Yelp review sentiment analysis

Big Data for Business Decisions 20564

Group assignment

Group # 11

Christian Gaggiotti 3119139

Pier Francesco Prodani 3004937

Riccardo Milani 3000753

Valeri Mdivani 3101259

Table of Contents

1. Introduction

Nowadays, consumer trends and preferences are changing and since the catering business has continued to grow over the years it requires business to meet the consumer demands for staying afloat. It has become crucial for restaurants and any small business to have a good image on the internet for attracting new clients and for differentiating among many competitors on the market.

Therefore, consumer and businesses today have many online platforms, like Yelp, where people can share their opinions and which business can use for differentiating themselves to the new clientele.

Yelp is one of the largest platforms for sharing crowd-sources business reviews with more than 200 million reviews on its website. It allows users to submit textual reviews and scores on a one to five start rating system. Yelp also allows business owners to interact with its customers, by having separate app for business for responding to and investigating each review. This ability to interact and analyze does give restaurants a great opportunity to showcase their commitment and customer service.

Nevertheless, it's impossible to react to each negative point, since the costs of particular changes often don't justify the benefits, especially for the small local businesses. So, we think that It is going to be a great tool for business owners to have a better understanding of those reviews for identifying cornerstone points, which are crucial for the improvement of consumer satisfaction.

Nevertheless, it's impossible to react to each negative point, since the costs of particular changes often don't justify the benefits, especially for the small local businesses. So, we think that It is going to be a great tool for business owners to have a better understanding of those reviews for identifying cornerstone points, which are crucial for the improvement of consumer satisfaction.

1.1. Research Goals

Therefore, we download a huge and very comprehensive Yelp dataset, which can be easily found in Kaggle at: https://www.kaggle.com/yelp-dataset/yelp-dataset. Our aim is to build some sort of model through which it would be possible to easily state what is going on in a big sample of reviews in terms of main drivers of the ratings. We will do so by carrying out a sentiment analysis on said reviews and basically understanding where those sentiments come from. It will become much clearer throughout the report.

In this way, we will investigate first of all if we can find any difference between the sentiments of the reviews of different categories of restaurants, which would allow us to run an analysis on what kind of service people expect depending on the restaurant they are going to, and how this affects their reviews. This analysis can be useful for a potential entrepreneur which is going to open a restaurant and might ask himself: what should I care more about? Should I invest a lot on the location or is it better if I put all my initial capital into food quality?

Will the answer depend on the type of restaurant? We think so, but we will find out during the analysis.

Also, we can go back to the original purpose and switch the analysis to the single restaurant level. Using the same logic, we will try to find a way to quickly answer to questions like, what is performing well or bad in the restaurant? What do people mostly care about? How can the quality of the restaurant be increased from a customer's perspective?

2. Data collection & database description

We collect the data used for our project from kaggle.com. In particular, we are extracting two datasets, belonging to a group of five datasets, which contain a detailed information on Yelp reviews.

The datasets we use are respectively named “Business”, and “Reviews”.

Within the “Business” Dataset we find several different details on a wide range of Yelp’s partners (ranging from restaurants to lawyers’ offices or mortgages brokers, just to name a few of them). For instance, we can find some main features of the businesses, including details about the addresses, the Yelp review counts, ratings, the geographical location, and even some business-specific attributes such as dog allowance, wheelchair accessibility, Cryptocurrency acceptance, the presence of drive-through and many more.

Inside Dataset for “Reviews”, we find the textual bodies of the Yelp’s available reviews. Those are provided with some useful details, such as the business ID they are referring to, the ID of the user who wrote a certain review, the final grade of a review, the date and time when the review was written, how useful for the other clients the review was.

2.1 Data cleaning

We now start describing the data cleaning process we performed on the dataset “Business”. We start by importing the dataset, which originally was in the JSON format, through the stream_in function:

```
Business = stream_in(file("yelp_academic_dataset_business.json"))
```

Once we convert the dataset into a data table, we should eliminate the columns providing unnecessary information for our analysis. We hence obtain a dataset with only 11 columns, instead of the 58 initial ones.

Afterwards, we are using the function str_split to extract the single categories from the strings under the column “categories”, creating a list including all the categories of the businesses in the dataset:

```
Categories = str_split(business_11columns$categories,",")
categories = as.data.frame(unlist(categories))
```

As in the dataset are present many additional categories besides restaurants, we clean the dataset keeping only the relevant activities:

```
business_rest=business_11columns[business_11columns$categories %like% "Restaura
nts",]
```

Once our dataset only includes restaurant activities, in order to identify different categories of restaurants, we are removing the words “food” and “restaurants” from the category’s column, because every entry in our dataset included those words in the string related with its category:

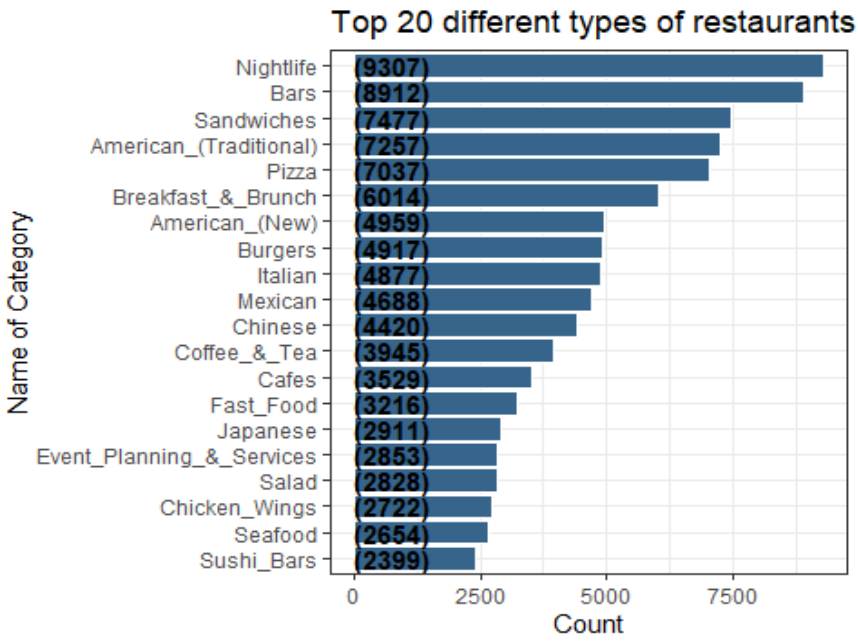
```
categories=data.frame(lapply(categories,function(x){gsub("Restaurants","",x)}))
business_semifinal=data.frame(lapply(business_rest,function(x){gsub("Restaurant
s, ","",x)}))
```

Finally, it is necessary to eliminate the spaces between words belonging to the same category of restaurants, so that we can joint them together as a single category. In order to do so, first we have to remove the spaces

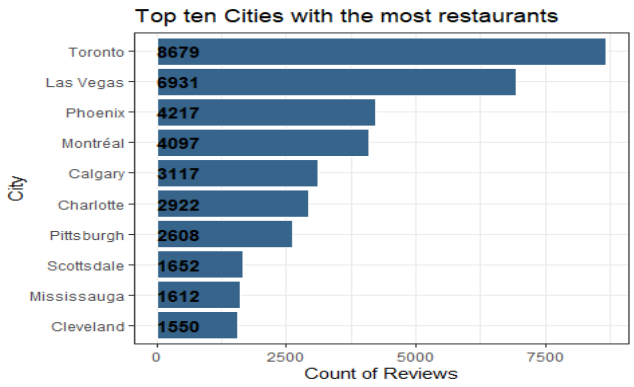
after commas (which divided different categories of restaurants). Then, we can proceed substituting every remaining space with an underscore, therefore obtaining categories such as “sushi_bars”, “fast_food”:

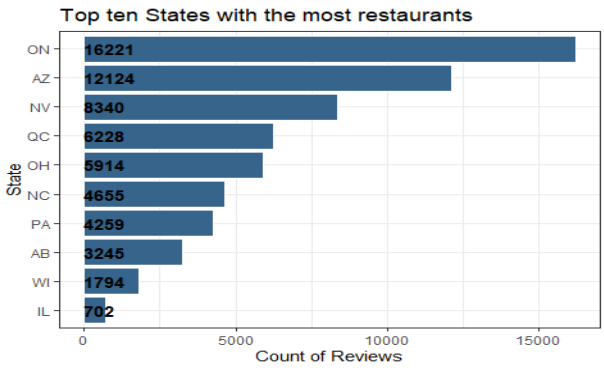
```
new_nospaces=lapply(business_final$categories,function(x){gsub(",     ",",",x)})
new_nospaces=lapply(new_nospaces,function(x){gsub(" ","_",x)}
```

We can then obtain a graph showing all the different categories of restaurants:

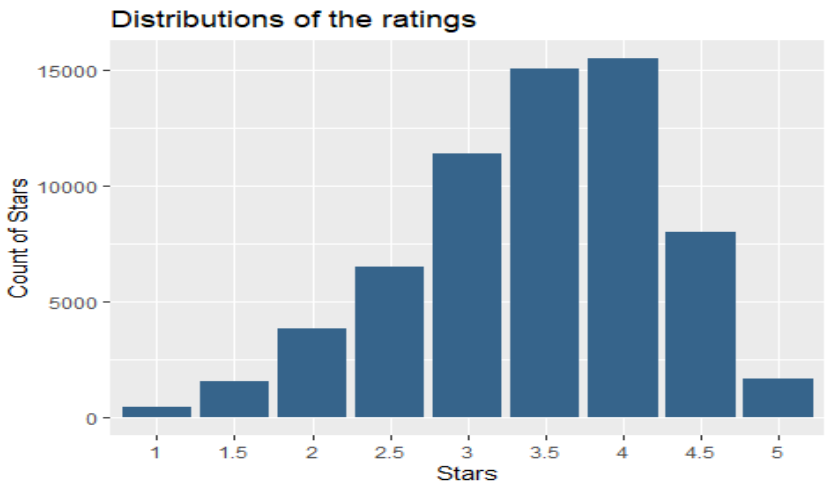


We can then create two more graphs, representing respectively the top ten states with the highest number of restaurants, and the top ten cities with the highest number of restaurants:





In conclusion we are going to create a histogram to represent the average rating distribution of the restaurants:



We are choosing to carry on our analysis specifically on Las Vegas restaurants, for two main reasons:

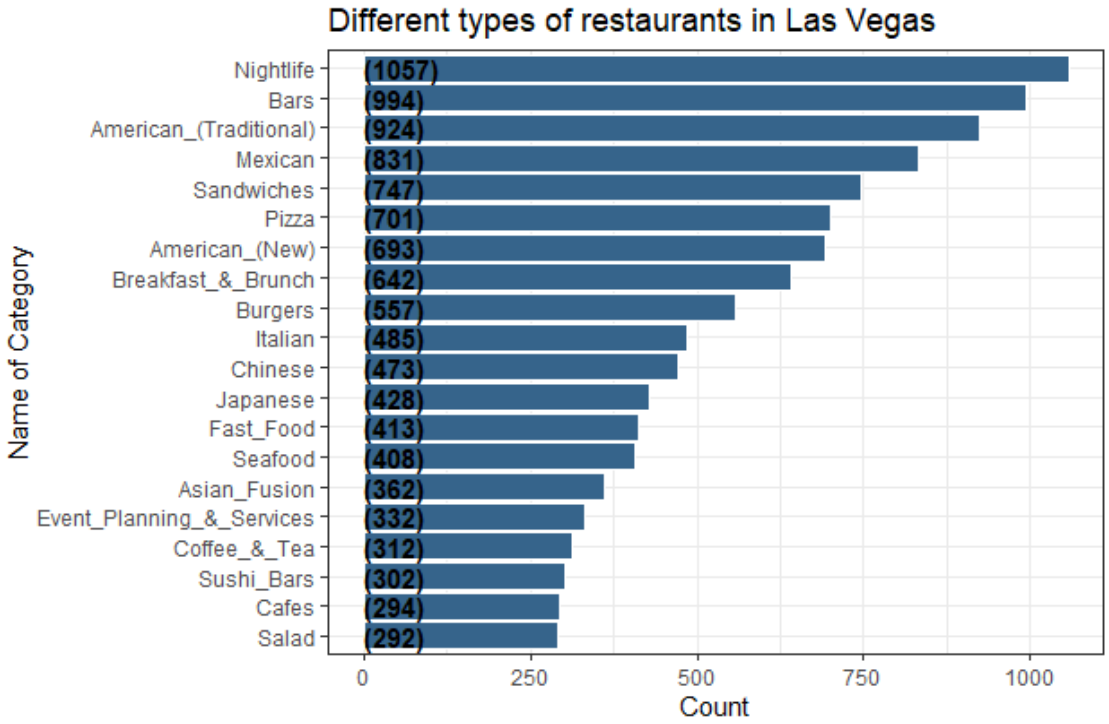
- Las Vegas is the US city with the highest number of restaurants
- we expect Las Vegas to exhibit a heterogeneous set of categories, with very different ratings as well.

2.2 Business sample preparation

To construct our sample, we create a dataset including only the restaurants in Las Vegas:

```
business_vegas=business_superfinal[business_superfinal$city == "Las Vegas",]
```

We then create a graph showing the count of each kind of restaurant in Las Vegas:



Among the different categories, we select three of them so that they are as different as possible from each other. In this way, we are capturing a heterogeneous sample, with many different and contrastive reviews. The categories we selected are “Italian”, “Chinese” and “Fast-food” .

For getting rid of the restaurants whose categories are described as a compound name, including one of the four words above but not fully coinciding with any of them, we wrote a function (below the one referring to the word “Chinese”) to remove the compound categories (such as “Chinese_Fast_Food”, “Internet_Cafes” and so on):

```
business_italian=business_vegas[business_vegas$categories   %like%   "Italian",]
business_italian=data.frame(lapply(business_italian,function(x){gsub("Italian_"
,"",x)}))
```

Afterwards we substitute the column “Categories” with the names of the three different categories we selected in the following way:

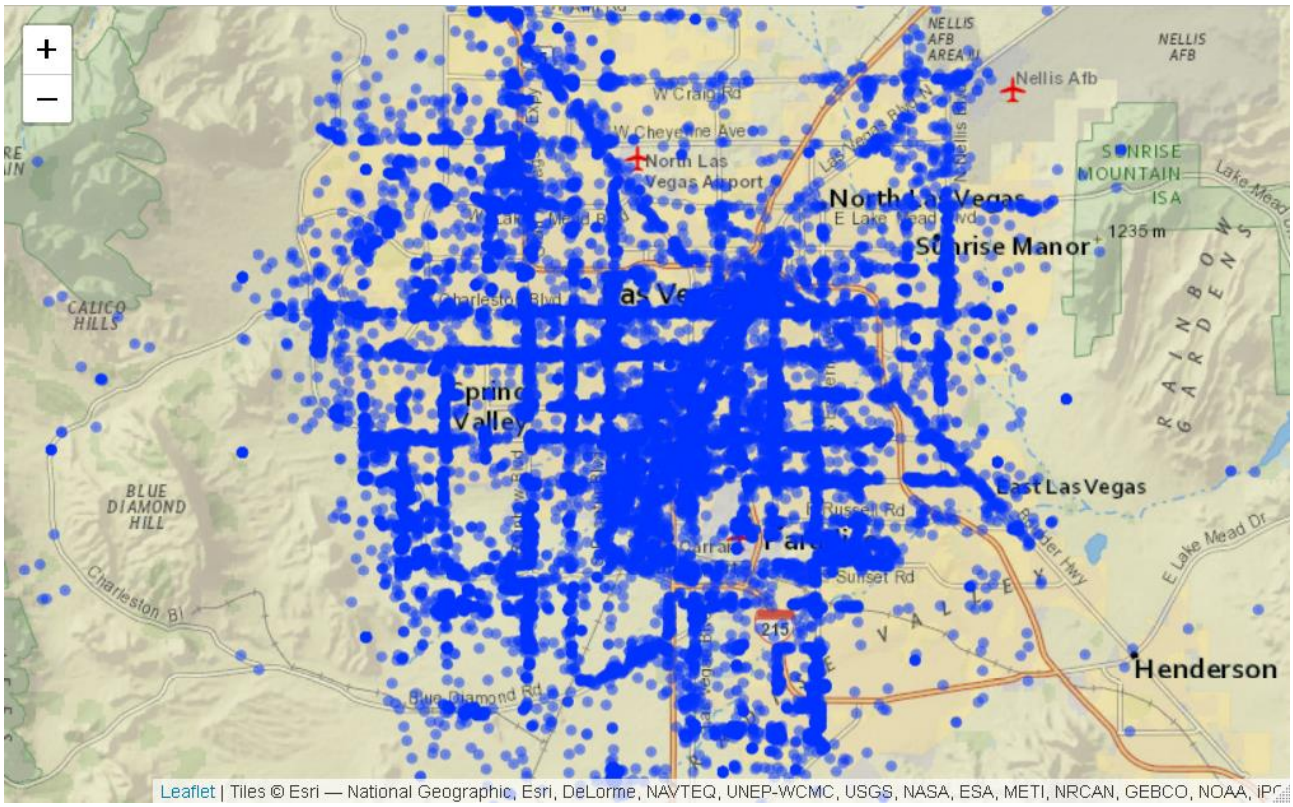
```
business_chinese$categories=strrep("Chinese", 1)
```

Furthermore, both for the sake of simplicity and to avoid overlaps, we remove from our dataset the restaurants which include two or more of the kinds of restaurants we selected in their category. To do so, firstly we are merging the three subsets and removing all the overlaps. Then, we divide them again into separate data frames:

```
business_3types=rbind(business_italian,business_chinese,business_ffood)
business_3types_noduplicates=business_3types[!duplicated(business_3types$busine
ss_id),]
```

```
Business_chinese_final=business_3types_noduplicates[business_3types_noduplicate
s$categories == "Chinese",]
```

In conclusion, to double-check that the selected restaurants are actually located in Las Vegas, we visualized them in a map, through the package “Leaflet”, utilising their latitude and longitude, which were available in the original dataset:



3. Textual Analysis

3.1. Reviews sample preparation

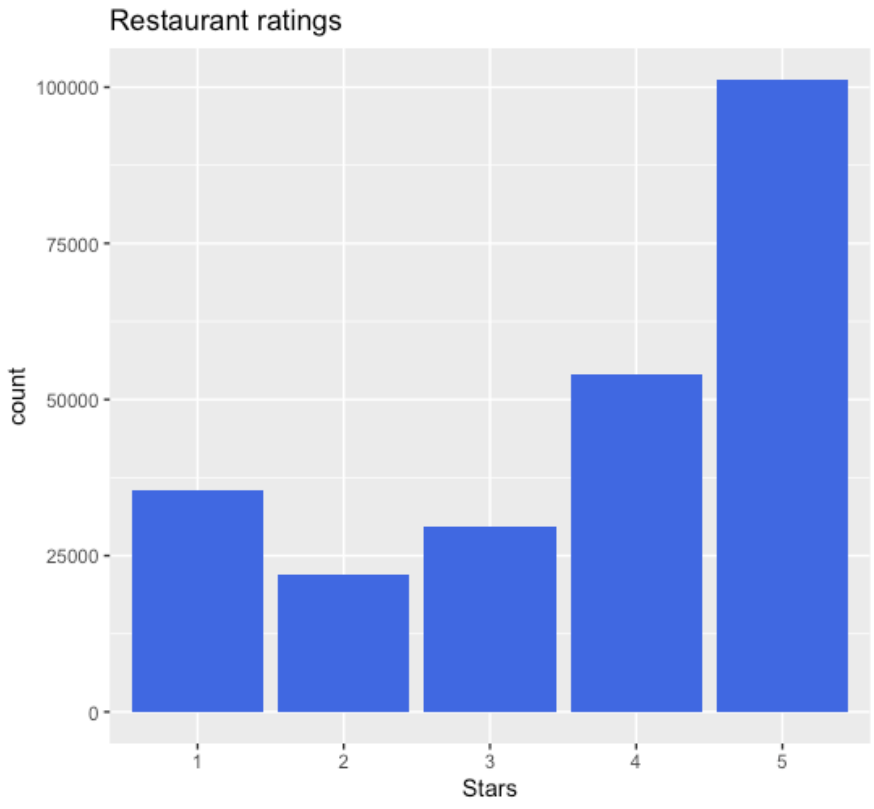
In this part we move on to the natural language processing part of the review texts. The first step is to import the necessary textual data. However, since texts of reviews in our case are part of a very large dataset in a json format, we are going to open the file in tranches and import only the columns of our interest, containing review texts, rating scores and respective IDs for reviews and businesses.

```
Rev_1 <- NULL
stream_in(
  file("yelp_academic_dataset_review.json"),
  pagesize=100000,
  handler=function(x) {
    Rev_1 <<- rbind(Rev_1, x[,c("review_id","text")])})
```

And thereafter we use the (dplyr) library and left_join() function in order to match the reviews with correct business, based on their IDs.

The next step is to make the choice of categories of restaurants, based on the type of cuisine or pricing. Since the goal of the analysis is to look for significant drivers of restaurant ratings, we decided to choose Chinese, Italian and Fast-food restaurants since in our opinion the comparison of these 3 popular and also different categories is going to yield some significant results.

The graph below gives the overview of distribution of restaurants on the rating scale. It is crucial to have a significant number of restaurants in each rating bracket, to be able to make comparisons between the key drivers among different ratings.



From the histogram we can see that the number of restaurants is roughly divided in half with scores below and above 4,0. So, below we are going to refer to restaurants as Below average and Above average, with ratings of under and over 4,0 respectively.

After reading and cleaning the data we need to generate a corpus for the text analysis. Below is a summary of a generated corpus.

```
Corpus consisting of 242403 documents, showing 5 documents:

 Text Types Tokens Sentences     X stars category
text1   175    300        22 67500     5  chinese
text2   142    240        10 67501     3  chinese
text3   102    155        14 67502     4  chinese
text4    33     40         6 67503     5  chinese
text5    71     93         7 67504     5  chinese
```

As a next step, we calculate a Document-Feature-Matrix (DFM), which describes how frequently terms occur in the corpus and splits the text into its single terms (tokens). Here, we also decided to remove the tokens from DFM, of very low and high frequency, below 1% and above 80% frequency respectively.

```
Reviews_dfm_trim = dfm_trim( Reviews_dfm,
                             min_docfreq = 0.01,
                             max_docfreq = 0.80,
                             docfreq_type = "prop" )
```

This way we are removing any leftover redundant words that do not contribute to differences between categories.

Therefore, at this point we are able to create a graphical representation of the words used among different categories. Moreover, we are also splitting the DFM into two parts, in above average and below average scores.

```
Reviews_above_dfm_subset = dfm_subset( Reviews_dfm_trim, stars >= 4 )
Reviews_below_dfm_subset = dfm_subset( Reviews_dfm_trim, stars < 4 )
```

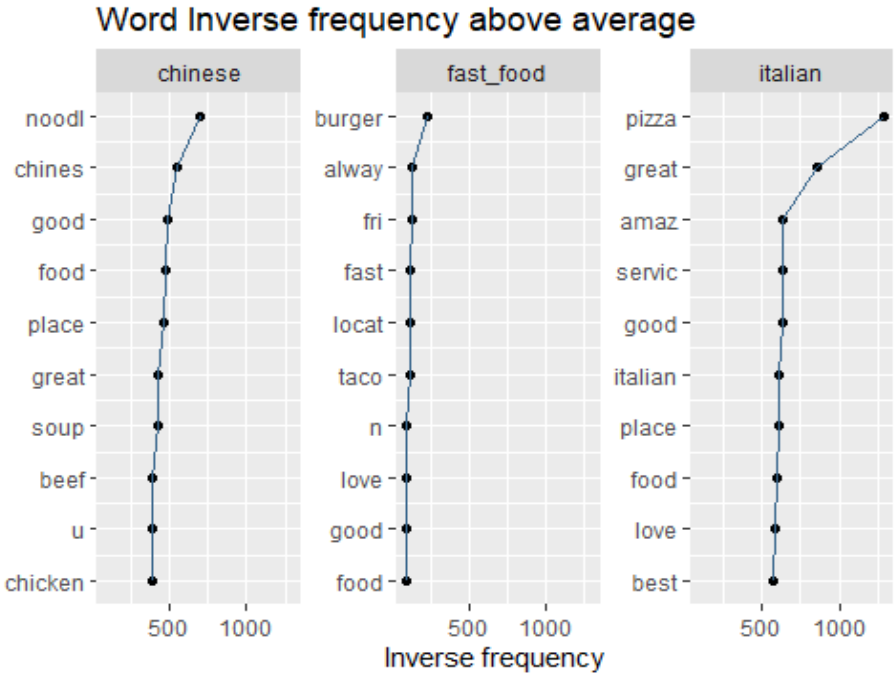
This way we are going to be able to make comparisons not only between the categories but also between restaurants of different qualities within the same category.

3.2. Preliminary textual analysis

To begin with, we started by plotting absolute frequency of words in different categories.

Nevertheless, plotting absolute frequency is not very helpful for making any conclusions since the values are not adjusted or weighted for the length of documents and fact that some words appear more frequently than others in general.

