Are Chicago-area Chinese restaurants more or less likely to have Wi-Fi than Chicago-area Indian restaurants?

Obinna Olajide Mogbogu

04/16/2015

Contents

1 Introduction **3**

2 Data **4**
2.1 Obtain Data 4
2.2 Scrub Data 7
2.3 Explore Data 9

3 Results: Tables **11**

4 Results: Graphs **13**

5 Conclusion **15**

List of Tables

1 **Chinese Restaurants Ratings** 12
2 **Indian Restaurants Ratings** 12
3 **Average Restaurants Ratings** 12

List of Figures

1 Average Restaurants Ratings Bar Graph 14
2 Average Restaurants Ratings Line Graph 14

1 Introduction

Today, there are about 616,000 restaurants in the United States (US). The city of Chicago, in the state of Illinois, is the third most populated city in the United States after New York City and Los Angeles. There are over 12,000 restaurants in Chicago. Over the years, the need for Wi-Fi services in restaurants by customers have grown. It has been debated in several research study if Wi-Fi services by any means does add to the customer satisfaction experience. Unfortunately, Wi-Fi services have had little or no contribution to the customer satisfaction experience in restaurants; possibly not entirely true. However, many restaurants in the US like Starbucks, and McDonalds continue to offer Wi-Fi services to their customers; possibly to enhance their customer satisfaction experience or simply for other reasons that are in the best interest of the business itself. Yet it is likely that some customers for example students, are likely to rate a restaurants high because it provides Wi-Fi services 'would you not?'. Again it is not always the case that customers will rate a restaurant high solely because of the provision of Wi-Fi services. Nonetheless, for the purpose of this tutorial, our OSEMN project will assume that a restaurant is only likely to have Wi-Fi services if it has a high customer rating. In order to know the customer ratings of restaurants we shall need to have the data of the restaurants; one of we can get hold of such data is by accessing local businesses via an API data source called Yelp.

In this report we shall access the data of two Chicago-area restaurants- Indian and Chinese- to determine which one of them is likely to have Wi-Fi services. We shall base our findings on the customer ratings for the restaurants in both of the Chicago-area restaurants. All the data for this report is retreived using the Yelp API. Although, non of the data collected via the Yelp API have any data on the Wi-Fi availability for a given restaurant, which could have made the task a lot easier in determining which of the restaurants will have Wi-Fi services or not. Also, some of the data record retrieved via the Yelp API consist of restaurants which are no longer active or owned by a business owner. Obviously, we do not want to base our report on any of such restaurants, therefore, we will learn few techniques in this tutorial on how to eliminate such records. The tutorial in our OSEMN project report, consist of steps required to learn how to obtain the data from the Yelp API data source, and scrubbing or cleaning the data that was retrieved via the Yelp API data source so that it can be displayed in a table and graph format. Let us begin. [1]

[1] The complete code for the 'Data' section tutorial can be found in '497projectRepo/OsemnProject/Data' directory as **indianGathering.R and chineseGathering.R** files. The CSV files created are located in the '497projectRepo/OsemnProject/Data/-file.CSV' directory as **Chinese.csv and Indian.csv** files. Also the complete R code related to the 'Scrub Data' section of this tutorial is located in the '497projectRepo/OsemnProject/Analysis' dir as **plot1.R** file.

2 Data

2.1 Obtain Data

There are ten steps required in order to obtain data from the Yelp API. These steps should be followed in the order outlined below[2]. Please note that The steps outlined in this section, are only guidelines on how to obtain data for Chicago-area Indian restaurant category[3].

Step One

A snippet of the R code used throughout this tutorial are shown in a grey box like this

```
require(httr)
require(httpuv)
require(jsonlite)

consumerKey ← "jgqw_ZH9fjPoZmvvqWcNaA"
consumerSecret ← "nxkGdieAY1LN8Z6KeZukrZRsfjg"
token ← "4gMVscnxTU30_tf8xs0YcRN3McHff5oc"
token_secret ← "5qF1VnhF10CFh_zPdx_6ga13aOA"
```

In step one we begin by obtaining the data we need via the Yelp API data source using the token/secret keys that will be avalilable to you after you have signed up[4].

