Predicting Restaurant Popularity in Metropolitan Areas from Yelp Statistics

Claire Yang, Hayden Sansum, Nikhil Vanderklaauw

Project Overview

To use data from Yelp to discover what factors result in a "popular" restaurant and whether these factors can be used to predict the popularity of previously unseen restaurants. Popularity in this instance is defined as the rate of both number of check-ins and number of new reviews over a given time period (weekly/monthly). This will give an approximation as to how actively customers are visiting a given restaurant. Factors such as the physical facilities, operating characteristics and customer behaviours will be included as predictors.

In order to keep data sizes manageable we will be focusing on restaurants and food business in the Las Vegas area.

Data Sources

Yelp review data - https://www.yelp.com/dataset/download

A useful metadata dictionary table can be found in the appendix of this report, whilst variables have been named appropriately for their data, please see the metadata dictionary for further clarifications of variable intent (Appendix section 1.1).

Hypotheses

What attributes of restaurants and reviews contribute to the 'popularity' of restaurants?

- Restaurants with a higher than average proportion of tagged reviews - aka reviews with "cool", "funny" or "useful" reviews will be more popular due to the increased engagement these reviews might have.
- Having a higher proportion of reviews by 'influencers' (defined by criteria such as number of fans, elite status) will increase the popularity of a restaurant.
- Restaurants with a greater number of attributes (parking, wifi, etc.) will be more popular.

Further possible questions:

- Can future popularity be predicted for new unseen restaurants previously unseen restaurants?
- Can trends in restaurant type over time (American food, Bar food, Tapas etc.) be studied from the data to understand how tastes have changed?

Data Preprocessing

NOTE: Much of the data processing was performed in Python for performance reasons as many of the initial JSON datasets were >5GB in size. This created issues with storing data in RAM and hence the approach used was to yield chunks of data at a time, storing only subsets. The python data processing code can be found in the Appendix (section 1.3) for further details and to ensure proper recording and reproducibility of code from this report.

Flow

The data came in multiple JSON files each explaining it's own subset of the overall Yelp dataset. For our purposes the most relevant tables are businessess (a full list of all businesses with unique *business_id*), reviews (a full list of all reviews for those business with a link to the the user who wrote it: unique *business_id*, unique *review_id*, unique *user_id*) and users (all the user information for those reviews: unique *user_id*).

In order to begin modelling we first had to combine all these tables into a clean dataset. To do this though as the tables were too large to keep in memory we decided to cut down the scope from all USA food businesses to just Las Vegas, which was the largest city for number of food businesses, and so this reduced the initial business list down to just 30,000 unique businesses. From here the data processing was as follows:

1. Run through all Reviews in chunks and keep only those corresponding to the 30,000 unique businesses. (Later on in the business processing this would be reduced again to just food businesses).
2. Resolve a list of unique review_ids from those remaining reviews
3. Run through all Users in chunks and keep only those corresponding to the remaining reviews.
4. Clean and aggregate the business categorical and attribute information (see below)
5. Clean and aggregate the review data to business level - calculate sum and mean statistics for counts such as "cool" and "funny" reviews
6. Clean the user table and compute relevant statistics such as 'outlier' reviewers and 'elite' reviewers
7. Left join the user data onto the review data (keep all rows from review and join in user information)
8. Left join the user/review table onto the business table

These steps can be seen in the flow diagram below (figure 1).

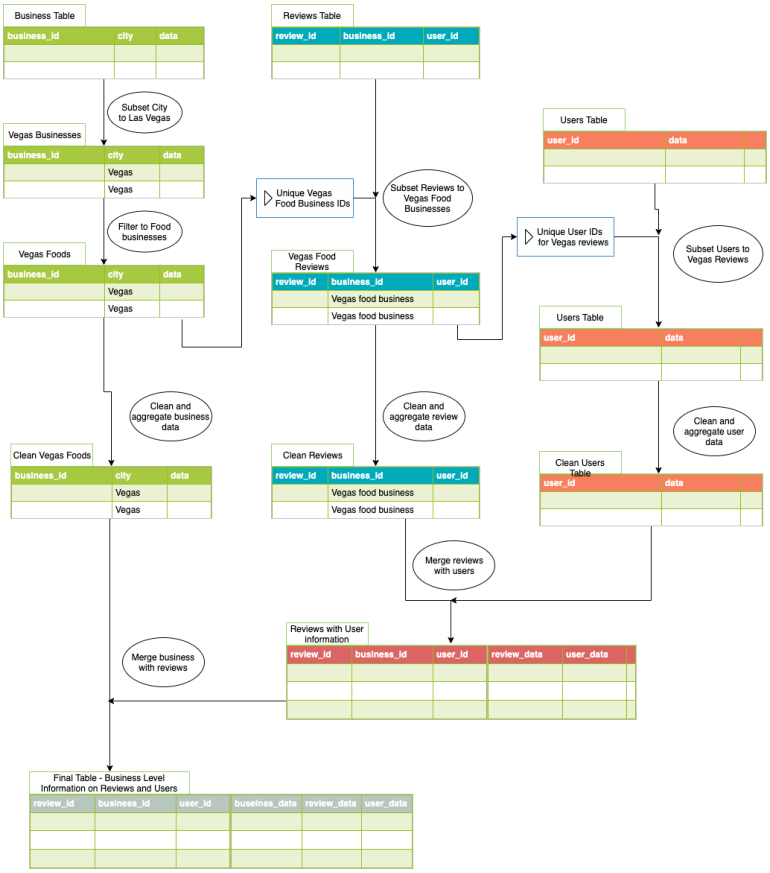


Figure 1: Top Food Related Business Categories

Business Data Processing

After filtering to the city of Las Vegas, and only businesses that were open, we were left with 23,793 businesses.

Looking at the categories broadly, we get an initial breakdown by count:

Table 1: Initial category frequency counts

Category	Frequency
Restaurants	4,221
Shopping	4,144
Home Services	3,508
Health & Medical	2,927

The category variable is a comma separated list of categories the business fits into, of varying specificity. Including Restaurant, Food, and also more specifics like cuisine and specific dishes (eg Chimney Cakes, Churros).