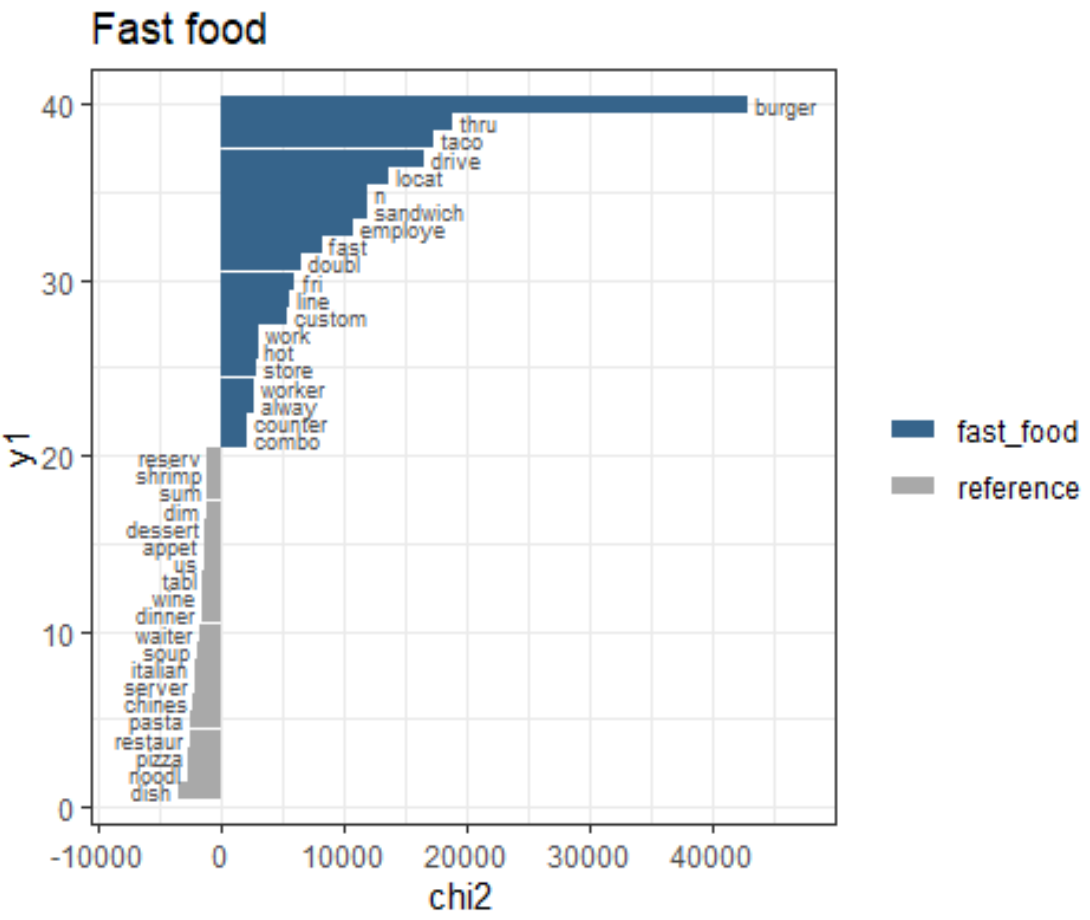
As a solution, we are going to create Inverse frequency plots. The term frequency–inverse document frequency (Tf-Idf) reflects an importance of a word to a document in a corpus. The Inverse document frequency (idf) gives us the scale of information provided by the word and the high score in Tf-Idf is reached by high term frequency and a low document frequency of the term, therefore filtering out the common words.



First of all, we observe from the td-idf graphs that apart from dishes specific to each cuisine, service and place have a very high importance for Italian and Chinese restaurants. The difference between the two is also clear, since main staples and ingredients of Chinese cuisine like noodles, soups, beef and etc. have a high importance while for Italian restaurants we can observe more value in adjectives describing overall high quality and excellence rather than particular ingredients

Among reviews of Fast-food restaurants, we also see a lesser importance of particular items of the menu, but rather higher value for speed, location and consistency.

Furthermore, we are building graphs for each chosen category, based on the Keyness. A measure of keyness gives us the distinguishing features of categories, by comparing the word frequencies in texts of chosen category against their expected frequencies derived in a larger corpus. As a result, we receive the visual representation of words as being important to the context of reviews.



As we can see from the example of a Keyness plot for Fast-food restaurants, the words mostly used in the reference sample are mostly the types of dishes and services offered in specific categories. The results from the graph also further support the data from Inverse frequency charts, that words relating to employees, location and speed are much more likely to play key role in Fast food category. Whereas, among Italian and Chinese restaurants reviews relating to cuisine specific dishes and service quality have much higher importance than for Fast food places.

4. Sentiment analysis

4.1. Sentimentr package

In order to carry out the sentiment analysis, we decided to use the "Sentimentr" package. Our choice was based on the fact that Sentimentr combines strong accuracy with high computing power. In order to define the final sentiment, Sentimentr uses the following innovative approach, which operates at a sentence level.

First, the text is divided into sentences, and each sentence is broken into an ordered bag of words, where punctuation is removed, with the exception of pause punctuations. The words in each sentence are searched and compared to a dictionary of polarized words. We used the default dictionary, which is the Jockers (2017) dictionary.

To every positive and negative word is assigned respectively a score of +1 and -1. For each word a polarized context cluster is created, taking into account the four previous words and the two subsequent ones. The sentiment of the cluster is obtained considering initial polarized value, adjusted by different factors.

These factors are the locations of punctuation denoting a pause, and the role of the other words included in the cluster, also called valence-shifters. These valence-shifters can act as neutral, negator, amplifier or de-amplifier of the initial tone. Amplifiers increase the polarity, and they become de-amplifiers if the context cluster contains an odd number of negators. They of course work to decrease the polarity. Negators acts on amplifiers/de-amplifiers as discussed but also flip the sign of the polarized word.

The adversative conjunctions (i.e., “but”, “however”, and “although”) also weight the context cluster. An adversative conjunction before the polarized word up-weights the cluster, while an adversative conjunction after the polarized word down-weights the cluster. This corresponds to the belief that an adversative conjunction makes the next clause of greater values while lowering the value placed on the prior clause.

Last, these context clusters are summed up and divided by the square root of the word count yielding an unbounded polarity score for each sentence. To obtain the sentiment at a document level, we can just take the mean of all the sentiment scores of the sentences.

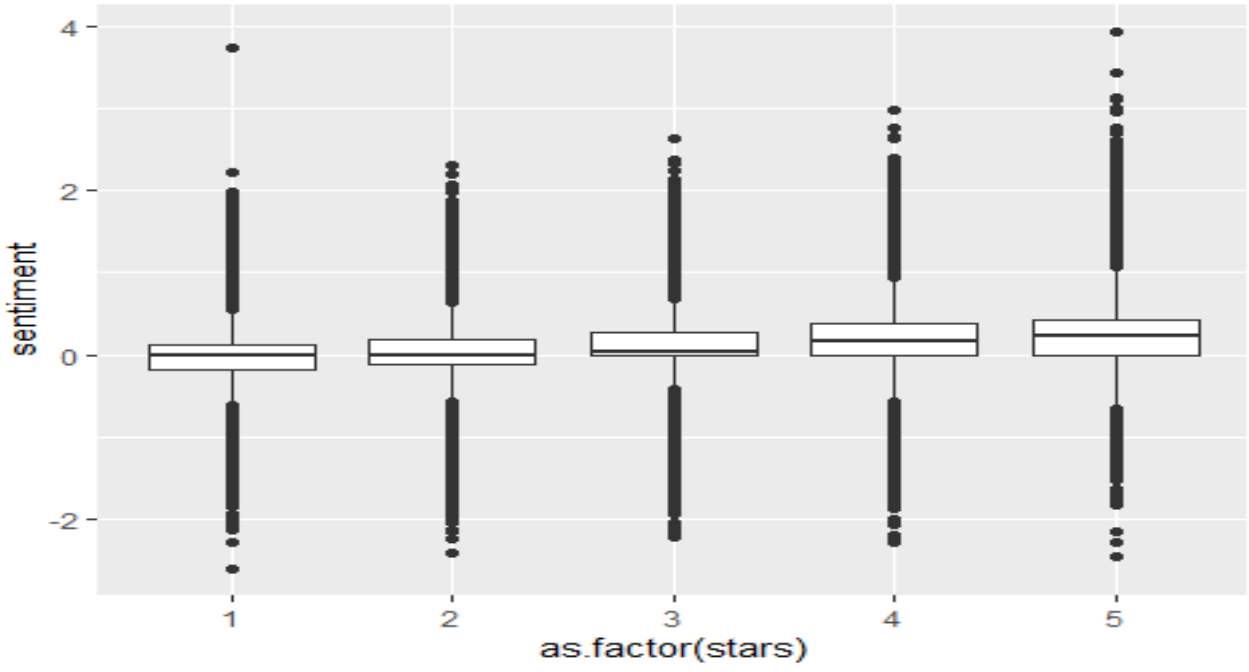
In order to have more reliable results, we decided to adopt a slightly different way to compute the mean, which basically downweighs the sentences whose sentiment is equal to 0. This is useful in the context of language where we don't want the neutral sentences to have such a strong influence on the general sentiment of the discourse with multiple sentences.

We then ran the sentiment analysis at both document and sentence level:

```
Reviews_total_sent=sentiment(get_sentences(Reviews_final))
Reviews_sent_bydoc = sentiment_by(get_sentences(Reviews_final))
```

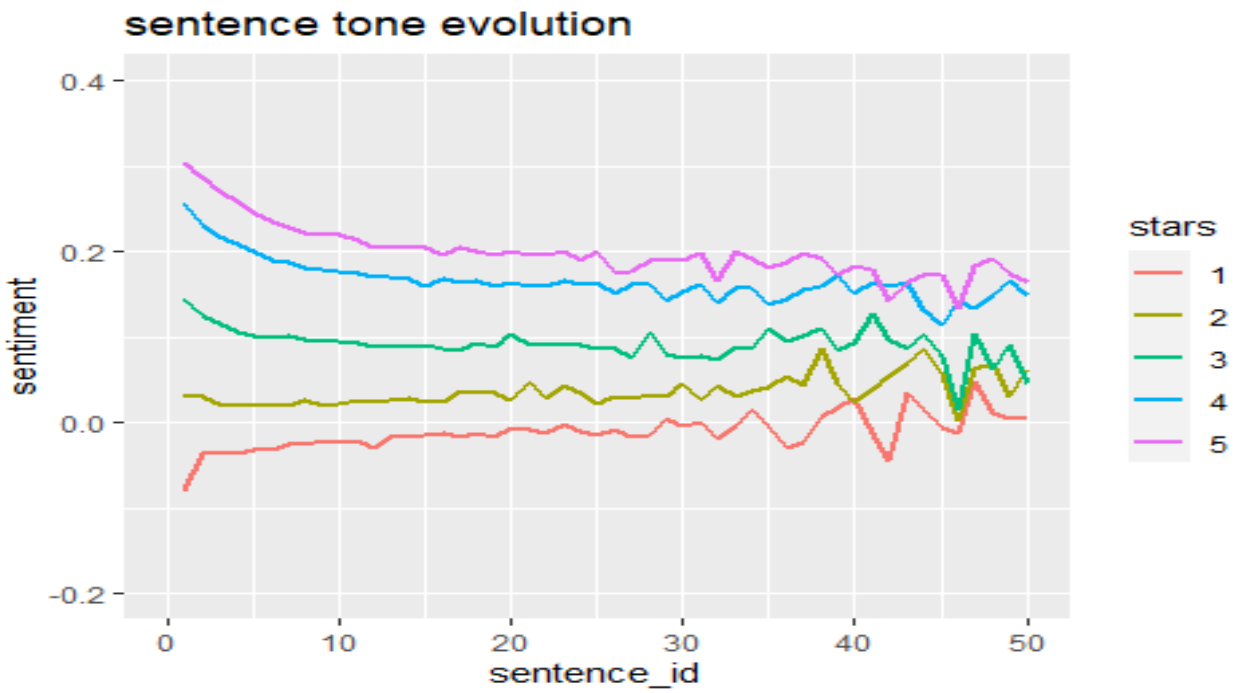
4.2. Sentiment analysis on the whole sample

In order to start the investigation of our sample, we wanted to have a picture on the overall results. Therefore, we created a boxplot showing the distribution of the sentiment at a sentence level, according to the stars of the reviews.



We can immediately see that the sentiment is increasing according to the stars, as predicted. However, the changes do not seem to be very relevant, but this is partially due to the presence of the outliers which make the graph scale quite wide. Indeed, we can observe a huge dispersion in the sentiments.

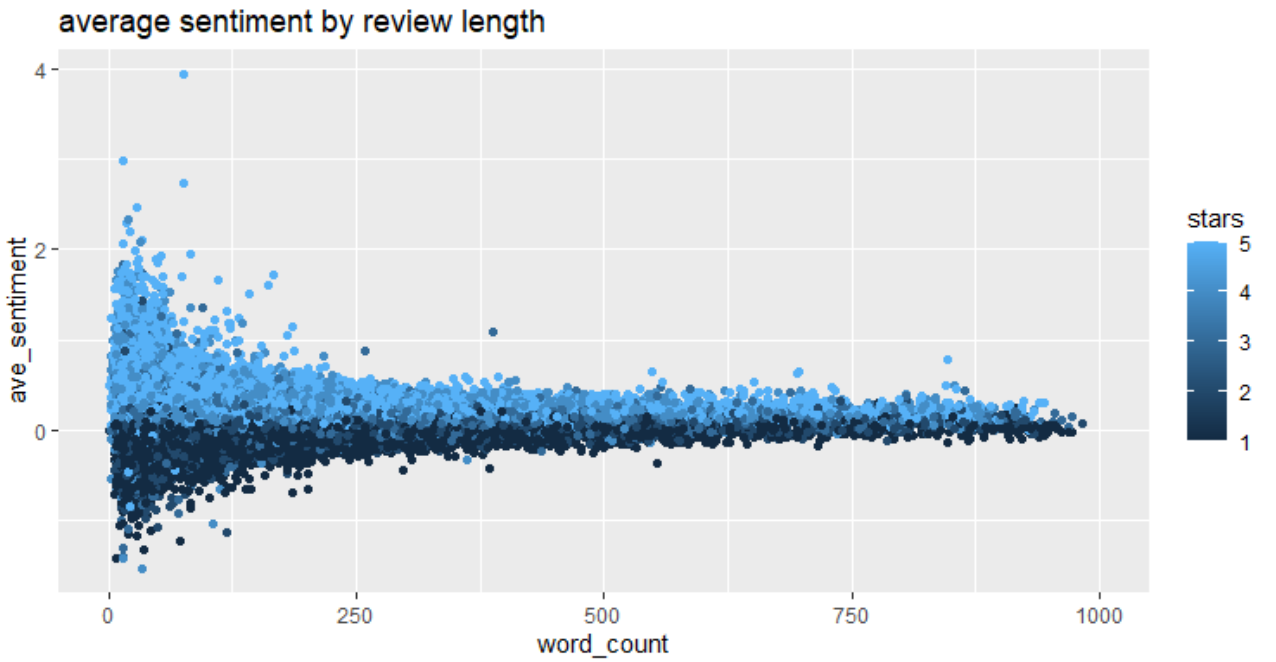
Now we want to see if we are able to find any particular pattern in our data. In order to do so we start by looking at the evolution of the tone throughout the length of the reviews. We then plot the evolution of the average tone at a sentence level according to the ordinal number of the sentence:



The first point we can observe is that as the ordinal number of the sentence increases, its average tone tends to converge towards some sort of mean value. This means that, for the low-rated reviews, as the ordinal number of the sentence increases, its average tone is going to increase. The average tone of the high-rated reviews instead, decreases as its ordinal number increases.

However, we see that the evolution of the average tone of a sentence follows unpredictable patterns when the ordinal number of the sentence becomes very high: this is due to the lack of a sufficient number of observations with a very high number of sentences. For this reason, we partially cut the last part of the graph, as the results would not be meaningful at all.

These results lead us to investigate whether we can find a correlation between the average sentiment of a reviews and its length. In order to do so we create a scatterplot associating the review sentiment with its number of words:



As expected, there seems to be an inverse pattern in the evolution of the sentiment, depending on the stars received by the reviews: for the low-rated reviews, the average sentiment is increasing, while for the high-rated reviews it is decreasing. Moreover, we can notice that, as we could expect from the previous graph, most if not all the dispersion is concentrated in the initial part of the graph, where we have shorter reviews.

Indeed, we deduce that, on average, the most extremes tones are found in shorter reviews. As the reviews become longer, the tone tends to be mitigated.

Hence, we can suppose that on average, most relevant comments are made at the very beginning of a review.

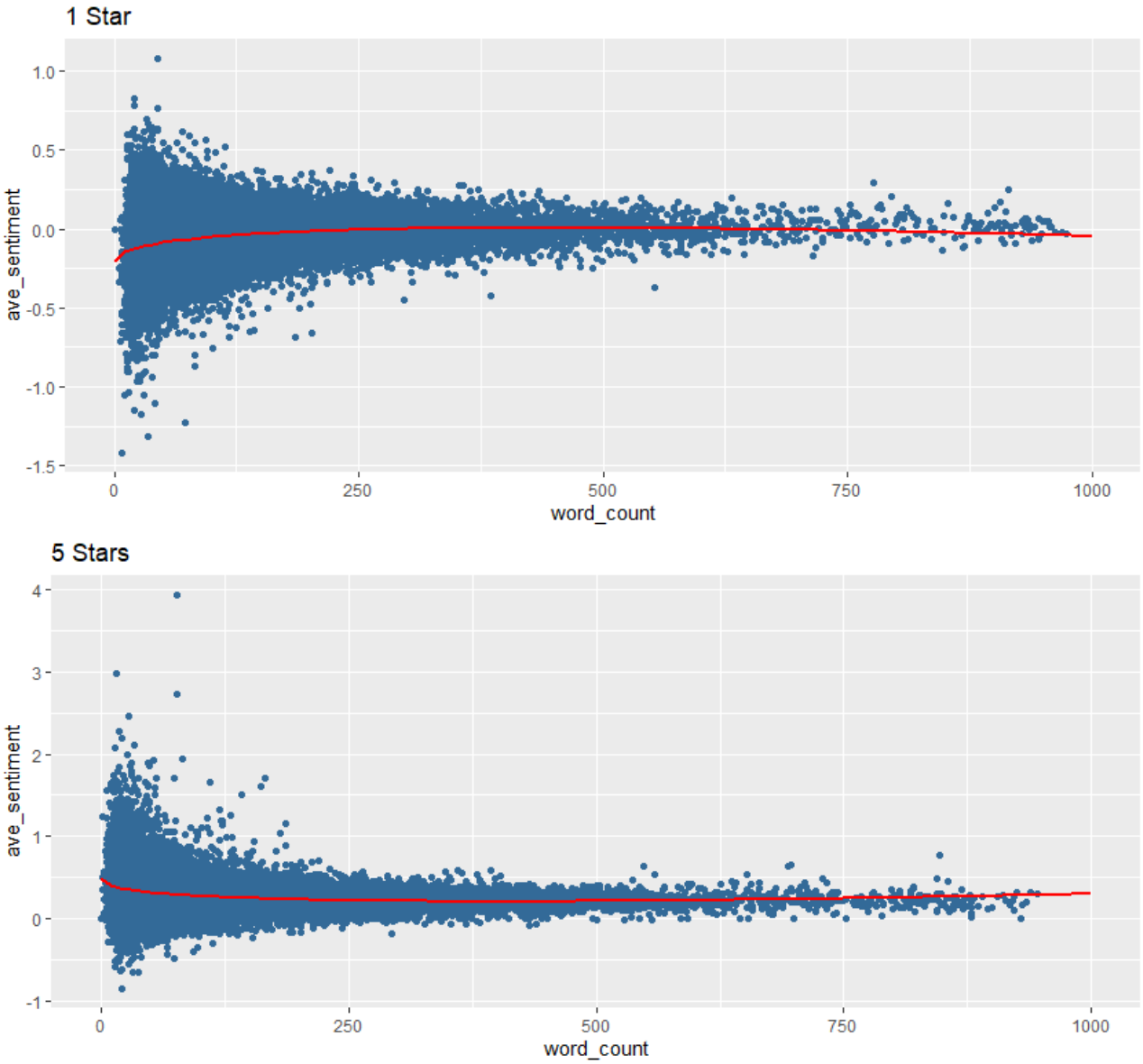
However, if we consider separately each subset according to a single star, we see that the distribution for smaller reviews is way more spread than it seemed before. Furthermore, we take as example the extreme rated reviews (1-star and 5-star) and we try to investigate the relationship between the sentiment and the number of words in a review. In order to do so, we regress the sentiments of the reviews on the number of words of those reviews. The regression which better fits our data has resulted to be the following one:

```
reg_5 = lm(ave_sentiment ~ word_count + I(word_count^0.5),
      data =  subset(Reviews_sent_bydoc, stars==5))
```

As we can guess from the graph, there 2 main components which drive the relationship:

1) Indeed, there is an inverse relationship, which means that in high-rated reviews, as the number of words increases the sentiment decreases, for low-rated reviews instead, as the number of words increases, the sentiment increases;

2) As the number of words increases, the marginal effect of an additional word on the sentiment of the



However, both the regressions have a low R^2 (**see page 55 R markdown**) and do not seem to explain a lot of the variance of the model.

In fact, especially by paying attention to the sentiment distribution of the short reviews, there seems to be an incredibly high number of outliers that make the pattern much less predictable. In particular, there are many reviews which show an extreme and opposite sentiment to what you would expect according to the rating: there are many 1 star-rated review which show a sentiment 4 times higher than the average 5-star review, and many 5 star-reviews which have very negative sentiment.

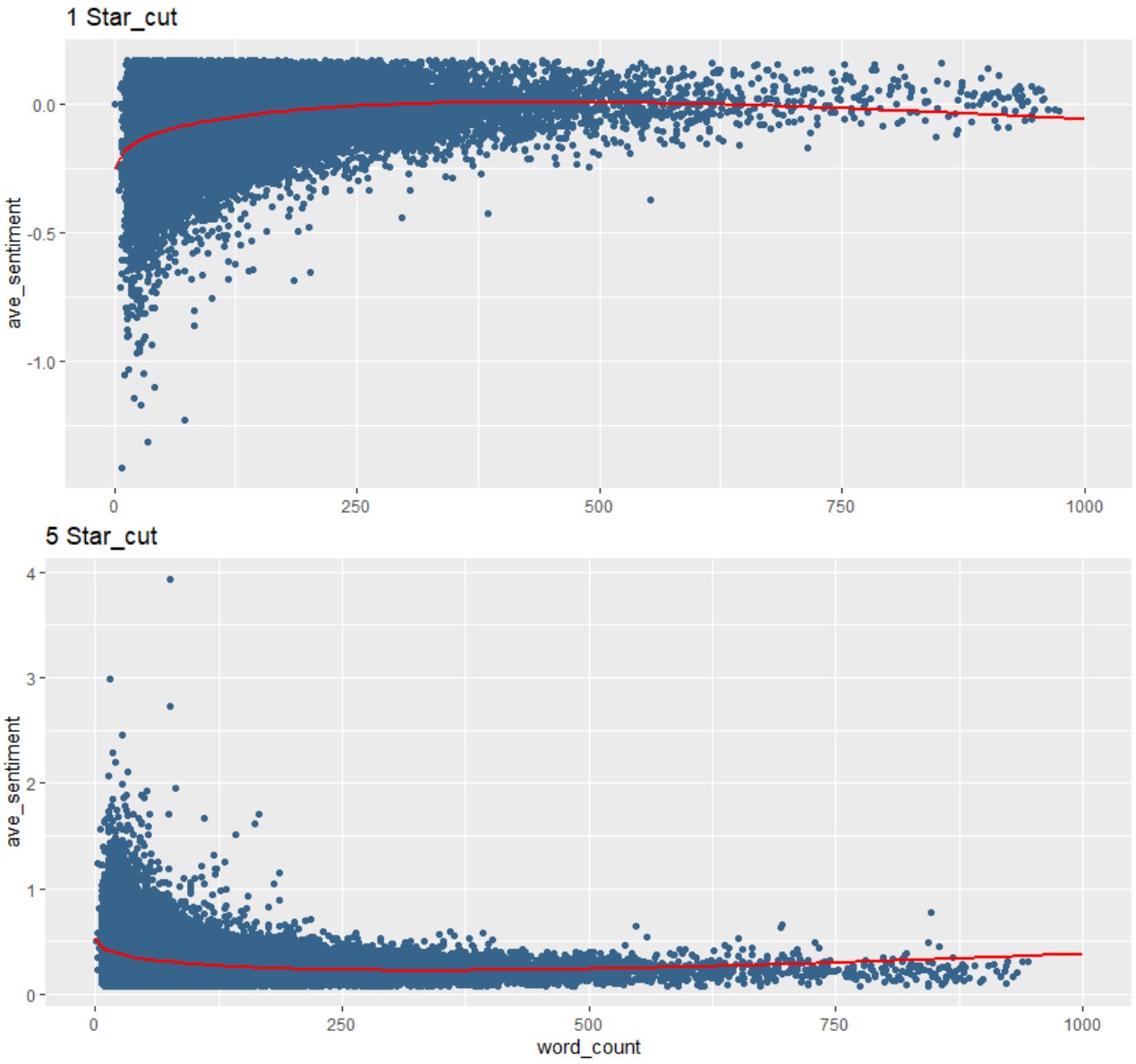
We therefore decide to have a look at those observations, and we note that they are caused by the fact that our model is not able to properly valuate certain reviews. The most recursive reasons are for example the fact the sometimes users write with an ironic tone, and this is not of course recognized by the system. Moreover, it might happen that the user compares his bad experience with a very good one (or vice versa), using positive words which are not related with the restaurant he is reviewing. Finally, it is clear that some

users made a mistake, when typing the number of stars, as the reviews are totally in contrast with the number of stars given by the users. At page 57 of the final R markdown you can find some examples taken from the best 1% of the 1-star reviews.

We believe that those misclassified reviews might have a negative impact on our analysis, and we then decide to get rid of them. In order to do it, we cut the distribution of each single star-subset. For 1 and 2 stars we keep only the observations falling inside the 95% quintile range. For 4 and 5 we cut the observations falling inside the 5% quintile. For 3, we keep the observations falling between the 2.5% and the 97.5% quintiles.

```
subset1 = subset1[ave_sentiment < quantile(ave_sentiment, 0.95),]
subset2 = subset2[ave_sentiment < quantile(ave_sentiment, 0.95), ]
subset3 = subset3[ave_sentiment > quantile(ave_sentiment, 0.025) &
ave_sentiment < quantile(ave_sentiment, 0.975), ]
subset4 = subset4[ave_sentiment > quantile(ave_sentiment, 0.05),]
subset5 = subset5[ave_sentiment > quantile(ave_sentiment, 0.05), ]
```

If we run again the regressions on the new scatterplots, we notice that the distributions are less spread and that our regressions better describe the relationship of the number of words on the review sentiment, achieving higher R^2 **(see page 58 R markdown**)



At this point, we merge together all the new subsets, cleared of the extreme outliers, and we come up with what will be our final dataset of reviews for our analysis.

4.3. Sectorial breakdown and drivers model

Now we want to dig deeper into the sentiments associated with each of the four restaurants previously selected: “Chinese”, “Italian”, “Fast Food”, “Cafes”.

