Write Up

Jessica Yu and Kenny Chen (Group C)

12/8/2019

Introduction

According to the Centers for Disease Control, about 48 million people in the United States will get food poisoning each year. While many people will look at websites like Yelp to look at restaurant reviews, seldom look for information regarding a restaurant's cleanliness or health code violations. With the ever rising costs of health care, can we really afford to take chances and dine at restaurants that could possibly put our lives at risk?

In our project, we aim to combine data regarding restaurant health inspection violations alongside reviews data to see if there are any benefits to seeing both types of data at once. We ultimately created a shiny app to help display the two datasets together in hopes of helping consumers make better decisions of where to eat.

Data

Our restaurant reviews comes from the Yelp Open Dataset, which contains almost 7 million reviews on ~200,000 restaurants across the United States. While there are different files within the Yelp Dataset, we used the business and review files. These can be downloaded from https://www.yelp.com/dataset/challenge. Warning: these files are quite large.

The Yelp dataset contains data for restaurants all across the U.S; we limited our focus to just Las Vegas, NV due to the high volumes of visitors and overall popularity. From there, we obtained Las Vegas restaurant inspections data which is available publicly through https://opendataportal-lasvegas.opendata.arcgis.com/datasets/restaurant-inspections-open-data . We accessed this data on November 26th 2019 but it is updated weekly.

```
allBusinesses <- stream_in(file("business.json"))
allReviews <- stream_in(file("review.json"))

allInspections <- read_csv("Restaurant_Inspections_Open_Data.csv")
```

After reading our files, we can proceed with filtering and cleaning. This entails limiting the Yelp data to only Las Vegas restaurants, as well as removing extraneous characters from names and addresses in both datasets. We also chose to convert names and addresses to lowercase to establishing consistency and hopefully make the matching process simpler.

The inspection data was also modified to fix errors in the longitude variable and include only the most recent inspection information.

```
# get restaurants
index <- grep("Restaurants", allBusinesses$categories)
allRestaurants <- allBusinesses[index,]
# get las vegas
vegasRestaurants <- allRestaurants %>%
  filter(city == "Las Vegas")
#write a file of just vegas restaurants
saveRDS(vegasRestaurants,"VegasRestaurants.Rds")

#cleaning vegas restaurants
cleanVegasRestaurants <- vegasRestaurants %>%
```

```
  filter(is_open == 1) %>%
  mutate(name = tolower(name),
         name = gsub("[&\\+]","and", name),
         name = gsub("[,-\\/]"," ", name),
         name = gsub("['\\.\\(\\)\\|]","", name),
         address = tolower(address),
         address = gsub("[,\\.]","",address)) %>%
  select(-attributes, -hours)
#removing attributes because it is a dataframe within a list


#cleaning vegas inspections
cleanInspections <- allInspections %>%
  mutate(Restaurant_Name = tolower(Restaurant_Name),
         Restaurant_Name = gsub("[&\\+]","and",Restaurant_Name),
         Restaurant_Name = gsub("[,-\\/]"," ", Restaurant_Name),
         Restaurant_Name = gsub("['\\.\\(\\)]", "", Restaurant_Name),
         Restaurant_Name = gsub("@","at", Restaurant_Name),
         Address = tolower(Address),
         Address = gsub("-"," ", Address)) %>%
  separate(Location_1, c("iLatitude","iLongitude"), sep=",") %>%
  mutate(iLatitude = as.numeric(gsub("\\(","", iLatitude)),
         iLongitude = as.numeric(gsub("\\)","", iLongitude)),
         iLongitude = if(iLongitude > 0){ iLongitude*(-1)})

# getting the most recent inspections
currentInspections <- cleanInspections %>%
  group_by(Restaurant_Name, Address) %>%
  arrange(desc(Inspection_Time)) %>%
  top_n(1)
```

Once the data were cleaned, we attempted to join the two datasets together through a common variable such as name or address. However, we ran into a few issues. Many restaurants's names and addresses did not match exactly with one another.

