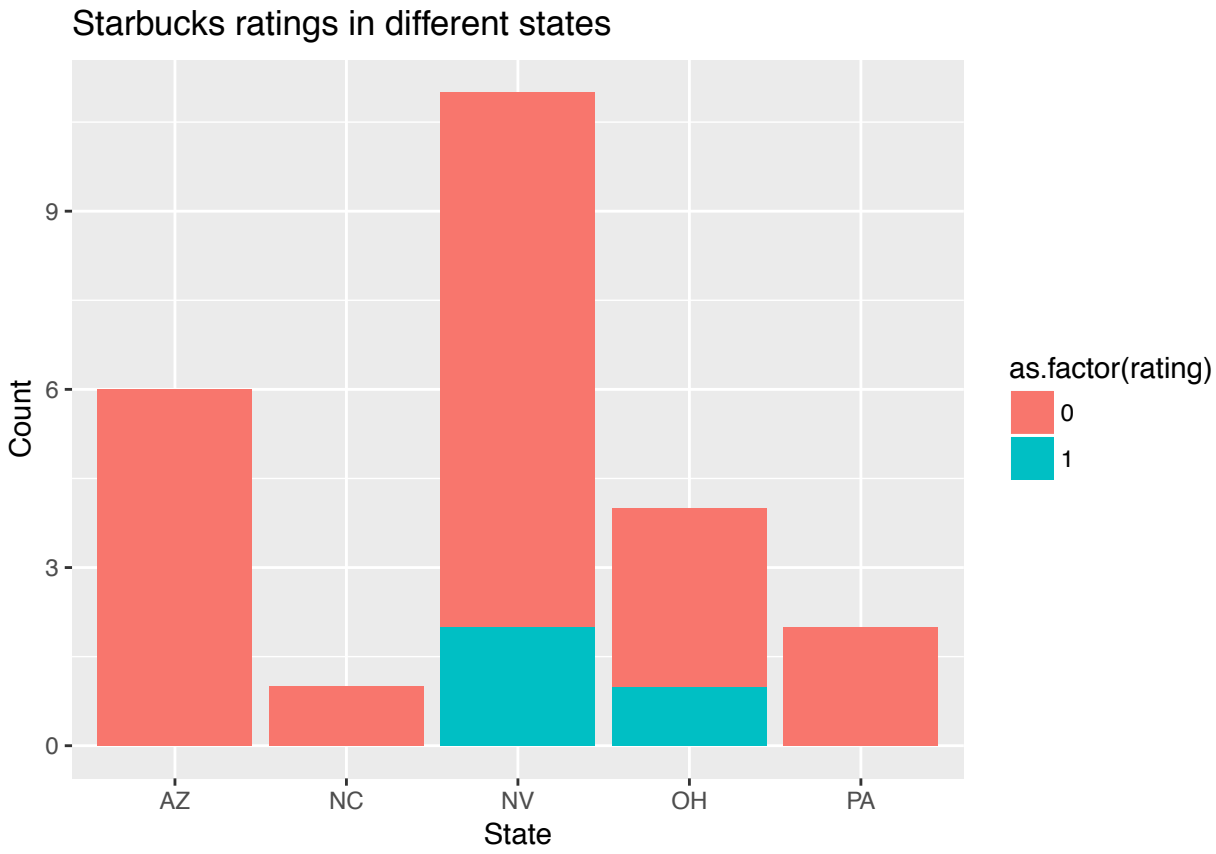
```
##      rating                                name      postal_code
##  Min.   :0.0000   Starbucks                  : 24   Min.   :15017
##  1st Qu.:0.0000   Taco Bell                  :  9   1st Qu.:44070
##  Median :1.0000   Five Guys Burgers and Fries:  7   Median :85019
##  Mean   :0.5507   Chipotle Mexican Grill     :  6   Mean   :63716
##  3rd Qu.:1.0000   Domino's Pizza             :  6   3rd Qu.:85378
##  Max.   :1.0000   Pizza Hut                  :  6   Max.   :89183
##                   (Other)                    :612
##          city         state        latitude       longitude
##  Las Vegas :124   AZ     :219   Min.   :33.23   Min.   :-115.35
##  Phoenix   : 74   NV     :153   1st Qu.:33.61   1st Qu.:-112.29
##  Charlotte : 56   OH     : 94   Median :36.06   Median :-111.88
##  Pittsburgh: 48   NC     : 76   Mean   :36.88   Mean   : -99.56
##  Madison   : 29   PA     : 71   3rd Qu.:40.45   3rd Qu.: -81.41
##  Tempe     : 29   WI     : 45   Max.   :43.25   Max.   : -79.67
##  (Other)   :310   (Other): 12
##   review_count             cuisine    WheelchairAccessible   WiFi
##  Min.   :   3.00   Bars        :136   no : 31              free:402
```

```
##  1st Qu.:  15.00   Coffee & Tea: 79   yes:639             no  :263
##  Median :  35.50   Fast Food   : 63                      paid:  5
##  Mean   :  81.75   Chinese     : 62
##  3rd Qu.:  96.75   Pizza       : 58
##  Max.   :1996.00   Sandwiches  : 54
##                    (Other)     :218
##  BusinessAcceptsCreditCards          Alcohol        NoiseLevel
##  no :  9                     beer_and_wine: 76   average  :515
##  yes:661                     full_bar     :223   loud     : 49
##                              none         :371   quiet    : 88
##                                                  very_loud: 18
##
##
##
##  RestaurantsPriceRange2 RestaurantsAttire    Smoking
##  Min.   :1.00           casual:665        no     :200
##  1st Qu.:1.00           dressy:  4        outdoor:447
##  Median :1.00           formal:  1        yes    : 23
##  Mean   :1.49
##  3rd Qu.:2.00
##  Max.   :4.00
##
##  RestaurantsReservations OutdoorSeating GoodForKids avg_word_count
##  no :536                 no :368        no : 83     Min.   : 28.00
##  yes:134                 yes:302        yes:587     1st Qu.: 80.67
##                                                     Median : 95.38
##                                                     Mean   : 97.00
##                                                     3rd Qu.:112.03
##                                                     Max.   :308.00
##
```

Comparing the same chain restaurant Starbucks across the country

Here, we see that there are 24 Starbucks (25 if including a Starbucks in Mandarin hotel) located all across the country. It may be useful to compare how the same restaurant chain performs at different locations. In the visualization below, we see that Starbucks is mostly rated as 'bad' independent of its location.

```
starbucks <- train[train$name == 'Starbucks',]

ggplot(starbucks, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Starbucks ratings in different states')
```




```
# ggsave(filename = 'starbucks.png', plot = last_plot(),
#   scale = 1, width = par("din")[1], height = par("din")[2], units = c("in",
#           "cm", "mm"), dpi = 300)
```

Restaurant ratings in different states based on cuisine

```
library(gridExtra)

levels(train$cuisine)
```

```
##  [1] "Bakeries"           "Bars"               "Breakfast & Brunch"
##  [4] "Burgers"            "Chinese"            "Coffee & Tea"
##  [7] "Fast Food"          "Italian"            "Mexican"
## [10] "Pizza"              "Sandwiches"
```

```
bakeries <- train[train$cuisine == 'Bakeries',]
bars <- train[train$cuisine == 'Bars',]
Italian <- train[train$cuisine == 'Italian', ]
Pizza <- train[train$cuisine == 'Pizza', ]
fast.food <- train[train$cuisine == 'Fast Food', ]
ct <- train[train$cuisine == 'Coffee & Tea', ]
Sandwiches <- train[train$cuisine == 'Sandwiches', ]
Mexican <- train[train$cuisine == 'Mexican', ]
bb <- train[train$cuisine == 'Breakfast & Brunch', ]
Chinese <- train[train$cuisine == 'Chinese', ]
```

```r
Burgers <- train[train$cuisine == 'Burgers', ]


p <- list()


p[[1]] <- ggplot(bakeries, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Bakeries rating counts')

p[[2]] <- ggplot(bars, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Bars rating counts')

p[[3]] <- ggplot(Italian, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Italian rating counts')

p[[4]] <- ggplot(Pizza, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Pizza rating counts')

p[[5]] <- ggplot(fast.food, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Fast Food rating counts')

p[[6]] <- ggplot(ct, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Coffee & Tea rating counts')

p[[7]] <- ggplot(Sandwiches, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Sandwiches rating counts')

p[[8]] <- ggplot(Mexican, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Mexican rating counts')

p[[9]] <- ggplot(bb, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Brunch & Breakfast rating counts')

p[[10]] <- ggplot(Chinese, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Chinese rating counts')

p[[11]] <- ggplot(Burgers, aes(x = as.factor(state), fill = as.factor(rating)))+
  geom_bar()+
  labs(x = 'State', y = 'Count', title = 'Burgers rating counts')

do.call(grid.arrange, p)
```

If we look at a particular type of cuisine, certain states tend to give better rating on a single type of cuisine compared to the rests.

- Bakeries: Arizona 89% good;
- Coffee and tea: Ohio 86% good;
- Sandwiches: Pennsylvania 83% bad, Wiscosin 80% bad, Arizona 71% good, Nevada 67% bad;
- Chinese: Wisconsin 100% good, Arizona 69% good;
- Burgers: Pennsylvania 100% good, Arizona 80% good, Wisconsin 80% good

```
# percentage good/bad rating in each state for selected cuisines
cuisine.selected <- c('Bakeries', 'Coffee & Tea', 'Sandwiches', 'Chinese', 'Burgers')

# calculate percentages
for(i in 1:length(cuisine.selected)){
  percentage.good <- list()
  percentage.bad <- list()
  for(j in 1:length(states)){
    data <- train[train$state == states[[j]] & train$cuisine == cuisine.selected[[i]],]
    percentage.good[[j]] <- mean(data$rating == 1)
    percentage.bad[[j]] <- mean(data$rating == 0)
  }
```

```
  cat('Cuisine:', cuisine.selected[[i]], '\n')
  print(data.frame(state = states,
                   percentage.good = round(as.numeric(percentage.good),2),
                   percentage.bad = round(as.numeric(percentage.bad),2)))
  cat('\n')
}
```

```
## Cuisine: Bakeries
##   state percentage.good percentage.bad
## 1    AZ            0.89           0.11
## 2    IL             NaN            NaN
## 3    NC            0.60           0.40
## 4    NV            0.67           0.33
## 5    OH            0.67           0.33
## 6    PA            1.00           0.00
## 7    SC             NaN            NaN
## 8    WI            0.33           0.67
##
## Cuisine: Coffee & Tea
##   state percentage.good percentage.bad
## 1    AZ            0.39           0.61
## 2    IL            1.00           0.00
## 3    NC            0.33           0.67
## 4    NV            0.39           0.61
## 5    OH            0.86           0.14
## 6    PA            0.56           0.44
## 7    SC             NaN            NaN
## 8    WI            0.50           0.50
##
## Cuisine: Sandwiches
##   state percentage.good percentage.bad
## 1    AZ            0.71           0.29
## 2    IL            1.00           0.00
## 3    NC            0.50           0.50
## 4    NV            0.33           0.67
## 5    OH            0.64           0.36
## 6    PA            0.17           0.83
## 7    SC            0.00           1.00
## 8    WI            0.20           0.80
##
## Cuisine: Chinese
##   state percentage.good percentage.bad
## 1    AZ            0.69           0.31
## 2    IL             NaN            NaN
## 3    NC            0.44           0.56
## 4    NV            0.40           0.60
## 5    OH            0.50           0.50
## 6    PA            0.50           0.50
## 7    SC            0.00           1.00
## 8    WI            1.00           0.00
##
## Cuisine: Burgers
##   state percentage.good percentage.bad
## 1    AZ            0.80           0.20
```

```
## 2    IL            NaN           NaN
## 3    NC           0.50          0.50
## 4    NV           0.50          0.50
## 5    OH           0.33          0.67
## 6    PA           1.00          0.00
## 7    SC            NaN           NaN
## 8    WI           0.80          0.20
```

---

300 WORDS SUMMARY:

The relationship between a restaurant's rating and its location

Within each state, a restaurant's rating is largely independent of its location. Figure 1 shows the scatter plot of restaurants' ratings in every state. There is no particular cluster or pattern relating a restaurant's location to its rating within each state.