Our aim is to analyse the differences between those kinds of restaurants, looking for the main drivers of the reviews rating. We are going to do so by considering the sentiment associated with each star-subset of reviews, isolating step by step each potential driver, and finally seeing how they affect the overall sentiment.

The Sentimentr is much suited for this matter. In fact, it allows us to easily compute the average sentiment corresponding to each group of reviews with the same star. First of all we try it on the whole sample:

```
Reviews_sent_bystar=with(Reviews_final2,
                         sentiment_by(get_sentences(text),list(stars)))
```

We then obtain the following table in which we can see again that the average sentiment increases as the number of stars obviously increases:

```
##    stars word_count        sd ave_sentiment
## 1:     1    4095093 0.2853795   -0.05608484
## 2:     2    2864679 0.3000446    0.02361396
## 3:     3    3737936 0.3040734    0.13807745
## 4:     4    5944146 0.3110667    0.26173232
## 5:     5    8147310 0.3032402    0.32271646
```

Now we focus on the single categories of restaurants, starting from the Chinese one.

We run the previous code on the sub-sample made only by Chinese observations, and we get a similar table. Then we identify a potential driver, for example food. We manually write a list of all the potential words which can be found in the sentences related to food, for example: “Food”, “Noodles”, “Ramen”, and so on.

We split our dataset of reviews into sentences and we transform it into a corpus object. We write a code to detect the presence of at least one of such words in each sentence of our corpus, and we create another corpus with only those sentences.

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent),
"grill|chees…

Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
```

Finally, we transform it into a dataframe and we run the Sentimentr code grouping by stars. In this way, we can approximate the average sentiment associated with that driver (Food), as the number of stars changes.

We repeat this process with a list of drivers we selected: “Food”, “Price”, “Location”, “Service”, “Timing”, “Vegan”, “Delivery”, “Drink”. “Food”, “Price”, “Location”, “Service”, and “Drink” are self-explanatory, while, with “Timing”, we mean the punctuality of the service, with “Vegan” we refer to the presence of vegan dishes offered by the restaurants, with “Delivery” we investigate the quality of the delivery or drive-in services.

Ultimately, we aggregate all these results in a single table which shows the average sentiment per star of each driver and of the whole Chinese sub-sample. We then repeat this process for the other categories of restaurants, obtaining four analogous tables. Here follows an analysis of what we found with Chinese restaurants at first:

```
##      stars       total        food   location        price     service
## [1,]     1 -0.06699585 -0.03270563 -0.0445212 -0.049974652 -0.10292773
## [2,]     2  0.01276642  0.02976214  0.0563343  0.001134725  0.01648748
## [3,]     3  0.13256649  0.13408710  0.1616312  0.125059213  0.18122735
## [4,]     4  0.25196593  0.24891951  0.2616788  0.282672257  0.37089363
## [5,]     5  0.32158406  0.31797054  0.3351764  0.392472076  0.48433992
##             timing       vegan   delivery       drink
## [1,] -0.009368046 -0.00769091 0.02569809 -0.01668528
## [2,]  0.040076430  0.04543234 0.08644224  0.03787670
## [3,]  0.128934287  0.11514344 0.14234641  0.11026702
## [4,]  0.228800530  0.18676340 0.25373371  0.19619349
## [5,]  0.289949519  0.23695201 0.32819367  0.27143475
```

To carry out our analysis, we mainly focus on the comparison between the total average sentiment by stars and the average sentiment by stars per driver. In particular, we pay attention to the highest and lowest ratings sentiments, to how they compare with the average sentiments, and to how wide their spread is. By spread we mean the differences between the average sentiment of the high-rated reviews and the low-rated ones. In fact, the wider the spread in both directions, the more important that driver is supposed to be in the assignment of the rating.

Of course, it is also important to compare the drivers among themselves. This is because the comparison with the total average might be biased by the fact that the total average sentiment is not fully explained by the few drivers we selected. In addition, we should remember, from the previous analysis, that the most extreme sentiments are found in the shorter reviews, where it is probably harder to isolate even just one driver. Hence, in order to consider a certain driver relatively more important than the others, we should not just focus on the comparison with the overall sentiment. This is because it might often happen that a driver whose average sentiment does not exhibit a widespread, compared to the total averages, it is instead way more relevant if compared with the other drivers we selected.

For example, looking at the table, we now consider “Service”. It is clear that the average sentiment for the one-star reviews is lower than the overall one, and that the average sentiment for the five-star reviews is way higher than the overall one. This spread is also the highest among all the remaining drivers. This means that the service is a huge discriminant in the rating assignment. On average, if you are opening a Chinese restaurant, in order to achieve a high rating, you should really pay attention to the service, because, on average, the high-rated reviews display a very high sentiment, with regards to the service. On the contrary, we can observe the exact opposite for the low-rated reviews: on average, a low-rated review is associated with a very low service sentiment.

Following the same reasoning, other important discriminant factors, are, in order: “Price”, “Location”, “Food”. For example, we see that interestingly in the context of Chinese restaurants customers care more about what they believe to be the fair price, rather than the actual quality of the food. This might be partially due also to the fact that the food quality of the Chinese restaurants is perceived to be relatively constant.

Finally, a good delivery service might be another booster to obtain a good review, while not acting much on the downside.

Let's now examine the Italian restaurants:

```
##      stars       total         food    location       price     service
## [1,]     1 -0.04465663 -0.005087015 -0.03060802 -0.03544714 -0.07294874
## [2,]     2  0.03240266  0.064069868  0.07608219  0.01130238  0.03944641
## [3,]     3  0.14306959  0.162423771  0.17197767  0.10290910  0.18104872
## [4,]     4  0.26779345  0.291193989  0.27282443  0.27309693  0.36802920
## [5,]     5  0.32203982  0.339900294  0.32754101  0.37779252  0.45842871
##           timing      vegan   delivery       drink
## [1,] 0.001035016 0.02584474 0.03628824 0.006288931
## [2,] 0.049744772 0.09226504 0.08602326 0.054443254
## [3,] 0.126288916 0.14597006 0.14183844 0.139343412
## [4,] 0.233250426 0.22420310 0.23137247 0.242638346
## [5,] 0.284765468 0.25848794 0.28607495 0.314236061
```

We can immediately see that the average sentiments of the whole sample, are higher than the total Chinese ones, reflecting that expected higher standard of the restaurants.

However, when it comes to our drivers, the picture is quite similar to the previous one. "Service" is again the most important factor by far, with "Price", "Location" and "Food" following in this order. Nevertheless, "Service" and "Price" show a less widespread than in the Chinese restaurants, and this aspect, combined with the fact that the total spread is only slightly lower, might suggest that in the Italian restaurants' customers give less importance to these two aspects.

On the contrary, "Food" and "Drink" become more important factors in high-rated restaurants. This was highly predictable, since in high-end Italian restaurants we would expect an important offer of sophisticated dishes and high-quality wines.

We now want to have a look at Fast Foods:

```
##      stars        total         food    location       price     service
## [1,]     1 -0.071426078 -0.01111171 -0.06627970 -0.03947255 -0.13558909
## [2,]     2  0.005507767  0.04716082  0.04438389  0.01777350  0.01782191
## [3,]     3  0.129395269  0.16014795  0.15540960  0.12479959  0.20105240
## [4,]     4  0.264815104  0.28916948  0.28056813  0.29127976  0.42021378
## [5,]     5  0.335300794  0.37150346  0.35816477  0.39967679  0.50445154
##           timing        vegan    delivery        drink
## [1,] -0.03224146 -0.004356434 -0.02965883 -0.017924742
## [2,]  0.03008229  0.035122219  0.02570417  0.006258592
## [3,]  0.13733305  0.146346996  0.10879511  0.102009483
## [4,]  0.24756533  0.225592176  0.20370123  0.210401299
## [5,]  0.30909336  0.298170628  0.29105068  0.276018340
```

The total spread is wider; hence we will expect wider spreads also in our drivers. The most important one is still "Service", while for the remaining ones we get mixed results. In the upper side, the high-rated restaurants, "Price" is still the most relevant, followed respectively by "Food" and "Location".

However, for low-rated restaurants, the second most important factor seems to be "Location", followed by "Price", "Timing" and "Delivery". This is probably due to the fact that in Fast Foods, customers on average care more about getting their food as soon as possible, and about being able to eat their food at home (or in the car) in the easiest possible way. The quality of the food, instead, does not represent a significant down-weight factor.

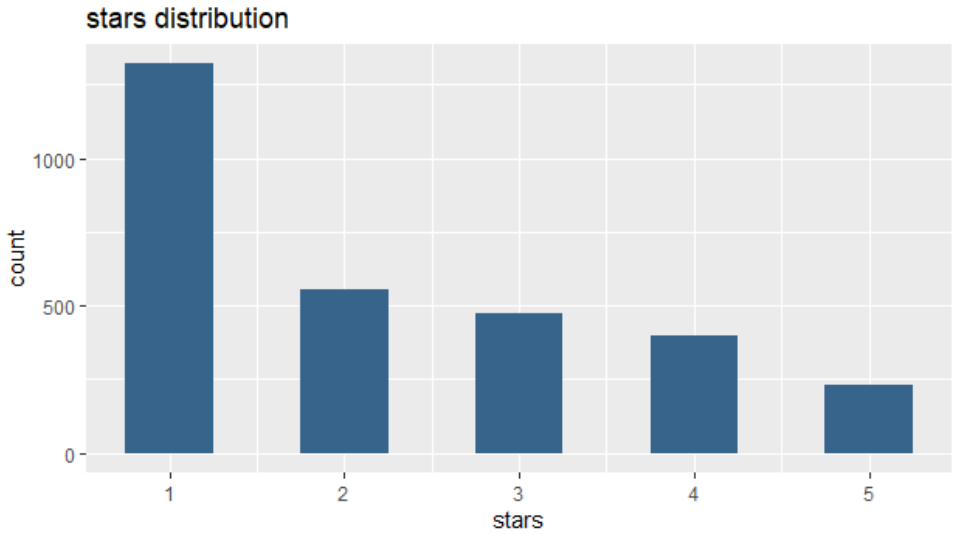
One last thing we notice, is that “Vegan” becomes way more important, compared to the other two categories of restaurants.

4.4. Analysis on a single restaurant (model application)

In this final part we want to try to implement our model of sentiment analysis, taking into consideration a single restaurant, in order to understand which are the main drivers of its performance.

The business we selected is “Circus Circus Las Vegas Hotel and Casino”, which belongs to the Italian restaurants sub-sample. Our choice was driven by the following reasons:

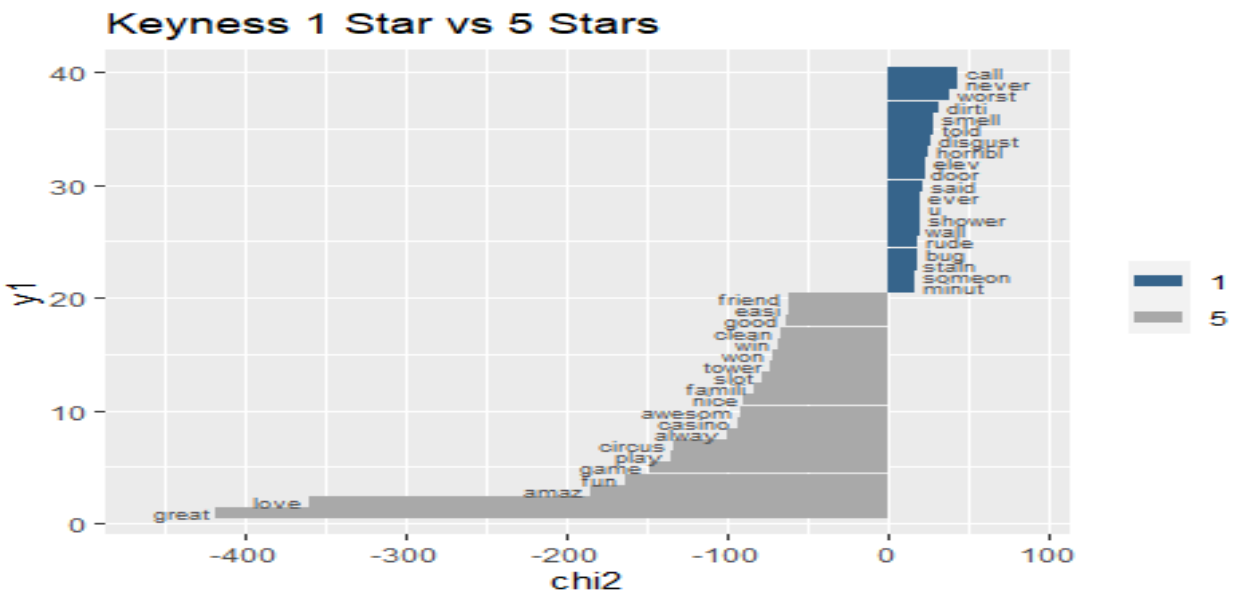
1) Availability of a very high number of reviews on this restaurant (2977);
2) Large number of negative reviews, and it would be useful to detect the main weaknesses (or strengths) of its performance, without fully reading almost three thousand reviews;



3) Even if belonging to the Italian restaurants’ subset, as you can guess from its name, “Circus Circus Las Vegas Hotel and Casino” is also a hotel, providing their customers with a wide range of services. This would make our analysis even more interesting, investigating even further drivers.

After having extracted the sample of reviews, we want to have a clearer picture of our sample before running the sentiment analysis.

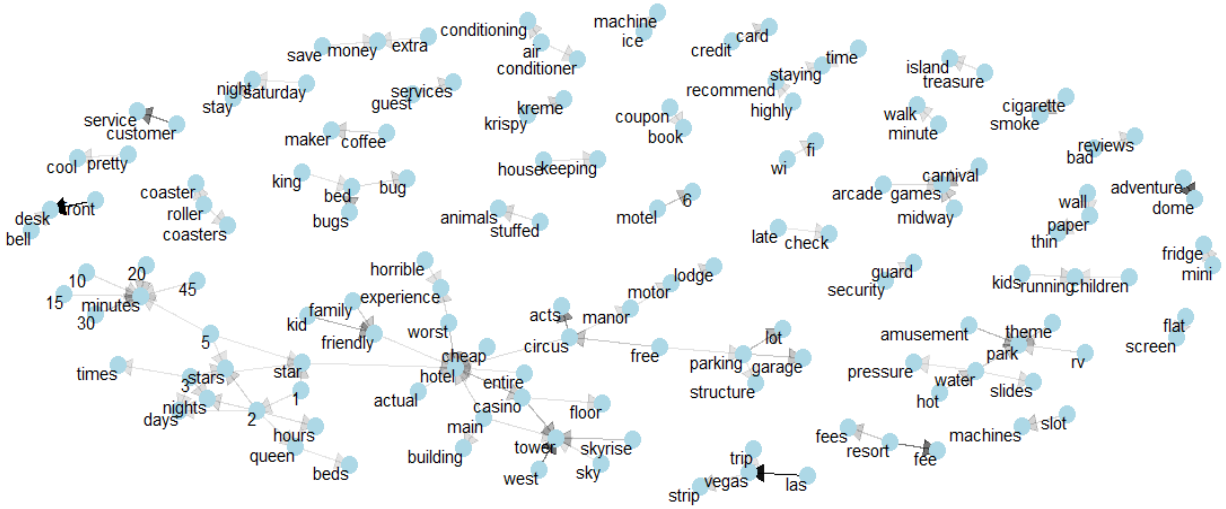
First, we want to look at the words’ frequencies, comparing the extreme reviews of 1 and 5 stars. We then decide to display a Keyness graph, which shows the words relatively more frequent in one of the two rating subsets (1 and 5 stars). In order to do it in a more reliable way, we eliminate numbers, punctuation, symbols, and stopwords. We also eliminate the words with a frequency lower than 1% and higher than 80%, to get rid of meaningless words, such as typing errors and words appearing too many times.



The fact that some words are cut, such as “amaz” comes from the stemming procedure that we ran, reducing inflected words to their word stem, base or root form. Some words are of course common adjectives which express very positive or negative feelings, such as “great”, “love” for the 5-star reviews, “worst” and “horrible” for the 1-star reviews.

However, we can already gain some insights on the main strengths and weaknesses of the business. Some recurring topics for the 1-star reviews seem to be related to the location, with words such as “dirty”, “smell”, “elevator”, “shower”. For what concerns 5-star reviews, some features related to the entertainment seem to be very much appreciated, like “game”, “casino”, “tower”, “play”.

Now we want to focus instead on the relationship between words. In order to do so, we work with bigrams, i.e. associations of two words, and we see which are the most recurring ones. We then display them on a network graph:



We can immediately notice that there is again a huge importance of the entertainment side, linked to the presence of an amusement park inside the structure, expressed through words such as: “tower”, “skyrise”, “carnival”, “adventure dome”, “island treasure”. Some relevant concerns seem to be related to the poor cleaning of the room. Other meaningful relationships which cannot be clearly classified, are linked to: timing, pricing, parking, wi-fi, air conditioning, security and customer service.

Now we are ready to implement the sentiment analysis, updated with all the information found in this preliminary analysis. This means that, in addition to the drivers already used, we decide to add some specific drivers: “Rooms”, which reflects the quality and the cleaning of the rooms in the hotel; “Parking”, which reflects the functioning of the parking lot; “Entertainment”, which captures the tones related with the amusement park, the casino and other entertaining activities; “Security”, which is about the level of security in the building.

```
table

##      stars        total        food       price    location        rooms
## [1,]     1 -0.088580653 -0.06097373 -0.04818259 -0.07849066 -0.117227535
## [2,]     2  0.008938912  0.03072043  0.01653637  0.00587059 -0.004063179
## [3,]     3  0.103557782  0.11839546  0.09242402  0.10644461  0.101806202
## [4,]     4  0.207921544  0.24907552  0.19250673  0.19259688  0.229298086
## [5,]     5  0.264101951  0.29090691  0.29822003  0.24164767  0.306298692
##          service      timing       drink     parking entertainment    security
## [1,] -0.08862360 -0.03330987 -0.04922915 -0.11189458   -0.03031075 -0.09168327
## [2,]  0.03216975  0.02102203  0.01881521 -0.03026853    0.05469081  0.03171747
## [3,]  0.16512995  0.08925188  0.08668056  0.06460098    0.13671110  0.08731685
## [4,]  0.29337919  0.16921764  0.17788587  0.15832870    0.21985563  0.25775111
## [5,]  0.45637194  0.20805118  0.29850616  0.27788558    0.26124097  0.22803531
```

As we can expect from the average star rate of this business, which is around 2, the average sentiments are quite low, compared to the sectorial averages that we found in the previous part of our analysis.

First of all, the most discriminant factors, the ones with the highest spread, are “Service” above all, but also “Rooms”, and “Food”. Secondly, let’s look at the negative drivers. Even though all the drivers show low sentiments for the low ratings, there are 3 of them whose sentiment is way lower than the total average. They are “Rooms” and “Parking” (which show negative sentiments even for the 2-star reviews), and “Security”. Finally, by looking at the high-rated reviews, we can observe that the more positive drivers are “Service”, “Rooms”, and “Food” (which displays a quite high sentiment even in the 4-star reviews).

All In all, these results can be interpreted by the owners of Circus Las Vegas Hotel and Casino in the following ways:

- If the owner wants to increase the average rating, he should focus first of all on “Rooms”, “Parking” and “Security”, since they are the factors which mostly push down the sentiment in the negative reviews. Hence, he/she should manage to provide their guests with cleaner and better rooms, a more efficient parking service, and with a safer overall environment.
- The owner should notice that when he/she receives high grades, it is mostly due to the high sentiment in “Service” above all, and “Rooms”. However, the same factors display very negative sentiments for the low rated reviews. This means that, the customers of Circus Circus Las Vegas Hotel and Casino are very sensitive to room quality and customer service. Indeed, even though in this fields there seems to be many issues, especially in rooms, when the staff does instead a good job, the sentiment goes up by a lot, way more than any other driver, in particular with the service. Therefore, a lot of attention should be paid to these aspects, trying to provide a constantly good performance.
- Even though they are not impressive, “Food”, “Drink” and “Price” show constantly higher average sentiment compared to the other drivers. This means that in the quality of food and drink, and in their prices, the business is doing relatively well.

- Surprisingly, we cannot say much about “Entertainment”, whose sentiment is aligned with the average.
- “Timing” shows a very narrow spread, and almost constant sentiment. This is due either to the fact that performance of the business is very constant and modest in this matter, or to the fact that customers do not care much about it and do not mention it very often with extreme tones in the reviews.

5. Conclusions

To summarize our findings, in our preliminary analysis of the whole reviews sample we find some significant relation between the length of the document and its sentiment. It is very likely, indeed, that most extreme tones, hence most extreme comments, are expressed through shorter reviews. As the review gets longer, the overall tone tends to mitigate. This might also be linked to the fact that most important things tend to be mentioned first. Moreover, it is not as easy to find specific patterns for subsamples of reviews belonging to the same star score, but this might be due to some kind of misspecification of the model we used, and also to some recurring errors of the data itself. Regardless, this helped us to clean the dataset in a more efficient way, in order to have it as ready as possible for our core analysis.

The different factors driving reviews' stars can be studied through some sort of diversified sentiment analysis which considers the sample as a whole and then step by step isolates each single factor. We carry out said analysis with some specific method we propose, and we apply it to three main restaurant categories: Chinese, Italian and Fast Foods. We find that in general the quality of the service is a huge discriminant in the rating assignment. In addition, in Chinese restaurants customers care more about the price rather than the quality of the food. Italian restaurants have overall higher sentiments, and an increase importance of the quality of food and drink. Fast foods give more mixed results, but we can infer that here customers care more about a good delivery and drive-in service than the quality of the food. Also, quite surprisingly, the presence of vegan food in the menu becomes more important.

Following the same procedure, we can work at a single restaurant level and try to build a model which helps the owner in the proper management of the business. We carry out the same analysis on an Italian restaurant located in an hotel, trying to make it as tailored as possible to that specific case to obtain more meaningful results, and again we get satisfying outcomes. We can say that we are able to efficiently suggest what the owner should focus on to increase the performance of its business.

5.1. Limitations and further suggestions

The model still presents some limitations, like the fact that results are not always crystal clear and easy to interpret. This issue could be at least partially solved looking for a dictionary, which is at the base of the sentiment analysis, always more accurate and pertinent to our specific dataset. Also, additional time can be spent increasing the number of drivers and improving their quality in terms of specific words associated to each driver.

As a nice way to improve the model, we suggest adding the temporal component to the reviews, meaning the day in which they were written. By doing so you can first of all run some analysis on the evolution of the sentiments throughout time, maybe to see what is changing in the way customers write reviews about the restaurants. This would provide important information beside the mere change of the average star score. Furthermore, you can build a similar sentiment model weighting each review, and thus the relative sentiments, by how far back in time it has been written. Of course, more recent reviews will have a higher weight. This is in order to have a model that is even better able to capture the actual drivers of today's average sentiments of the restaurant, and to provide more accurate information to the owner.

6. R code markdown

Data collection and sample preparation

```
library(jsonlite)

## Warning: package 'jsonlite' was built under R version 4.0.3

library(data.table)

## Warning: package 'data.table' was built under R version 4.0.3

library(stringr)

## Warning: package 'stringr' was built under R version 4.0.3

library(igraph)

## Warning: package 'igraph' was built under R version 4.0.3

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(ggraph)

## Warning: package 'ggraph' was built under R version 4.0.3

## Loading required package: ggplot2

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.3

## -- Attaching packages ----------------------- tidyverse 1.3.0 --

## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.2     v dplyr   1.0.2
## v readr   1.4.0     v forcats 0.5.0

## Warning: package 'tidyr' was built under R version 4.0.3

## Warning: package 'readr' was built under R version 4.0.3

## Warning: package 'purrr' was built under R version 4.0.3

## Warning: package 'dplyr' was built under R version 4.0.3

## Warning: package 'forcats' was built under R version 4.0.3
```

```
## -- Conflicts --------------------------- tidyverse_conflicts() --
## x dplyr::as_data_frame() masks tibble::as_data_frame(), igraph::as_dat
a_frame()
## x dplyr::between()       masks data.table::between()
## x purrr::compose()       masks igraph::compose()
## x tidyr::crossing()      masks igraph::crossing()
## x dplyr::filter()        masks stats::filter()
## x dplyr::first()         masks data.table::first()
## x purrr::flatten()       masks jsonlite::flatten()
## x dplyr::groups()        masks igraph::groups()
## x dplyr::lag()           masks stats::lag()
## x dplyr::last()          masks data.table::last()
## x purrr::simplify()      masks igraph::simplify()
## x purrr::transpose()     masks data.table::transpose()
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:igraph':
##
##     %--%, union
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(DT)
```

```
## Warning: package 'DT' was built under R version 4.0.3
```

```
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(leaflet)
```

```
## Warning: package 'leaflet' was built under R version 4.0.3
```

#import the dataset business and convert it into datatable#

```
setwd("C:/Users/chris/Desktop/Bocconi/Big Data for Business Decisions/Gro
up Work")
Business = stream_in(file("yelp_academic_dataset_business.json"))

## opening file input connection.

## closing file input connection.

setDT(Business)

## Warning in setDT(Business): Some columns are a multi-column type (such
as a
## matrix column): [12, 14]. setDT will retain these columns as-is but su
bsequent
## operations like grouping and joining may fail. Please consider as.data
.table()
## instead which will create a new column for each embedded column.
```

#delete all the unnecessary columns#

```
business_11columns=subset(Business, select=c(business_id, name, address,
city, state, postal_code, latitude, longitude, stars, review_count, categ
ories))
head(business_11columns)

##               business_id                         name
address
## 1: f9NumwFMBDn751xgFiRbNA    The Range At Lake Norman           10913
Bailey Rd
## 2: Yzvjg0SayhoZgCljUJRF9Q          Carlos Santo, NMD 8880 E Via Linda
, Ste 107
## 3: XNoUzKckATkOD1hP6vghZg                      Felinus     3554 Rue Not
re-Dame O
## 4: 6OAZjbxqM5ol29BuHsil3w        Nevada House of Hose            1015
Sharp Cir
## 5: 51M2Kk903DFYI6gnB5I6SQ     USE MY GUY SERVICES LLC        4827 E Do
wning Cir
## 6: cKyLV5oWZJ2NudWgqs8VZw Oasis Auto Center - Gilbert 1720 W Elliot Rd
, Ste 105
##               city state postal_code latitude  longitude stars review_
count
## 1:       Cornelius    NC       28031 35.46272  -80.85261   3.5
36
## 2:      Scottsdale    AZ       85258 33.56940 -111.89026   5.0
4
## 3:        Montreal    QC     H4C 1P4 45.47998  -73.58007   5.0
5
## 4: North Las Vegas    NV       89030 36.21973 -115.12773   2.5
3
## 5:            Mesa    AZ       85205 33.42807 -111.72665   4.5
26
## 6:         Gilbert    AZ       85233 33.35040 -111.82714   4.5
```