Step Two

```
myapp = oauth_app("YELP", key=consumerKey, secret=consumerSecret)
sig=sign_oauth1.0(myapp, token=token,token_secret=token_secret)
```

In step two, you will use your token/secret key to authenticate an API requests[5].

Step Three

The # signs in the code will be used throughout this tutorial to show what the Chinese restaurant variable will be like

```
#For the Chinese version, replace
#indian%restaurant -- chinese%restaurant

URL.API ← "http://api.yelp.com/v2/search/?"
limit ← 20
location ← "Chicago"
term ← "indian%restaurant"
```

In step three, we begin by specifying the search parameters that we shall use to make our API request. The Yelp API uses the **/v2/search** command; a 'GET' request command used for searching local businesses. Other search parameters such as term, location, limit will be specified in order to give us a more refined

[2]The Yelp API have access to the data of over 50 million local business, such as restaurants. You will need to sign up and get verified before you could use the Yelp API data source to access data of businesses

[3]Please refer to indianGathering.R and chineseGathering.R in the 'Analysis/Data/fileRgathering' directory for the complete code used in completing the OSEMN project

[4]www.yelp.com/developers

[5]Yelp uses OAuth 1.0a for authenticating API requests as per the OAuth specification

search. The **limit** parameter is used for specifying the number of businesses (in this case, restaurants) results to return. The **location** parameter is used for speficify the neighborhood, address or city of the business. Also, a **term** parameter is specified, which we shall use to make a more in-depth search. For example if we want to search for 'food', we will specify the term to be food; in this case the Yelp API will search for businesses related to food.

Step Four

```
yelpurl ← paste0(URL.API,"limit=",limit,"&location=",location,"&
    term=",term)
```

In step four we combined all our specified search parameters into a single **URL** object, which we shall use to make the request to the Yelp API data source.

Step Five

```
locationdata=GET(yelpurl, sig)
locationdataContent = content(locationdata)
```

In step five, we use the URL specified in step four, to query the Yelp API data source. The **GET** Yelp API command, is used along with the token/secret keys to authenticate the URL request we had specified. The **content** command simply holds the details of all the returned data. The data type for the 'locationdataContent' object is a 'list'. We shall convert this object to a **JSON** object in the next step.

Step Six

```
jsonLocationdataContent ← toJSON(locationdataContent)
locationdataList=fromJSON(jsonLocationdataContent)
df ← data.frame(locationdataList)
```

In step six, we used the **toJSON** command to convert our 'locationdataContent' object to a JSON object. After that, we then use the **fromJSON** command to convert our new JSON object back to a list object. The object 'df' will be used to store a dataframe object of our new list object. At this point, we have managed to retrieve the entire dataset using the search parameter that we specified in step 3 and 4. The result of this step produces a dataframe that contains multiple column headers. In the next step we shall see how to select the required columns that we need for the OSEMN project.

Step Seven

```
[1] businessRating ← df[,"businesses.rating"]
[2] businessName ← df[,"businesses.name"]
[3] businessReviewCount ← df[,"businesses.review_count"]
[4] businessLocation ← df[,"businesses.location"]
[5]  businessCity ← businessLocation[,"city"]
[6]  businessState ← businessLocation[,"state_code"]
```

```
businessCountry ← businessLocation[,"country_code"]
businessIsClosedStatus ← df[,"businesses.is_closed"]
businessIsClaimedStatus ← df[,"businesses.is_claimed"]
```

In step seven, line 1 through 9 uses the dataset stored in the 'df' dataframe to select the column headers that we need. In line 4 through we used the businessLocation object to select the city, state, and country.

Step Eight

```
library(jsonlite)
businessRatingjson ← toJSON(businessRating, pretty=TRUE)
businessNamejson ← toJSON(businessName, pretty=TRUE)
businessReviewCountjson ← toJSON(businessReviewCount, pretty=TRUE)
businessCityjson ← toJSON(businessCity, pretty=TRUE)
businessStatejson ← toJSON(businessState, pretty=TRUE)
businessCountryjson ← toJSON(businessCountry, pretty=TRUE)
businessIsClosedStatusjson ← toJSON(businessIsClosedStatus, pretty
    =TRUE)
businessIsClaimedStatusjson ← toJSON(businessIsClaimedStatus,
    pretty=TRUE)
```

In step eight, we shall convert the list objects that we created in the previous step to JSON objects. In order to achieve this, we will use the **toJSON** command again with the argument *pretty* set to TRUE.