However, **a restaurant's rating is dependent on the state it is located in.** In **North Carolina, Nevada, Pennsylvania**, and **Wisconsin**, the number of restaurants rated as `good` and `bad` are similar in numbers. In **Arizona** and **Ohio**, restaurants are rated `good` ~**1.5x more often** than `bad`, while in **Illinois** restaurants are overwhelmingly rated as `good`, and in **South Carolina** overwhelmingly rated as `bad`. However, there are too few data points in either state to draw conclusive statements. This statistics is summarized in Figure 2.

Performance of Starbucks across the country

To find out if location truely affects a restaurant's rating, it is perhaps fair to compare how one specific restaurant is rated in different locations. Here, we compare the ratings of 25 **Starbucks** across the country. Figure 3 shows that Starbucks is mostly rated as 'bad' independent of its location.

Cuisine specific restaurant rating vs location

So far we have looked at all restaurants as a whole. If instead, we were to examine restaurants of a particular type of cuisine, we realize that **certain states tend to give overwhelmingly better ratings for a particular type of cuisine** compared to other states. The bar charts in Figure 4 shows that restaurants serving **bakery goods, coffee & tea, sandwiches, Chinese**, and **burgers** have state-dependent rating behavior. Below is the summary statistics of those that are out of the ordinary:

- **Bakeries**: Arizona 89% good;
- **Coffee and tea**: Ohio 86% good;
- **Sandwiches**: Pennsylvania 83% bad, Wisconsin 80% bad, Arizona 71% good;
- **Chinese**: Wisconsin 100% good;
- **Burgers**: Pennsylvania 100% good, Arizona 80% good, Wisconsin 80% good

Furthur analysis

We also consider the possibility that a restaurant's rating may be related to its proximity to places of interest, town centers, highways, train/bus stations, etc.. However, there isn't enough information in the dataset for us to perform such analysis. From the map plots in Figure 1, we do have two minor observations: **restaurants**