Yelp provides a list of possible categories on their site. We went through this and manually extracted 416 "food and bar related" categories. We then processed through the category variables, filtering food related businesses and extracting the first more descriptive category the business fell into (ie cuisine) so that the category variable is a single label. If no further descriptive categories were provided, "restaurant" or "bar" was returned.

We are then left with 5,573 food related businesses in Las Vegas, with 171 unique top categories.

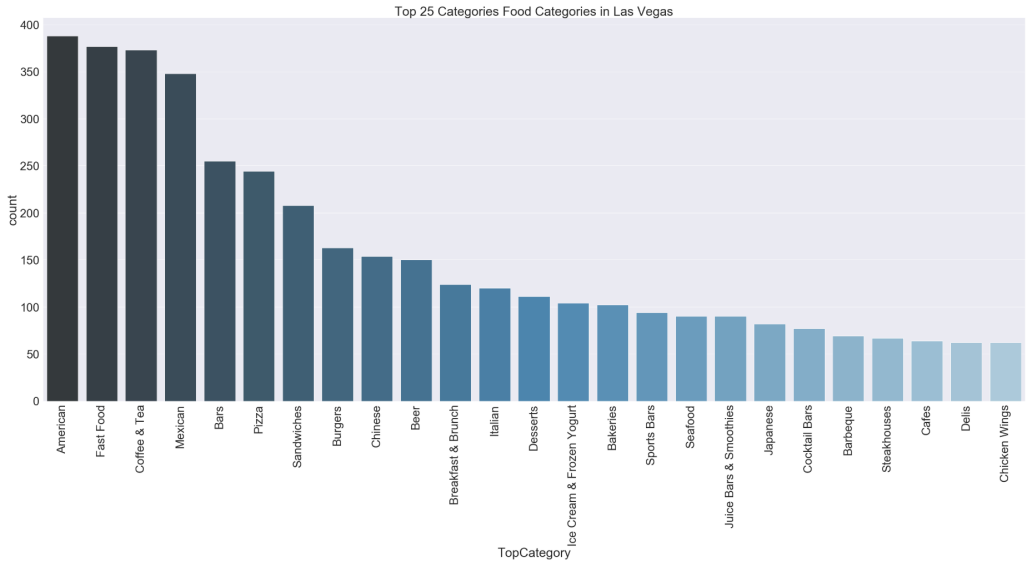


Figure 2: Top Food Related Business Categories (seaborn)

We also had to clean the JSON attribute dictionary and determine the most common attributes available between all businesses.

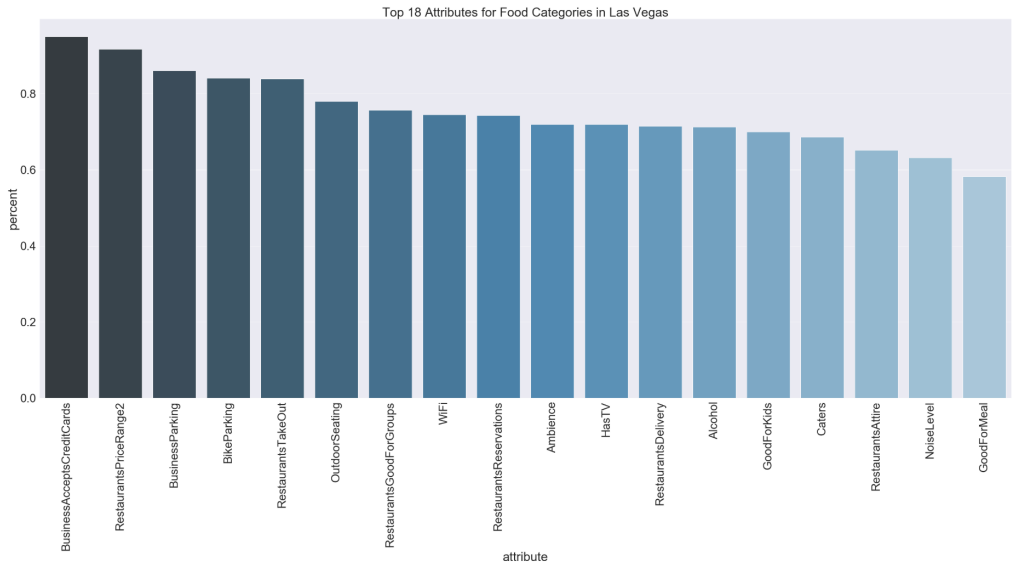


Figure 3: Top Business Attributes (seaborn)

After filtering out the non-relevant attributes, we also needed to one-hot encode the attribtues that weren't true/false like abiance (which includes: "divey","classy","hipster" etc.). We were unable to leave these as one column factor. After addressing NaNs vs. blank objects, we were left with the final predictors for each business (as seen in the final table metadata appendix section).

User Data Processing

As the user data is already at a single row of data per user there was no aggregation required. However to get the most out of the users table we need relevant statistics that will eventually be aggregated up to business level for the final dataset. Hence much of the original user information would be of little use at this higher aggregate level.

To overcome this we decided to focus on a number of key user statistics that would be helpful at business level:

- Users with "outlier" number of friends/fans
- Users with elite status
- Users friend network
- Users activity

NA issues in User Data

One of the biggest issues with processing the user data is that the data is not a perfect subset of the reviews data. Our initial hypothesis on the data is that Yelp had provided a subset of businesses throughout the US and their relevant reviews and user data. However this was incorrect. Whilst the review data did match up well to the businesses, the users did not match up anywhere near so neatly to the reviews table.

In fact on for most businesses reviews we only have around 65-70% of the user data. The remaining 30-35% of user information is entirely missing from the provided data. In essence this means we are missing data on a third of the users who wrote the reviews on our businesses of interest.