Step Nine

```
Ratings← fromJSON(businessRatingjson)
Name ← fromJSON(businessNamejson)
Reviews ← fromJSON(businessReviewCountjson)
City ← fromJSON(businessCityjson)
State ← fromJSON(businessStatejson)
Country ← fromJSON(businessCountryjson)
isClosed ← fromJSON(businessIsClosedStatusjson)
isClaimed ← fromJSON(businessIsClaimedStatusjson)
newCSV.data ← data.frame(Name, Reviews, Ratings, City, State,
    Country, isClosed, isClaimed)
```

In step nine, just as we have done in the previous step but now differently, we shall convert each JSON object that was created to a 'matrix' object. Line 1 through 8 converts each JSON object to a matrix oject. In Line 9 we used all the newly created matrix objects to create a new dataframe object called 'newCSV.data'. In the next step we shall use the dataframe object to create a **CSV** file.

Step Ten

```
#For the Chinese version, replace
#Indian.csv -- Chinese.csv

write.csv(newCSV.data, file="Indian.csv")
```

A CSV file or a Comma Seperated File can be used for storing our dataset in a tables document. Storing your data in a CSV format is good as it works well with R and can make your project more reproducible. In order to create and store data as a CSV file, we will need a dataframe (newCSV.data) object and a name (Indian.csv) for which we would want to save the CSV file[6].

This concludes the procedures required to obtain the data from the Yelp API data source. Please make sure that you follow this step in the order given in order to obtain the data that you need.

2.2 Scrub Data

This section consists of six steps, which you will need in order to clean the data that was retreived from the Yelp API data source. Remember our data was stored in a CSV file called Indian.csv and Chinese.csv for the Chinese version. Please note that depending on the state of your data, the levels of cleaning mentioned here may or may not apply to data retrieved from other data source. The steps here are only intended for cleaning data retrieved from the Yelp API data source[7].

Step One

```
#For the Chinese version, replace
#Indian.csv -- Chinese.csv

myIndianData ← read.csv("Data/file_CSV/Indian.csv")
```

Before we can clean our data, we need to have it available. From the previous section- Obtain Data- we have been able to obtain the data from the Yelp API and saved the data in a CSV file called Indian.csv. In this step, we will use the **read.csv** command, to load the CSV file into R.

Step Two

```
#For the Chinese version, replace
#myIndianData with my myChineseData

myIndianData$Name ← as.character(myIndianData$Name)
myIndianData$City ← as.character(myIndianData$City)
myIndianData$State ← as.character(myIndianData$State)
myIndianData$Country ← as.character(myIndianData$Country)
myIndianData$isClosed ← as.character(myIndianData$isClosed)
myIndianData$isClaimed ← as.character(myIndianData$isClaimed)
myIndianData$Ratings ← as.numeric(myIndianData$Ratings)
myIndianData$Reviews ← as.numeric(myIndianData$Reviews)
```

[6]Please refer to Indian.csv and Chinese.csv file in the 'Analysis/Data/file.CSV' directory.

[7]For more on data cleaning, please read Chapter 7 of the text: 'Reproducible Research with R and RStudio' by Christopher Gandrud

Now that we have our CSV file loaded in R, what we need to do next is to format our column headers. Sometimes your column headers may not be formatted properly. In such cases, any statistical anlaysis performed on the columns may lead to the wrong result. Thus, in step two, we want to ensure that our data set's columns have the right data types. We used the **as.character** command to convert our data sets columns- Name, City, State, Country, isClosed and isClaimed- to character data types. Likewise, we used the **as.numeric** command to convert our data sets columns- Ratings, and Reviews- to numerical data types.