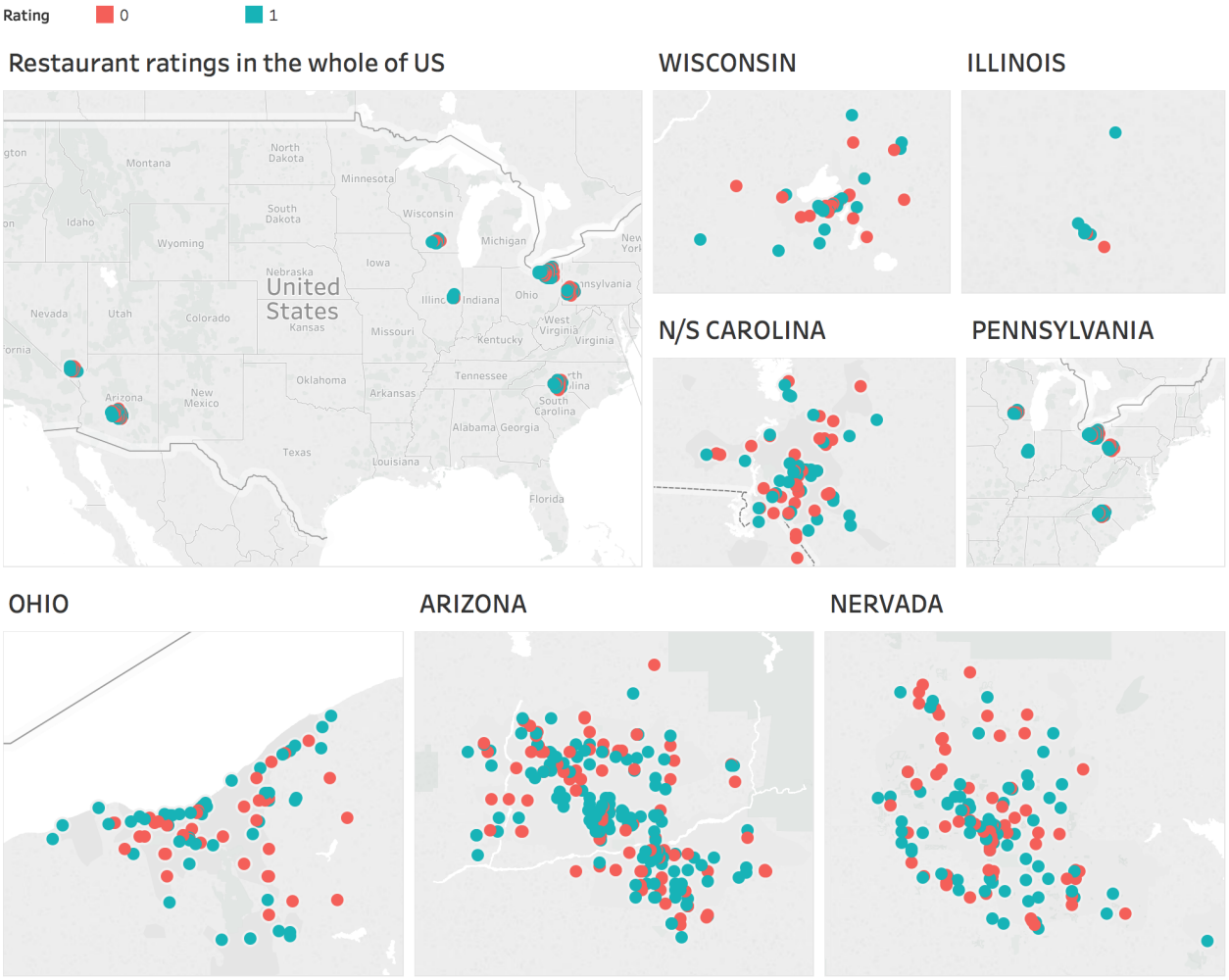


Figure 1: Scatter plot of restaurant ratings according to states

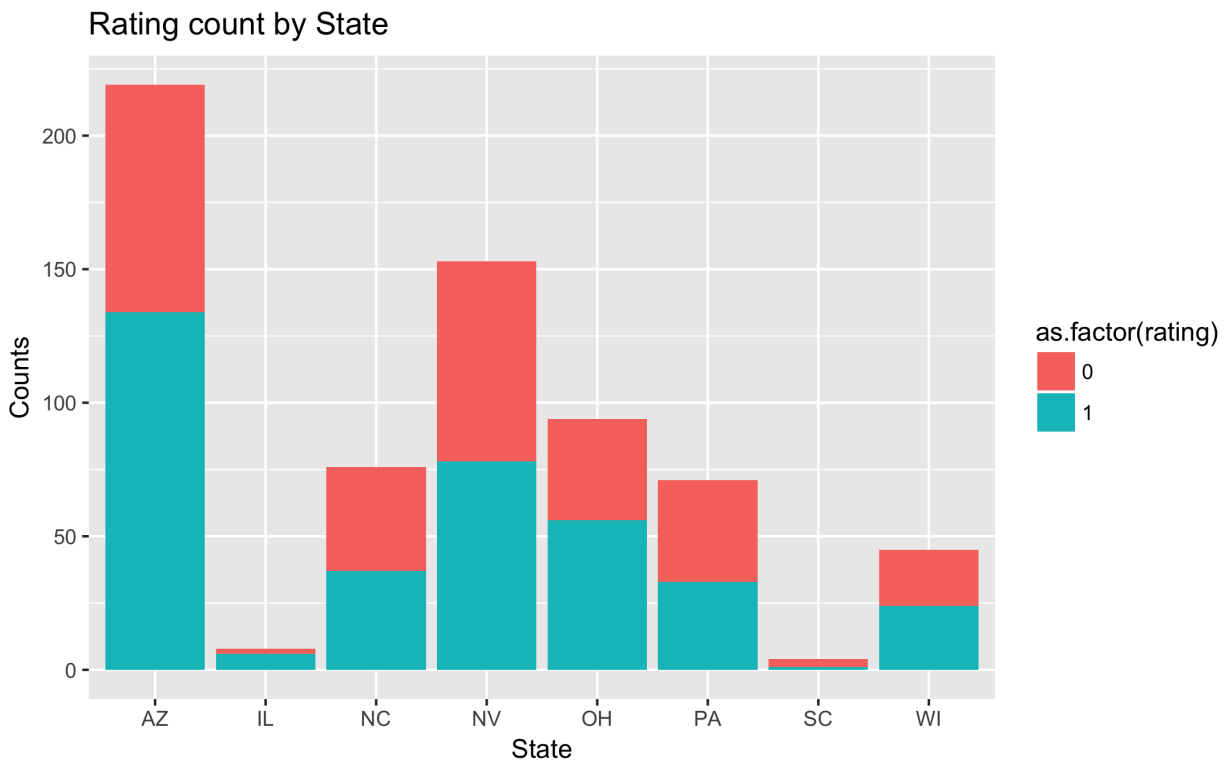


Figure 2: Rating count by state

Figure 3: Starbucks ratings across different states

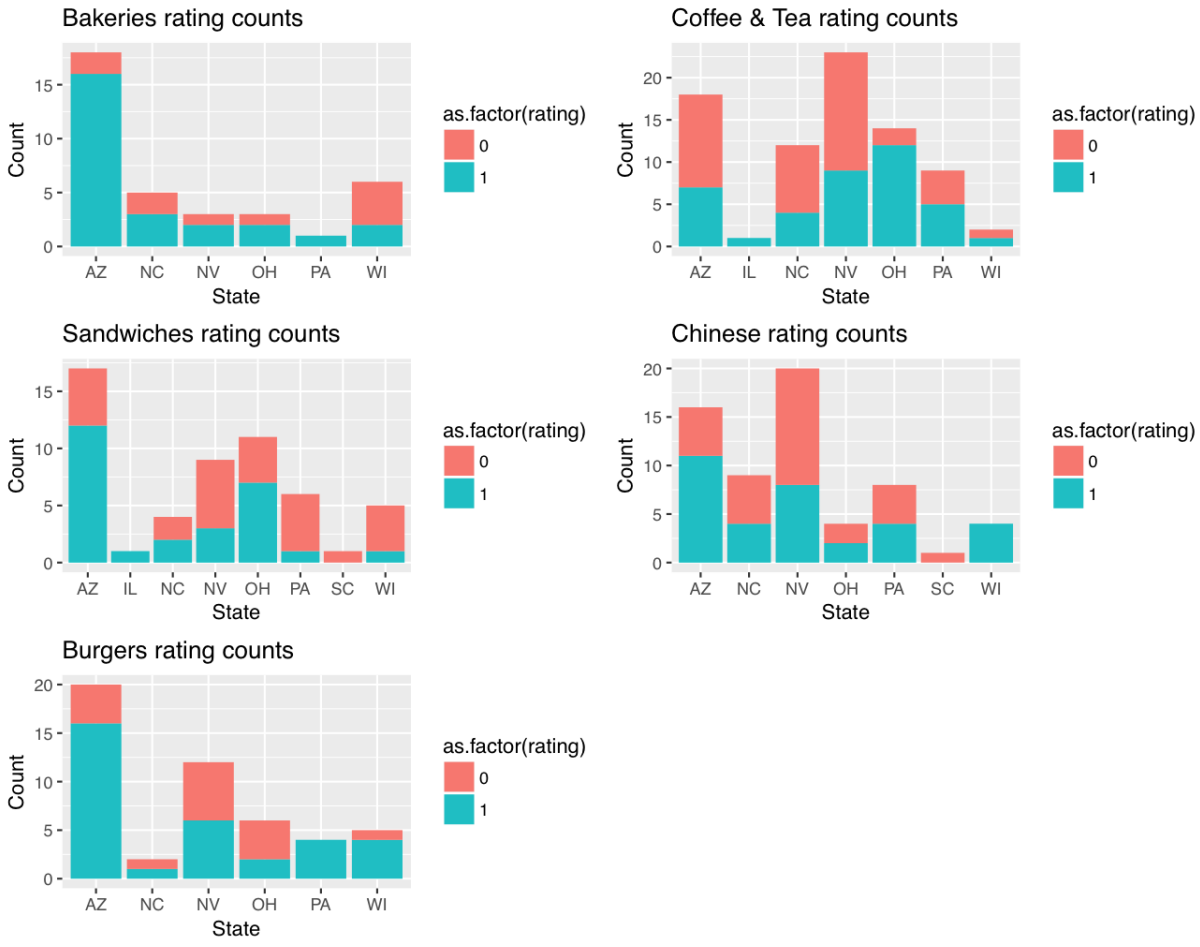


Figure 4: Cuisines-specific rating counts

located along Lake Erie in Ohio tend to have `good` ratings, while **restaurants located along the Vegas strip in Nevada tend to have `bad` ratings** (not shown in figure). This goes to show that a restaurant's rating can be related to other attributes that are not available in our current dataset. We would be able to push further our analysis on the relationship between a restaurant's location and its rating should we have more information on these restaurants' proximities to nearby places of interest.

Problem 2 [35 points]

This problem is concerned with predicting a restaurant's rating based on text features. We'll consider the Multinomial-Dirichlet Bayesian model to describe the distribution of text features and finally to predict the binary ratings.

Probability model: Let $(y_1^g, y_2^g, \ldots, y_{100}^g)$ denote the total counts of the 100 dictionary words across the reviews for "good" restaurants, and $(y_1^b, y_2^b, \ldots, y_{100}^b)$ denote the total counts of the 100 dictionary words across the reviews for "bad" restaurants. We assume the following *multinomial* likelihood model:

$$p(y_1^g, y_2^g, \ldots, y_{100}^g \,|\, \theta_1^g, \ldots, \theta_{100}^g) \;\propto\; (\theta_1^g)^{y_1^g} (\theta_2^g)^{y_2^g} \ldots (\theta_{100}^g)^{y_{100}^g}$$

$$p(y_1^b, y_2^b, \ldots, y_{100}^b \,|\, \theta_1^b, \ldots, \theta_{100}^b) \;\propto\; (\theta_1^b)^{y_1^b} (\theta_2^b)^{y_2^b} \ldots (\theta_{100}^b)^{y_{100}^b}.$$

The model parameters $(\theta_1^g, \ldots, \theta_{100}^g)$ and $(\theta_1^b, \ldots, \theta_{100}^b)$ are assumed to follow a *Dirichlet* prior distribution with parameter α. That is

$$p(\theta_1^g, \ldots, \theta_{100}^g) \;\propto\; (\theta_1^g)^\alpha \ldots (\theta_{100}^g)^\alpha$$
$$p(\theta_1^b, \ldots, \theta_{100}^b) \;\propto\; (\theta_1^b)^\alpha \ldots (\theta_{100}^b)^\alpha.$$

Hence we can interpret, for example, θ_5^g as the probability the word "perfect" is observed once in a review of "good" restaurants. For the purposes of this problem, set $\alpha = 2$.

(a) Describe briefly in words why the posterior distribution formed from a Dirichlet prior distribution and a multinomial likelihood is a Dirichlet posterior distribution? What are the parameters for the Dirichlet posterior distribution? [5 points]

Answer

The Dirichlet distribution is a conjugate prior to the multinomial likelihood function. The posterior distribution is $p(\theta|y)$ is proportional to $p(y|\theta)p(\theta)$ according to Bayes Rule. This means that if the prior distribution of the multinomial parameters is Dirichlet then the posterior distribution is also a Dirichlet distribution, but with different parameters from those of the prior. The advantage of using conjugate priors is that the posterior distribution is easy to compute (addition of exponential components as we will explain below), and in some sense it is possible to quantify how much our beliefs have changed after collecting the data.

Dirichlet prior distribution: Let's take for example the restaurants that are rated good. $\theta^g = (\theta_1^g, ..., \theta_{100}^g)$ is the vector of multinomial parameters (i.e. the probabilities of each word appearing in a restaurant that is reviewed as good). The probability of this vector follows a dirichlet distribution with a constant alpha as assumed in the question. This gives us:

$$p(\theta) = p(\theta_1^g, ..., \theta_{100}^g) \propto (\theta_1^g)^{\alpha-1}(\theta_2^g)^{\alpha-1}...(\theta_{100}^g)^{\alpha-1} \propto Dirichlet(\alpha, ..., \alpha)$$

Multinomial likelihood: The likelihood function is

$$p(y|\theta^g) = p(y_1^g, y_2^g, \ldots, y_{100}^g \,|\, \theta_1^g, \ldots, \theta_{100}^g) \;\propto\; (\theta_1^g)^{y_1^g} (\theta_2^g)^{y_2^g} \ldots (\theta_{100}^g)^{y_{100}^g}$$

Posterior distribution: The posterior distribution is proportional to, up to a constant, the product of the likelihood distribution $p(y|\theta)$ and the prior distribution $p(\theta)$, which gives us:

$$p(y|\theta)p(\theta) \propto [(\theta_1^g)^{y_1^g}(\theta_2^g)^{y_2^g}\dots(\theta_{100}^g)^{y_{100}^g}][(\theta_1^g)^{\alpha-1}(\theta_2^g)^{\alpha-1}...(\theta_{100})^{\alpha-1}] = (\theta_1^g)^{(\alpha-1+y_1^g)}(\theta_2^g)^{(\alpha-1+y_2^g)}...(\theta_{100})^{(\alpha-1+y_{100}^g)}$$

We can easily tell that the posterior distribution is also a Dirichlet distribution of $(\theta_1^g, ..., \theta_{100}^g)$ with new posterior alphas $(\alpha, \alpha, ..., \alpha) + (y_1^g, y_2^g, ..., y_{100}^g)$, where $\alpha = 2$. The same works for those with bad restaurant ratings.

To summarize, the parameters for the Dirichlet posterior distribution for restaurants rated good is $\alpha_{posterior}^g$, where $\alpha_{posterior}^g = [(2 + y_1^g), (2 + y_2^g), ..., (2 + y_{100}^g)]$; the parameters for the Dirichlet posterior distribution for restaurants rated bad is $\alpha_{posterior}^b = [(2 + y_1^b), (2 + y_2^b), ..., (2 + y_{100}^b)]$.

(b) From a Monte Carlo simulation of the Dirichlet posterior distribution for "good" restaurants, what is the posterior mean probability that the word "chocolate" is used? From a Monte Carlo simulation of the Dirichlet posterior distribution for bad restaurants, what is the posterior mean probability that the word "chocolate" is used? **Hint**: use the `rdirichlet` function in the `MCMCpack` library. [15 points]

Answer

```
library(MCMCpack)
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2017 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
set.seed(42)

## priors, likelihood, and posterior expressed as dirichlet distribution parameters
prior_alpha <- 2

likelihood_alpha.gd <- as.numeric(apply(
  train[train$rating == 1, seq(22, 121)], 2, sum))
likelihood_alpha.bad <- as.numeric(apply(
  train[train$rating == 0, seq(22, 121)], 2, sum))

posterior_alpha.gd <- likelihood_alpha.gd + prior_alpha
posterior_alpha.bad <- likelihood_alpha.bad + prior_alpha

# sample for posterior distribution using the dirichlet distribution
posterior_prob.gd <- rdirichlet(10000, posterior_alpha.gd)
posterior_prob.bad <- rdirichlet(10000, posterior_alpha.bad)

cat('Posterior mean probability that "chocolate" appears in "good" restaurants:',
    apply(posterior_prob.gd, 2, mean)[2], '\n')
```

```
## Posterior mean probability that "chocolate" appears in "good" restaurants: 0.01723619
```

```
cat('Posterior mean probability that "chocolate" appears in "bad" restaurants:',
    apply(posterior_prob.bad, 2, mean)[2])
```

```
## Posterior mean probability that "chocolate" appears in "bad" restaurants: 0.005544536
```

(c) For the restaurants in the test data set, estimate the probability based on the results of the Dirichlet-Multinomial model that each is good versus bad. You may want to apply the function `posterior_pA` provided below (in `midterm-1.Rmd`). Create a visual summary relating the estimated probabilities and the actual binary ratings in the test data. [15 points]

Answer

```
# confusion matrix, as proportions.
confmat <- function(actual, predicted) {
    addmargins(
        prop.table(
            table(actual, predicted),
            margin = 1),
        margin = 2)
}
```

```
# sum over columns of word count
yA = as.numeric(apply(train[train$rating == 1, -seq(1, 21)], 2, sum)) # good
yB = as.numeric(apply(train[train$rating == 0, -seq(1, 21)], 2, sum)) # bad

labels <- test[, 1] # rating of test set
features <- test[, -seq(1, 21)]

n.test = nrow(test)
dirichlet.probs = rep(NA, n.test)

for(i in 1:n.test){
  y_til = as.numeric(as.character(features[i, ]))
  dirichlet.probs[i] = posterior_pA(alpha = 2, yA = yA, yB=yB, y_til = y_til)
}

# # Estimated probability of good
# cat('List of estimated probability that restaurant is good:')
# dirichlet.probs
#
# # Estimated probability of bad
# cat('List of estimated probability that restaurants is bad:')
# 1-dirichlet.probs

cat('Classification statistics on the test set')
```

```
## Classification statistics on the test set
```

```
table(labels, dirichlet.probs > 0.5)
```

```
##
## labels FALSE TRUE
##      0   111   35
##      1    34  148
```

```
# Accuracy tests
cat('\nConfusion Matrix\n')
```

```
##
## Confusion Matrix
```

```
confmat(labels, round(dirichlet.probs))
```

```
##       predicted
## actual         0         1       Sum
##      0 0.7602740 0.2397260 1.0000000
##      1 0.1868132 0.8131868 1.0000000
```