For example, the Yelp dataset has a restaurant as "Cleaver - Butchered Meats, Seafood & Cocktails", while the Inspections dataset has the same place as " CLEAVER RESTAURANT". Similar scenarios surfaced for address as well. Surprisingly, the latitudes and longitudes had descrepancies too, making the joining process much more complicated.

However, through the use of the Fuzzyjoin package, we were able to create a function that matched restaurants together using the smallest distances between strings. To further ensure accurate matches, we created an additional function to see if the latitude and longitude deviated by a difference of less than 0.00005

```
#checks if the string distance between two names is <5
match_fun_stringdist <- function(v1, v2) {
  dists <- stringdist(v1, v2, method = "osa")
  ret <- dplyr::data_frame(include = (dists <= 5))
  ret[["distance_col"]] <- dists
  ret
}
# checks if the coordinate difference is <0.00005
match_location <- function(l1,l2) {
   diff <- abs(l1 - l2)
  ret <- dplyr::data_frame(include = (diff < 0.0005))
```

```
  ret[["diff"]] <- diff
  ret
}
```

With our functions created, we can now use fuzzy_inner_join to match the restaurants from Yelp to the corresponding ones in the inspections dataset using our own criteria.

```
#using fuzzyjoin to match restaurants with inspections
matches <- fuzzy_inner_join(cleanVegasRestaurants, currentInspections,
                by = list(x = c("name",
                                "address",
                                "latitude",
                                "longitude"),
                          y = c("Restaurant_Name",
                                "Address",
                                "iLatitude",
                                "iLongitude")),
                match_fun = list(match_fun_stringdist,
                                 match_fun_stringdist,
                                 match_location,
                                 match_location))

# csv file of businesses along with inspections
write.csv(matches, "matches.csv")
```

After joining the two datasets we got 475 matches; after manually checking the matches, we removed only 8 erroneous matches.

With these matches, we can finally add the reviews from Yelp. Since the newly created matches dataset has the buisness ID variable from Yelp, we can easily left join the reviews onto the matches.

```
cleanedMatches <- read_csv("cleanedMatches.csv")

matches_reviews <- cleanedMatches %>%
  left_join(allReviews, by = "business_id")
```

In our final step, we updated the dataset to contain each restaurant's original name, since it was decapitalized earlier in the initial data cleaning process.

```
#getting back the original names and address
matches_reviews_cleaned <- matches_reviews %>%
  left_join(vegasRestaurants, by = "business_id") %>%
  select(-attributes, -hours)

saveRDS(matches_reviews_cleaned, "matches_reviews_fixed.Rds")
```

Moving forward, we will be using the "matches_reviews_fixed.Rds" file as our dataset!

Sentiment Analysis and Text Mining

Now that we have our desired dataset, we can move forward and generate some insights from the actual reviews themselves using sentiment analysis. While it is hard and time consuming to pour over hundreds of reviews to get a general consensus, we can perhaps use data science to quickly summarize the top words and get a quick picture of what "vibes" a restaurant has.

From the tidytext package, we decided to use "bing" sentiment lexicon to match our reviews to. The bing

lexicon contains a set of English words that are either labeled positive or negative. We can go through our reviews to see which words will match to the ones in bing and obtain the top 3 most commonly used positive and negative words per restaurant.

```
library(tidytext)
bing <- get_sentiments("bing")

#load dataset
sentiment <- readRDS("matches_reviews_fixed.Rds")

head(sentiment %>%
       select(Location_Name,text))
```

```
## # A tibble: 6 x 2
##   Location_Name      text
##   <chr>              <chr>
## 1 Roberto's Taco Sh~ I've been eating here for 4 years and everything I've eate~
## 2 Roberto's Taco Sh~ "I like Roberto's, been to several other location, this on~
## 3 Roberto's Taco Sh~ "I give this Roberto's a 3 star. I love the food, but the ~
## 4 Roberto's Taco Sh~ "I dont know why they cant clean the place up a little bit~
## 5 Roberto's Taco Sh~ "Price: $9.00 for 3 tacos.\n\nFood Quality: Tortillas are ~
## 6 Roberto's Taco Sh~ Mess load of Nachos(carne asada) came jam packed with lots~
```