Aggregating the data from individual review to business level overcomes a lot of the processing issues (in terms of NAs) although we do need to be aware that our user information is not built upon complete data. An example:

Business_id	Review_id	User_id	Number_of_friends
Restaurant_1	Review 1	UserA	10
Restaurant_1	Review 2	UserB	50
Restaurant_1	Review 3	NA	NA
Restaurant_1	Review 4	UserC	90

So when this data is aggregated up to business level there aren't any overall NAs - we have enough data to compute the average statistics we need for the analysis:

Business_id	Total_Number_of_friends
Restaurant_1	150

However this means that we are assuming that our data on users is missing totally at random and that on average our missing data should not skew or effect the statistics we compute for each business. In this case the missing user could have had 10,000 friends and therefore the reviews would be reaching a much larger audience than we would see in the data. There is however no way to impute this data and so this is an assumption we have to live with and will be noted alongside any results.

EDA

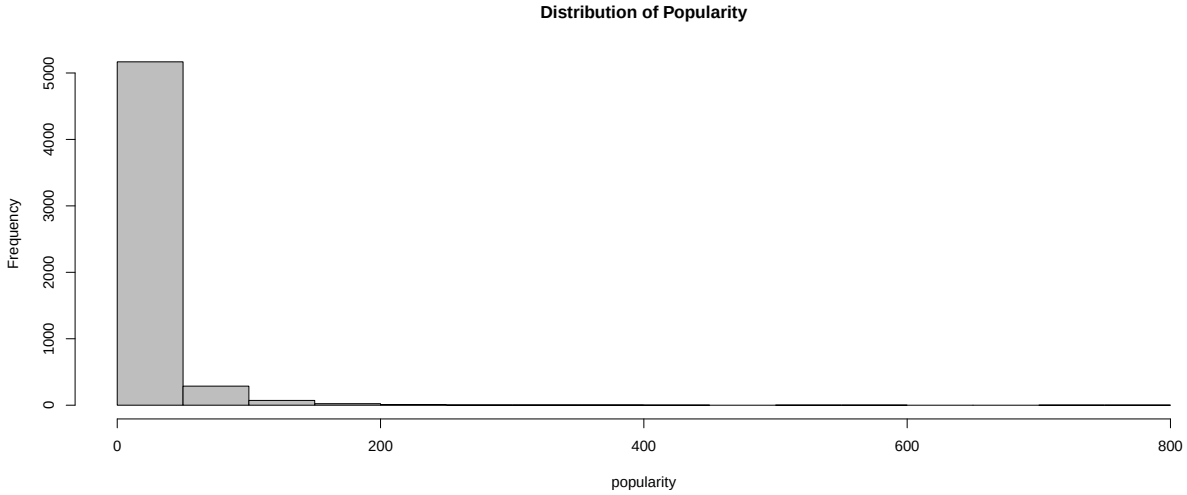
Response Variable

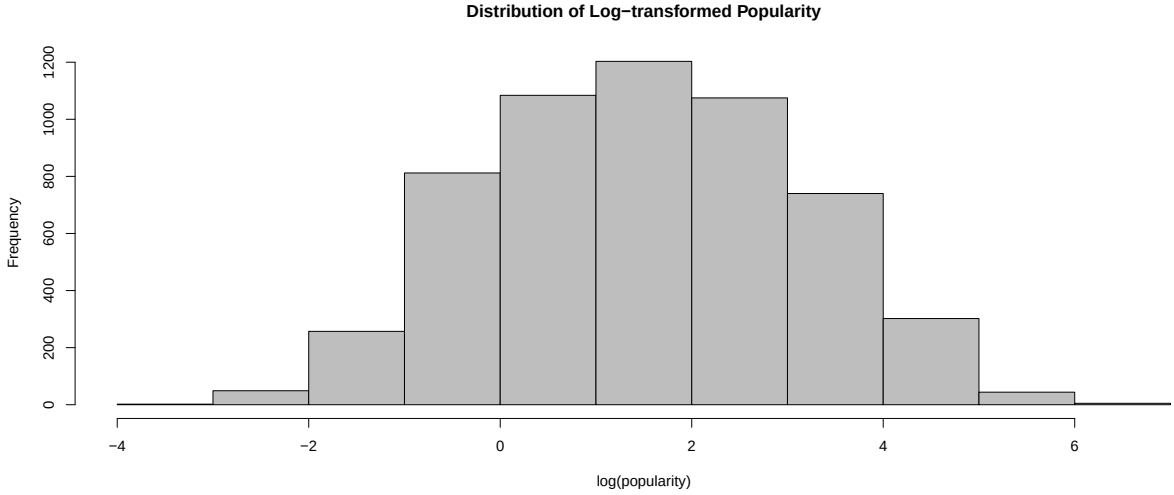
We wish to learn what attributes of a restaurant contribute to the popularity of a restaurant. Here we define 'popularity' of a restaurant to encompass: 1) frequency of visits to the restaurant as proxied by the number of check-in's and 2) number of Yelp reviews the restaurant received. Based on this definition, the 'popularity' measure of a restaurant was constructed as follows:

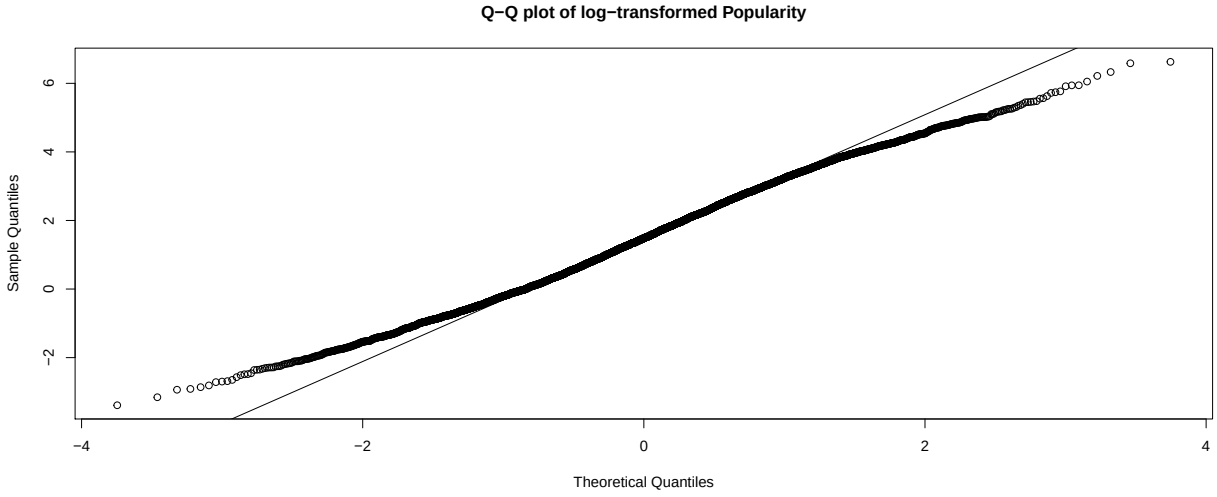
$$\frac{\text{Number of check-in's+Number of Yelp reviews}}{\text{Number of months open}}$$