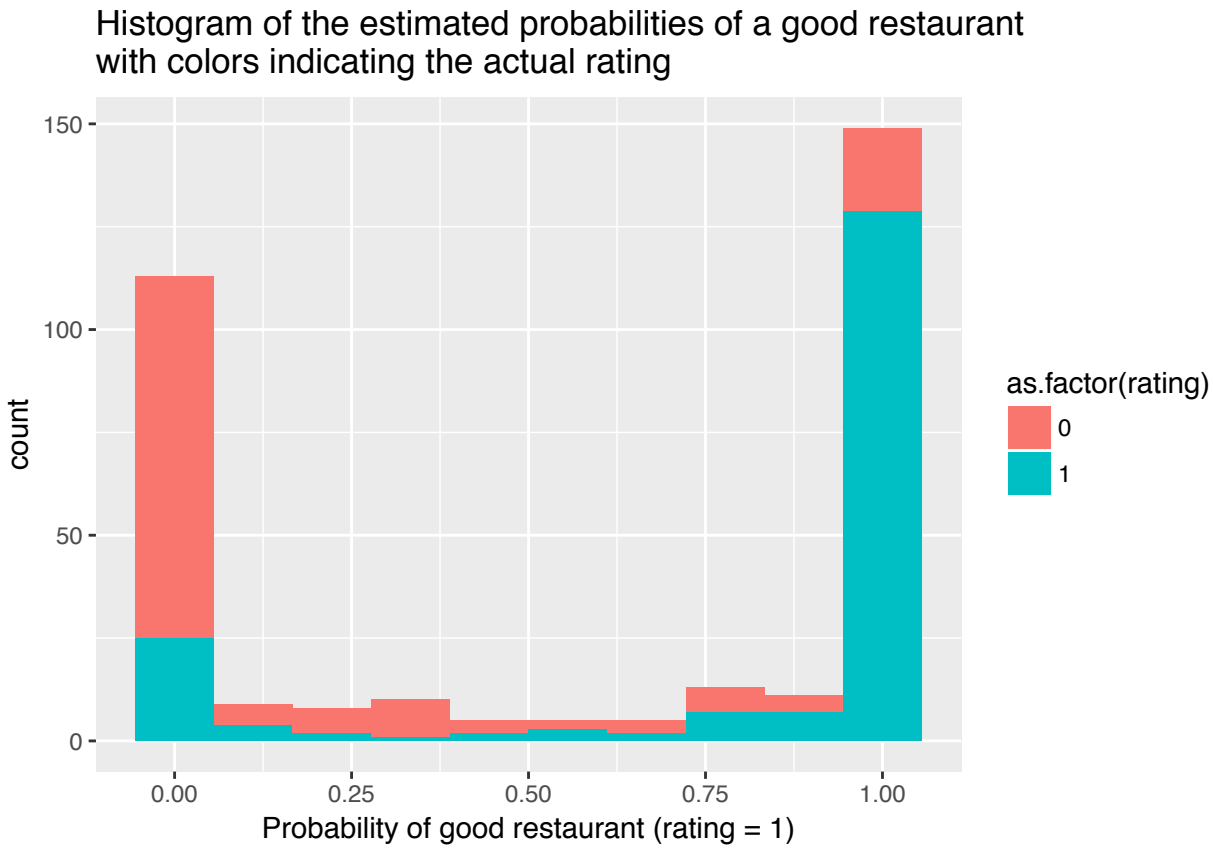
```
cat('\nOverall Accuracy\n')
```

```
##
## Overall Accuracy
```

```
mean(labels == round(dirichlet.probs))
```

```
## [1] 0.7896341
```

```
# visualization
test_ <- test
test_$prob <- dirichlet.probs
test_$pred <- round(dirichlet.probs)

ggplot(test_, aes(x = prob, fill = as.factor(rating)))+
  geom_histogram(bins = 10)+
  labs(x = 'Probability of good restaurant (rating = 1)', y = 'count',
       title = 'Histogram of the estimated probabilities of a good restaurant
with colors indicating the actual rating')
```



Problem 3 [45 points]

This problem is concerned with modeling a restaurant's rating on factors other than word occurrences.

(a) Construct a model for the probability a restaurant is rated "good" as a function of latitude and longitude, average word count, and business attributes. Include quantitative predictor variables as smoothed terms as you see appropriate. You may use default tuning parameter. Summarize the results of the model. Does evidence exist that the smoothed terms are significantly non-linear? Produce visual displays to support your conclusions. [20 points]

Answer

Data exploration

First, we explore to see if there is an obvious dependency of good or bad ratings on any of the business attributes.

```
predictors <- colnames(train)[3:20]

p <- list()

for (i in 1:length(predictors)){
p[[i]]<-(ggplot(train, aes_string(x = predictors[i], fill = 'as.factor(rating)'))+
  geom_bar())
}

p[[19]] <- ggplot(train, aes(x = avg_word_count, fill = as.factor(rating)))+
  geom_histogram(bins = 30)

do.call(grid.arrange,p)
```

From the graphs above, there isn't any particular business attribute that stands out at skewing the rating to either good or bad. Hence, we try to include all business attributes in the following.

Building a GAM model

GAM model 1: mod.gam1

We first consider including all business attributes, and apply smooth functions on continuous variables.

```
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.14
```

```
mod.gam1 <- gam(rating ~ s(latitude) * s(longitude) + s(avg_word_count) + s(review_count)
                + state + cuisine + WheelchairAccessible + WiFi +
                  BusinessAcceptsCreditCards + Alcohol + NoiseLevel +
                  factor(RestaurantsPriceRange2) + RestaurantsAttire +
                  Smoking + RestaurantsReservations + OutdoorSeating + GoodForKids,
                data = train, family=binomial(link = "logit"))

pred.gam1 <- round(predict(mod.gam1, newdata=test, type="response"))

cat('Confusion matrix\n')
```

```
## Confusion matrix
```

```
confmat(test$rating, pred.gam1)
```

```
##       predicted
## actual         0         1       Sum
##      0 0.5136986 0.4863014 1.0000000
##      1 0.3131868 0.6868132 1.0000000
```

```
cat('\nOverall accuracy\n')
```