Taking a quick look at our dataset, we can see that there are multiple reviews for each restaurant. To make the analysis easier, we can collapse all the reviews of one restaurant into a single observation that we can then work with.

```
sentimentsCollapsed <- sentiment%>%
select(business_id,text)%>%
group_by(business_id)%>%
summarize(text=paste(text,collapse=" // "))
```

Once we have the collapsed reviews, we can create a function that will obtain the top 3 most frequently used positive and negative words within a restaurant's reviews.

```
goodBad <- function(text) {
words <- tibble(text) %>%
  unnest_tokens(word, text)
positive <- words %>%
  count(word, sort = T) %>%
  inner_join(bing) %>%
  filter(sentiment == "positive") %>%
  head(3)
negative <- words %>%
  count(word, sort = T) %>%
  inner_join(bing) %>%
  filter(sentiment == "negative") %>%
  head(3)
c <- rbind(positive, negative)
c
}
```

Here's an example of our function running on a single restaurant. We can see here that the most positive words "good", "great", and "delicious" appear much more frequently than the negative words "bad", "disappointed", and "cold".

```
sentimentsCollapsedExcerpt <- sentimentsCollapsed %>%
  head(1)

sentimentsCollapsedExcerpt %>%
  group_by(business_id) %>%
  do(goodBad(sentimentsCollapsedExcerpt$text))
```

```
## # A tibble: 6 x 4
## # Groups:   business_id [1]
##   business_id            word            n sentiment
##   <chr>                  <chr>       <int> <chr>
## 1 _0x7W6fizaPP76xNBxBLAQ good          200 positive
## 2 _0x7W6fizaPP76xNBxBLAQ great         122 positive
## 3 _0x7W6fizaPP76xNBxBLAQ delicious     106 positive
## 4 _0x7W6fizaPP76xNBxBLAQ bad            20 negative
## 5 _0x7W6fizaPP76xNBxBLAQ disappointed   19 negative
## 6 _0x7W6fizaPP76xNBxBLAQ cold           14 negative
```

In order to further extend the visualization of our reviews data, we can also create word clouds to highlight the most frequently used words in a chunk of text. Again, this can help us get a quick summary of what a restaurant's image is like (and if there is anything that could be concerning).

Using the wordcloud package, we can generate the top 30 most frequently used words for each restaurant.

```
library(wordcloud)
wordcloud(sentimentsCollapsedExcerpt$text, max.words = 30, scale = c(10, .1),
colors = topo.colors(n = 30), random.color = TRUE)
```