Step Three

```
#For the Chinese version, replace
#myIndianData -- myChineseData
#myIndianData_City.gr -- myChineseData_City.gr
#myIndianData_isClosed.gr -- myChineseData_isClosed.gr
#myIndianData_isClaimed.gr -- myChineseData_isClaimed.gr

[1] myIndianData_City.gr ← myIndianData[grep("Chicago",
    myIndianData$City),]
[2] myIndianData_isClosed.gr ← myIndianData_City.gr[grep("FALSE",
    myIndianData_City.gr$isClosed),]
[3] myIndianData_isClaimed.gr ← myIndianData_isClosed.gr[grep("
    TRUE", myIndianData_isClosed.gr$isClaimed),]
```

In step three, we continued in our cleaning/scrubbing process by specifying some criterial that we need in order to select the rows/records from the data set. We used the **grep** command, to achieve this task. In the first line, we used the grep command to select restaurants rows that only have 'Chicago' as its city. In the second line, we used the grep command to select restaurants that are still opened/active; therefore 'isClosed equals to FALSE'. In the third line, we used the grep command to select only restaurants that have a business owner; therefore 'isClaimed equals to TRUE'. The final object from this performing the steps in this section, will create a dataframe object called **myIndianData.isClaimed.gr**, which will be a record of Chicago-area Indian restaurants that are currenlty active and owned by a business owner

Step Four

```
#For the Chinese version, replace
#myIndian.subset -- myChinese.subset
#(myIndianData_isClaimed.gr -- myChineseData_isClaimed.gr

myIndian.subset ← subset(myIndianData_isClaimed.gr, Reviews ≥ 100
    & Reviews ≤ 899)
```

This step could be omitted but please use it in order to follow up with the procedures in the OSEMN project that is been discussed in this tutorial

In step four we specified a logical criterial, which even helps us to refine the restaurants records that we want. We used the *Review* column of our 'myIndianData.isClaimed.gr' dataframe object to specify a criterial. The sole purpose of doing this, is to make sure our OSEMN project is not unbiased in any way. It will be unfair, to have restaurants with very low or very high reviews alongside

other restaurants that do not match up equally, respectively. Therefore, we specifed a range of dataset that can help eliminate the possibility of such from happening.

Step Five

```
#For the Chinese version, replace
#myIndian.subset -- myChinese.subset
#myIndian.selection -- myChinese.selection

myIndian.selection ← data.frame(myIndian.subset$Name,
    myIndian.subset$Ratings)
```

In this step, we select the column that we want to work with and create a new dataframe object called **myIndian.selection**. This time, we only need the 'Name' and 'Ratings' column from our dataframe object **myIndian.subset**.

Step Six

```
#For the Chinese version, replace
#myIndian.New -- myChinese.New
#myIndian.selection -- myChinese.selection
#myIndian.subset.Name -- myChinese.subset.Name
#myIndian.subset.Ratings -- myChinese.subset.Ratings

[1] library(plyr)
[2] myIndian.New ← rename(myIndian.selection, replace=c("
    myIndian.subset.Name"="Name", "myIndian.subset.Ratings"="
    Ratings"))
[3] myIndian.New$Name ← as.character(myIndian.New$Name)
```

Our final step for the scrubbing procedure, is to create a new dataframe object called **myIndian.New**, which renames our dataframe object's columns 'myIndian.selection.Name' and 'myIndian.selection.Ratings' as 'Name' and 'Ratings' respectively. The third line in this step further re-format one of the column header's data type to 'character' using the **as.character** command.

This concludes the scrubbing procedures for the OSEMN project. Now is a good time to grab a cup of coffee if your eyes are begining to fail you

2.3 Explore Data

Use the class, str and summary command to explore the data more.

Exploring with the Class command

```
#For the Chinese version, replace
#myIndian.New -- myChinese.New

> class(myIndian.New$Name)
[1] "character"
> class(myIndian.New$Ratings)
[1] "numeric"
> class(myIndian.New)
[1] "data.frame"
```

The above R code uses the **class** command to show the data type for columns in our dataframe object **myIndian.New**. As we can see, the Name column of our dataframe object is a 'character' data type; the Rating column is a 'numeric' data type and the entire dataframe object itself is as we expect, a dataframe data type.