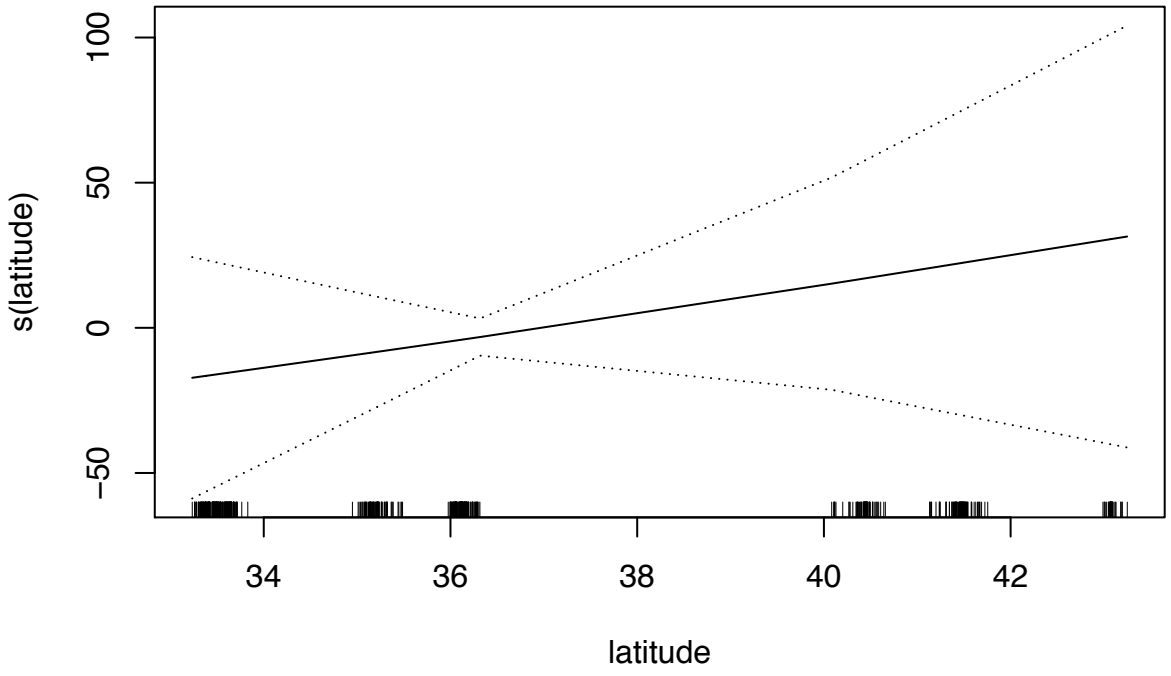
```
##
## Overall accuracy
```

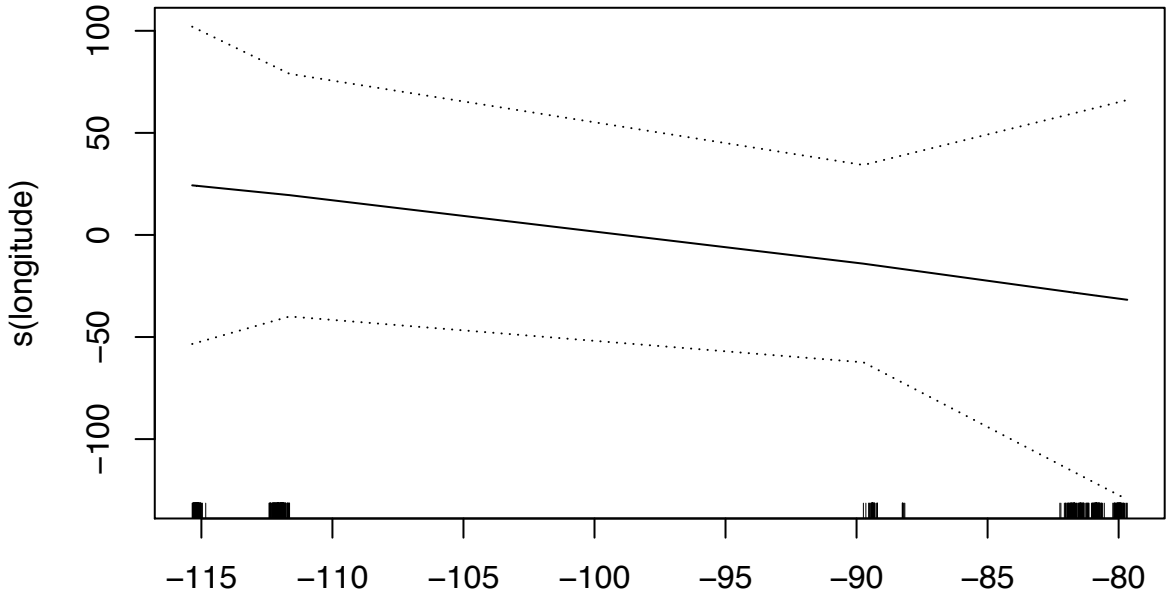
```
mean(test$rating == pred.gam1)
```

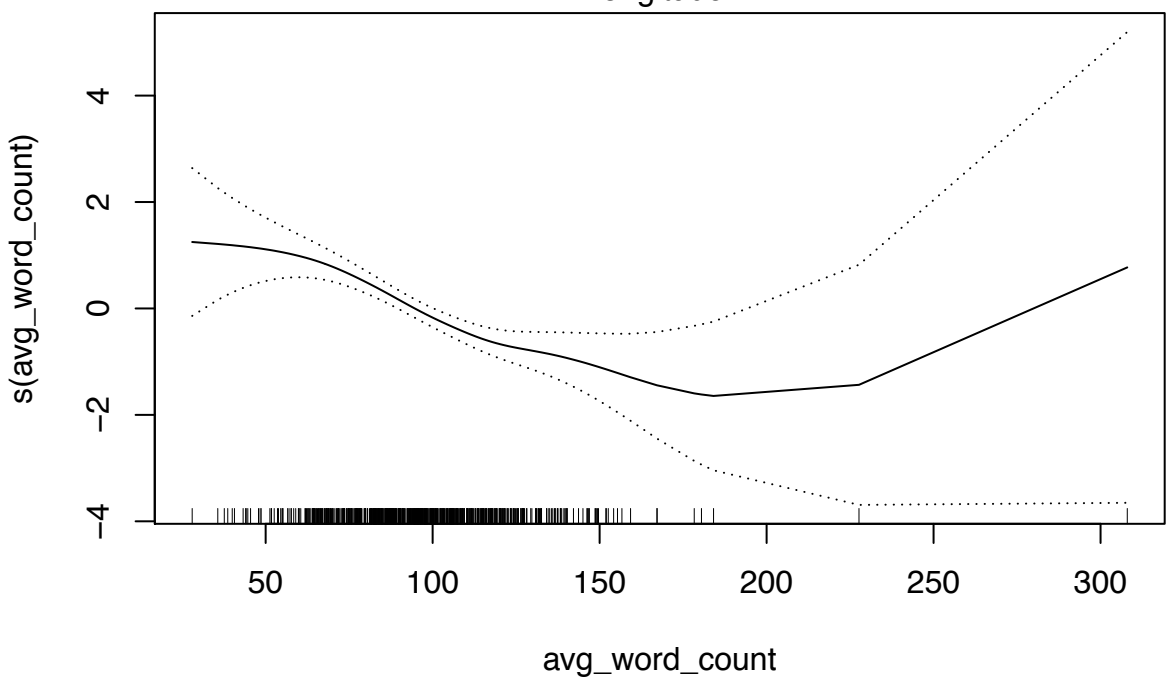
```
## [1] 0.6097561
```

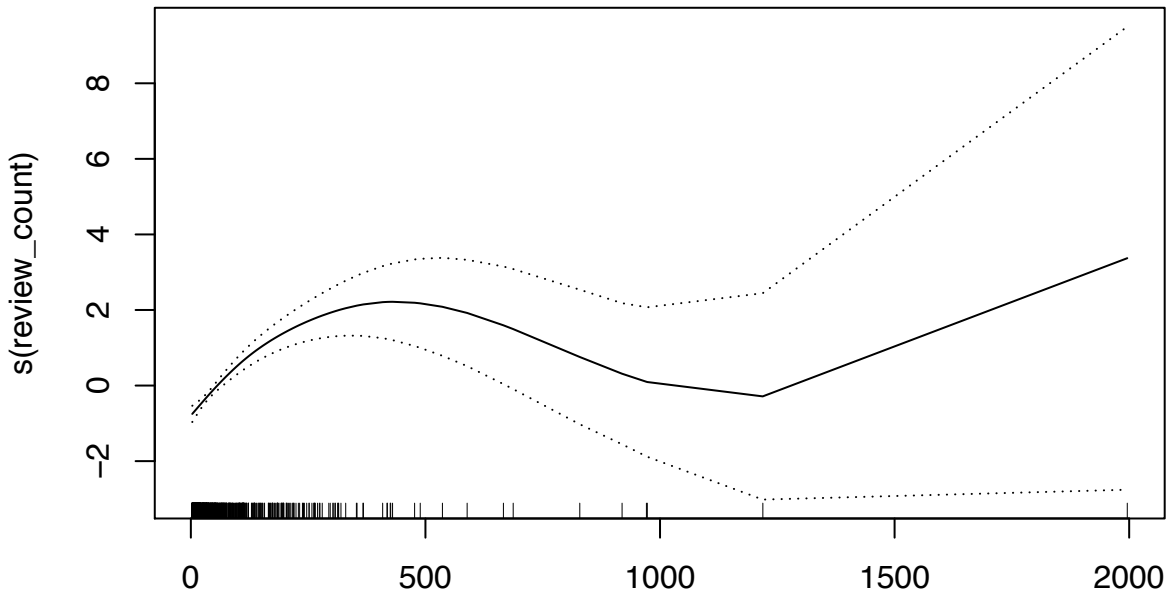
```
plot(mod.gam1, se=TRUE)
```

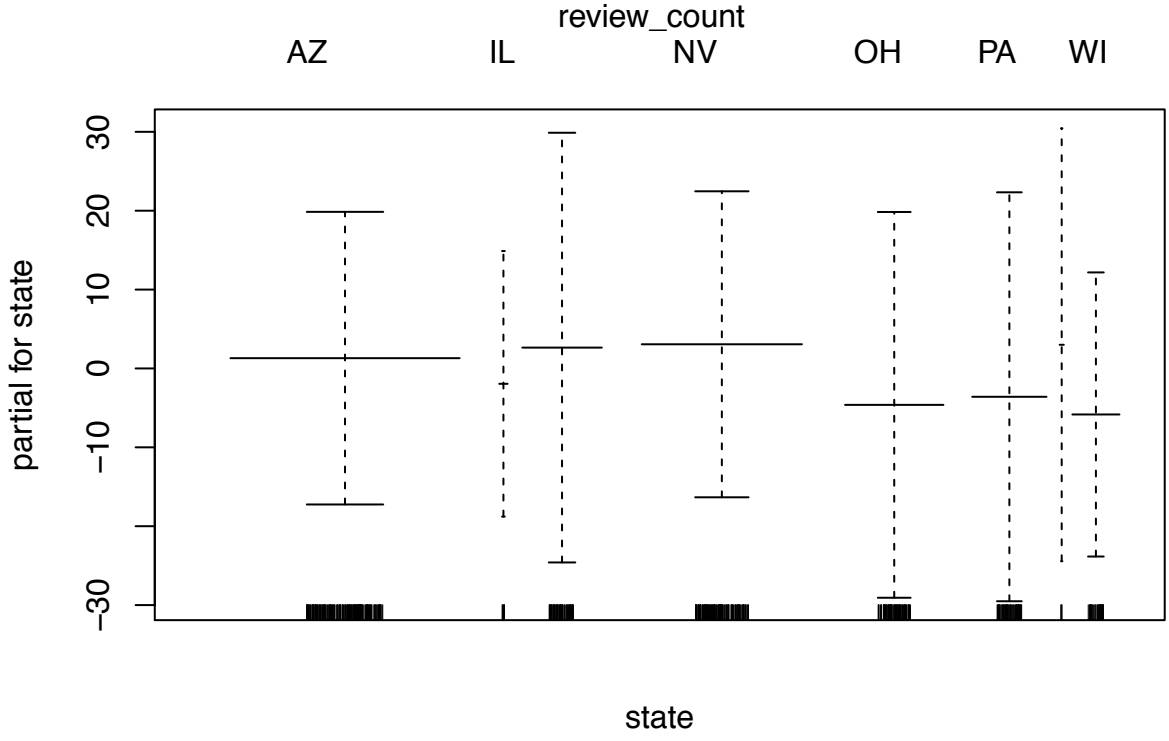
```
## Warning in preplot.gam(x, terms = terms): No terms saved for "a:b" style
## interaction terms
```