```
38
##                                                                    cat
egories
## 1:                       Active Life, Gun/Rifle Ranges, Guns & Ammo, S
hopping
## 2:        Health & Medical, Fitness & Instruction, Yoga, Active Life,
Pilates
## 3:                                              Pets, Pet Services, Pet G
roomers
## 4: Hardware Stores, Home Services, Building Supplies, Home & Garden, S
hopping
## 5:               Home Services, Plumbing, Electricians, Handyman, Cont
ractors
## 6:          Auto Repair, Automotive, Oil Change Stations, Transmission
Repair
```

#create a list of all the words contained in the column categories#

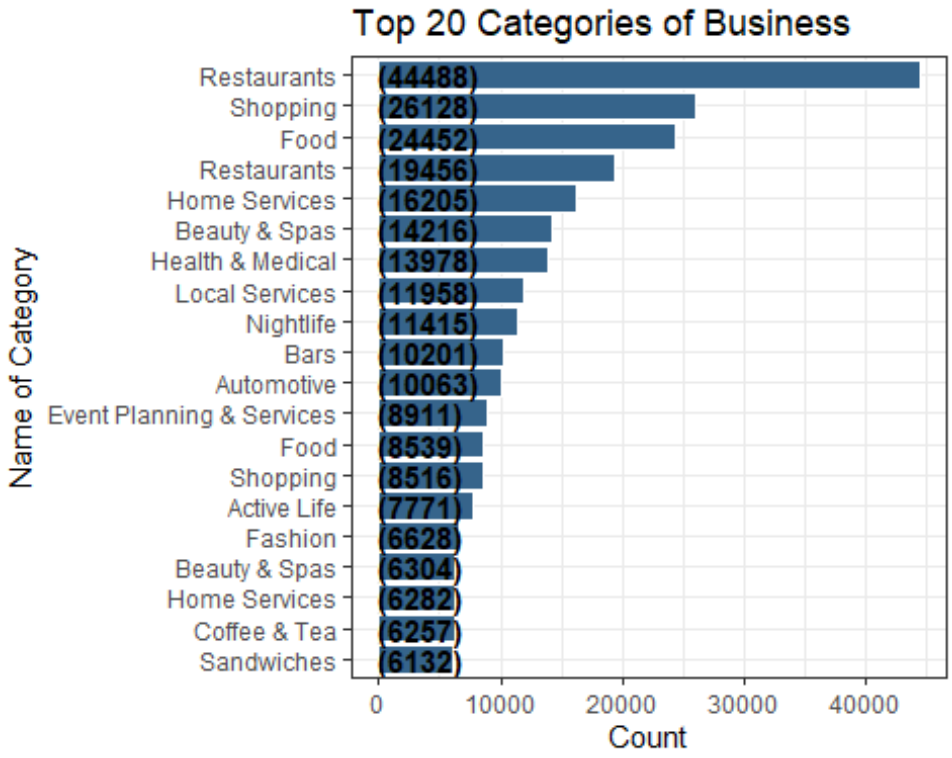
```
categories = str_split(business_11columns$categories,",")
categories = as.data.frame(unlist(categories))
colnames(categories) = c("Name")
head(categories)
```

```
##                     Name
## 1            Active Life
## 2       Gun/Rifle Ranges
## 3            Guns & Ammo
## 4               Shopping
## 5       Health & Medical
## 6  Fitness & Instruction
```

#group and plot the top 20 categories#

```
categories %>%
  group_by(Name) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  ungroup() %>%
  mutate(Name = reorder(Name,Count)) %>%
  head(20) %>%


  ggplot(aes(x = Name,y = Count)) +
  geom_bar(stat='identity',colour="white",fill="steelblue4") +
  geom_text(aes(x = Name, y = 1, label = paste0("(",Count,")",sep="")),
            hjust=0, vjust=.5, size = 4, colour = 'black',
            fontface = 'bold') +
  labs(x = 'Name of Category', y = 'Count',
       title = 'Top 20 Categories of Business') +
  coord_flip() +
  theme_bw()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



#remove all the categories different from restaurants#

```
business_rest=business_11columns[business_11columns$categories %like% "Restaurants",]
```

#remove the word restaurants from the categories#

```
categories=data.frame(lapply(categories,function(x){gsub("Restaurants","",x)}))
business_semifinal=data.frame(lapply(business_rest,function(x){gsub("Restaurants, ","",x)}))
business_final=data.frame(lapply(business_semifinal,function(x){gsub(", Restaurants","",x)}))
```

#remove all the spaces and substitute them with underscores#

```
new_nospaces=lapply(business_final$categories,function(x){gsub(", ",",",x)})
new_nospaces=lapply(new_nospaces,function(x){gsub(" ","_",x)})
business_final$categories=new_nospaces
```

#remove the word food from the categories column#

```
business_superfinal=data.frame(lapply(business_final,function(x){gsub("Food,","",x)}))
business_superfinal=data.frame(lapply(business_superfinal,function(x){gsub(",Food","",x)}))
```

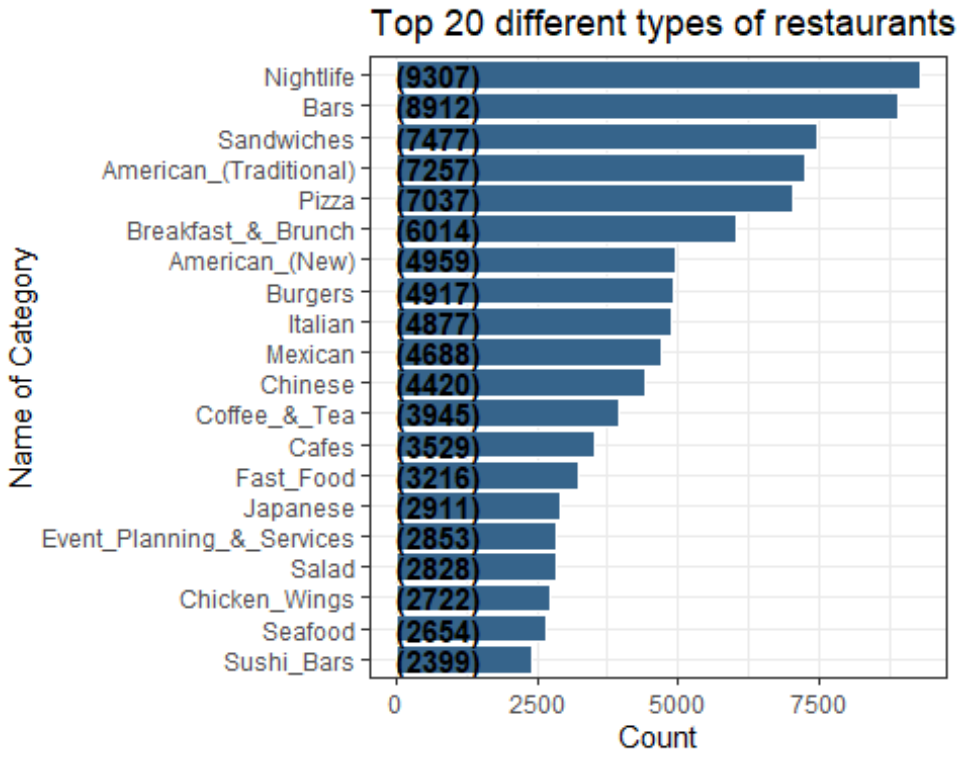
#create a new dataframe with all the cleaned categories#

```
categories_newversion = str_split(business_superfinal$categories,",")
categories_newversion = as.data.frame(unlist(categories_newversion))
colnames(categories_newversion) = c("Name1")
```

#list and plot the new cleaned categories#

```
categories_newversion %>%
  group_by(Name1) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  ungroup() %>%
  mutate(Name = reorder(Name1,Count)) %>%
  head(20)  %>%

  ggplot(aes(x = Name,y = Count)) +
  geom_bar(stat='identity',colour="white",fill="steelblue4") +
  geom_text(aes(x = Name, y = 1, label = paste0("(",Count,")",sep="")),
            hjust=0, vjust=.5, size = 4, colour = 'black',
            fontface = 'bold') +
  labs(x = 'Name of Category', y = 'Count',
       title = 'Top 20 different types of restaurants') +
  coord_flip() +
  theme_bw()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

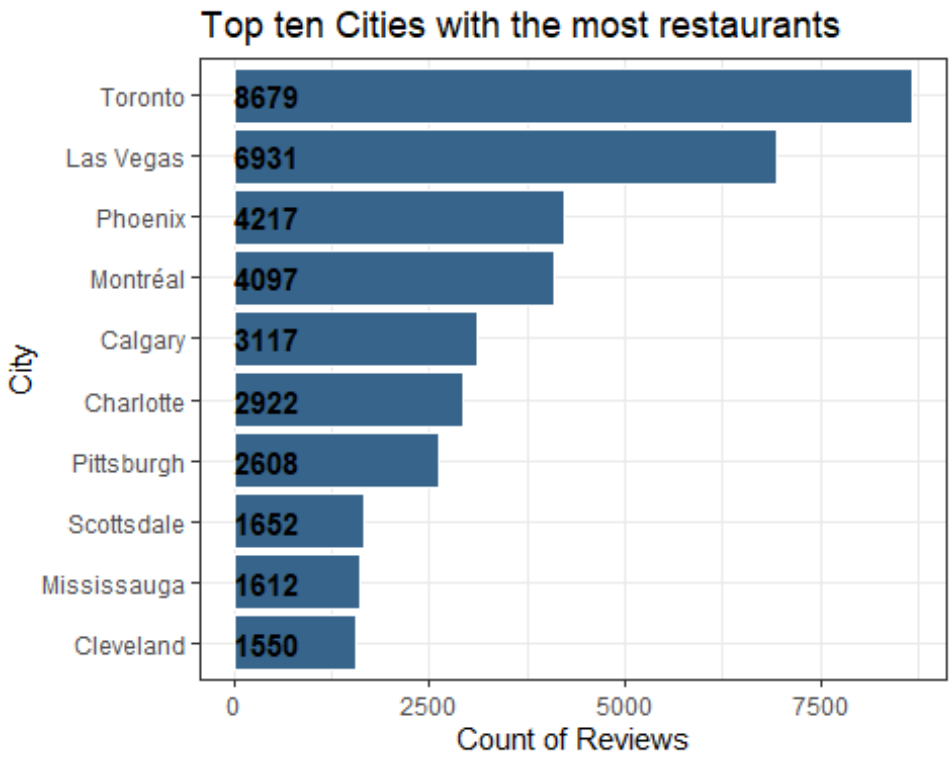


#plot an istogram with the 10 most common cities#

```
business_superfinal %>%
  group_by(city) %>%
  summarise(Count = n()) %>%
```

```
  arrange(desc(Count)) %>%
  ungroup() %>%
  mutate(City = reorder(city,Count)) %>%
  head(10) %>%

  ggplot(aes(x = City,y = Count)) +
  geom_bar(stat='identity',colour="white",fill="steelblue4") +
  geom_text(aes(x = City, y = 1, label = paste0(round(Count),sep="")),
            hjust=0, vjust=.5, size = 4, colour = 'black',
            fontface = 'bold') +
  labs(x = 'City', y = 'Count of Reviews',
       title = 'Top ten Cities with the most restaurants') +
  coord_flip() +
  theme_bw()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

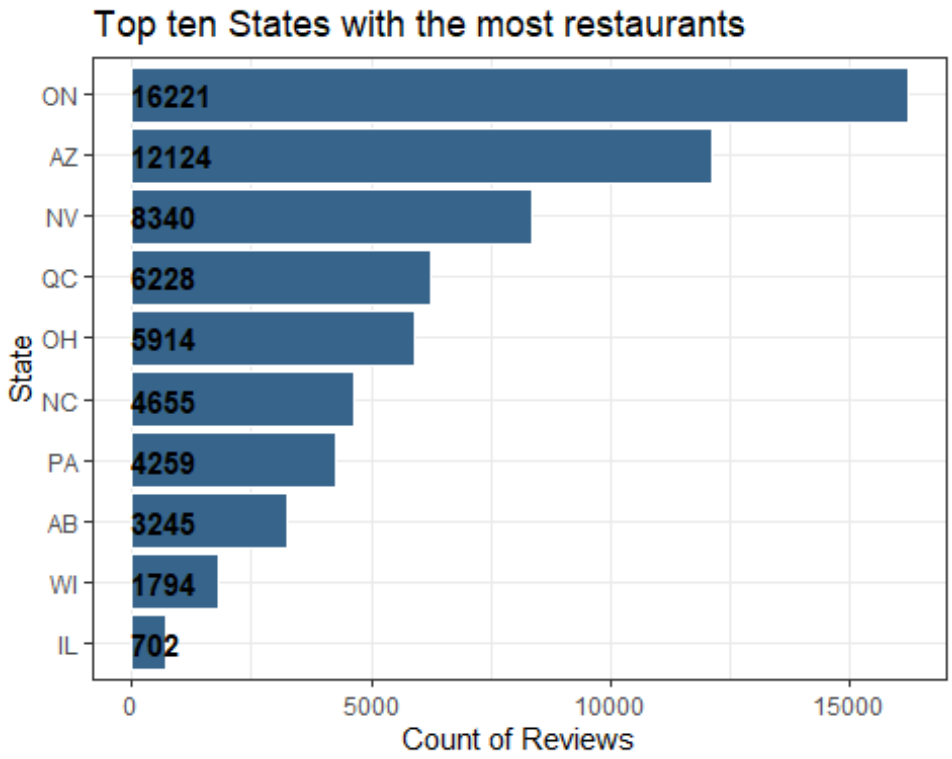


#plot an istogram with the 10 most common states#

```
business_superfinal %>%
  group_by(state) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  ungroup() %>%
  mutate(State = reorder(state,Count)) %>%
  head(10) %>%

  ggplot(aes(x = State,y = Count)) +
  geom_bar(stat='identity',colour="white",fill="steelblue4") +
  geom_text(aes(x = State, y = 1, label = paste0(round(Count),sep="")),
```

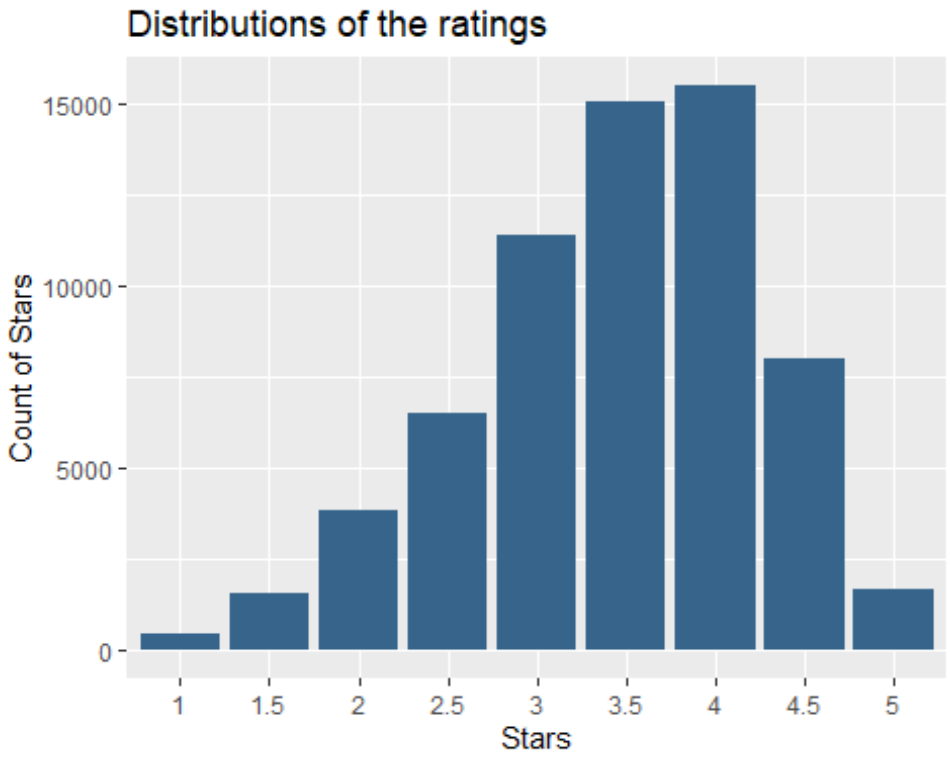
```
            hjust=0, vjust=.5, size = 4, colour = 'black',
            fontface = 'bold') +
  labs(x = 'State', y = 'Count of Reviews',
       title = 'Top ten States with the most restaurants') +
  coord_flip() +
  theme_bw()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



#plot of the distribution of the stars#

```
business_superfinal %>% ggplot(aes(x=as.factor(stars))) +
  geom_bar(fill="steelblue4") +
  labs(x = 'Stars', y = 'Count of Stars',
       title = 'Distributions of the ratings')
```



#create a dataset with just vegas restaurants#

```
business_vegas=business_superfinal[business_superfinal$city == "Las Vegas
",]
```

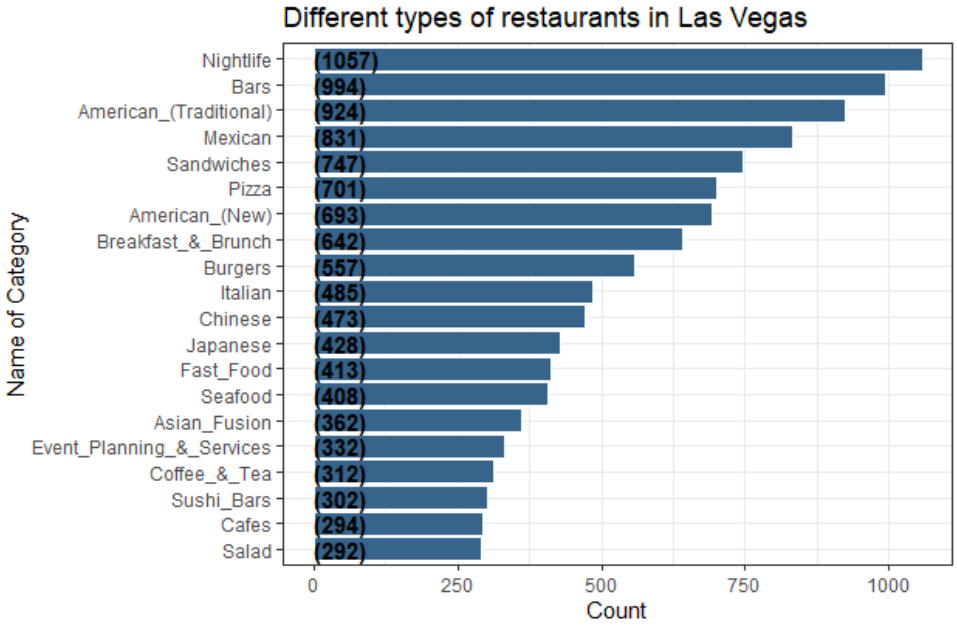
#list the categories of restaurants in vegas#

```
categories_vegas = str_split(business_vegas$categories,",")
categories_vegas = as.data.frame(unlist(categories_vegas))
colnames(categories_vegas) = c("Name2")
```

#plot the vegas categories#

```
categories_vegas %>%
  group_by(Name2) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  ungroup() %>%
  mutate(Name = reorder(Name2,Count)) %>%
  head(20)  %>%

  ggplot(aes(x = Name,y = Count)) +
  geom_bar(stat='identity',colour="white",fill="steelblue4") +
  geom_text(aes(x = Name, y = 1, label = paste0("(",Count,")",sep="")),
            hjust=0, vjust=.5, size = 4, colour = 'black',
            fontface = 'bold') +
  labs(x = 'Name of Category', y = 'Count',
       title = 'Different types of restaurants in Las Vegas') +
  coord_flip() +
  theme_bw()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



#pick 3 different categories in vegas: Chinese, Italian and Fast Food#

#clean the categories columns and delete the words different from the selected categories#

```
business_chinese=business_vegas[business_vegas$categories %like% "Chinese
",]
business_chinese=data.frame(lapply(business_chinese,function(x){gsub("Chi
nese_","",x)}))
business_chinese=data.frame(lapply(business_chinese,function(x){gsub("_Ch
inese","",x)}))
business_chinese=business_chinese[business_chinese$categories %like% "Chi
nese",]

business_italian=business_vegas[business_vegas$categories %like% "Italian
",]
business_italian=data.frame(lapply(business_italian,function(x){gsub("Ita
lian_","",x)}))
business_italian=data.frame(lapply(business_italian,function(x){gsub("_It
alian","",x)}))
business_italian=business_italian[business_italian$categories %like% "Ita
lian",]

business_ffood=business_vegas[business_vegas$categories %like% "Fast_Food
",]
business_ffood=data.frame(lapply(business_ffood,function(x){gsub("Fast_Fo
od_","",x)}))
business_ffood=data.frame(lapply(business_ffood,function(x){gsub("_Fast_F
ood","",x)}))
business_ffood=business_ffood[business_ffood$categories %like% "Fast_Food
",]
```

#substitute the useless categories with only a single word of the selected ones#

```
business_chinese$categories=strrep("Chinese", 1)
business_italian$categories=strrep("Italian", 1)
business_ffood$categories=strrep("Fast_Food", 1)
```

#aggregate the datasets to remove duplicates with 2 or more categories#

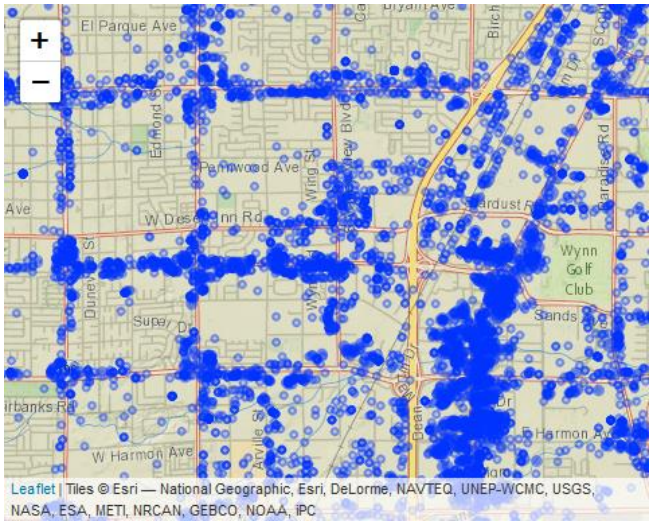
```
business_3types=rbind(business_italian,business_chinese,business_ffood)
business_3types_noduplicates=business_3types[!duplicated(business_3types$
business_id),]
```

#split again into the 3 datasets#

```
Business_chinese_final=business_3types_noduplicates[business_3types_nodup
licates$categories == "Chinese",]
Business_italian_final=business_3types_noduplicates[business_3types_nodup
licates$categories == "Italian",]
Business_ffood_final=business_3types_noduplicates[business_3types_nodupli
cates$categories == "Fast_Food",]
```

#map of the restaurants in vegas#

```
LasvegasCoords = Business %>% filter(city == "Las Vegas")

center_lon = median(LasvegasCoords$longitude,na.rm = TRUE)
center_lat = median(LasvegasCoords$latitude,na.rm = TRUE)

leaflet(LasvegasCoords) %>% addProviderTiles("Esri.NatGeoWorldMap") %>%
  addCircles(lng = ~longitude, lat = ~latitude,radius = ~sqrt(review_coun
t))  %>%
  setView(lng=center_lon, lat=center_lat,zoom = 13)
```



#review part and choice of the seingle restaurant for the last analysis#
#Substitute characters with numerical values in the column review_count#

```
Business_chinese_final$review_count=as.numeric(Business_chinese_final$rev
iew_count)
Business_italian_final$review_count=as.numeric(Business_italian_final$rev
iew_count)
```

```
Business_ffood_final$review_count=as.numeric(Business_ffood_final$review_
count)
```

#order the dataset by most reviewed#

```
Business_ffood_final=Business_ffood_final[order(Business_ffood_final$revi
ew_count, decreasing = TRUE),]
Business_italian_final=Business_italian_final[order(Business_italian_fina
l$review_count, decreasing = TRUE),]
Business_chinese_final=Business_chinese_final[order(Business_chinese_fina
l$review_count, decreasing = TRUE),]
```

#extract the most reviewed restaurant for each category#

```
head(Business_ffood_final[1:3,])
```

```
##                 business_id                  name               addres
s
## 642  nVAJZ6BJ9PPlxDXn976R6A       In-N-Out Burger   4888 Dean Martin D
r
## 982  T0Uw6vwwfO3el29wBoDamQ       Pink's Hot Dogs 3667 Las Vegas Blvd
S
## 2461 5ma1OUQWy_Ds80xUiP7JPQ Fuku Burger Chinatown     3429 S Jones Blv
d
##           city state postal_code   latitude    longitude stars review_
count
## 642  Las Vegas    NV       89103 36.1013195 -115.1821862     4
1164
## 982  Las Vegas    NV       89109  36.109413  -115.172353     3
940
## 2461 Las Vegas    NV       89146  36.126649  -115.225307   4.5
630
##      categories
## 642   Fast_Food
## 982   Fast_Food
## 2461  Fast_Food
```

```
head(Business_italian_final[1:3,])
```

```
##                business_id                                        name
## 494 G-5kEa6E6PD5fkBRuA7k9Q                                      Giada
## 400 6Q7-wkCPc1KF75jZLOTcMw Circus Circus Las Vegas Hotel and Casino
## 279 NvKNe9DnQavC9GstglcBJQ                           Grand Lux Cafe
##                       address      city state postal_code      latitud
e
## 494 3595 Las Vegas Blvd South Las Vegas    NV       89109 36.115059207
2
## 400     2880 S Las Vegas Blvd Las Vegas    NV       89109    36.137641
6
## 279     3355 Las Vegas Blvd S Las Vegas    NV       89109     36.12208
9
##           longitude stars review_count categories
```

```
## 494 -115.1721090426   3.5         3421    Italian
## 400    -115.1653861     2         3037    Italian
## 279     -115.168031     4         3015    Italian
```

```
head(Business_chinese_final[1:3,])
```

```
##                 business_id                   name
## 2901 yfxDa8RFOvJPQh0rNtakHA           Pho Kim Long
## 3171 pH0BLkL4cbxKzu471VZnuA SUSHISAMBA - Las Vegas
## 1751 X8c23dur0ll2D9XTu-I8Qg     Hakkasan Nightclub
##                                            address      city state
## 2901                           4029 Spring Mountain Rd Las Vegas    NV
## 3171                             3327 Las Vegas Blvd S Las Vegas    NV
## 1751 Mgm Grand Hotel And Casino, 3799 Las Vegas Blvd S Las Vegas    NV
##      postal_code      latitude       longitude stars review_count cate
gories
## 2901       89102 36.1261943318  -115.193445287   3.5         3398    C
hinese
## 3171       89109 36.1245818974 -115.1675528772     4         2636    C
hinese
## 1751       89109     36.101375     -115.172452   2.5         1914    C
hinese
```