Exploring with the str command

```
#For the Chinese version, replace
#myIndian.New -- myChinese.New

> str(myIndian.New$Name)
 chr [1:14] "Naansense" "India House Restaurant" "Rangoli" "Tandoor
      Char House" ...
> str(myIndian.New$Ratings)
 num [1:14] 4 4 4 4 3.5 3.5 4 3.5 3.5 3.5 ...
> str(myIndian.New)
'data.frame':  14 obs. of  2 variables:
 $ Name   : chr  "Naansense" "India House Restaurant" "Rangoli" "
     Tandoor Char House" ...
 $ Ratings: num  4 4 4 4 3.5 3.5 4 3.5 3.5 3.5 ...
```

The **str** command helps you see the number of obsevations and columns for any given data set as well as their data type. As we can see both the 'Name and Ratings including the myIndian.New dataframe itself' consist of **14** observations; with myIndian.New dataframe object having two variables or columns-Name and Ratings'[8].

Exploring with the summary command

```
#For the Chinese version, replace
#myIndian.New -- myChinese.New

> summary(myIndian.New$Name)
   Length     Class      Mode
       14 character character
> summary(myIndian.New$Ratings)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    3.5     3.5     3.5     3.8     4.0     4.5
> summary(myIndian.New)
     Name               Ratings
 Length:14          Min.   :3.5
 Class :character   1st Qu.:3.5
 Mode  :character   Median :3.5
                    Mean   :3.8
                    3rd Qu.:4.0
                    Max.   :4.5
```

The **summary** command in R not only allows you to see the data type of the R objects but also their mininum and maximum allowable values as well as other statistical measurements. The minimum and maximum allowable values for our Ratings object in the myIndian.New dataset is 3.5 and 4.5 respectively. The

[8]Please note that, our scrubbing procedures refined our data set and as a result, we ended up from 20 to 14 observations

average or mean value for the Ratings object is 3.8 approximately. Executing the 'summary' command on the myIndian.New dataset will give you statistical summary of the individuaul columns (i.e. Name and Ratings) contained in the dataset.

3 Results: Tables

In this section, we shall be doing more exploring on our dataset. In other words, we will be creating three separate tables for our data sets.

Table 1 shows the names of the 14 Chicago-area Chinese restaurants we selected in this tutorial for the OSEMN project. Likewise, **Table 2** shows the names of the 14 Chicago-area Indian restaurants. For each of these two tables (i.e. Table 1 and Table 2), the average rating was computed and shown in a third table- **Table 3**. One way to determine if a restaurants is likely to have Wi-Fi or not is to look at the average customer ratings. The average customer rating of a restaurant could reveal many things about the reception of the restaurants by customers. If the availability of Wi-Fi in restaurants is part of the reason to high customer ratings in restaurants, then it will make sense to suggest that a restaurant with a high average customer rating should likely have Wi-Fi services[9].

R is a wonderful tool for carrying out statistical measurment on data used in research study. R equips its users with many functions such as sum, mean, mode, median, max, min etc, which can help a researcher compute just about any statistical analysis on his/her data.In this tutorial, we will use the **mean** function to compute the average customer ratings for the Chinese and Indian restaurants displayed in table 1 and table 2 respectively. The syntax of the mean function is

$$R_{numObj} = mean(R_{vectObj})$$

where R_{numObj} is a numeric object of the calculated mean and $R_{vectObj}$ is a vector objecting consisiting of numerical values.

The result of the mean function for each restuarant's customer ratings is displayed in **Table 3: Average Restaurants Rating.** In table 3, you will notice that the average customer ratings for both restaurants are approximately the same. This clearly tells us that both of these restaurants are either likely to have or not have Wi-Fi services. There is no significant difference in average customer ratings between the two restaurants that will suggest that either of the restaurants will likely have Wi-Fi or not.