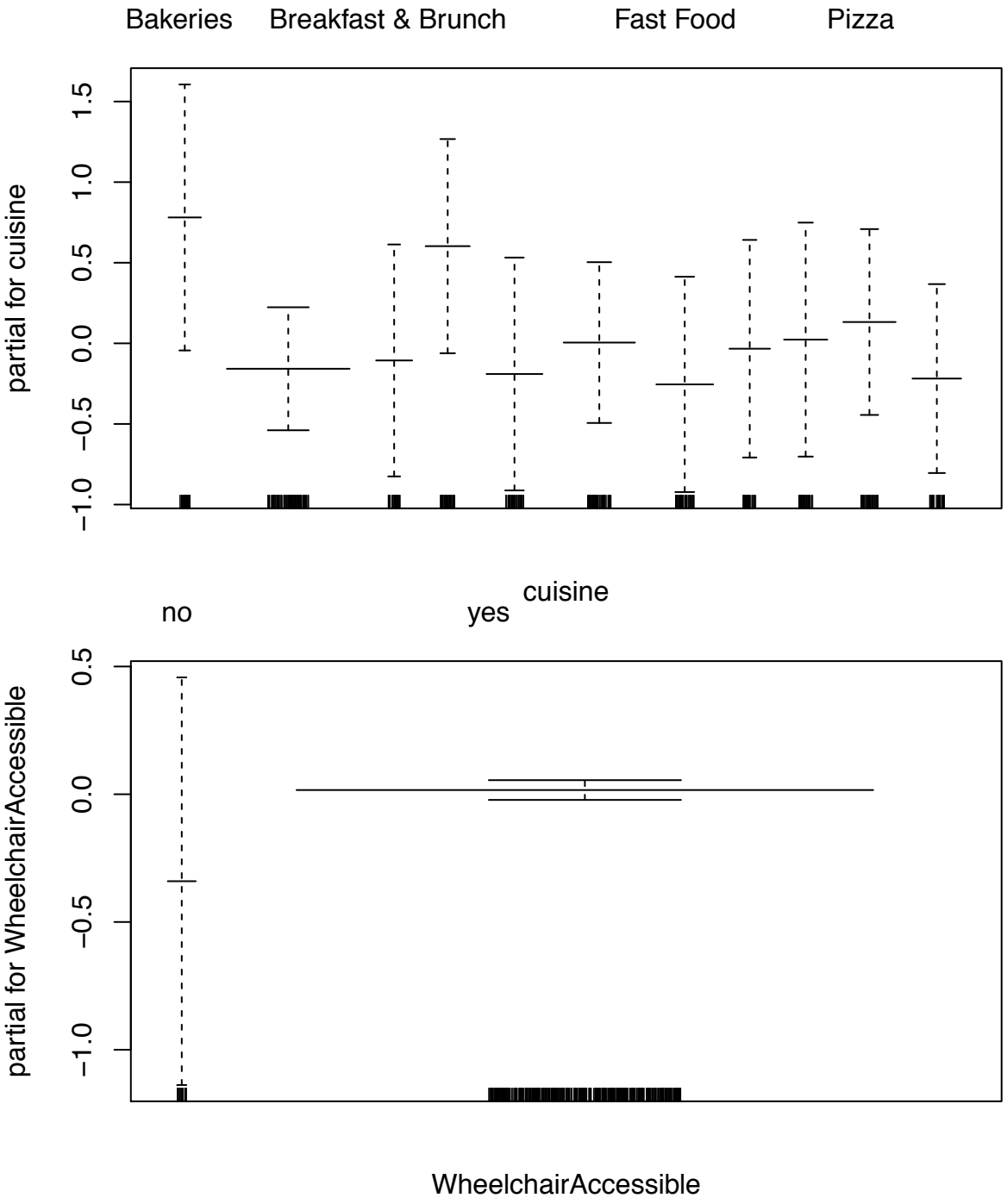


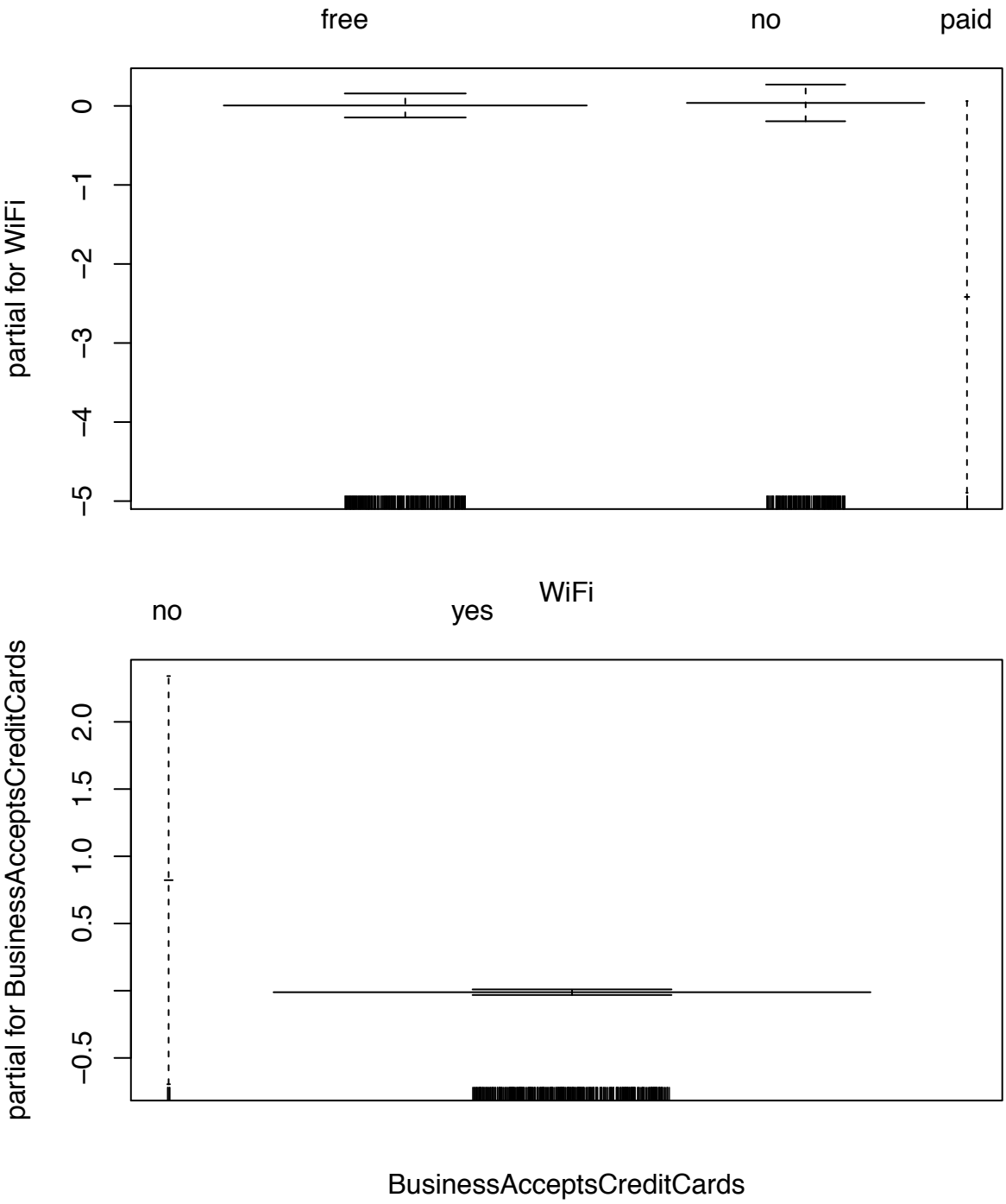

s(longitude)
100
50
0
−50
−100
−115
−110
−105
−100
−95
−90
−85
−80
longitude

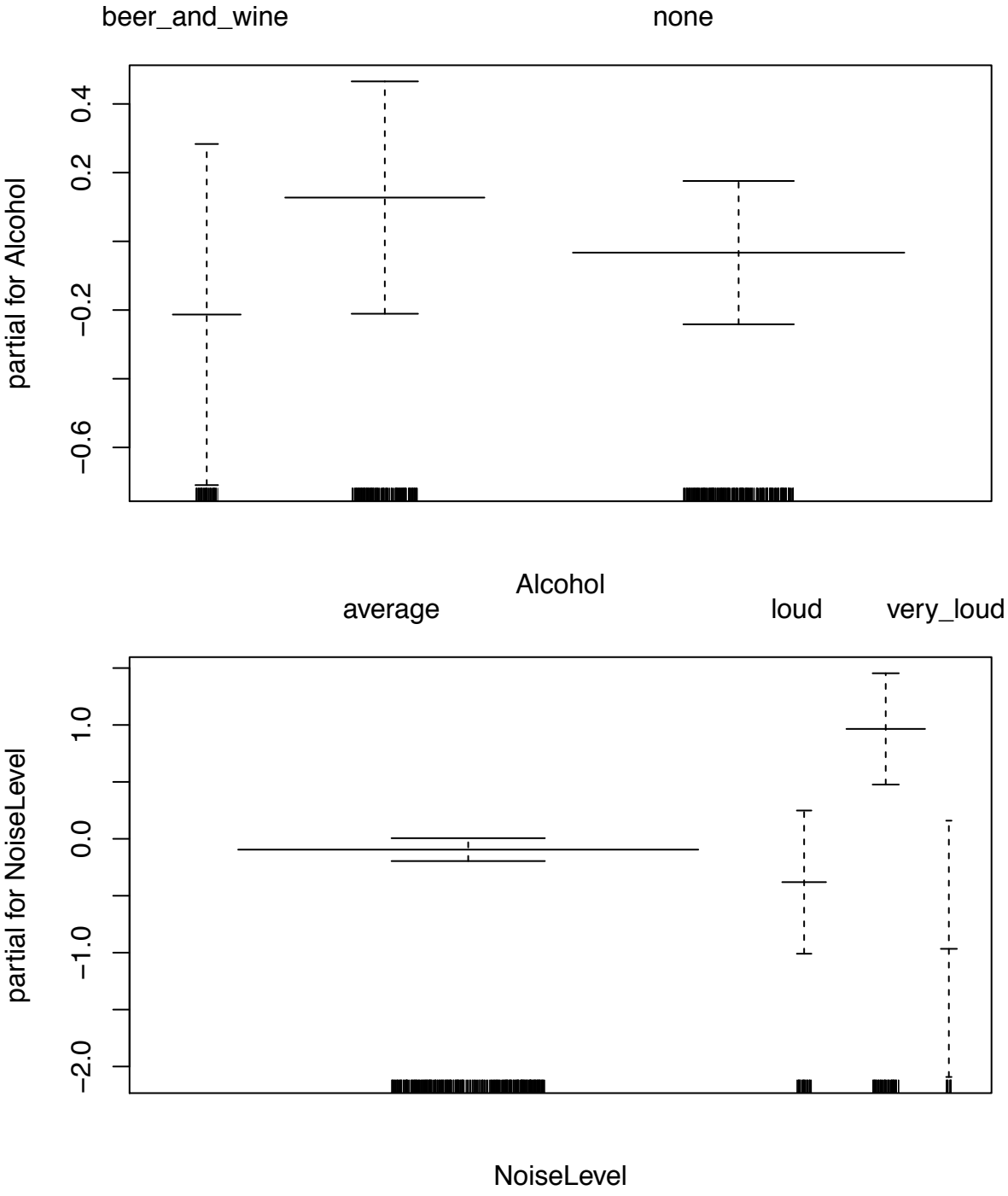

s(avg_word_count)
4
2
0
−2
−4
50
100
150
200
250
300
avg_word_count

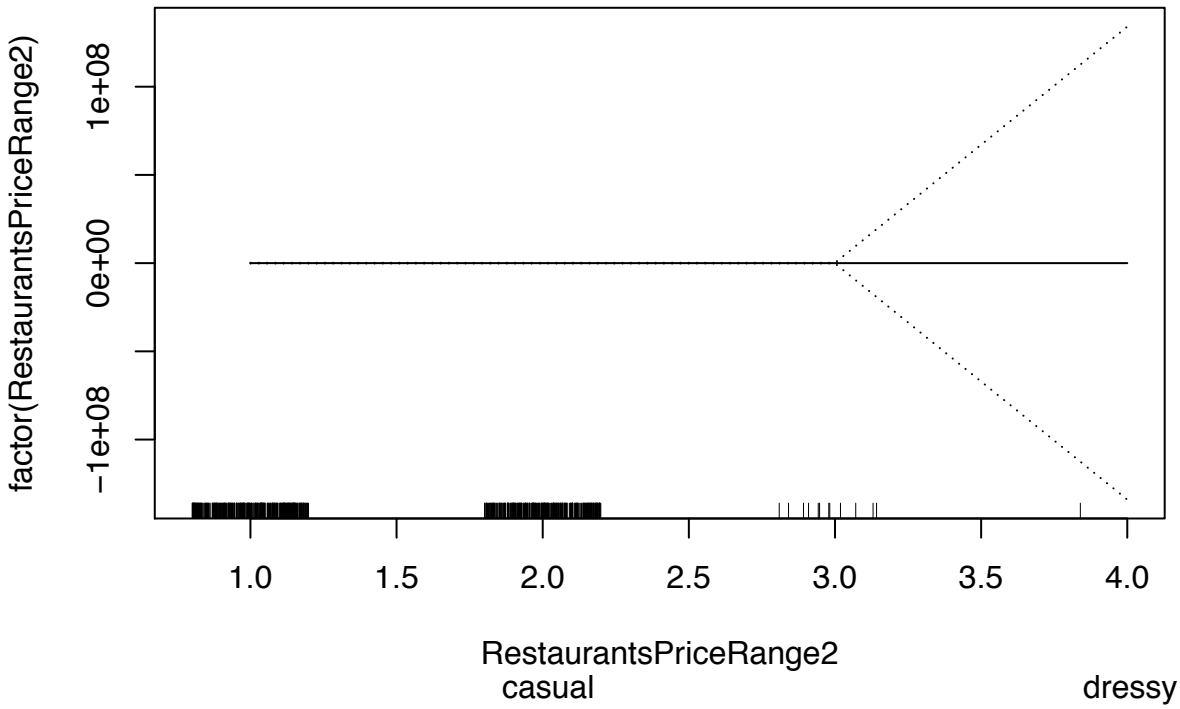

s(review_count)
8
6
4
2
0
−2
0
500
1000
1500
2000
review_count

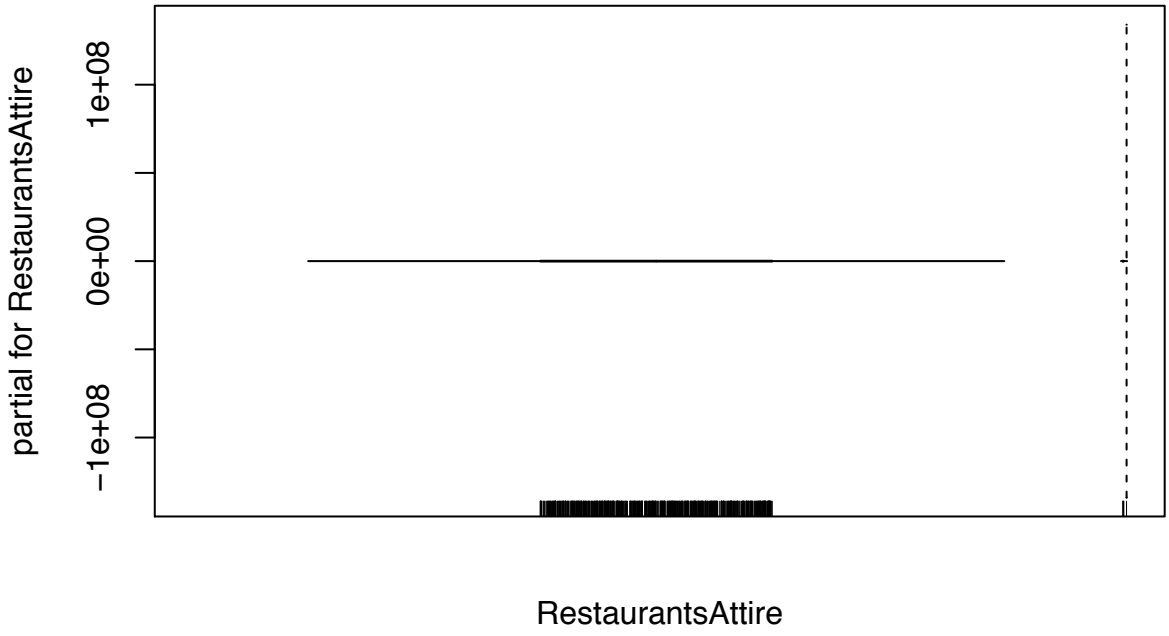

AZ
IL
NV
OH
PA
WI