#clean global environment#

```
library(gdata)
```

```
## Warning: package 'gdata' was built under R version 4.0.3
```

```
## Warning in system(cmd, intern = intern, wait = wait | intern,
## show.output.on.console = wait, : running command 'C:\Windows\system32\
cmd.exe /c
## ftype perl' had status 2

## Warning in system(cmd, intern = intern, wait = wait | intern,
## show.output.on.console = wait, : running command 'C:\Windows\system32\
cmd.exe /c
## ftype perl' had status 2

## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLSX' (Excel 2007+) files.

##

## gdata: Run the function 'installXLSXsupport()'
## gdata: to automatically download and install the perl
## gdata: libaries needed to support Excel XLS and XLSX formats.

##
## Attaching package: 'gdata'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine

## The following objects are masked from 'package:dplyr':
##
##     combine, first, last

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

## The following objects are masked from 'package:data.table':
##
##     first, last

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##     startsWith

keep(Business_ffood_final,Business_chinese_final,Business_italian_final,s
ure=TRUE)
```

Review sample preparation

#we open the review file by tranches, opening only the columns of interest#

```
Rev_1 <- NULL
stream_in(
  file("yelp_academic_dataset_review.json"),
  pagesize=100000,
  handler=function(x) {
    Rev_1 <<- rbind(Rev_1, x[,c("review_id","text")])})

## using a custom handler function.

## opening file input connection.

## closing file input connection.

Rev_2 <- NULL
stream_in(
  file("yelp_academic_dataset_review.json"),
  pagesize=100000,
  handler=function(x) {
    Rev_2 <<- rbind(Rev_2, x[,c("review_id","business_id","stars")])})
```

```
## using a custom handler function.
## opening file input connection.

## closing file input connection.
```

#merging the two tranches to get the final dataset#

```
library(dplyr)
Reviews <- left_join(Rev_1, Rev_2, by = "review_id")
rm(Rev_1)
rm(Rev_2)
```

#chinese

```
businessid_chinese=Business_chinese_final["business_id"]
Reviews_chinese=left_join(businessid_chinese,Reviews,by="business_id")
```

#ffood

```
businessid_ffood=Business_ffood_final["business_id"]
Reviews_ffood=left_join(businessid_ffood,Reviews,by="business_id")
```

#italian

```
businessid_italian=Business_italian_final["business_id"]
Reviews_italian=left_join(businessid_italian,Reviews,by="business_id")

rm(Business_chinese_final)
rm(Business_ffood_final)
rm(Business_italian_final)
rm(businessid_chinese)
rm(businessid_ffood)
rm(businessid_italian)
```

#we add an additional column with the type of restaurant# #we will use it later on in the textual analysis#

```
Reviews_chinese$category=strrep("chinese",1)
Reviews_ffood$category=strrep("fast_food",1)
Reviews_italian$category=strrep("italian",1)
```

#aggregate the 3 datasets in one single dataset#

```
Reviews_final=rbind(Reviews_chinese,Reviews_ffood,Reviews_italian)

rm(Reviews_chinese)
rm(Reviews_ffood)
rm(Reviews_italian)
```

#erase not needed column#

```
Reviews_final=subset(Reviews_final,select = -c(review_id))
head(Reviews_final)
```

```
##              business_id
## 1 yfxDa8RFOvJPQh0rNtakHA
## 2 yfxDa8RFOvJPQh0rNtakHA
## 3 yfxDa8RFOvJPQh0rNtakHA
## 4 yfxDa8RFOvJPQh0rNtakHA
## 5 yfxDa8RFOvJPQh0rNtakHA
## 6 yfxDa8RFOvJPQh0rNtakHA
##
text
## 1
I've been here several times because it's the go to place for my family g
rowing up. I'm a bigger fan of the rice dishes, especially com ga roti (r
oasted chicken and rice)  than the soups. I had bun bo hue for the first
time at the place and the broth was watered down. They didn't give any po
rks blood either which is a classic component for this dish (even if some
are repulsed by it). They barely also give any meat. The pho was better b
ut I wouldn't call it the best. It does its job if you're craving pho aft
er a night out of having fun though. 3 Stars for its decency and the sent
imental value the place has for my fam.
## 2
This place deserve 5 stars. Service was good, clean and most important th
e pho was good. Tasty soup base; not salty. And generous portion.
## 3
My first time in this restaurant... The pho was not good but they charged
me $10.00 because I wanted my rare beef on the side ( not in bowl ).. And
I just waisted my money on the deep fried tofu too...
## 4
I'm a frequent visitor of Pho Kim Long but they're not normally my go-to
spot. The convenience of being open 24 hours is typically what wins me ov
er. \n\nI am deeply addicted to the eggs rolls. Those lettuce wrapped egg
rolls do it in for me all the time! My girlfriend usually gets the pho. I
'm not a fan of pho so this review is not for the pho alone. I've had the
duck a few times here and it's a hit or miss. I'll either get all bone or
all skin. \n\nThe service is less than mediocre. You wait to be seated an
d the second you're seated you're rushed to put in your drink order. The
second your drinks arrive they pressure you to put in your food order. It
can get a tad uncomfortable.
## 5
The Pho was amazing, their friedrice is equally delicious, would definetl
y go back when i am back in Vegas!!!
## 6 I heard a lot about this place and had to check it out. The restaura
nt is clean but the smell of the Vietnamese incense were way too strong.
The staff were friend and spoke english and Vietnamese. \n\nI was sorta d
isappointed by the food. It was okay for being in Nevada. I got the pho d
ac biet and it was missing a lot of items that are usually in traditional
pho dac biet. The broth was very bland and not very flavorful. It wasn't
bad but it needed a little bit more flavor. However I did get eggrolls (c
ha gio). They came out pretty good. They were hot and crunchy. \n\nThe fo
od is overpriced for typical vietnamese food but judging from the locatio
n I guess it is acceptable. If you are from SoCal the food and price at t
```

```
he restaurant are not comparable to areas like Little Saigon or Alhambra.
\n\nOverall it was a decent experience. I wouldn't recommend it for peopl
e who know traditional vietnamese food, but I do recommend it for people
who are trying vietnamese food for the first time.
##   stars category
## 1     3  chinese
## 2     5  chinese
## 3     1  chinese
## 4     3  chinese
## 5     5  chinese
## 6     3  chinese
```

Preliminary text analysis

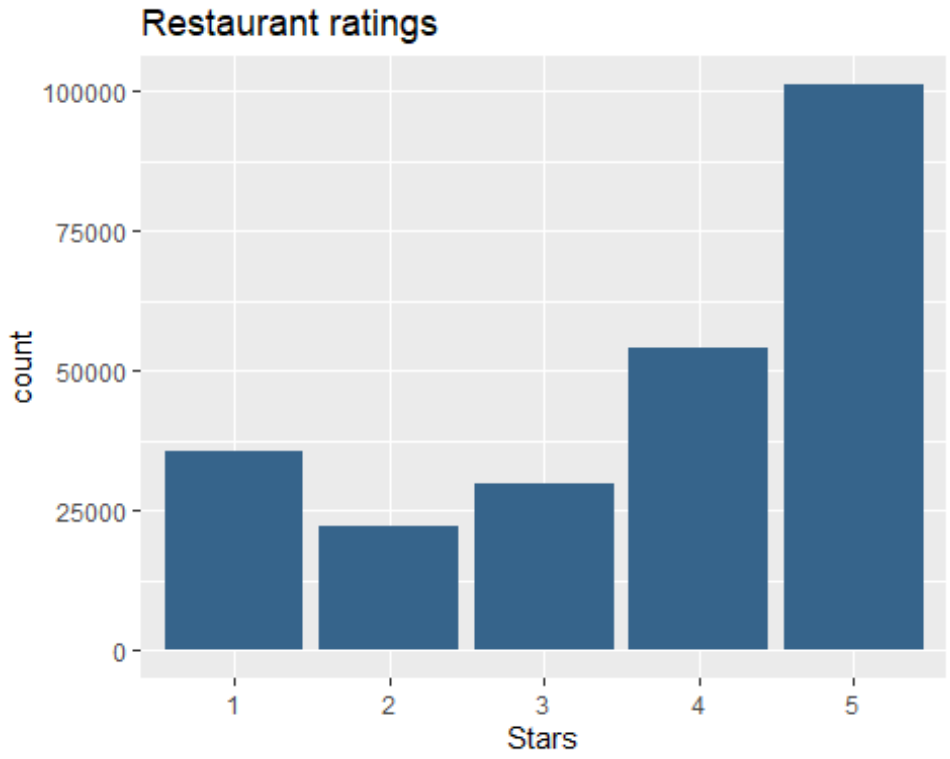
```
library( data.table )
library( readtext )

## Warning: package 'readtext' was built under R version 4.0.3

library( quanteda )

## Warning: package 'quanteda' was built under R version 4.0.3

## Package version: 2.1.2

## Parallel computing: 2 of 8 threads used.

## See https://quanteda.io for tutorials and examples.

##
## Attaching package: 'quanteda'

## The following object is masked from 'package:igraph':
##
##     as.igraph

## The following object is masked from 'package:utils':
##
##     View

library( topicmodels )

## Warning: package 'topicmodels' was built under R version 4.0.3

library( ggplot2 )
library( gghighlight )

## Warning: package 'gghighlight' was built under R version 4.0.3

library( stringr )
library( textstem )

## Warning: package 'textstem' was built under R version 4.0.3

## Loading required package: koRpus.lang.en
```

```
## Warning: package 'koRpus.lang.en' was built under R version 4.0.3

## Loading required package: koRpus

## Warning: package 'koRpus' was built under R version 4.0.3

## Loading required package: sylly

## Warning: package 'sylly' was built under R version 4.0.3

## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()

##
## Attaching package: 'koRpus'

## The following objects are masked from 'package:quanteda':
##
##     tokens, types

## The following object is masked from 'package:readr':
##
##     tokenize

library( lexicon )

## Warning: package 'lexicon' was built under R version 4.0.3

library( sentimentr )

## Warning: package 'sentimentr' was built under R version 4.0.3

##
## Attaching package: 'sentimentr'

## The following object is masked from 'package:lexicon':
##
##     available_data

library( ggridges )
library( lemon )

## Warning: package 'lemon' was built under R version 4.0.3

##
## Attaching package: 'lemon'

## The following object is masked from 'package:purrr':
##
##     %||%
```

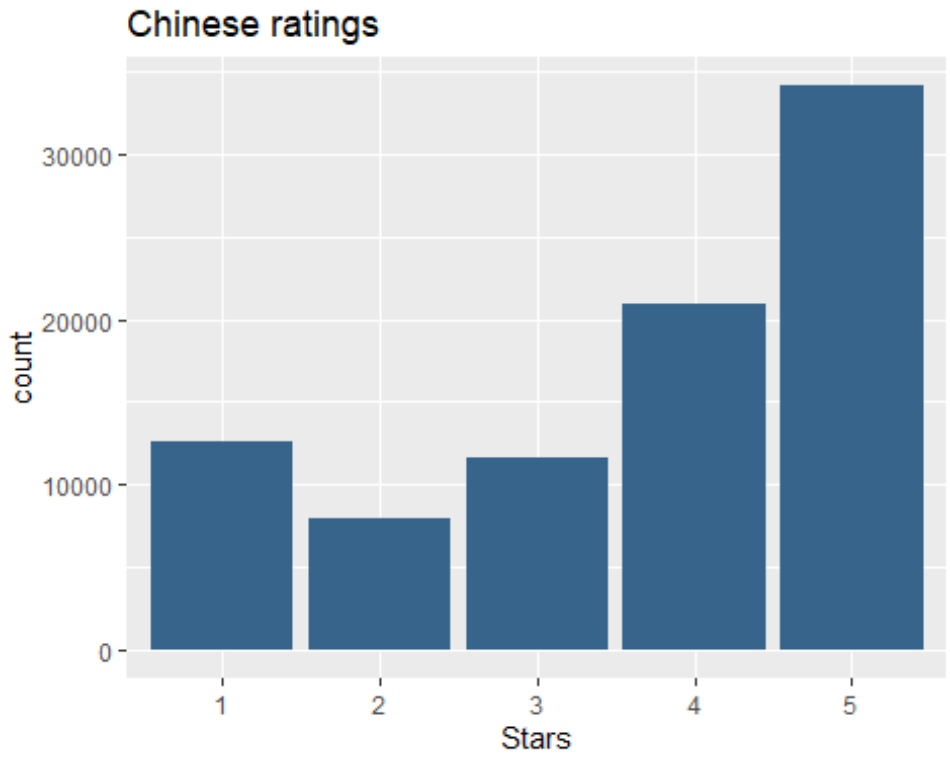
```
## The following objects are masked from 'package:ggplot2':
##
##     CoordCartesian, element_render

library( patchwork )
```

#star distribution of each category#

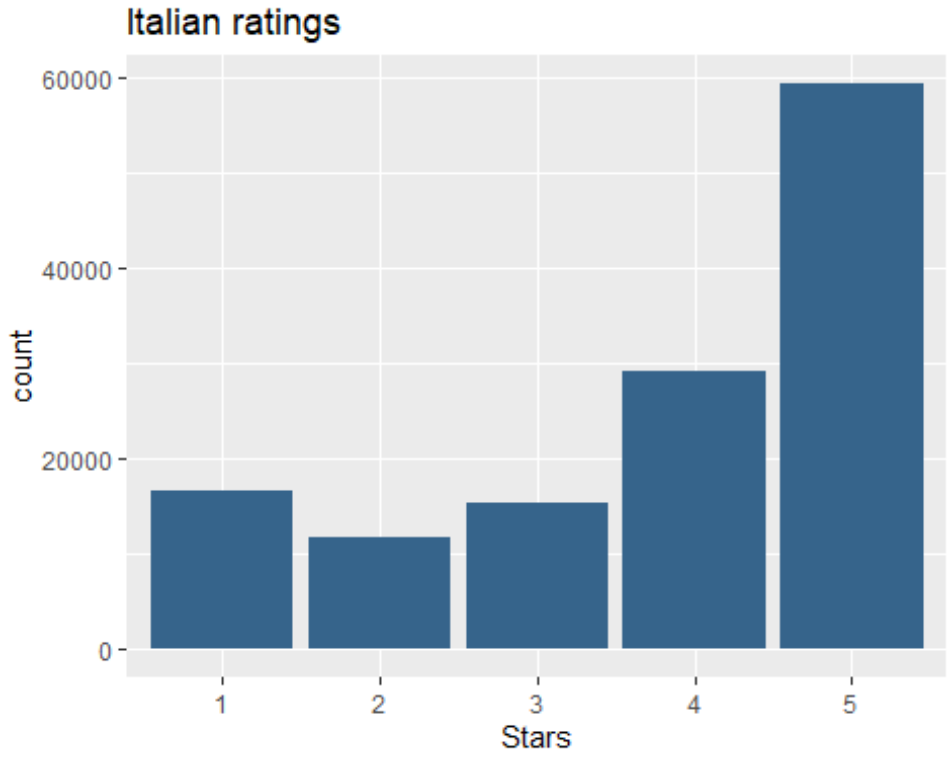
```
star_restaurants <- Reviews_final %>% ggplot(aes(x=as.factor(stars))) + g
eom_bar(fill = "steelblue4") + labs( x = "Stars") + ggtitle("Restaurant r
atings")
star_restaurants
```



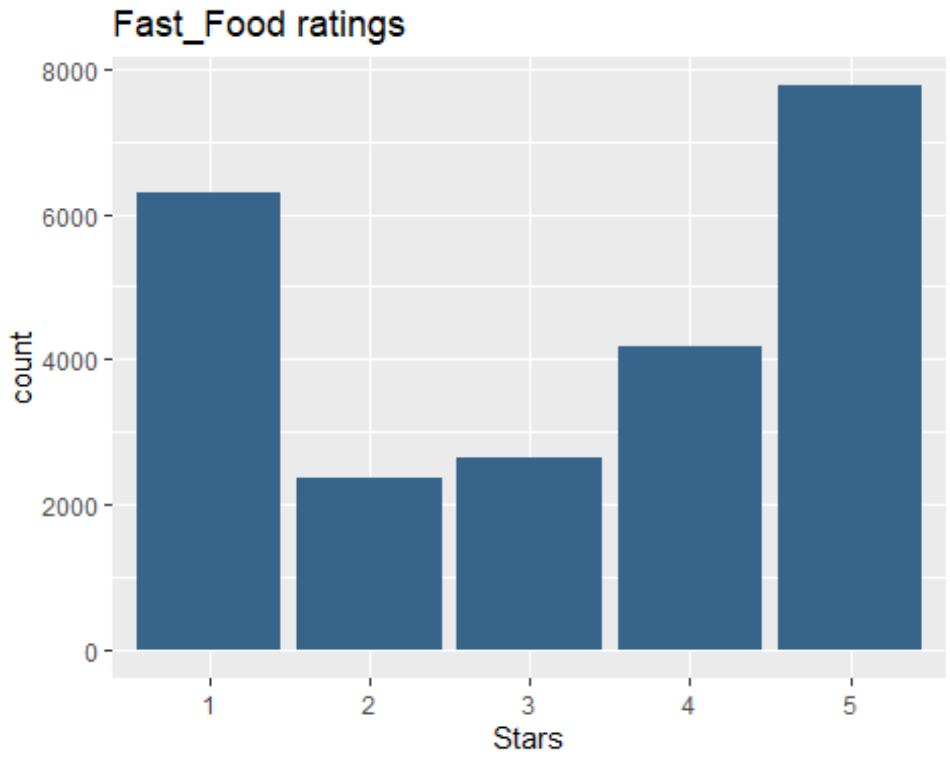

```
star_restaurants %+% subset(Reviews_final, category=="chinese") + ggtitle
("Chinese ratings")
```




```
star_restaurants %+% subset(Reviews_final, category=="italian") + ggtitle
("Italian ratings")
```




```
star_restaurants %+% subset(Reviews_final, category=="fast_food") + ggtit
le("Fast_Food ratings")
```



#create a corpus#

```
Reviews_corp=corpus(Reviews_final)
summary(Reviews_corp, 5)

## Corpus consisting of 242403 documents, showing 5 documents:
##
##   Text Types Tokens Sentences             business_id stars category
##  text1    97    136         8 yfxDa8RFOvJPQh0rNtakHA     3  chinese
##  text2    25     30         4 yfxDa8RFOvJPQh0rNtakHA     5  chinese
##  text3    38     51         3 yfxDa8RFOvJPQh0rNtakHA     1  chinese
##  text4    98    148        12 yfxDa8RFOvJPQh0rNtakHA     3  chinese
##  text5    20     24         1 yfxDa8RFOvJPQh0rNtakHA     5  chinese
```

#create tokens and remove stopwords#

```
Reviews_tokens = quanteda::tokens( Reviews_corp,
                                   remove_numbers = TRUE,
                                   remove_punct = TRUE,
                                   remove_symbols = TRUE,
                                   remove_url = TRUE,
                                   split_hyphens = TRUE,
                                   include_docvars = TRUE,)

Reviews_tokens=tokens_tolower(Reviews_tokens)
Reviews_tokens=tokens_remove(Reviews_tokens,stopwords())
```

#create dfm#

```
Reviews_dfm = dfm( Reviews_tokens,
                   tolower = TRUE,
                   stem = TRUE,)
```

#removing words appearing too many or too little times#

```
Reviews_dfm_trim = dfm_trim( Reviews_dfm,
                             min_docfreq = 0.01,
                             max_docfreq = 0.80,
                             docfreq_type = "prop" )

Reviews_dfm_trim=dfm_subset(Reviews_dfm_trim, ntoken(Reviews_dfm_trim)>0)
head( Reviews_dfm_trim, n = 10, nf = 10 )
```

```
## Document-feature matrix of: 10 documents, 10 features (77.0% sparse) a
nd 3 docvars.
##        features
## docs    sever time go place famili fan rice dish especi roast
##   text1     1    2  1     3      1   1    2    2      1     1
##   text2     0    0  0     1      0   0    0    0      0     0
##   text3     0    1  0     0      0   0    0    0      0     0
##   text4     0    2  1     0      0   1    0    0      0     0
##   text5     0    0  1     0      0   0    0    0      0     0
##   text6     0    1  0     1      0   0    0    0      0     0
## [ reached max_ndoc ... 4 more documents ]
```

#we divide the dataset in 2 parts#
#Above average rating reviews with 4 or more stars# #below average rating reviews with less than 4 stars#

```
Reviews_above_dfm_subset = dfm_subset( Reviews_dfm_trim, stars >= 4 )
Reviews_below_dfm_subset = dfm_subset( Reviews_dfm_trim, stars < 4 )

ndoc(Reviews_above_dfm_subset)
```

```
## [1] 155276
```

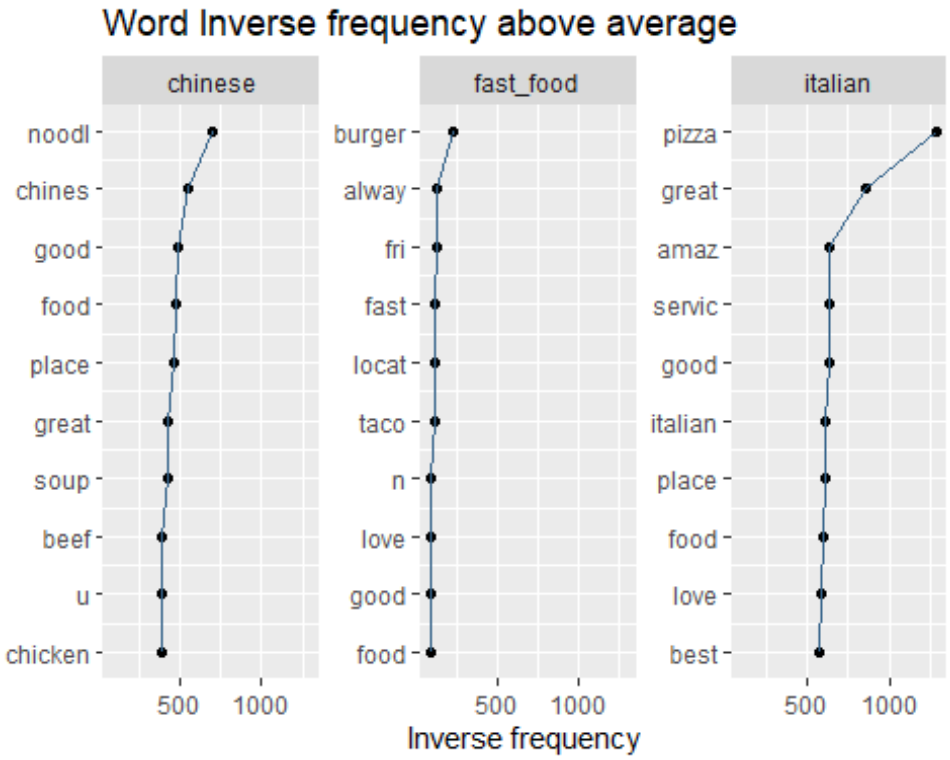
```
ndoc(Reviews_below_dfm_subset)
```

```
## [1] 87106
```

#most frequent words#

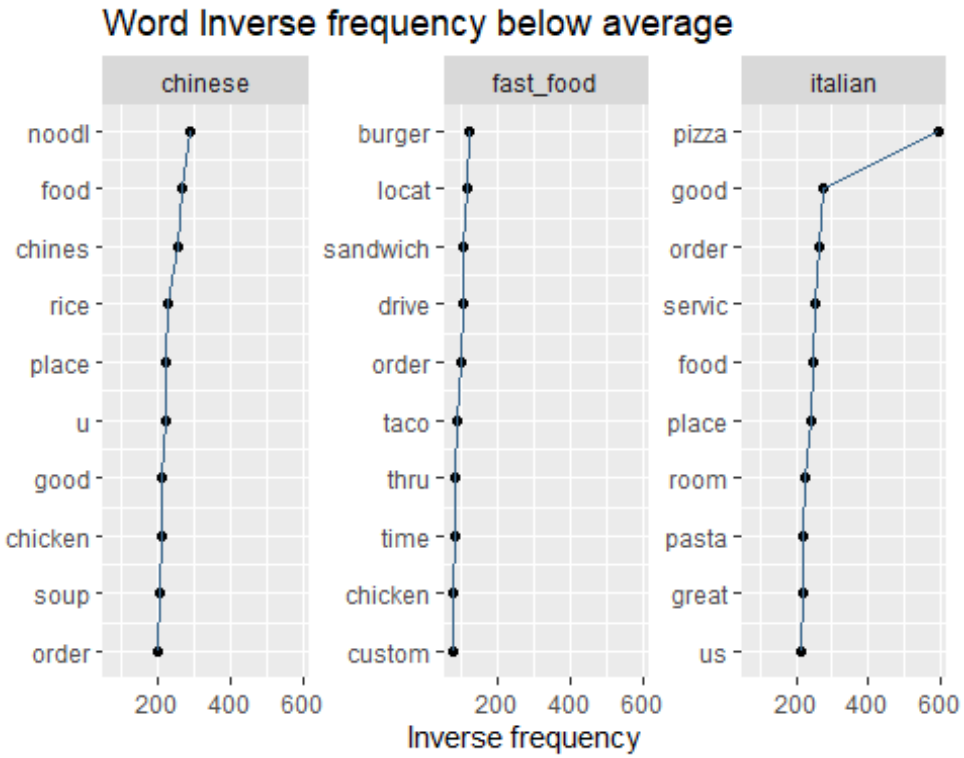
```
Reviews_above_dfm_subset_inverse = dfm_tfidf( Reviews_above_dfm_subset, s
cheme_tf = "prop" )
Reviews_above_dfm_subset_inverse_freq = textstat_frequency( Reviews_above
_dfm_subset_inverse,
                                                            n = 10,
                                                            groups = "cat
egory",
                                                            force = TRUE
)
```

```
ggplot( data = Reviews_above_dfm_subset_inverse_freq,
        aes( x = nrow( Reviews_above_dfm_subset_inverse_freq ):1, y = fre
quency ) ) +
  geom_point() +
  geom_line( color = "steelblue4" ) +
  facet_wrap(~ group, scales = "free_y" ) +
  coord_flip() +
  scale_x_continuous( breaks = nrow( Reviews_above_dfm_subset_inverse_fre
q ):1,
                      labels = Reviews_above_dfm_subset_inverse_freq$feat
ure ) +
  labs( x = NULL, y = "Inverse frequency") +
  ggtitle("Word Inverse frequency above average")
```