The total sum of check-in's and reviews for a business is divided by how long the business has been open to make the popularity measure comparable across businesses that have been open for different durations. The Yelp dataset does not provide information about how long a business has been open; the number of months a business has been open was approximated by computing the difference in the latest and oldest dates of reviews or check-in's.

popularity
Min. : 0.0337
1st Qu.: 1.3124
Median : 4.4023
Mean : 14.9258
3rd Qu.: 14.8738
Max. :754.5333







The median popularity of restaurants is approximately 4.41 and 75% of restaurants have a popularity measure below 14.94. There are 630 restaurants considered to have outlier values for popularity measure (outliers defined as values above 3rd Quantile + 1.5*IQR or 35.2158594). The response value is highly right-skewed;

the normality assumption on whcih the linear model relies upon will most likely be violated. The response variable could be log-transformed, yielding a symmetric distribution.

Exploring Predictors

Not all features included in the dataset may be suitable for predicting popularity of restaurants: for example, certain features may have too many missing observations or several features may be highly correlated with one another. Here we explore 1) how certain features are distributed across businesses 2) missingness and potential multicollinearity issues in predictors 2) relationship between popularity and the predictors.

Here we've included just the first four predictors within the dataset as an example - however a comprehensive list of all feature distributions can be found in the Appendix (section 1.2).

X	address	business_id	city
Min. : 0	: 119	_-3aZXuecjOSjramI1IBkA: 1	Las Vegas:5573
1st Qu.:1393	5757 Wayne Newton Blvd: 52	_-DzOH5j-PbRrMf_Nm6w-Q: 1	NA
Median :2786	3799 Las Vegas Blvd S : 32	_-o4WsGS2yhFneiuMTNdSA: 1	NA
Mean :2786	3355 Las Vegas Blvd S : 27	_-QrdVYJbebaC2qschZ6iQ: 1	NA
3rd Qu.:4179	3570 Las Vegas Blvd S : 24	_0x7W6fizaPP76xNBxBLAQ: 1	NA
Max. :5572	3131 Las Vegas Blvd S : 23	_0ZIFTvfcA3UETO_S_JTNA: 1	NA
NA	(Other) :5296	(Other) :5567	NA

Distribution of Predictors across Businesses

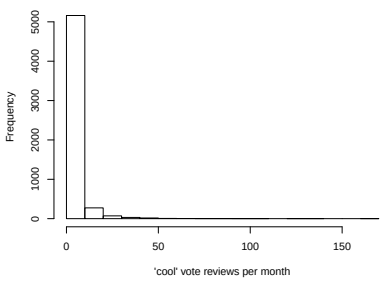
TopCategory variable indicates the 'main' cuisine or food cateogry a business serves. We explore how many different categories exist and the number of businesses associated for each of the categories

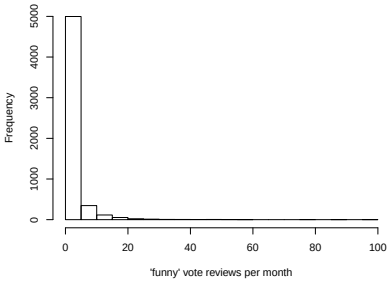
	Category	Frequency
3	American	388
64	Fast Food	377
43	Coffee & Tea	373
109	Mexican	348
11	Bars	255
122	Pizza	244
132	Sandwiches	208
26	Burgers	163
39	Chinese	154
14	Beer	150
21	Breakfast & Brunch	124
94	Italian	120
54	Desserts	111
88	Ice Cream & Frozen Yogurt	104
9	Bakeries	102

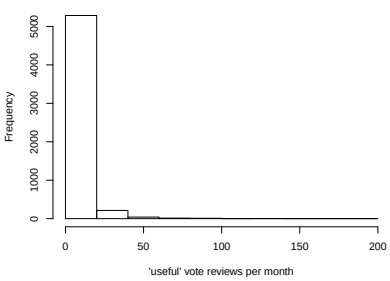
There are 171 different food categories. We may wish to merge categories for which there are only a few businesses into 'Other' category.

Exploring Relationship Predictors and Response

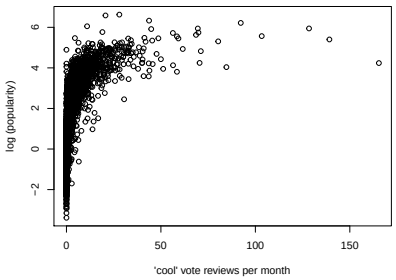
Here we explore the relationship bewteen quantitaive predictors and the response. We are interested in learning whether there appears to be a relationship between the predictors and the response and if so whether or not the relationship is linear. **Note: in all plots below, popularity variable has been log-transformed. The predictors are also heavily right-skewed and we may consider log-transforming the predictors as well. For some predictors, the minimum value observed is zero; a small constant should be added to all values of the observation when performing log-transformation**