partial for state
30
20
10
0
−10
−30
state

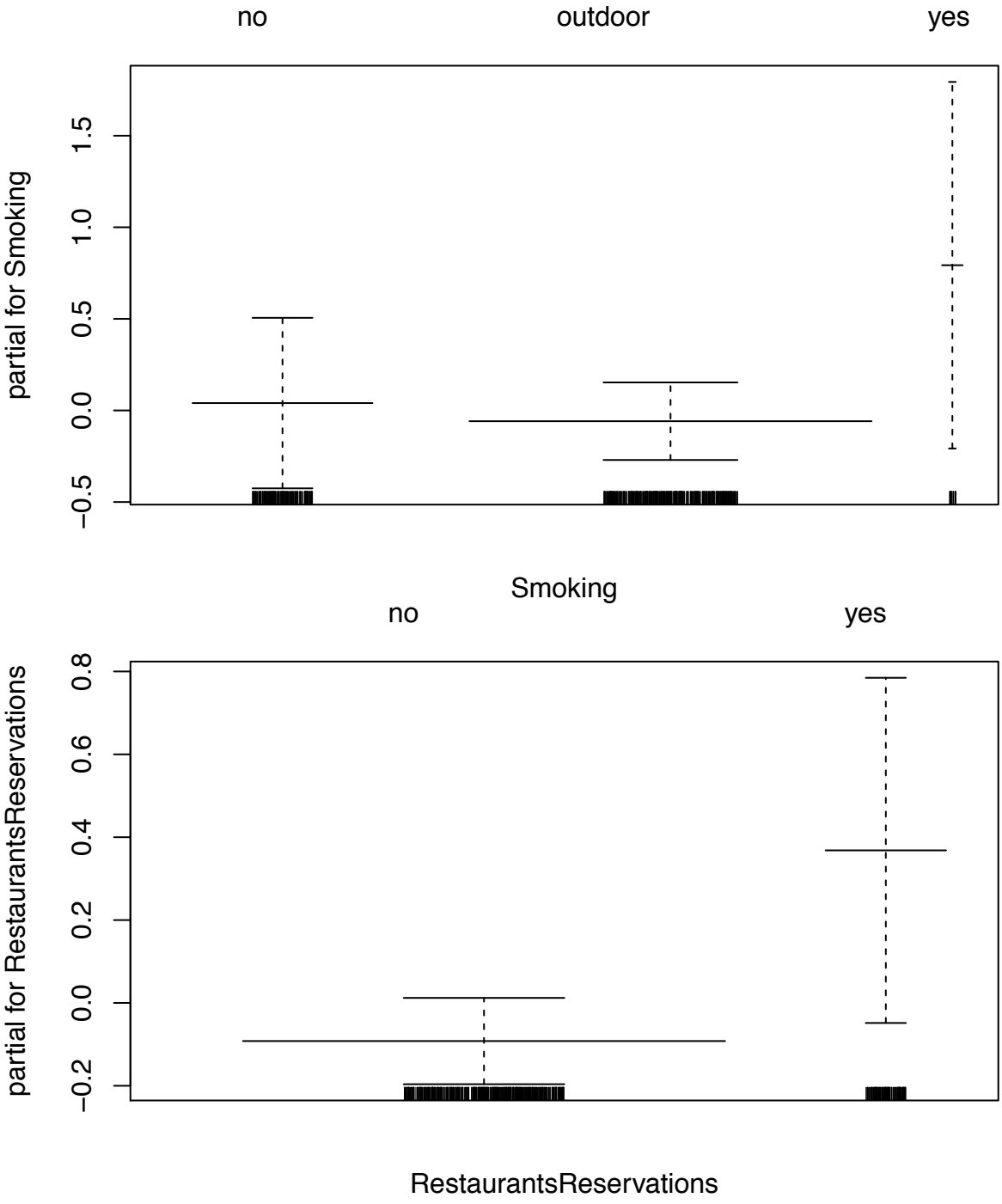

Bakeries
Breakfast & Brunch
Fast Food
Pizza
partial for cuisine
−1.0
−0.5
0.0
0.5
1.0
1.5
cuisine
no
yes
partial for WheelchairAccessible
−1.0
−0.5
0.0
0.5
WheelchairAccessible

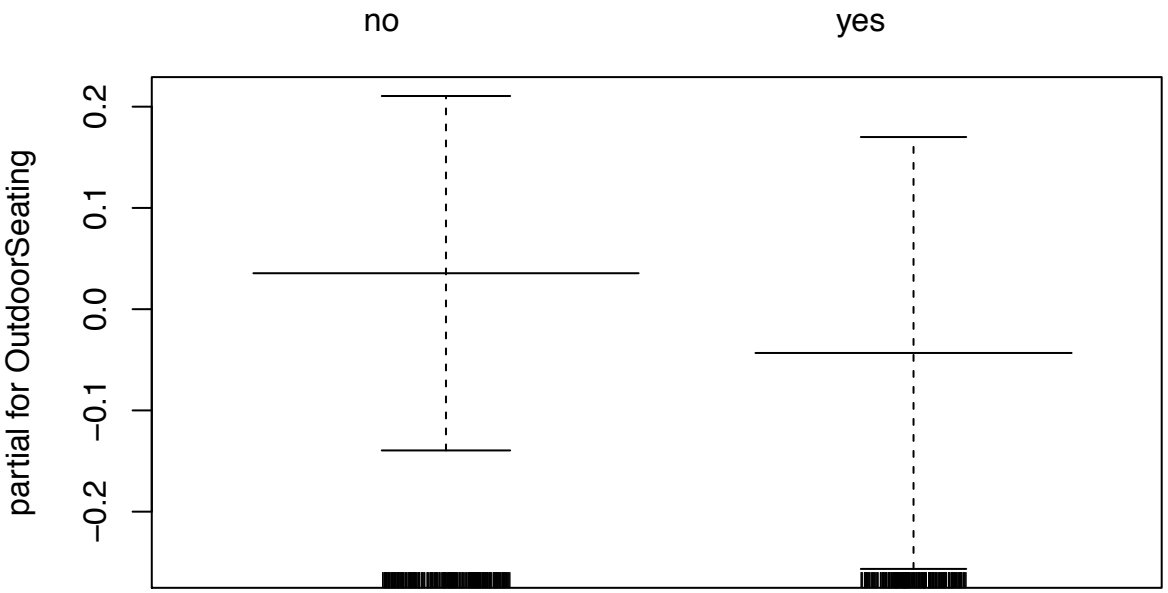

free
no
paid
partial for WiFi
0
−1
−2
−3
−4
−5
WiFi
no
yes
partial for BusinessAcceptsCreditCards
2.0
1.5
1.0
0.5
−0.5
BusinessAcceptsCreditCards