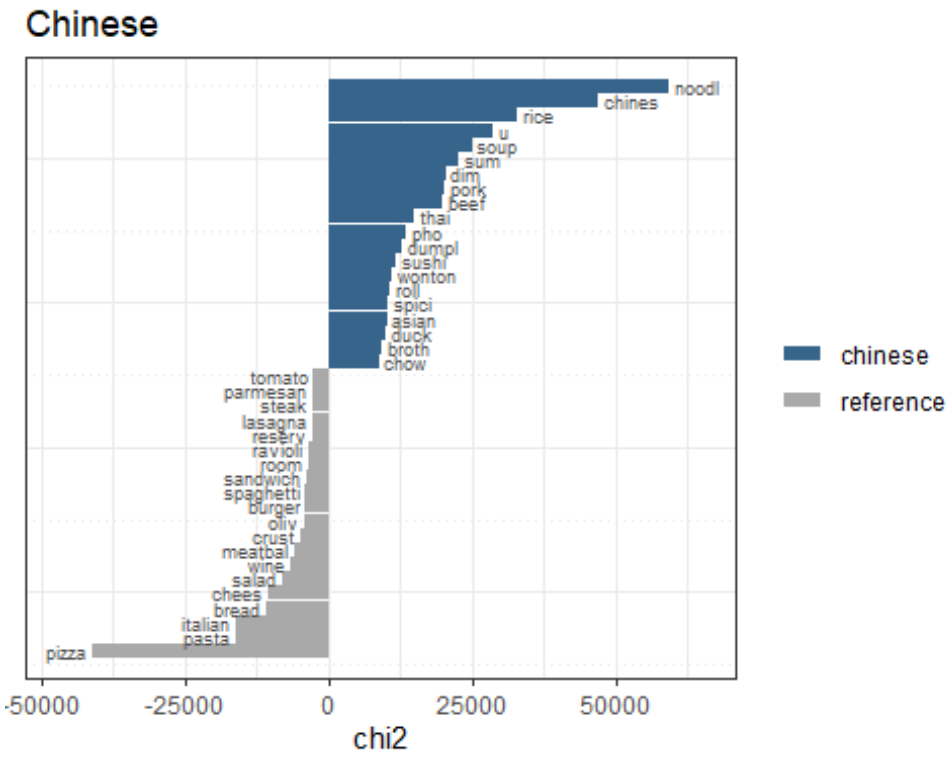

```
Reviews_below_dfm_subset_inverse = dfm_tfidf( Reviews_below_dfm_subset, s
cheme_tf = "prop" )
Reviews_below_dfm_subset_inverse_freq = textstat_frequency( Reviews_below
_dfm_subset_inverse,
                                                            n = 10,
                                                            groups = "cat
egory",
                                                            force = TRUE
)
ggplot( data = Reviews_below_dfm_subset_inverse_freq,
        aes( x = nrow( Reviews_below_dfm_subset_inverse_freq ):1, y = fre
quency ) ) +
  geom_point() +
  geom_line( color = "steelblue4" ) +
  facet_wrap(~ group, scales = "free_y" ) +
```

```
  coord_flip() +
  scale_x_continuous( breaks = nrow( Reviews_below_dfm_subset_inverse_fre
q ):1,
                      labels = Reviews_below_dfm_subset_inverse_freq$feat
ure ) +
  labs( x = NULL, y = "Inverse frequency") +
  ggtitle("Word Inverse frequency below average")
```

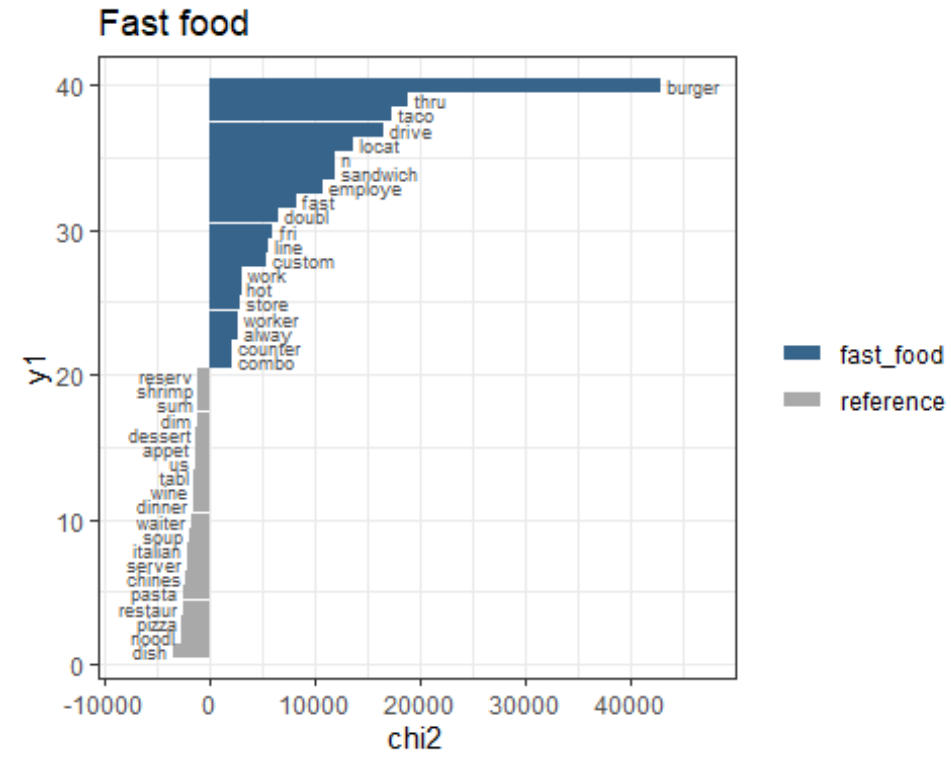


Keyness analysis

```
dfm_categories = dfm( Reviews_dfm_trim, groups = "category")
key_categories_chinese = textstat_keyness( dfm_categories, target = "chin
ese" )
key_categories_fastfood = textstat_keyness( dfm_categories, target = "fas
t_food" )
key_categories_italian = textstat_keyness( dfm_categories, target = "ital
ian" )

textplot_keyness( key_categories_chinese, labelsize = 2.5,
                  color = c( "steelblue4", "darkgray" ) ) +
  ggtitle( "Chinese" )
```

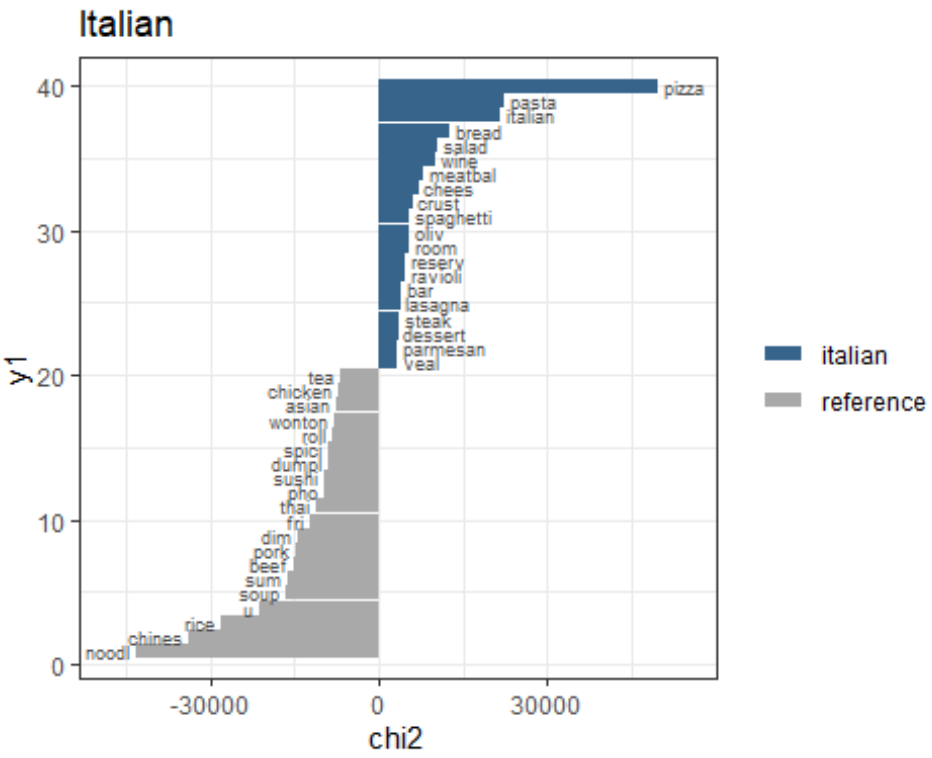



```
textplot_keyness( key_categories_fastfood, labelsize = 2.5,
                  color = c( "steelblue4", "darkgray" ) ) +
  ggtitle( "Fast food" ) +
  theme_bw()
```




```
textplot_keyness( key_categories_italian, labelsize = 2.5,
                  color = c( "steelblue4", "darkgray" ) ) +
```

```
  ggtitle( "Italian" ) +
  theme_bw()
```

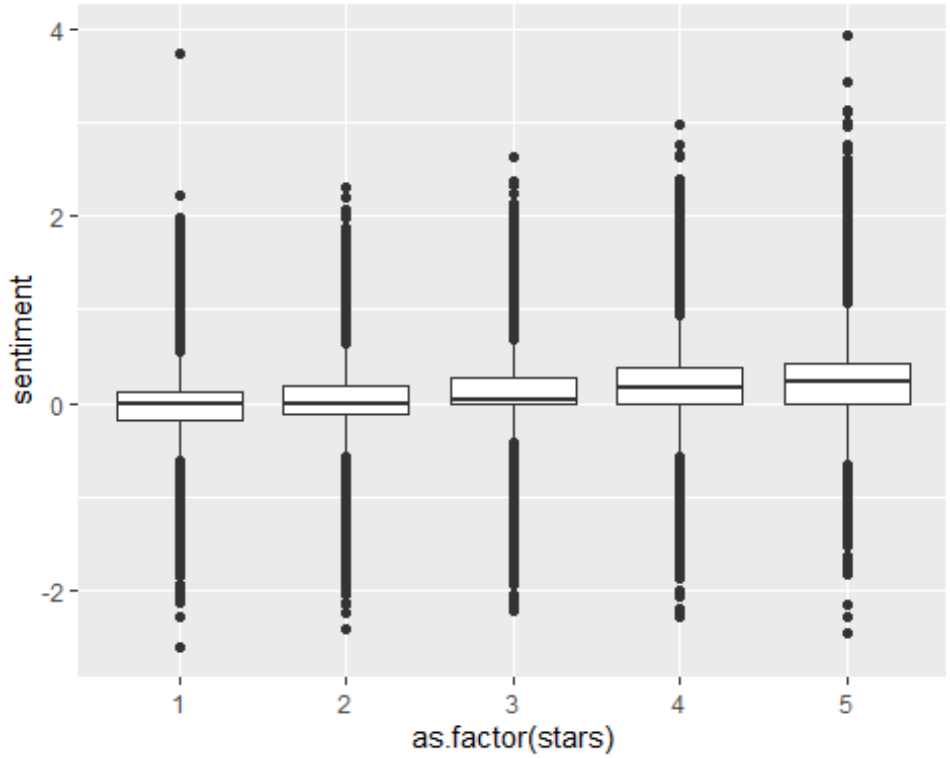


Preliminary sentiment analysis

```
library(sentimentr)
library( corpus )

## Warning: package 'corpus' was built under R version 4.0.3

library( stringr )
library( ggridges )
library( grid )
library( gridExtra )
library( GGally )

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

library( quantreg )

## Loading required package: SparseM

##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve
```

#tone evolution#
#means are calculated one by one from the initial output which already provides the number of the sentence

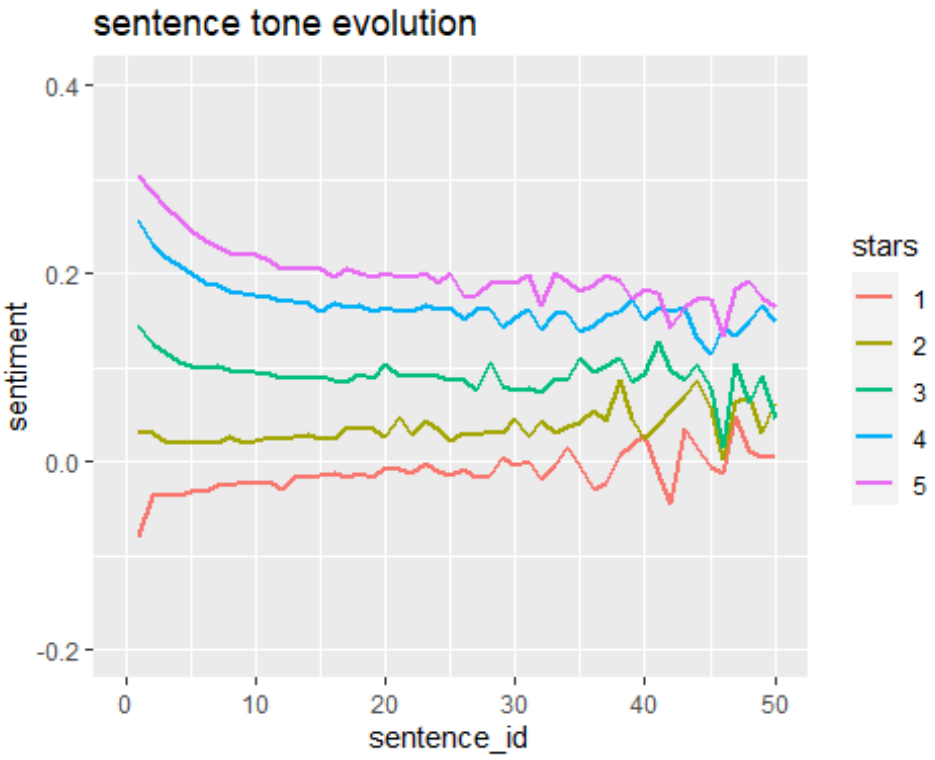
```
Reviews_final$text=tolower(Reviews_final$text)
Reviews_total_sent=sentiment(get_sentences(Reviews_final))
ggplot(Reviews_total_sent, aes( x = as.factor(stars), y = sentiment ) )+
  geom_boxplot()
```




```
means <- matrix(ncol=5, nrow=100)

for (p in 1:5) {

  for(i in 1:100){
    means[i,p] <- mean(Reviews_total_sent[stars==p & sentence_id==i,senti
ment])
  }
}

colnames(means)=c("S1","S2","S3","S4","S5")
means=as.data.frame(means)
means[,"sentence_id"]=1:100

ggplot() +
  geom_line( data = means,
             aes( x = sentence_id, y = S1 , color = "1" ),
             size = 1 ) +
  geom_line( data = means,
             aes( x = sentence_id, y = S2 , color = "2" ),
             size = 1 ) +
```

```
  geom_line( data = means,
             aes( x = sentence_id, y = S3 , color = "3" ),
             size = 1 ) +
  geom_line( data = means,
             aes( x = sentence_id, y = S4 , color = "4" ),
             size = 1 ) +
  geom_line( data = means,
             aes( x = sentence_id, y = S5 , color = "5" ),
             size = 1 ) +
  labs( x = "sentence_id", y = "sentiment",color="stars")+
  ggtitle( "sentence tone evolution" ) + xlim(0,50) + ylim(-0.20,0.40)+
  scale_fill_discrete(name = "stars")
```

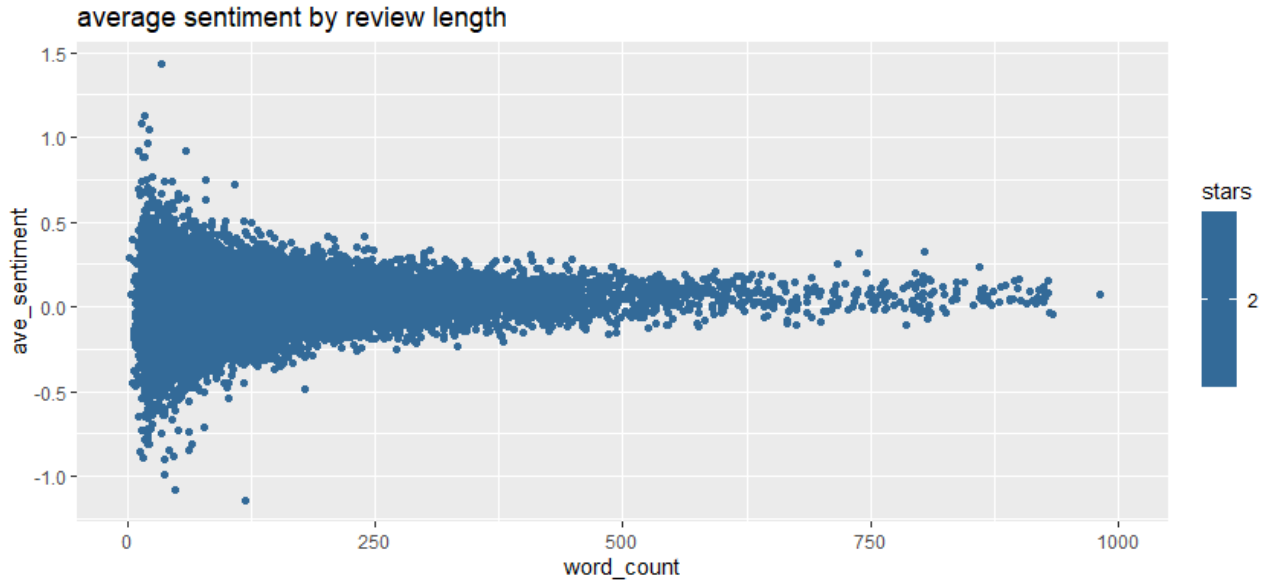
```
## Warning: Removed 50 row(s) containing missing values (geom_path).

## Warning: Removed 50 row(s) containing missing values (geom_path).

## Warning: Removed 50 row(s) containing missing values (geom_path).

## Warning: Removed 50 row(s) containing missing values (geom_path).

## Warning: Removed 50 row(s) containing missing values (geom_path).
```



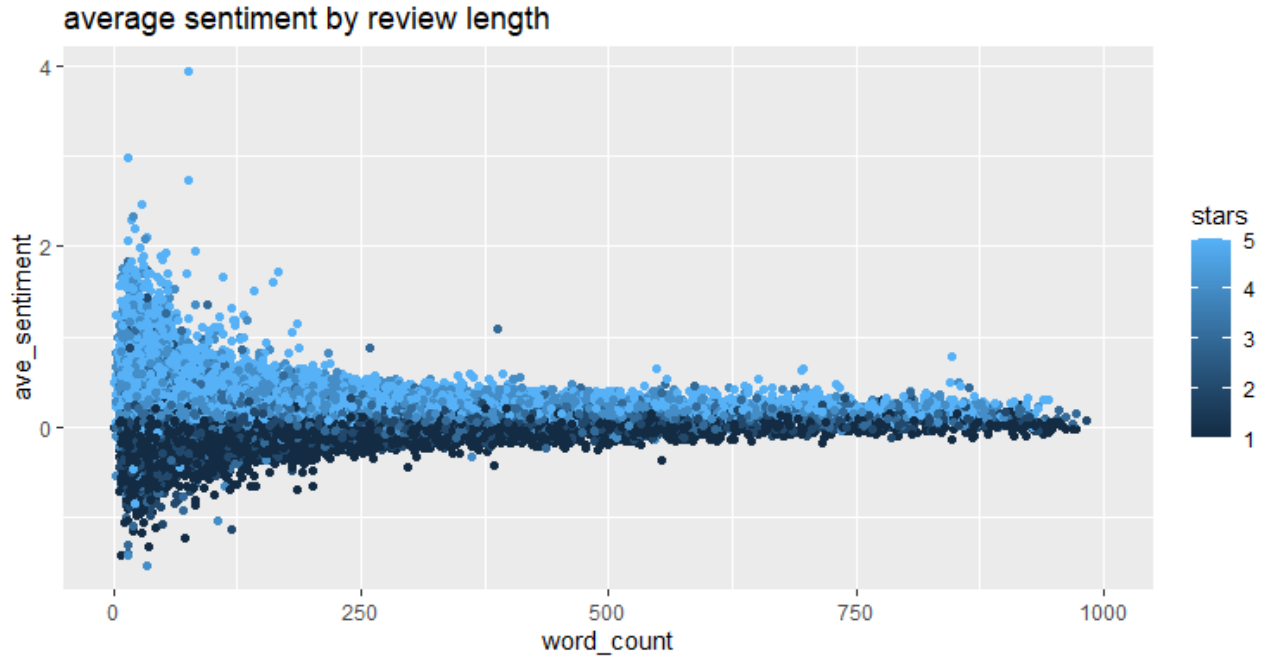
#scatterplots of average review sentiment vs number of words and associated regressions#

```
Reviews_sent_bydoc = sentiment_by(get_sentences(Reviews_final))
Reviews_sent_bydoc = cbind(Reviews_sent_bydoc,Reviews_final)

words_sent = ggplot(Reviews_sent_bydoc,
                    aes(x=word_count,y=ave_sentiment,color=stars))+
```

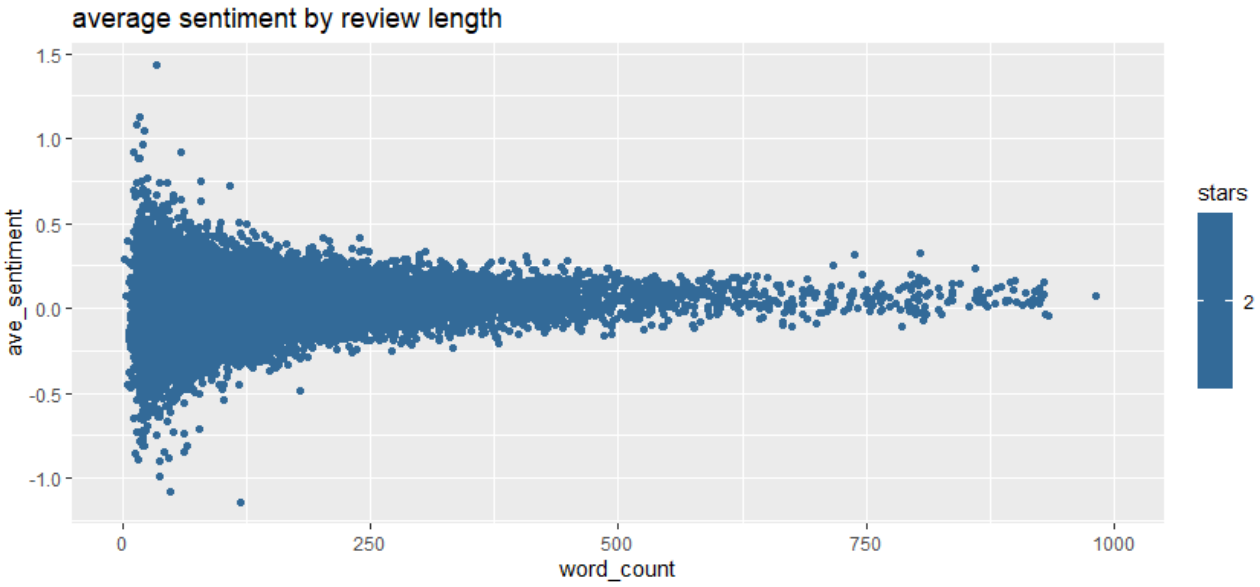
```
  geom_point()
```



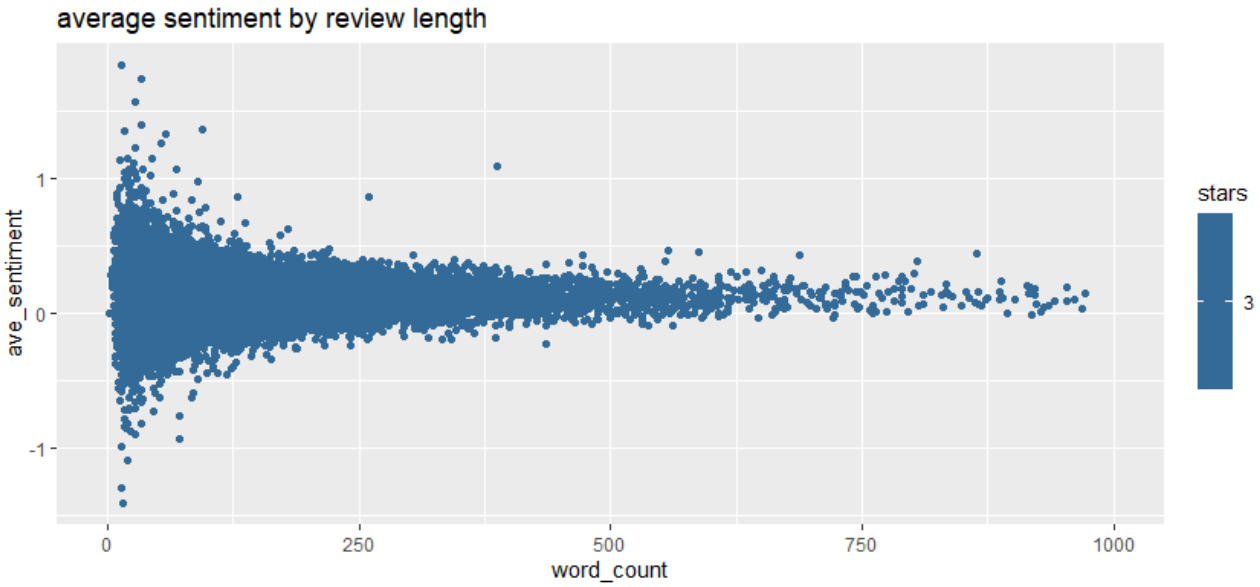

```
+ xlim(0,1000)
  ggtitle( "average sentiment by review length" )
words_sent
```