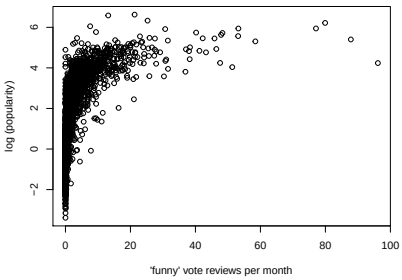


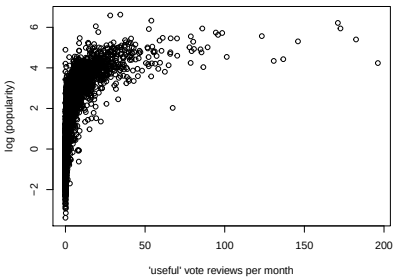


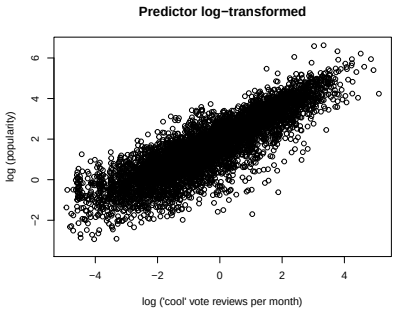


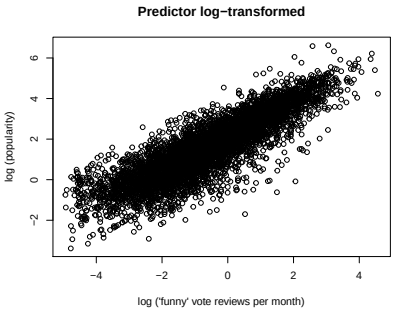
	cool_review	funny_review	useful_review
Min.	0.0000000	0.000000	0.0000000
1st Qu.	0.1475216	0.131739	0.3018946
Median	0.5982054	0.492126	1.1065131
Mean	2.9119535	2.004509	4.5916715
3rd Qu.	2.6119403	1.832173	4.3496681
Max.	165.3846154	96.153846	196.1538462