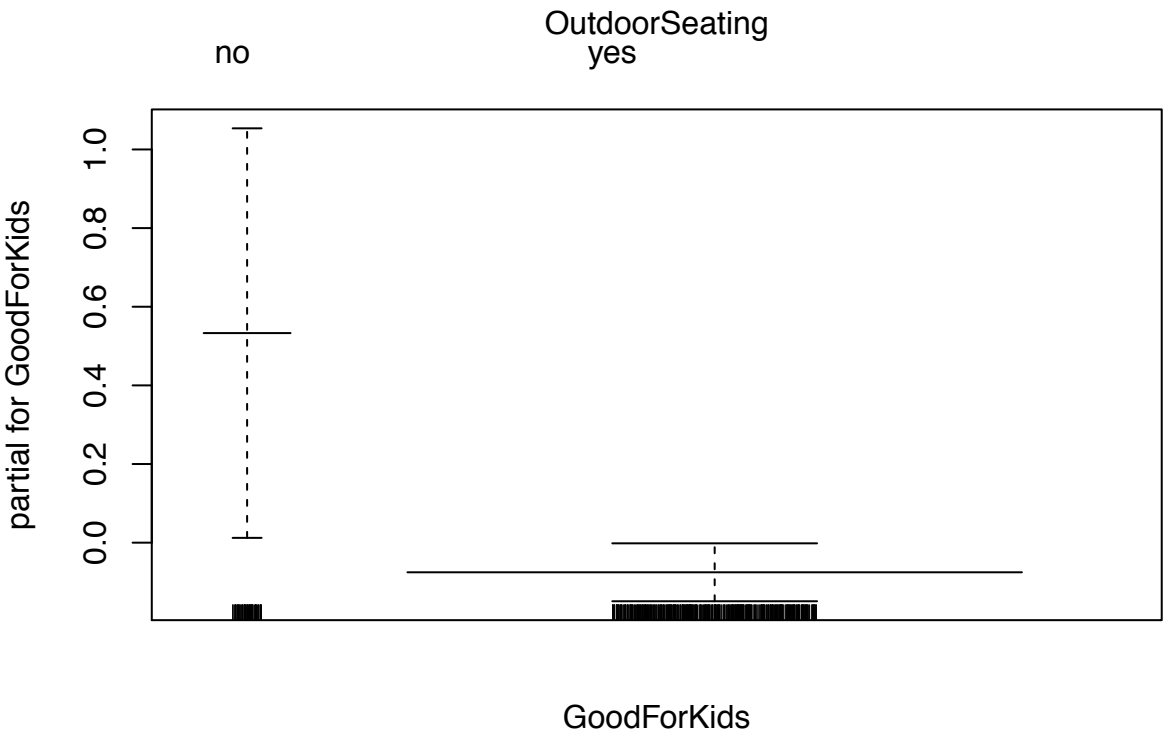

beer_and_wine
none
partial for Alcohol
0.4
0.2
−0.2
−0.6
Alcohol
average
loud
very_loud
partial for NoiseLevel
1.0
0.0
−1.0
−2.0
NoiseLevel


factor(RestaurantsPriceRange2)
1e+08
0e+00
−1e+08
1.0
1.5
2.0
2.5
3.0
3.5
4.0
RestaurantsPriceRange2
casual
dressy


partial for RestaurantsAttire
1e+08
0e+00
−1e+08
RestaurantsAttire


no
outdoor
yes
partial for Smoking
−0.5
0.0
0.5
1.0
1.5
Smoking
no
yes
partial for RestaurantsReservations
−0.2
0.0
0.2
0.4
0.6
0.8
RestaurantsReservations





The smoothed terms `latitude` and `longitude` are fairly linear here, which may indicate that we do not need to perform smooth on them. However, the interaction term of smoothed latitude and longitude is usually to account for spatial-autocorrelation, so we will keep them as they are. The other two variables `avg_word_count` and `review_count` show some degree of nonlinearity. We should keep the smoothing functions on them.

Building state-dependent GAM models

Since there seem to be a slightly higher percentage of good restaurantes in Arizona and Ohio, we try to build a different GAM model for each of these two states in the following, and examine if they perform better than

the general model mod.gam1 above.

```
# split datasets according to their states
y.train = split(train, train$state)
y.test = split(test, test$state)

summary(test$state)
```

```
##  AZ  IL  NC  NV  OH  PA  SC  WI
## 114   5  37  68  40  47   2  15
```

```
summary(train$state)
```

```
##  AZ  IL  NC  NV  OH  PA  SC  WI
## 219   8  76 153  94  71   4  45
```

(1) Building a model for Arizona only

```
mod.gam.az <- gam(rating ~ s(latitude) * s(longitude) + s(avg_word_count) + s(review_count)
                  + cuisine + WheelchairAccessible + WiFi + BusinessAcceptsCreditCards
                  + Alcohol + NoiseLevel + Smoking + RestaurantsReservations
                  + OutdoorSeating + GoodForKids,
                data = y.train$AZ, family=binomial(link = "logit"))

pred.gam.az <- round(predict(mod.gam.az, newdata=y.test$AZ, type="response"))

# Model performance using this state specific model
cat('Confusion matrix using pred.gam.az:\n')
```

```
## Confusion matrix using pred.gam.az:
```

```
confmat(y.test$AZ$rating, pred.gam.az)
```

```
##       predicted
## actual         0         1       Sum
##      0 0.5283019 0.4716981 1.0000000
##      1 0.2622951 0.7377049 1.0000000
```

```
cat('Overall accuracy using pred.gam.az:\n')
```

```
## Overall accuracy using pred.gam.az:
```

```
mean(y.test$AZ$rating == pred.gam.az)
```

```
## [1] 0.6403509
```

```
# Compare to results obtained using state-unspecific model mod.gam2
cat('Confusion matrix using mod.gam1:\n')
```

```
## Confusion matrix using mod.gam1:
```

```
confmat(test$rating[test$state == 'AZ'], pred.gam1[test$state == 'AZ'])
```

```
##       predicted
## actual         0         1       Sum
##      0 0.4339623 0.5660377 1.0000000
##      1 0.2131148 0.7868852 1.0000000
```

```
cat('Overall accuracy using mod.gam1:\n')
```

```
## Overall accuracy using mod.gam1:
```

```
mean(test$rating[test$state == 'AZ'] == pred.gam1[test$state == 'AZ'])
```

```
## [1] 0.622807
```

(2) Building a model for Ohio only

```
mod.gam.oh <- gam(rating ~ s(latitude) * s(longitude) + s(avg_word_count) + s(review_count)
                  + cuisine + WheelchairAccessible + WiFi + Alcohol
                  + factor(RestaurantsPriceRange2) + RestaurantsReservations
                  + OutdoorSeating + GoodForKids,
                data = y.train$OH, family=binomial(link = "logit"))

pred.gam.oh <- round(predict(mod.gam.oh, newdata=y.test$OH, type="response"))

# Model performance using this state specific model
cat('Confusion matrix using pred.gam.oh:')
```

```
## Confusion matrix using pred.gam.oh:
```

```
confmat(y.test$OH$rating, pred.gam.oh)
```

```
##       predicted
## actual         0         1       Sum
##      0 0.7058824 0.2941176 1.0000000
##      1 0.3478261 0.6521739 1.0000000
```

```
cat('Overall accuracy using pred.gam.oh:')
```

```
## Overall accuracy using pred.gam.oh:
```

```
mean(y.test$OH$rating == pred.gam.oh)
```

```
## [1] 0.675
```

```
# Compare to results obtained using state-unspecific model mod.gam2
cat('Confusion matrix using mod.gam1:')
```

```
## Confusion matrix using mod.gam1:
```

```
confmat(test$rating[test$state == 'OH'], pred.gam1[test$state == 'OH'])
```

```
##       predicted
## actual         0         1       Sum
##      0 0.5294118 0.4705882 1.0000000
##      1 0.2608696 0.7391304 1.0000000
```

```
cat('Overall accuracy using mod.gam1:')
```

```
## Overall accuracy using mod.gam1:
```

```
mean(test$rating[test$state == 'OH'] == pred.gam1[test$state == 'OH'])
```

```
## [1] 0.65
```

The two attemps above show that building a state-specific model does help to improve the accuracy slightly. However, as this is not required by the question, we will stick with mod.gam1 for the rest of the analysis.

(b) For your model in part (a), summarize the predictive ability by computing a misclassification rate. [10 points]

Answer

```
cat('Confusion matrix\n')
```

```
## Confusion matrix
```

```
confmat(test$rating, pred.gam1)
```

```
##       predicted
## actual         0         1       Sum
##      0 0.5136986 0.4863014 1.0000000
##      1 0.3131868 0.6868132 1.0000000
```

```
cat('Overall accuracy\n')
```

```
## Overall accuracy
```

```
mean(test$rating == pred.gam1)
```

```
## [1] 0.6097561
```

```
cat('Overall missclassification rate\n')
```

```
## Overall missclassification rate
```

```
1-mean(test$rating == pred.gam1)
```

```
## [1] 0.3902439
```

The overall misclassification rate is 0.390. The misclassification rate for 'good' restaurants is 0.313, the misclassification rate for 'bad' restaurants is 0.486.

(c) Consider a version of model (a) that does not include the `cuisine` predictor variable. Explain briefly how you would test in your model whether `cuisine` is an important predictor of the probability of a good restaurant rating. Perform the test, and explain your conclusions. [15 points]

Answer

```
mod.gam2 <- gam(rating ~ s(latitude) * s(longitude) + s(avg_word_count) + s(review_count)
                + state + WheelchairAccessible + WiFi + BusinessAcceptsCreditCards
                + Alcohol + NoiseLevel + factor(RestaurantsPriceRange2)
                + RestaurantsAttire + Smoking + RestaurantsReservations
                + OutdoorSeating + GoodForKids,
               data = train, family=binomial(link = "logit"))
pred.gam2 <- round(predict(mod.gam2, newdata=test, type="response"))

cat('Confusion matrix\n')
```

```
## Confusion matrix
```

```
confmat(test$rating, pred.gam2)
```

```
##       predicted
## actual         0         1       Sum
##      0 0.4863014 0.5136986 1.0000000
##      1 0.3131868 0.6868132 1.0000000
```

```r
cat('Overall accuracy\n')
```

```
## Overall accuracy
```

```r
mean(test$rating == pred.gam2)
```

```
## [1] 0.597561
```

To test if `cuisine` is an important predictor of the probability of a good restaurant, we compare the two models each excluding (mod.gam2) and including of the `cuisine` predictor (mod.gam1) using the analysis of variance (ANOVA) test. The null hypothesis is mod.gam2, and the alternative hypothesis is mod.gam1. If the resultant p-value is lower than the significance level of 0.05, we rejet the null hypothesis and conclude that the model inclusive of the `cuisine` predictor (mod.gam1) is better; if the p-value is higher than the significance level of 0.05, then there is not enough evidence to reject the null hypothesis, and we conclude that the model exclusive of the `cuisine` predictor (mod.gam2) is better.

```r
anova(mod.gam2, mod.gam1, test = 'Chi')
```

```
## Analysis of Deviance Table
##
## Model 1: rating ~ s(latitude) * s(longitude) + s(avg_word_count) + s(review_count) +
##     state + WheelchairAccessible + WiFi + BusinessAcceptsCreditCards +
##     Alcohol + NoiseLevel + factor(RestaurantsPriceRange2) + RestaurantsAttire +
##     Smoking + RestaurantsReservations + OutdoorSeating + GoodForKids
## Model 2: rating ~ s(latitude) * s(longitude) + s(avg_word_count) + s(review_count) +
##     state + cuisine + WheelchairAccessible + WiFi + BusinessAcceptsCreditCards +
##     Alcohol + NoiseLevel + factor(RestaurantsPriceRange2) + RestaurantsAttire +
##     Smoking + RestaurantsReservations + OutdoorSeating + GoodForKids
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       626      786.7
## 2       616      777.7 10   8.9972   0.5324
```

The p-value of $0.5324 >> 0.05$ is too large, we accept the null hypothesis and conclude that the model without cuisine (mod.gam2) is good enough. Hence, cuisine is not an important predictor of a restaurant's rating.