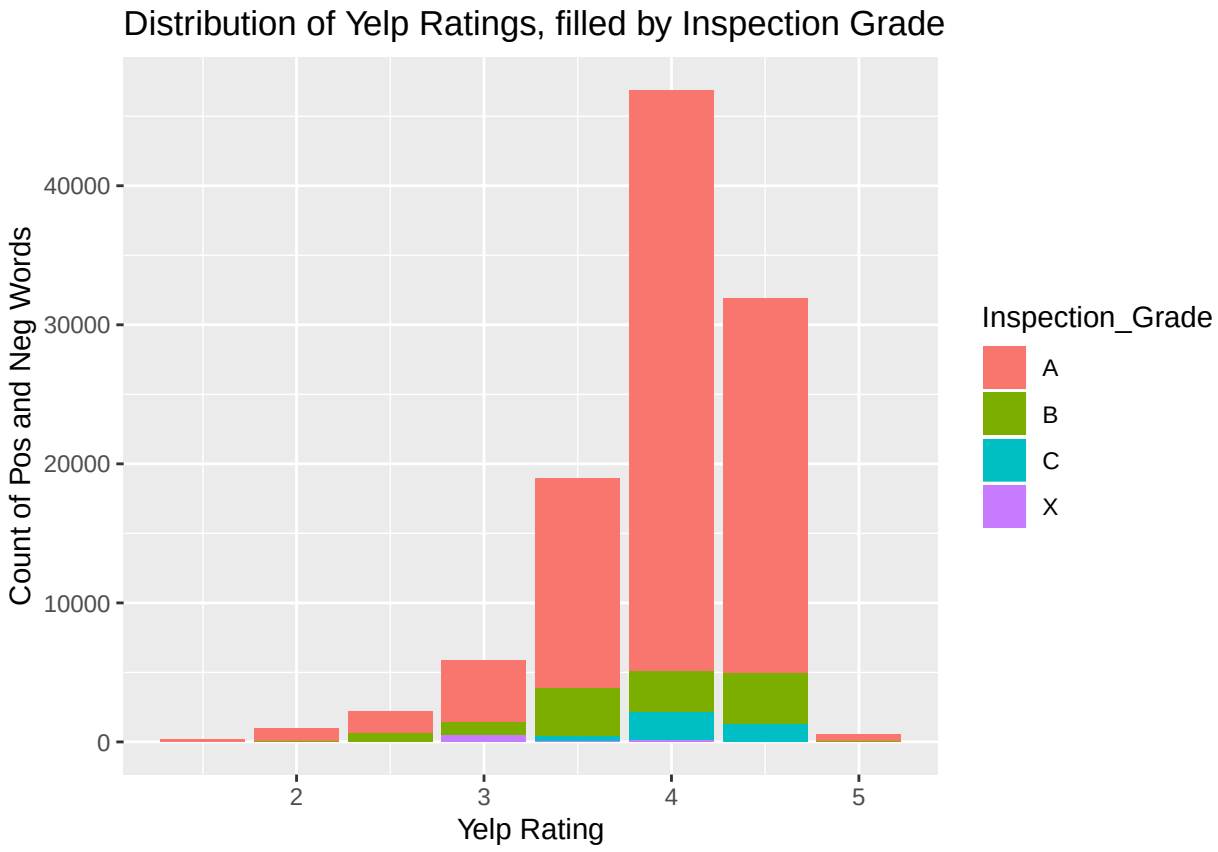
Shiny App and Insights

We created a shiny app to make our data interactive and navigable for users. It can be accessed here: https://stat231-groupc.shinyapps.io/Final/