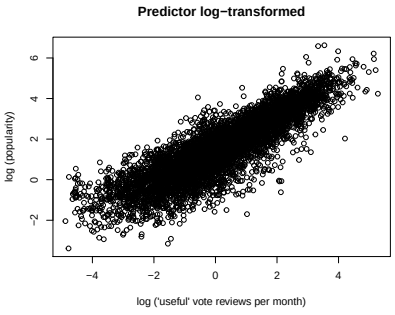






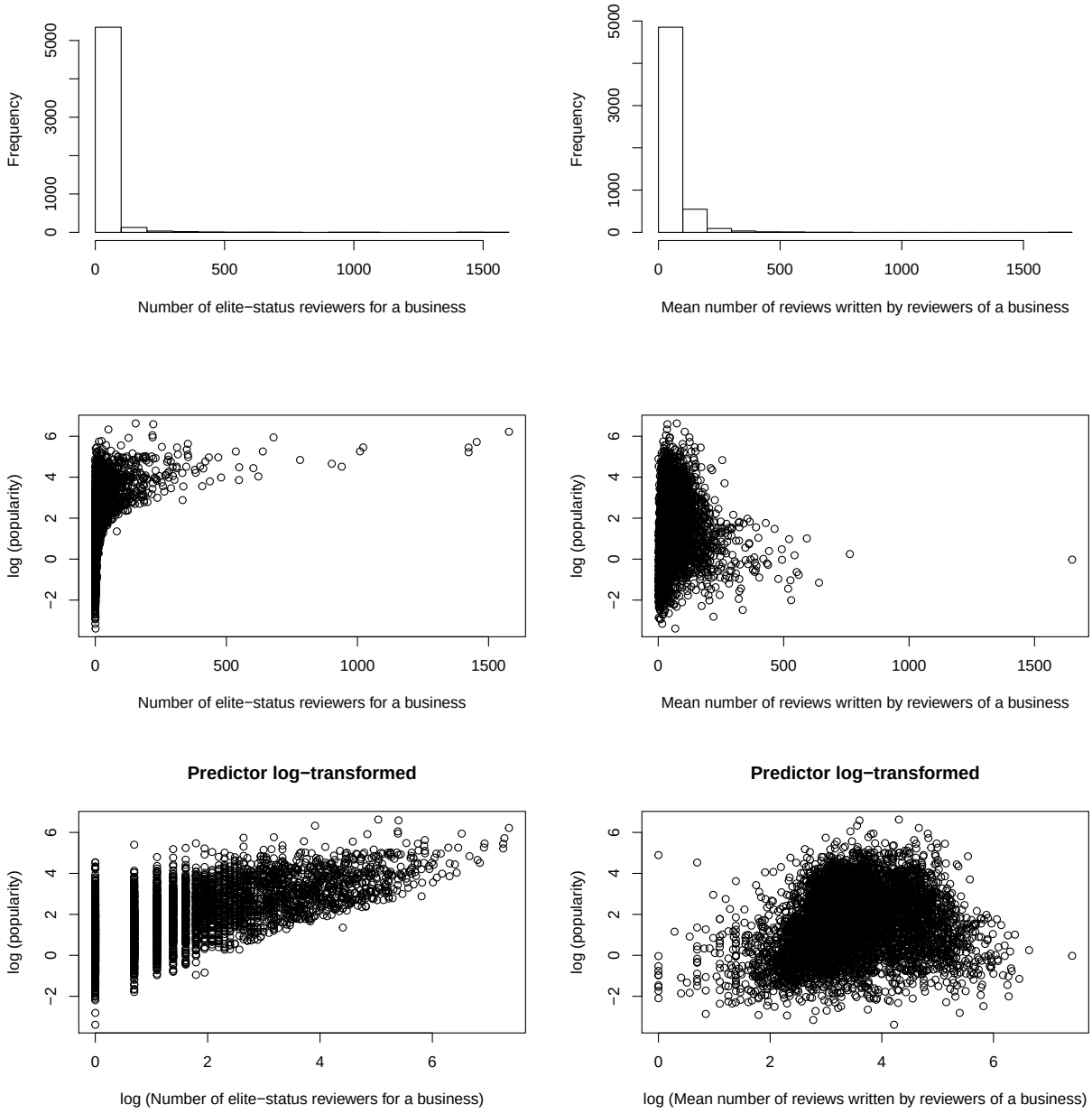






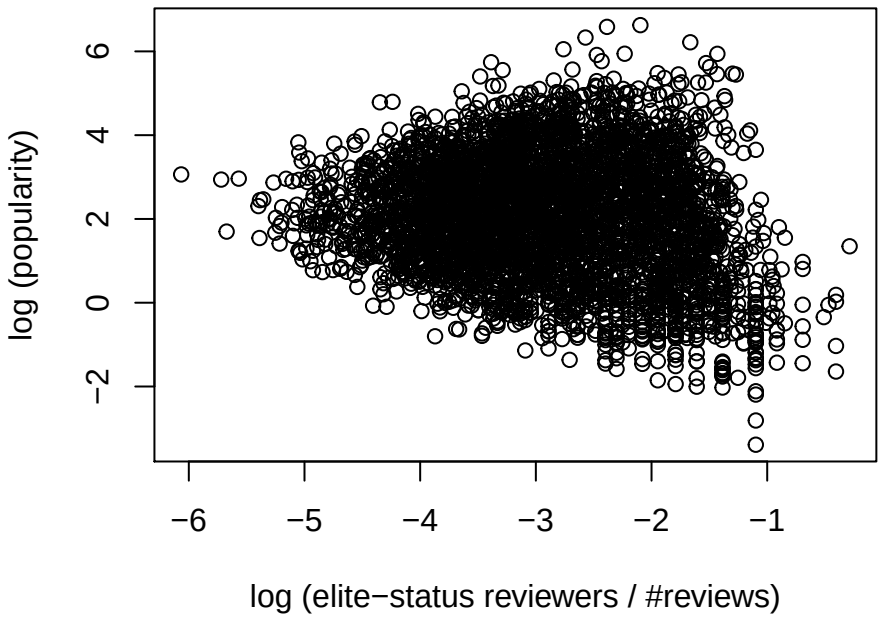
'cool', 'funny', and 'useful' votes are highly right-skewed; we may consider log-transforming these predictors as well. There appears to be a strong positive relationship between popularity and the number of 'cool',

'funny', and 'useful' votes reviews of a business receive per month.

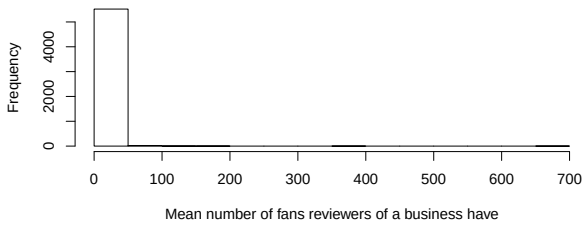


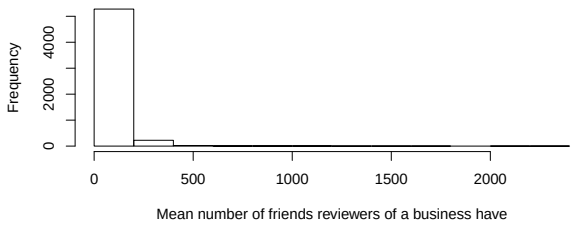
The number of elite-status reviewers and mean number of reviews written by reviewers are right-skewed; again, we may consider log-transforming these predictors.

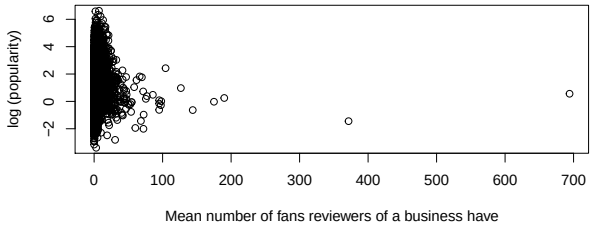
How 'active' reviewers of a business are, as proxied by the mean number of reviews written by reviewers of a business, may not be associated with the popularity of a business. There appears to be a positive association between the number of elite-status reviewers for a business and a business' popularity. This positive relationship may simply be due to the fact that businesses with a greater number of reviews (which was used to derive the popularity of a business) are also more likely to have a greater number elite-status reviewers. To correct for this, we should divide the number of elite-status reviewers for a business by the total number of reviews for a business.

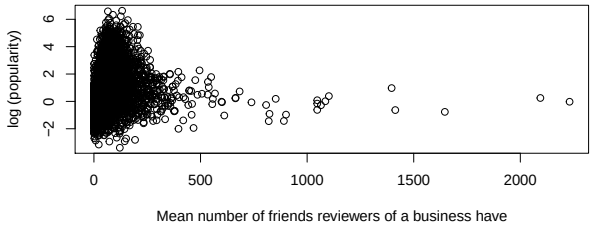


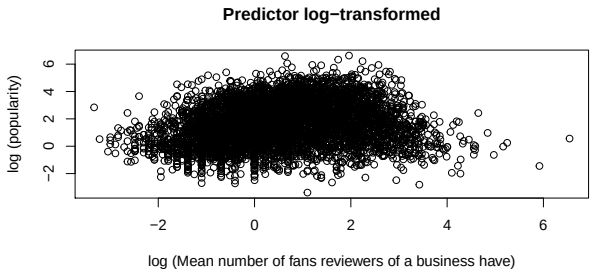
When we adjust the number of elite-status reviewers for a business by the number of reviews it received, we no longer see a strong positive relationship between the prevalence of elite-status reviewers and popularity of a business.