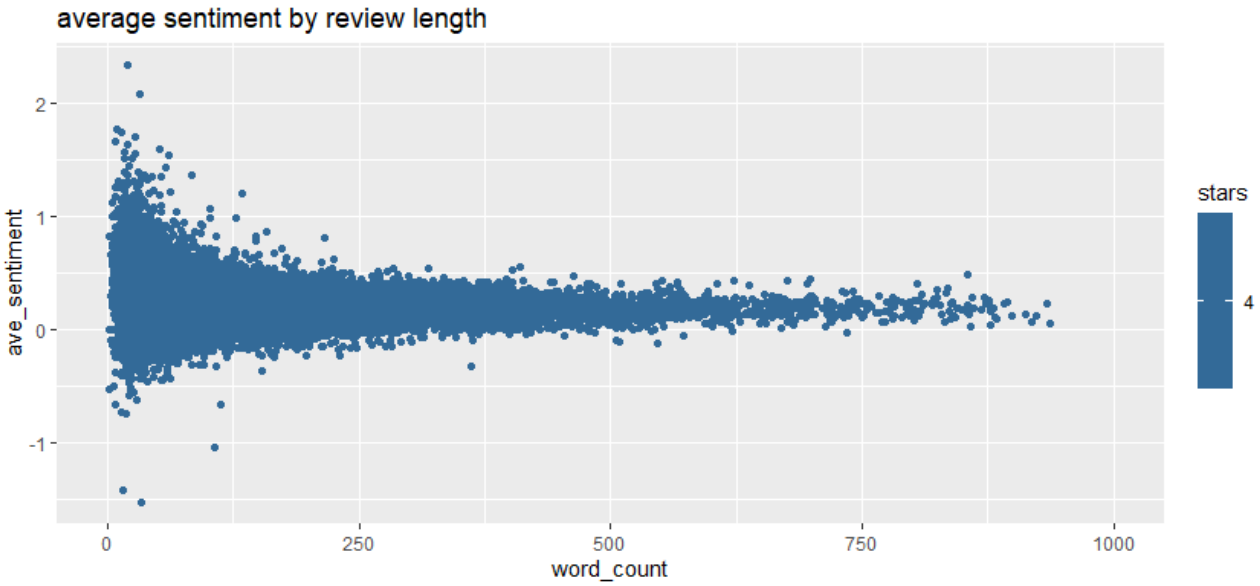

```
words_5 = words_sent %+% subset(Reviews_sent_bydoc, stars %in% 5) +
  theme(legend.position="none")+
  ggtitle( "5 Stars" )
words_1 = words_sent %+% subset(Reviews_sent_bydoc, stars %in% 1) +
  theme(legend.position="none")+
  ggtitle( "1 Star" )

words_sent %+% subset(Reviews_sent_bydoc, stars %in% 2)
```




```
words_sent %+% subset(Reviews_sent_bydoc, stars %in% 3)
```



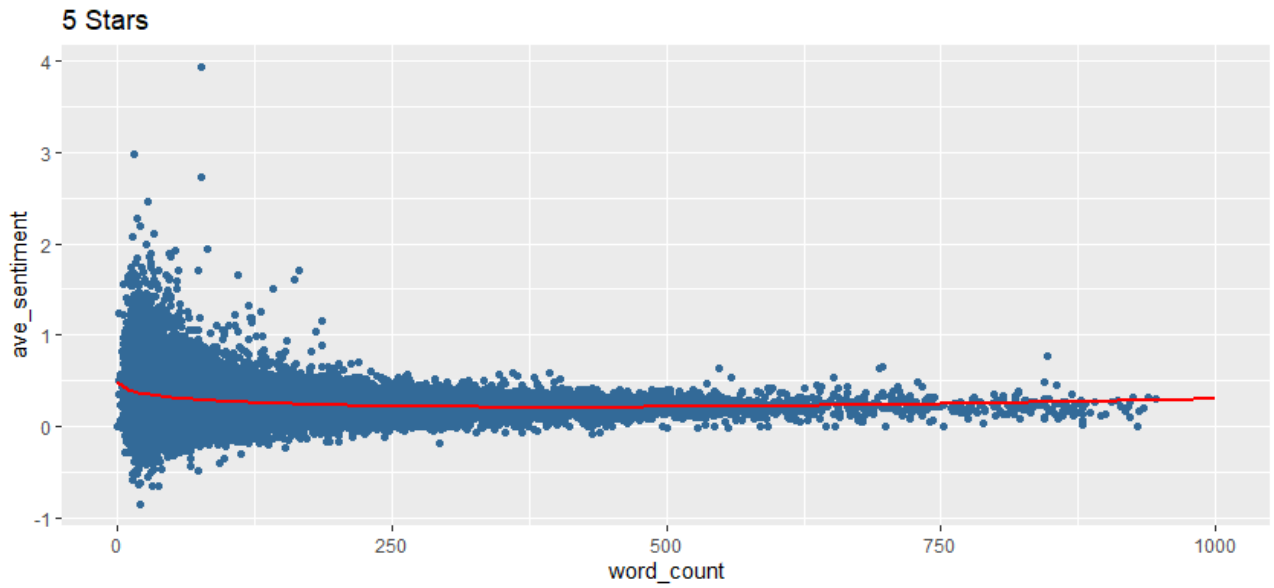

```
words_sent %+% subset(Reviews_sent_bydoc, stars %in% 4)
```



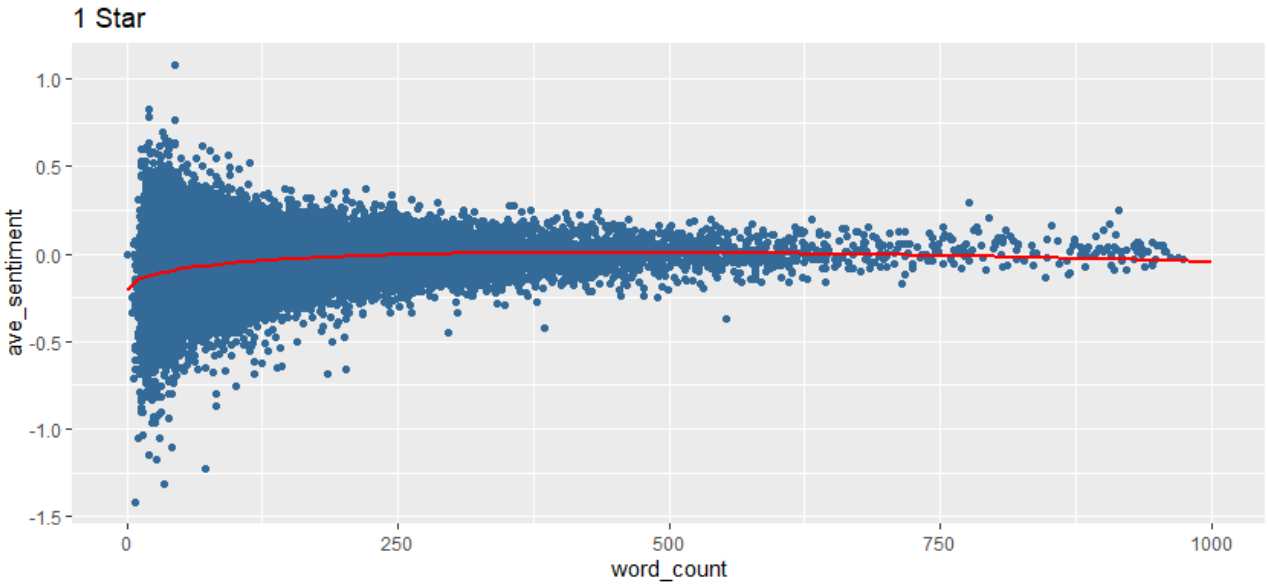

```
reg_5 = lm(ave_sentiment ~ word_count + I(word_count^0.5),
data =  subset(Reviews_sent_bydoc, stars==5))
summary(reg_5)

##
## Call:
## lm(formula = ave_sentiment ~ word_count + I(word_count^0.5),
##     data = subset(Reviews_sent_bydoc, stars == 5))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7698 -0.0923 -0.0052  0.0825  3.6398
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.4795225  0.0023446  204.52   <2e-16 ***
## word_count          0.0006813  0.0000188   36.25   <2e-16 ***
## I(word_count^0.5) -0.0272726  0.0004454  -61.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1604 on 101179 degrees of freedom
## Multiple R-squared:  0.08127,    Adjusted R-squared:  0.08125
## F-statistic:  4475 on 2 and 101179 DF,  p-value: < 2.2e-16

reg_1 = lm(ave_sentiment ~ word_count + I(word_count^0.5), data =  subset
(Reviews_sent_bydoc, stars==1))
summary(reg_1)

##
## Call:
## lm(formula = ave_sentiment ~ word_count + I(word_count^0.5),
##     data = subset(Reviews_sent_bydoc, stars == 1))
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.26261 -0.08601 -0.00014  0.08691  1.16991
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.048e-01  4.133e-03  -49.55   <2e-16 ***
## word_count        -4.779e-04  2.724e-05  -17.54   <2e-16 ***
## I(word_count^0.5)  2.017e-02  7.082e-04   28.48   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1479 on 35477 degrees of freedom
## Multiple R-squared:  0.06151,    Adjusted R-squared:  0.06146
## F-statistic:  1163 on 2 and 35477 DF,  p-value: < 2.2e-16
```

```
fun5 <- function(x) {
  reg_5$coefficients[1] + reg_5$coefficients[2]*x + reg_5$coefficients[3]
*x^0.5
}

fun1 <- function(x) {
  reg_1$coefficients[1] + reg_1$coefficients[2]*x + reg_1$coefficients[3]
*x^0.5
}

words_5 = words_5 + stat_function(fun = fun5 , color="red" , size=1)
words_1 = words_1 + stat_function(fun = fun1 , color="red" , size=1)

words_5
```




```
words_1
```



#quantile removal and regression on the new subsamples#
#in the first lines we show some examples of reviews which present errors like the ones we mentioned in the report

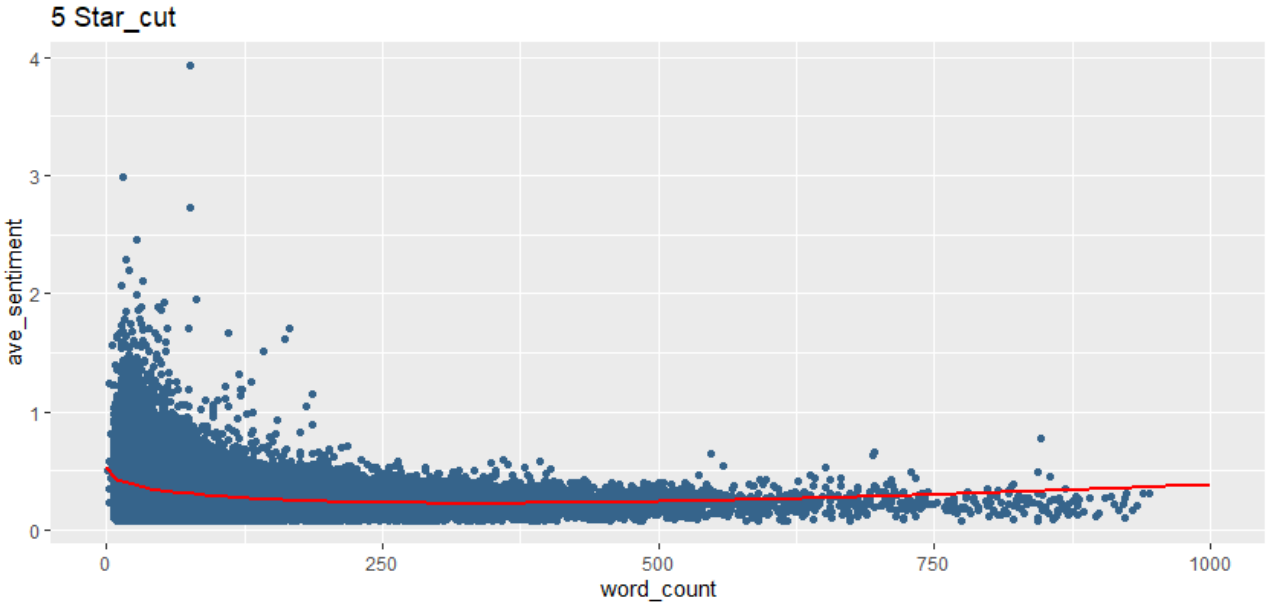
```
subset1 = subset(Reviews_sent_bydoc, stars==1)
sub = subset1[ave_sentiment > quantile(ave_sentiment, 0.90), ]
sub[1:2,6]

##
text
## 1: we normally stop by here to eat when we r in vegas. n this time we
have to give it 1 star. i'll explain it in a little bit down below. don't
get me wrong the food is good n the service is alright. but this time i h
ave to give this 1 star. we were there less than 10 mintues ago n my husb
and asked for a togo box n this server brought out the check n circle the
amount of it n have the guts to circle the  gratuity  part n told him thi
s is how much he should tip. who do that!!! that's pretty daring n scandl
eless.... if u are that desperate for money might as well just ask for $1
0 . normally our tip standard  is  15% but if we got really good service
we would do\n 20%+... but since she did that my husband still gave her $4
. but if it's up to to me i'll give her nothing... we even went up n told
the guy at the cash register . i believe he is the owner so hopefully he
will train his people again...
## 2:
everything was very good up until waitress hustle for tip and ask us to m
ake sure we follow tip/ gratuity guide and double checked it when we hand
ed the money to her that tip is taken cared of exactly as she asked.

subset1 = subset1[ave_sentiment < quantile(ave_sentiment, 0.95), ]
words_1_cut = ggplot(subset1,aes(x=word_count,y=ave_sentiment))+
  geom_point(color="steelblue4")+ xlim(0,1000) +
  ggtitle( "1 Star_cut" )
reg_1_cut = lm(ave_sentiment ~ word_count + I(word_count^0.5) , data =  s
ubset1)
summary(reg_1_cut)
```

```
##
## Call:
## lm(formula = ave_sentiment ~ word_count + I(word_count^0.5),
##     data = subset1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.22497 -0.07588  0.00689  0.08780  0.33865
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.536e-01  3.725e-03  -68.06   <2e-16 ***
## word_count        -5.894e-04  2.437e-05  -24.19   <2e-16 ***
## I(word_count^0.5)  2.468e-02  6.357e-04   38.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1298 on 33703 degrees of freedom
## Multiple R-squared:  0.1114, Adjusted R-squared:  0.1113
## F-statistic:  2112 on 2 and 33703 DF,  p-value: < 2.2e-16
```

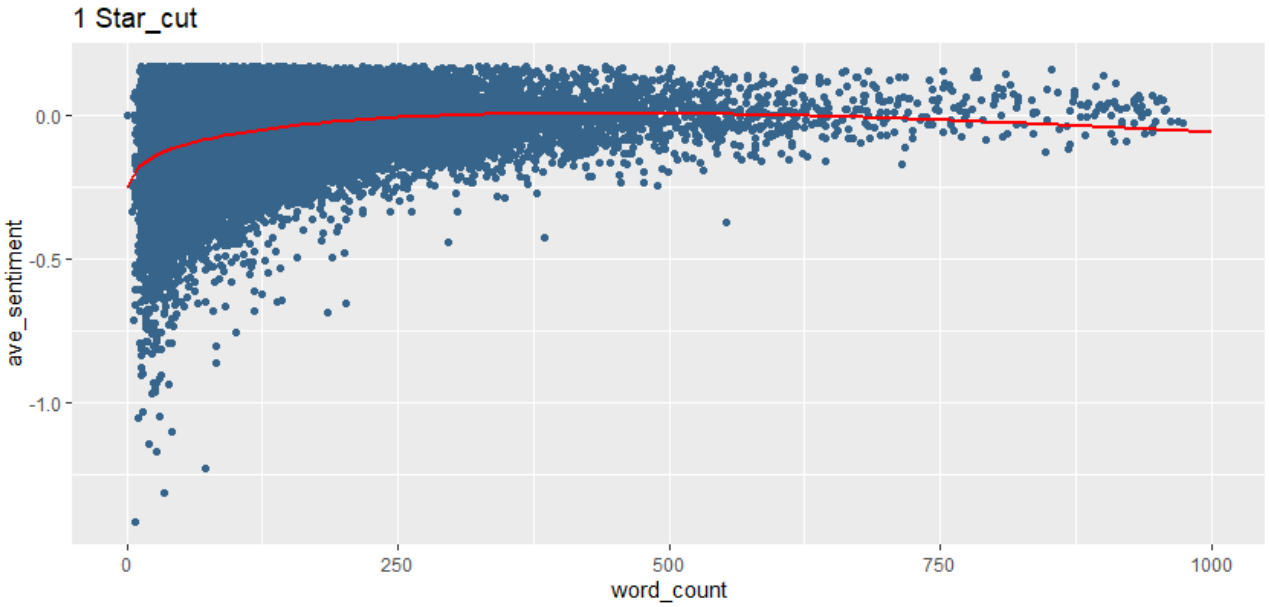
```
subset5 = subset(Reviews_sent_bydoc, stars==5)
subset5 = subset5[ave_sentiment > quantile(ave_sentiment, 0.05), ]
words_5_cut = ggplot(subset5,aes(x=word_count,y=ave_sentiment))+
  geom_point(color="steelblue4") + ggtitle( "5 Star_cut" ) + xlim(0,1000)
reg_5_cut = lm(ave_sentiment ~ word_count + I(word_count^0.5) , data =  s
ubset5)
summary(reg_5_cut)
```

```
##
## Call:
## lm(formula = ave_sentiment ~ word_count + I(word_count^0.5),
##     data = subset5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0701 -0.0931 -0.0126  0.0725  3.6270
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        5.262e-01  2.351e-03  223.86   <2e-16 ***
## word_count         9.043e-04  1.967e-05   45.97   <2e-16 ***
## I(word_count^0.5) -3.310e-02  4.577e-04  -72.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.145 on 96119 degrees of freedom
## Multiple R-squared:  0.1149, Adjusted R-squared:  0.1148
## F-statistic:  6236 on 2 and 96119 DF,  p-value: < 2.2e-16
```

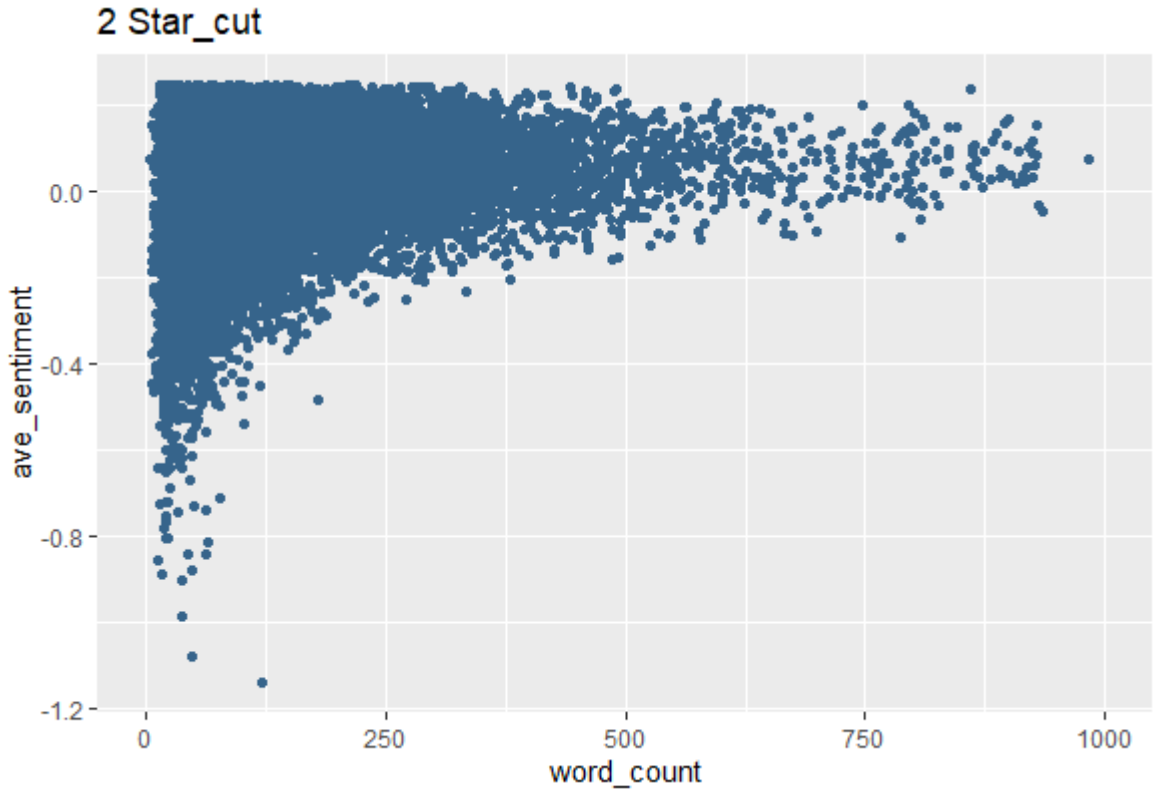
```
fun5_cut <- function(x) {
  reg_5_cut$coefficients[1] + reg_5_cut$coefficients[2]*x + reg_5_cut$coe
fficients[3]*x^0.5
}

fun1_cut <- function(x) {
  reg_1_cut$coefficients[1] + reg_1_cut$coefficients[2]*x + reg_1_cut$coe
fficients[3]*x^0.5
}

words_5_cut = words_5_cut + stat_function(fun = fun5_cut , color="red" ,
size=1)
words_1_cut = words_1_cut + stat_function(fun = fun1_cut , color="red" ,
size=1)

words_5_cut
```




```
words_1_cut
```

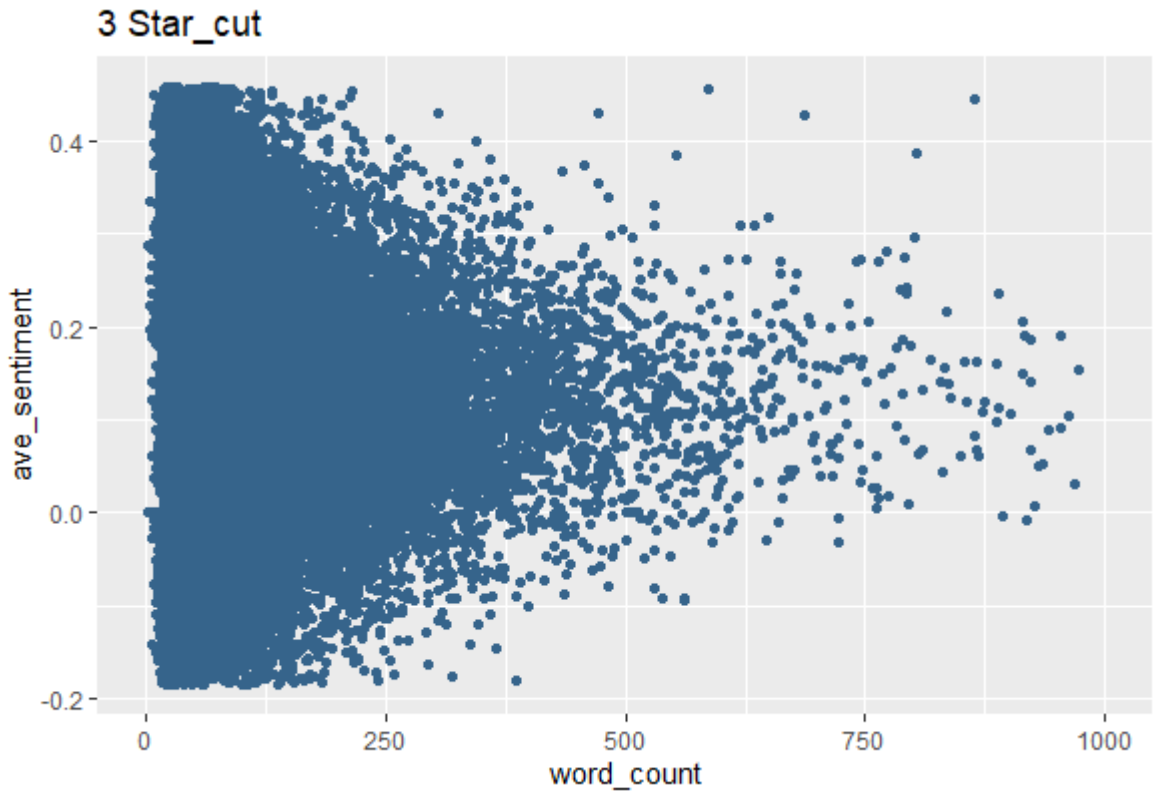



```
subset2 = subset(Reviews_sent_bydoc, stars==2)
subset2 = subset2[ave_sentiment < quantile(ave_sentiment, 0.95), ]
ggplot(subset2,aes(x=word_count,y=ave_sentiment))+
  geom_point(color="steelblue4") + ggtitle( "2 Star_cut" ) + xlim(0,1000)
```