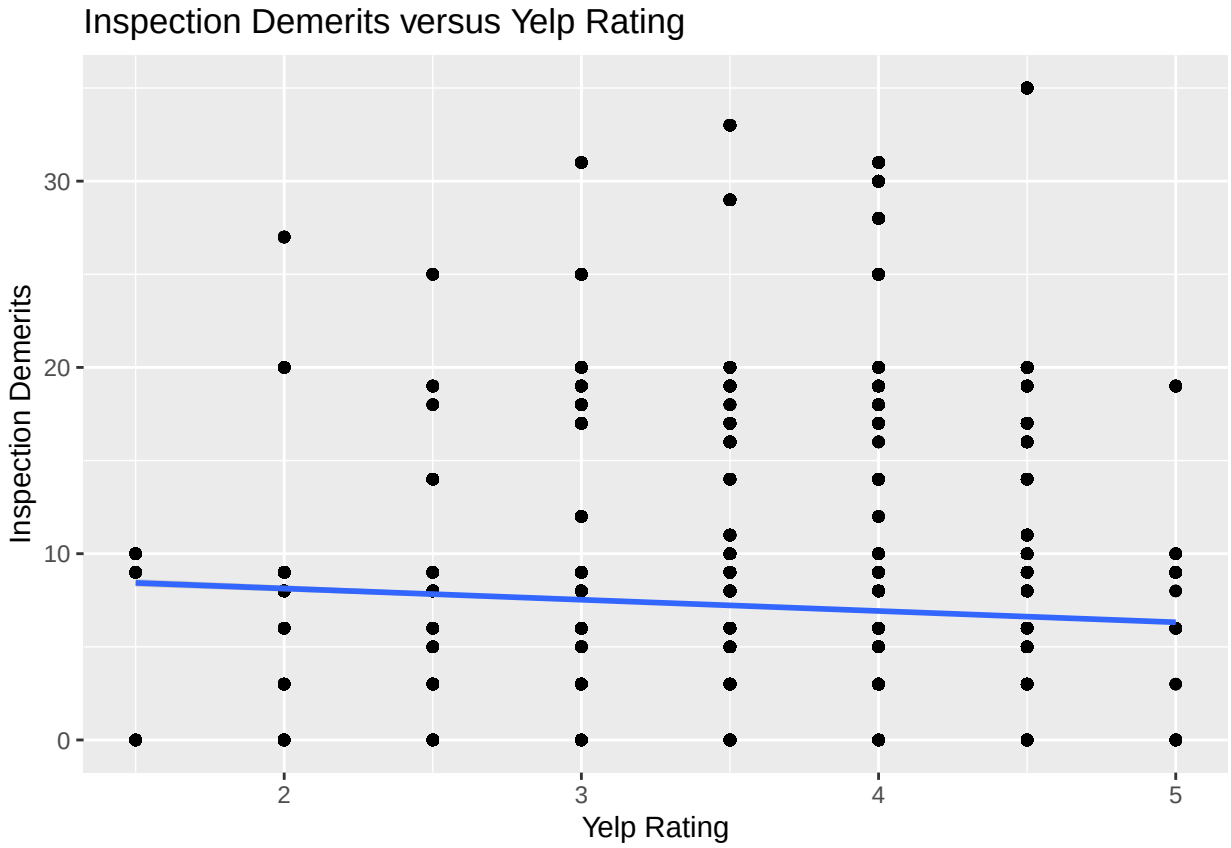
We started with a leaflet plot of all the restaurants from our data set, and allow for users to filter for specific criteria according to their preferences. Markers in blue represent restaurants with an 'A' rating, green for 'B', and red for 'C'. Once a restaurant is clicked, a word cloud alongside a bar graph with the top 3 negative and positive words are displayed. The average amount of stars, sanitation grade, and inspection demerits are all displayed as well as a sample of 5 reviews. All of these widgets allow the user to quickly explore a restaurant while also seeing information from the most current inspection.

After running some univariate and bivariate analysis, we were able to deduce a few observations. First, a majority of the ratings are around 3-4 stars, which is generally quite positive. A majority of the inspection grades were also an 'A', which perhaps suggests the possibility that the inspection grades are not as compelling as we originally thought. Lastly, the most surprising conclusion is that there is no strong negative correlation between yelp ratings and inspection demerits. This suggests that perhaps the cleanliness of a restaurant is not entirely reflected in reviews, which further supports our idea that both types of data are important for making decisions.

```
#more overall observations
ggplot(sentiment, aes(x = stars, fill = Inspection_Grade)) + geom_bar() + xlab("Yelp Rating") + ylab("C
```




```
ggplot(sentiment, aes(x = stars, y = Inspection_Demerits)) + geom_point() + geom_smooth(method = "lm") +
```



Limitations and Future Possibilities

Just because a restaurant is highly rated doesn't necessarily mean it's clean, and just because a restaurant recieves an A rating doesn't mean it's going to be a good restaurant. Both data are important, but can be so much more powerful when combined.

What we have shown through our analysis and shiny app is just the tip of the iceberg; the potential for new insights and information when combining sanitation inspections data with reviews is something that should be explored, especially by companies like Yelp that have such a lot of influence in our society today.

Our journey is not flawless; there are definetly some limitations to our project that hold us back from portraying the most accurate version of a restaurant. Given more time, we could have examined what specific violations each restaurant inccured which could help us get a better idea of a restaurant's cleanliness. Change over time is also an important factor to consider. We only looked at the current inspection grade but past violations as well as future ones are equally as important to consider.

In the future, it would be benefical to look for specific words pertaining to cleanliness and food safety within the Yelp reviews and see if those could help predict a restaurant's inspection rating. The possibilites are almost endless.

Ultimately, through this project and through data science in general, we can help make sense of the vast amounts of data around us and work towards creating a world of more informed citizens.