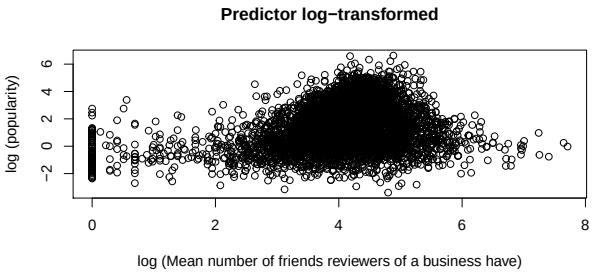




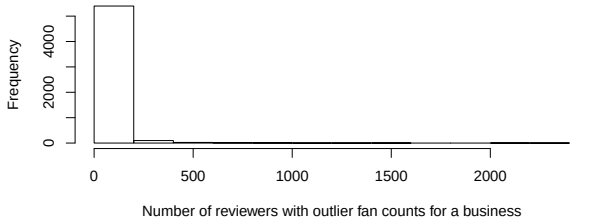


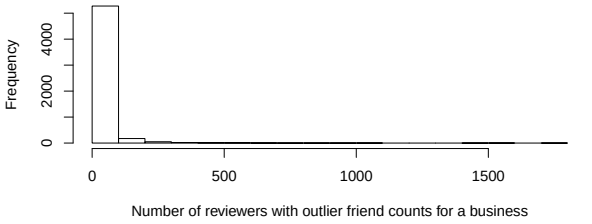


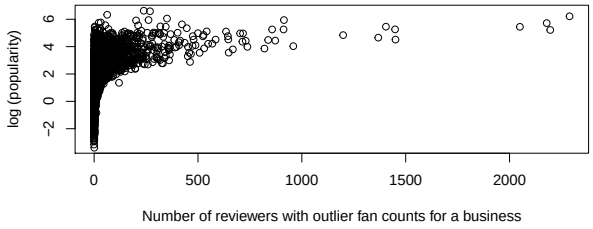


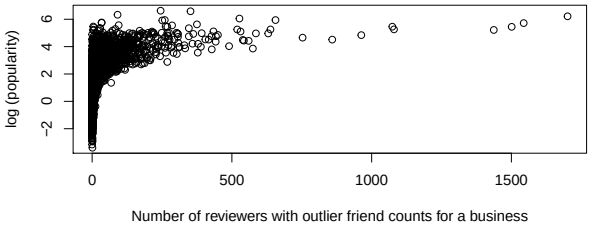


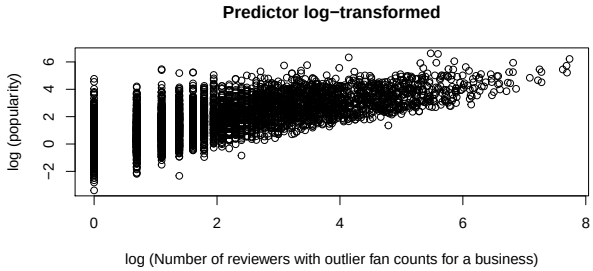
There does not appear to be a strong association between the mean number of fans/friends reviewers of a business have and the popularity of a business.

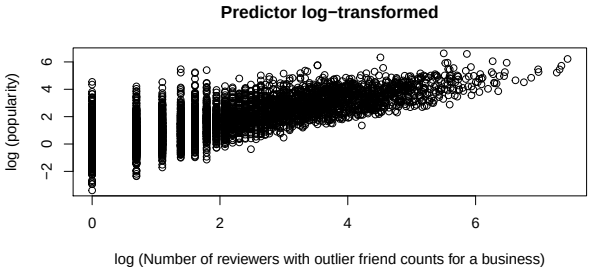




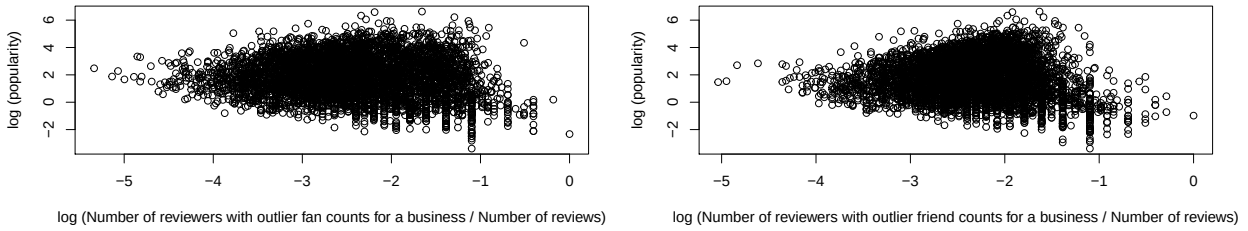








There appears to be a positive correlation between the number of reviewers with outlier fan/friend counts for a business and the popularity of a business: businesses that have greater of number reviewers with a lot of fans or friends tend to be more popular. Again, the positive correlation between the response and these predictors may simply be due to the fact that businesses that have more reviews (thus greater popularity) are more likely to have greater number of reviewers with outlier fan/friend counts. As before, we adjust the number of reviewers with outlier fan/friend counts by dividing by the total number of reviews for the business.



Now we no longer see as strong of a positive correlation between the number of reviewers with outlier fan/friends and popualrity.

Missing Values

There are missing values for multiple features - 26 businesses do not have any information about their reviewers - Missing values are prevalent for categorical variables (e.g. BikeParking, HasTV, Restaurants-GoodForGroups, etc.)