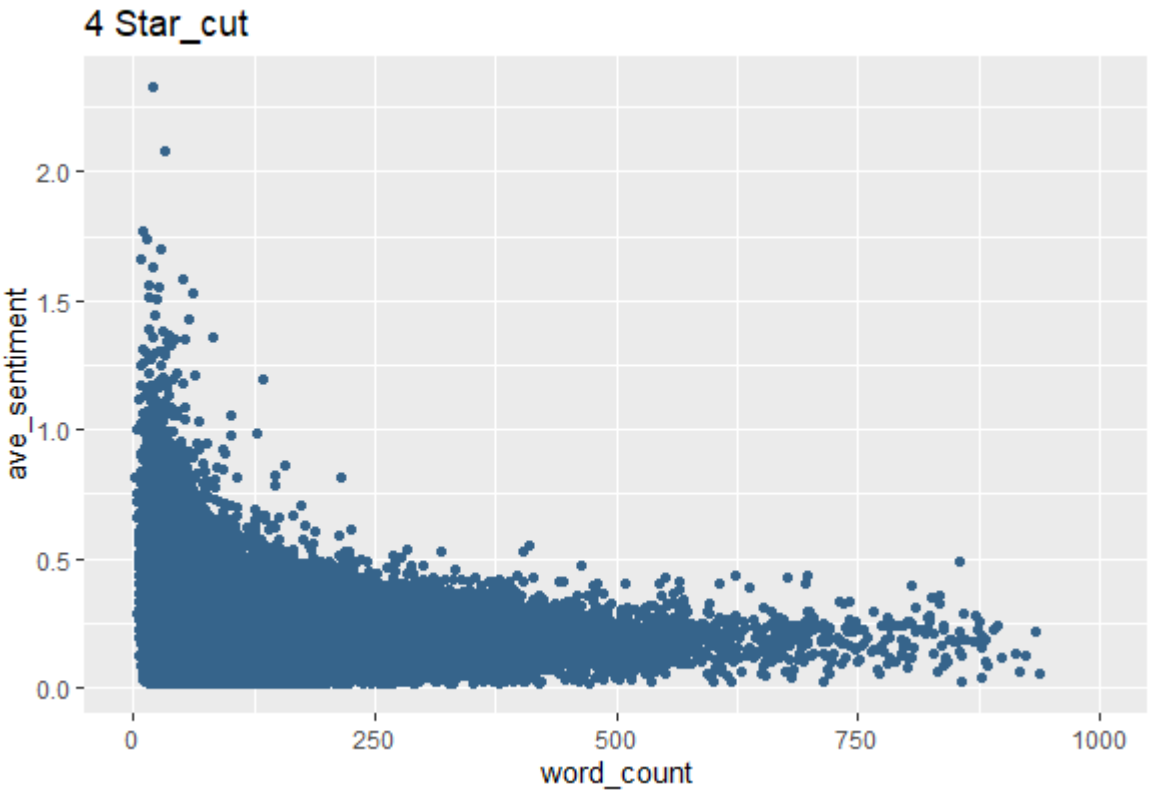



```
subset3 = subset(Reviews_sent_bydoc, stars==3)
subset3 = subset3[ave_sentiment > quantile(ave_sentiment, 0.025) & ave_se
ntiment < quantile(ave_sentiment, 0.975), ]
```

```
ggplot(subset3,aes(x=word_count,y=ave_sentiment))+
  geom_point(color="steelblue4") + ggtitle( "3 Star_cut" ) + xlim(0,1000)
```




```
subset4 = subset(Reviews_sent_bydoc, stars==4)
subset4 = subset4[ave_sentiment > quantile(ave_sentiment, 0.05), ]
ggplot(subset4,aes(x=word_count,y=ave_sentiment))+
  geom_point(color="steelblue4") + ggtitle( "4 Star_cut" ) + xlim(0,1000)
```



Sectorial analysis

```
Reviews_final2 = rbind(subset1,subset2,subset3,subset4,subset5)
Reviews_final2 = subset(Reviews_final2, select = c(business_id,text,stars
,category))

Reviews_sent_bystar=with(Reviews_final2,
                         sentiment_by(get_sentences(text),list(stars)))

Reviews_sent_bystar

##    stars word_count        sd ave_sentiment
## 1:     1    4095093 0.2853795   -0.05608484
## 2:     2    2864679 0.3000446    0.02361396
## 3:     3    3737936 0.3040734    0.13807745
## 4:     4    5944146 0.3110667    0.26173232
## 5:     5    8147310 0.3032402    0.32271646
```

#chinese#

```
Reviews_chi=Reviews_final2[category=="chinese",]
Reviews_chi_sent = get_sentences(Reviews_chi)
Reviews_chi_sent=corpus(Reviews_chi_sent)
```

#chinese_total

```
chi_total <- with(
  Reviews_chi,
  sentiment_by(
```

```
    get_sentences(text),
    list(stars)))
```

#chinese_food

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "grill|chee
s|steak|spicy|curry|eat|food|noodl|dim|sum|soup|tast|sushi|hot|cold|chick
en|rice|asian|ramen|pork|sauce|fish|soy|duck|thai|dish|crab|shrimp")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_food <- with(
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_price

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "price|bill
|charg|pay|expens|cheap|economic|card|cash")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_price <- with(
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_location

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "plac|locat
|area|zone|ambien|environm|music|atmos|neighb|street|dirty|clean|smell|vi
ew|hote")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_location <- with(
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_service

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "servic|ass
ist|help|kind|polit|rud|disrespect|unkind|mean|waiter|waitres|staf|chef|w
elcom")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_service <- with(
```

```
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_timing

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "tim|wait|e
xpect|speed|quick|punctual|delay|lat|slow")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_time <- with(
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_vegan

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "veg|bio|fr
uit|healt")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_veg <- with(
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_delivery

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "deliver|dr
ive-in|driv|transp|dist|pack|home")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_del <- with(
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#chinese_drink

```
containstarget = stringr::str_detect(texts(Reviews_chi_sent), "drin|wine|
beer|water|cup|glass|bever|vin|bar|liquor|alco|alch|sak|cok|fant|peps|tea
|drew|refill")
Reviews_corp_chi_keep = corpus_subset(Reviews_chi_sent, containstarget)
Reviews_corp_chi_keep=convert(Reviews_corp_chi_keep,"data.frame")

chi_dri <- with(
```

```
  Reviews_corp_chi_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian#

```
Reviews_ita=Reviews_final2[category=="italian",]
Reviews_ita_sent = get_sentences(Reviews_ita)
Reviews_ita_sent=corpus(Reviews_ita_sent)
```

#italian_total

```
ita_total <- with(
  Reviews_ita,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_food

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "food|chees
|tast|pizza|pasta|spaghet|polpet|meat|spicy|eat|soup|hot|cold|chicken|ric
e|pork|sauce|fish|duck|dish|crab|shrimp|risotto|carbonar|bologn|ragù|ragu
|lasagn|steak|grill")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_food <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_price

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "price|bill
|charg|pay|expens|cheap|economic|card|cashprice|bill|charg|pay|expens|che
ap|economic|card|cash")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_price <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_location

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "plac|locat
|area|zone|ambien|environm|music|atmos|neighb|street|dirty|clean|smell|vi
ew|hote")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
```

```
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_location <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_service

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "servic|ass
ist|help|kind|polit|rud|disrespect|unkind|mean|waiter|waitres|staf|chef|w
elcom")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_service <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_timing

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "tim|wait|e
xpect|speed|quick|punctual|delay|lat|slow")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_time <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_vegan

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "veg|bio|fr
uit|healt")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_veg <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_delivery

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "deliver|dr
ive-in|driv|transp|dist|pack|hom")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
```

```
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_del <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#italian_drink

```
containstarget = stringr::str_detect(texts(Reviews_ita_sent), "drin|wine|
beer|water|cup|glass|bever|vin|bar|liquor|alco|alch|sak|cok|fant|peps|tea
|drew|refill")
Reviews_corp_ita_keep = corpus_subset(Reviews_ita_sent, containstarget)
Reviews_corp_ita_keep=convert(Reviews_corp_ita_keep,"data.frame")

ita_dri <- with(
  Reviews_corp_ita_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food#

```
Reviews_ff=Reviews_final2[category=="fast_food",]
Reviews_ff_sent = get_sentences(Reviews_ff)
Reviews_ff_sent=corpus(Reviews_ff_sent)
```

#fast food_total

```
ff_total <- with(
  Reviews_ff,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_food

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "food|chees|
tast|meat|burger|frie|cheddar|sanwich|pizza|pasta|polpet|meat|spicy|eat|s
oup|hot|cold|chicken|rice|pork|sauce|fish|duck|dish|crab|shrimp|lasagn|st
eak|grill|lasagna|bolognes|onion|mushroom")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_food <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_price

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "price|bill|
charg|pay|expens|cheap|economic|card|cashprice|bill|charg|pay|expens|chea
p|economic|card|cash")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_price <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_location

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "plac|locat|
area|zone|ambien|environm|music|atmos|neighb|street|dirty|clean|smell|vie
w|hote")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_location <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_service

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "servic|assi
st|help|kind|polit|rud|disrespect|unkind|mean|waiter|waitres|staf|chef|we
lcom")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_service <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_timing

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "tim|wait|ex
pect|speed|quick|punctual|delay|lat|slow")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_time <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_vegan

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "veg|bio|fru
it|healt")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_veg <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_delivery

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "deliver|dri
ve-in|driv|transp|dist|pack|hom")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_del <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#fast food_drink

```
containstarget = stringr::str_detect(texts(Reviews_ff_sent), "drin|wine|b
eer|water|cup|glass|bever|vin|bar|liquor|alco|alch|sak|cok|fant|peps|tea|
drew|refill")
Reviews_corp_ff_keep = corpus_subset(Reviews_ff_sent, containstarget)
Reviews_corp_ff_keep=convert(Reviews_corp_ff_keep,"data.frame")

ff_dri <- with(
  Reviews_corp_ff_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#final tables#

```
chi_table = cbind(chi_total$stars,chi_total$ave_sentiment,chi_food$ave_se
ntiment,
                  chi_location$ave_sentiment,chi_price$ave_sentiment,chi_
service$ave_sentiment,
                  chi_time$ave_sentiment,chi_veg$ave_sentiment,chi_del$av
e_sentiment,
                  chi_dri$ave_sentiment)
colnames(chi_table) = c("stars","total","food","location","price","servic
e","timing",
                        "vegan","delivery","drink")
chi_table
```

```
##      stars       total        food   location        price     service
## [1,]     1 -0.06699585 -0.03270563 -0.0445212 -0.049974652 -0.10292773
## [2,]     2  0.01276642  0.02976214  0.0563343  0.001134725  0.01648748
## [3,]     3  0.13256649  0.13408710  0.1616312  0.125059213  0.18122735
## [4,]     4  0.25196593  0.24891951  0.2616788  0.282672257  0.37089363
## [5,]     5  0.32158406  0.31797054  0.3351764  0.392472076  0.48433992
##            timing       vegan   delivery       drink
## [1,] -0.009368046 -0.00769091 0.02569809 -0.01668528
## [2,]  0.040076430  0.04543234 0.08644224  0.03787670
## [3,]  0.128934287  0.11514344 0.14234641  0.11026702
## [4,]  0.228800530  0.18676340 0.25373371  0.19619349
## [5,]  0.289949519  0.23695201 0.32819367  0.27143475
```

```
ita_table = cbind(ita_total$stars,ita_total$ave_sentiment,ita_food$ave_se
ntiment,
                  ita_location$ave_sentiment,ita_price$ave_sentiment,ita_
service$ave_sentiment,
                  ita_time$ave_sentiment,ita_veg$ave_sentiment,ita_del$av
e_sentiment,
                  ita_dri$ave_sentiment)
colnames(ita_table) = c("stars","total","food","location","price","servic
e","timing",
                        "vegan","delivery","drink")

ita_table
```

```
##      stars       total         food    location       price     servic
e
## [1,]     1 -0.04465663 -0.005087015 -0.03060802 -0.03544714 -0.0729487
4
## [2,]     2  0.03240266  0.064069868  0.07608219  0.01130238  0.0394464
1
## [3,]     3  0.14306959  0.162423771  0.17197767  0.10290910  0.1810487
2
## [4,]     4  0.26779345  0.291193989  0.27282443  0.27309693  0.3680292
0
## [5,]     5  0.32203982  0.339900294  0.32754101  0.37779252  0.4584287
1
##           timing      vegan   delivery       drink
## [1,] 0.001035016 0.02584474 0.03628824 0.006288931
## [2,] 0.049744772 0.09226504 0.08602326 0.054443254
## [3,] 0.126288916 0.14597006 0.14183844 0.139343412
## [4,] 0.233250426 0.22420310 0.23137247 0.242638346
## [5,] 0.284765468 0.25848794 0.28607495 0.314236061
```

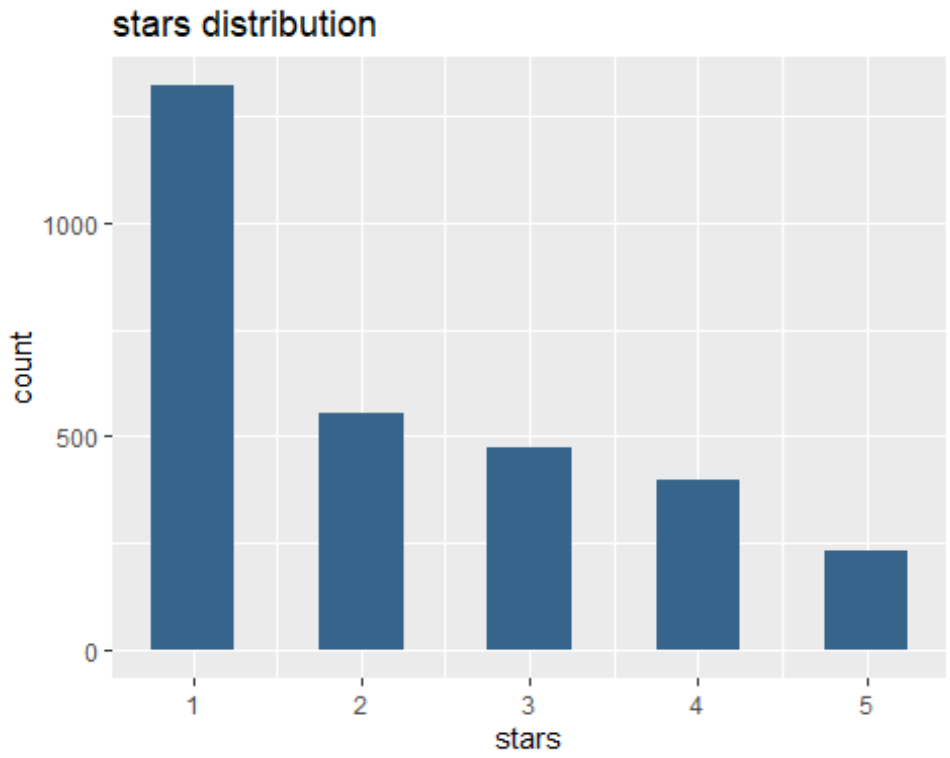
```
ff_table = cbind(ff_total$stars,ff_total$ave_sentiment,ff_food$ave_sentim
ent,
                 ff_location$ave_sentiment,ff_price$ave_sentiment,ff_serv
ice$ave_sentiment,
                 ff_time$ave_sentiment,ff_veg$ave_sentiment,ff_del$ave_se
ntiment,
```

```
                  ff_dri$ave_sentiment)
colnames(ff_table) = c("stars","total","food","location","price","service
","timing",
                       "vegan","delivery","drink")
ff_table
```

```
##      stars        total        food    location       price     servic
e
## [1,]     1 -0.071426078 -0.01111171 -0.06627970 -0.03947255 -0.1355890
9
## [2,]     2  0.005507767  0.04716082  0.04438389  0.01777350  0.0178219
1
## [3,]     3  0.129395269  0.16014795  0.15540960  0.12479959  0.2010524
0
## [4,]     4  0.264815104  0.28916948  0.28056813  0.29127976  0.4202137
8
## [5,]     5  0.335300794  0.37150346  0.35816477  0.39967679  0.5044515
4
##           timing        vegan    delivery        drink
## [1,] -0.03224146 -0.004356434 -0.02965883 -0.017924742
## [2,]  0.03008229  0.035122219  0.02570417  0.006258592
## [3,]  0.13733305  0.146346996  0.10879511  0.102009483
## [4,]  0.24756533  0.225592176  0.20370123  0.210401299
## [5,]  0.30909336  0.298170628  0.29105068  0.276018340
```

Single restaurant analysis

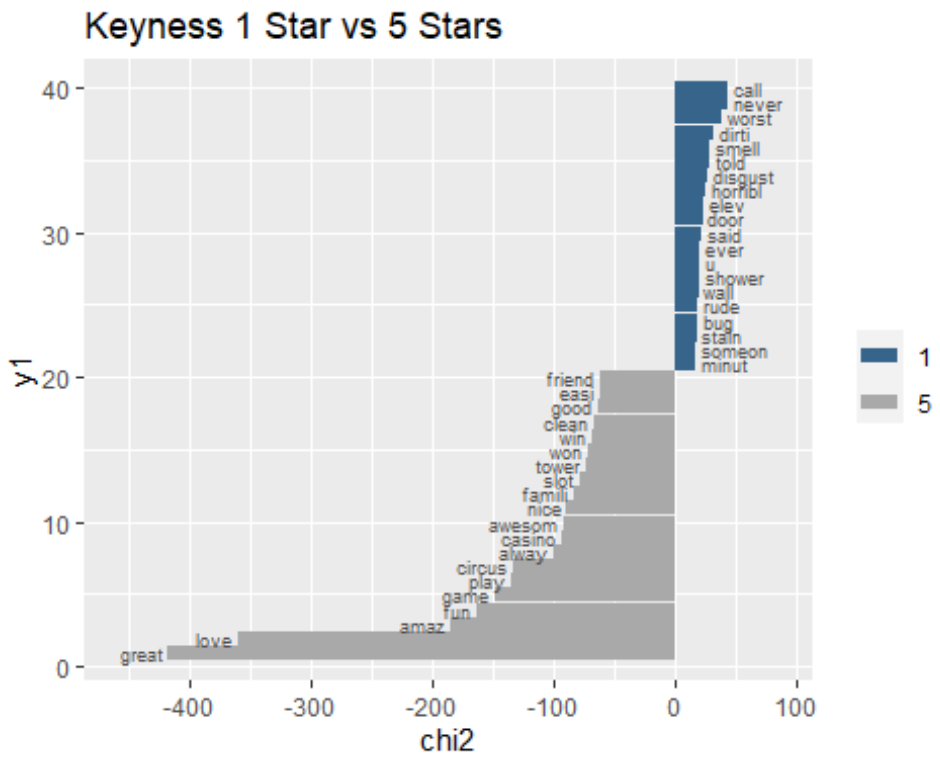
```
X="6Q7-wkCPc1KF75jZLOTcMw"

Reviews_rest = Reviews_final2[Reviews_final2$business_id==X,]
Reviews_rest=subset(Reviews_rest, select = -c(business_id,category))

ggplot(data=Reviews_rest , aes(stars))+
  geom_histogram(fill = "steelblue4",binwidth = 0.5)+
  ggtitle("stars distribution")
```



#Keyness 1vs5 stars#

```
Reviews_rest_corp = corpus(Reviews_rest)
Reviews_rest_tokens = quanteda::tokens( Reviews_rest_corp,
                                        remove_numbers = TRUE,
                                        remove_punct = TRUE,
                                        remove_symbols = TRUE,
                                        remove_url = TRUE,
                                        split_hyphens = TRUE,
                                        include_docvars = TRUE,)

Reviews_rest_tokens=tokens_remove(Reviews_rest_tokens,stopwords())
Reviews_rest_dfm = dfm( Reviews_rest_tokens,
                        stem = TRUE)

Reviews_rest_dfm = dfm_trim( Reviews_rest_dfm,
                             min_docfreq = 0.01,
                             max_docfreq = 0.80,
                             docfreq_type = "prop" )


Reviews_rest_dfm = dfm_subset(Reviews_rest_dfm, ntoken(Reviews_rest_dfm)>
0)
Reviews_rest_dfm = dfm_subset(Reviews_rest_dfm, stars==5 | stars==1)

dfm_1_vs_5 = dfm( Reviews_rest_dfm, groups = "stars")
key_1_vs_5 = textstat_keyness( dfm_1_vs_5, target = "1" )

textplot_keyness( key_1_vs_5, labelsize = 2.5,
                  color = c( "steelblue4", "darkgray" ) ) +
```

```
  ggtitle( "Keyness 1 Star vs 5 Stars" ) +
  theme_gray()
```



#bigram networks cloud#

```
library(tidytext)

## Warning: package 'tidytext' was built under R version 4.0.3

library(lubridate)
library(wordcloud)

## Warning: package 'wordcloud' was built under R version 4.0.3

## Loading required package: RColorBrewer

library(tm)

## Warning: package 'tm' was built under R version 4.0.3

## Loading required package: NLP

## Warning: package 'NLP' was built under R version 4.0.3

##
## Attaching package: 'NLP'

## The following objects are masked from 'package:quanteda':
##
##     meta, meta<-
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate

##
## Attaching package: 'tm'

## The following object is masked from 'package:koRpus':
##
##     readTagged

## The following objects are masked from 'package:quanteda':
##
##     as.DocumentTermMatrix, stopwords

library(SnowballC)

## Warning: package 'SnowballC' was built under R version 4.0.3

library(textcat)

## Warning: package 'textcat' was built under R version 4.0.3

count_bigrams <- function(dataset) {
  dataset %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    filter(!word1 %in% stop_words$word,
           !word2 %in% stop_words$word) %>%
    count(word1, word2, sort = TRUE)}

visualize_bigrams <- function(bigrams) {
  set.seed(2016)
  a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

  bigrams %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
    geom_edge_link(aes(edge_alpha = n), show.legend = FALSE, arrow = a) +
    geom_node_point(color = "lightblue", size = 5) +
    geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
    theme_void()}

bigrams_rest <- Reviews_rest %>%
  count_bigrams()

head(bigrams_rest)

##        word1  word2    n
## 1:    circus circus 1522
## 2:     front   desk  372
## 3:       las  vegas  344
## 4: adventure   dome  221
```

```
## 5:      west  tower  198
## 6:    resort    fee  197
```

```
bigrams_rest %>%
  filter(n > 20) %>%
  visualize_bigrams()
```

#sentiment analysis#

```
Reviews_rest_sent = get_sentences(Reviews_rest)
Reviews_rest_sent = corpus(Reviews_rest_sent)
```

#total

```
total <- with(
  Reviews_rest,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#food

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "food|chee
s|tast|pizza|pasta|spaghet|polpet|meat|spicy|eat|soup|hot|cold|chicken|ri
ce|pork|sauce|fish|duck|dish|crab|
                                    shrimp|risotto|carbonar|bologn|ragù|
ragu|lasagn|steak|grill|breakfast|buffe|dinner|lunch|mexican|nacho|taco|b
urrito|fajita|guacamole|bean|chilli|chili|
                                    empanada|chorizo|texmex|pinchos|pael
la|polpett|stake|filet|ribeye|sirloin")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
```

```
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

food <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#price

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "price|bil
l|charg|pay|expens|cheap|economic|card|cash|fee|coupon|credit")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

price <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#location

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "building|
plac|locat|area|zone|ambien|environm|music|atmos|neighb|street|view|hote"
)
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

location <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#rooms

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "dirty|cle
an|smell|bed|room|bug|fridge|tv|air|conditioning|shower|door|window|scree
n|
                                        soiled|filth|mud|dust|mucky|stink|wa
sh|polish|unstained|carpet|spac|balcon|terrace")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

rooms <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#service

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "bell|servic|assist|help|kind|polit|rud|disrespect|unkind|mean|waiter|waitres|staf|chef|welcom|customer|front|desk")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

service <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#timing

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "tim|wait|expect|speed|quick|punctual|delay|lat|slow|minut|hour")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

time <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#drink

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "drin|wine|beer|water|cup|glass|bever|vin|bar|liquor|alco|alch|sak|cok|fant|peps|tea|drew|refill|bar|cocktail")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

dri <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#parking

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "parking|car|bycicl|garag|motor|lodge")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

parking <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#entertainment

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "adventur|
dome|park|amusement|arcade|game|carnival|island|treasure|coaster
                                    roller|casino|slot|machine|gamble|st
rip|fun|win|tower|play|skyrise|cards|roulett")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

entertainment <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#security

```
containstarget = stringr::str_detect(texts(Reviews_rest_sent), "security|
guard|steal|rob|crim|safe|lock")
Reviews_corp_rest_keep = corpus_subset(Reviews_rest_sent, containstarget)
Reviews_corp_rest_keep=convert(Reviews_corp_rest_keep,"data.frame")

security <- with(
  Reviews_corp_rest_keep,
  sentiment_by(
    get_sentences(text),
    list(stars)))
```

#table#

```
table = cbind(total$stars,total$ave_sentiment,food$ave_sentiment,price$av
e_sentiment,
              location$ave_sentiment,rooms$ave_sentiment,service$ave_sent
iment,
              time$ave_sentiment, dri$ave_sentiment, parking$ave_sentimen
t,entertainment$ave_sentiment,
              security$ave_sentiment)

colnames(table) = c("stars","total","food","price","location","rooms","se
rvice","timing",
                    "drink","parking","entertainment","security")
table
```

```
##      stars        total        food       price    location        roo
ms
## [1,]     1 -0.088580653 -0.06097373 -0.04818259 -0.07849066 -0.1172275
35
## [2,]     2  0.008938912  0.03072043  0.01653637  0.00587059 -0.0040631
79
## [3,]     3  0.103557782  0.11839546  0.09242402  0.10644461  0.1018062
02
## [4,]     4  0.207921544  0.24907552  0.19250673  0.19259688  0.2292980
86
## [5,]     5  0.264101951  0.29090691  0.29822003  0.24164767  0.3062986
```

```
92
##          service      timing       drink     parking entertainment
security
## [1,] -0.08862360 -0.03330987 -0.04922915 -0.11189458   -0.03031075 -0.
09168327
## [2,]  0.03216975  0.02102203  0.01881521 -0.03026853    0.05469081  0.
03171747
## [3,]  0.16512995  0.08925188  0.08668056  0.06460098    0.13671110  0.
08731685
## [4,]  0.29337919  0.16921764  0.17788587  0.15832870    0.21985563  0.
25775111
## [5,]  0.45637194  0.20805118  0.29850616  0.27788558    0.26124097  0.
22803531
```

##end of script