[9]Sadly, this is not generally true as there are other factors such as good customer care service, more menu options, efficiency of staff and many other things that generally contributes to the average customer rating of restaurants

Table 1: **Chinese Restaurants Ratings**

Name	**Rating**
Lao Sze Chuan-Downtown	3.5
Chi Cafe	4
Shanghai Terrace at The Peninsula Chicago	4
Chengdu Impression	4
Yummy Yummy Asian Cuisine	4
MingHin Cuisine	4
Yum Cha Dim Sum Parlor	3.5
Fat Rice	4
Lao Hunan	3.5
Go 4 Food	4
Wow Bao	3.5
Take Me Out	4
MAK Restaurant	4
Wow Bao	3.5

Table 2: **Indian Restaurants Ratings**

Name	**Rating**
Naansense	4
India House Restaurant	4
Rangoli	4
Tandoor Char House	4
Bombay Wraps	3.5
Jaipur	3.5
Cumin	4
Mughal India	3.5
Gaylord Fine Indian Cuisine	3.5
The Indian Garden	3.5
Chicago Curry House	3.5
Nepal House - Indian and Nepali Cuisine	3.5
Luzzat	4.5
Ghareeb Nawaz Express	3.5

Table 3: **Average Restaurants Ratings**

Category	**Rating (*mean*)**
Indian Restaurant	3.75
Chinese Restaurant	3.82

4 Results: Graphs

In this section we will use the results shown in Table 3 to plot two distinct graphs using the **ggplot2** package in R. Figure 1 displayed above is a bar graph used to graphically show the relationship in rating between the average customer ratings between the Chicago area India restaurants and the Chicago area Chinese restaurants. The figure above, shows that both of these restaurants are likely to or not likely to have Wi-Fi. To some extent, perhaps the customer rating as we had thought had nothing to do with whether either of the restaurants will have Wi-Fi services or not.

2 Figure 2 below is a line graph, showing the same thing as Figure 1 but only differently. Similarly, both figures simply reperesents the average customer ratings of the two category of restaurants discussed throughout this tutorial. Again, we have used the results of the Tables in the previous section, to be able to display the average customer ratings for each of the restaurants- Chinese and Indian. In the conclusion section, I have place a measurement scale to help us elaborate more on the conclusions drawn from both of the figures displayed in this tutorial for the OSEMN project.

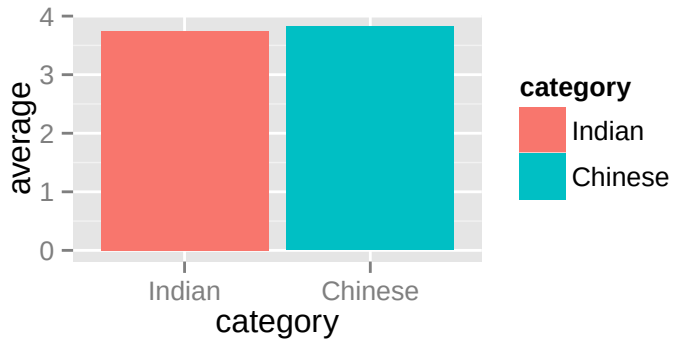

Figure 1: Average Restaurants Ratings Bar Graph

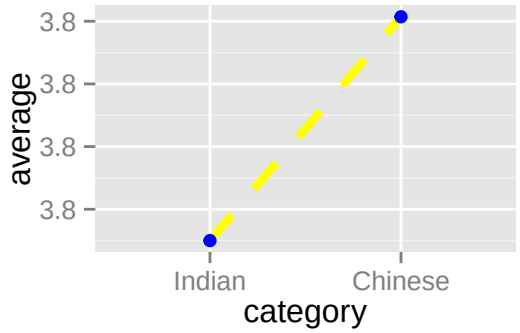

Figure 2: Average Restaurants Ratings Line Graph

5 Conclusion

In this section, we only summarize our results and findings from the two previous section. It is interesting to have found that the average customer ratings for both the Chicago area Indian restaurants and the Chicago are Chinese restaurants are nearly the same. It is for this reason, that making sense of their relationship in respect to whether any will likely have Wi-Fi service has been hard outline. Therefore, I have provided a custom scale (a nominal scale actually), which I have used to place the average ratings of both of this restaurants.

1 - Very unlikely
2 - unlikely
3 - Moderate
4 - likely
5 - Very likely

Using the above scale, we can see that, the average customer ratings of both the Chinese and Indian restaurants (i.e. 3.75 and 3.82) will both match up at **4, which is likely** when estimated. Thus, both restaurants are likely to have Wi-Fi.

This concludes the entire tutorial for our OSEMN project. Thank you!!!