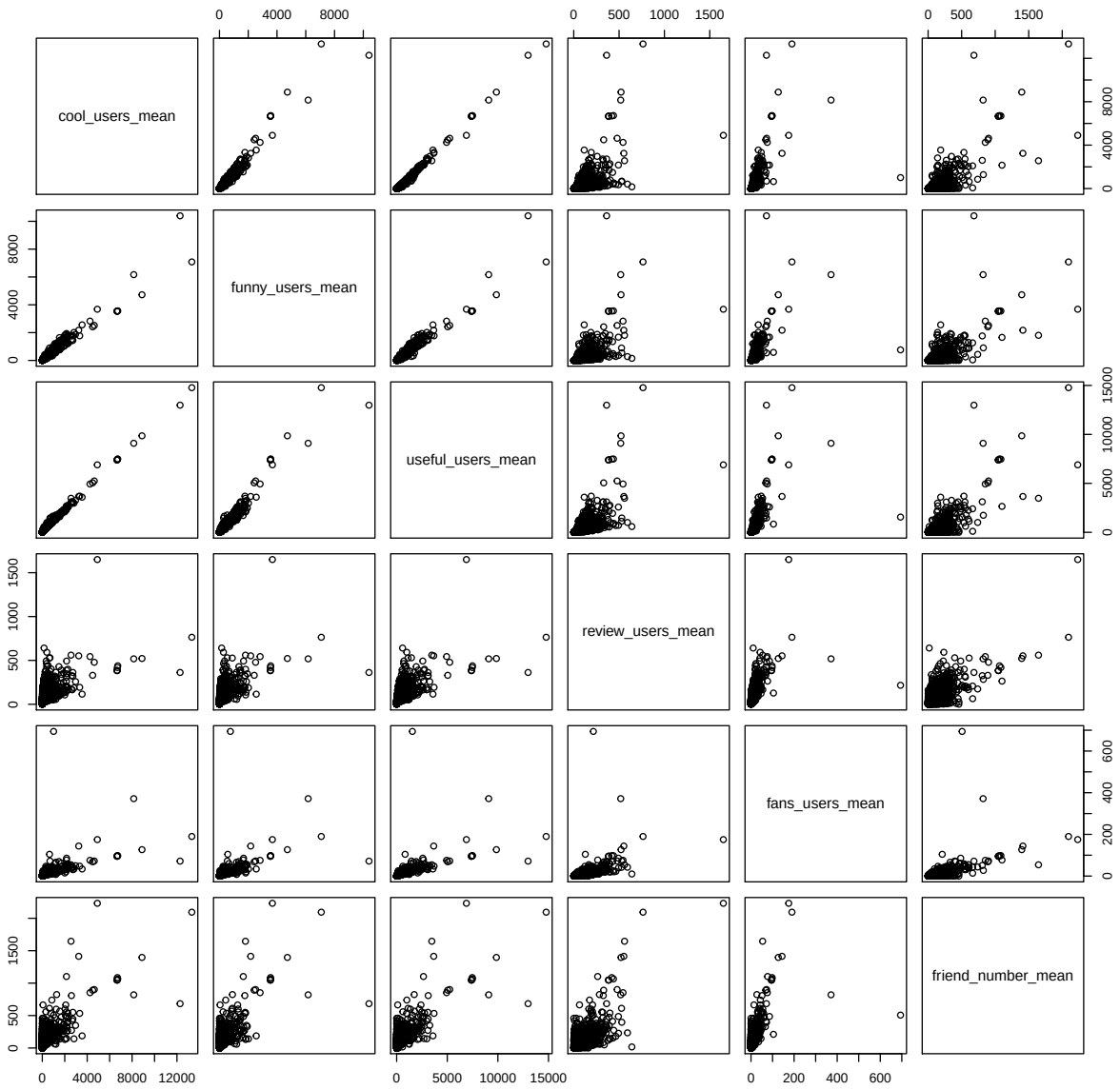
If we exclude businesses missing values for any of the features, we are left with 2548 businesses in the dataset. Inputing missing values in our categorical variables does not seem appropriate (for example, if a business has a NA for the 'GoodforKids' attribute, it does not seem reasonable to impute True or False). Given that our dataset is large enough even after dropping businesses with missing values for any of the attributes, we may leave the missing values as they are.

Correlations between Predictors

Reviewer related Variables

The dataset contains information about the reviewers of a business. For example, for each business we computed the mean number of reviews the business' reviewers have written on Yelp overall*, the mean number of fans the business' reviewers have, the mean number of 'cool' votes the business' reviewers gave on Yelp overall, etc. These variables are intended to capture how active or 'influential' reviewers of a business might be. We could imagine that the more active and influential the reviewers of a business are, the more popular the business. These reviewer related variables, however, may be highly correlated with each other.

*Each reviewer has a total number of reviews that he/she has written on Yelp, not just for that single business. We took these review counts and averaged across reviewers of a business.



There are very strong positive correlations between the reviewer related variables. Take for example, the greater the mean number of 'useful' votes a business' reviewers gave on Yelp overall, the greater the mean number of 'cool' and 'funny' votes they gave on Yelp overall. The mean number of friends and the mean number of fans a business' reviewers have are positively correlated with each other. The mean number of reviews by a business' reviewers on Yelp overall is also highly correlated with all other reviwer related variables. Given the strong positive correlation between the various reviewer related variables, we should consider using only a subset of them in our models.

Next Steps

Performing this EDA has illuminated some issues that we will resolve in our next iteration for modeling.

- Reduce number of categories of food businesses from 171 – putting non top ones into an "other" bucket
- Variables that are too correlated with review count will be converted to proportions
- Fan outlier count
- Friend outlier count
- Elite users sum
- In model, we will drop all of the sum aggregations in favor of means since we realized this is essentially duplicate information
- N/As: from users table and in business attributes, we are unable to robustly impute things like 0 or "false" so we will keep these missing values for now. Though these observations will be dropped from our model, we will still have ½ of the dataset remaining which is still a very strong samplesize.

Appendix

1.1 - Metadata

Table 8: Metadata for our final cleaned dataframe

	id	dataset	description
0	address	business	string
1	business_id	business	string, 22 character unique string business id
2	city	business	city (Las Vegas)
3	latitude	business	latitude
4	longitude	business	longitude
5	name	business	business name
6	postal_code	business	US postal code
7	review_count_business	business	number of reviews for the business
8	stars_business	business	star rating for the business
9	state	business	state
10	TopCategory	business	top category from within the list of possible business types
11	BikeParking	business	has bike parking y/n
12	RestaurantsPriceRange2	business	price range
13	HasTV	business	restaurant has TV
14	RestaurantsGoodForGroups	business	restaurant is good for groups
15	OutdoorSeating	business	outdoor seating
16	RestaurantsReservations	business	restaurants has reservations
17	GoodForKids	business	good for kids
18	Alcohol	business	alcohol served
19	NoiseLevel	business	noiseLevel
20	BusinessAcceptsCreditCards	business	accepts credit cards
21	WiFi	business	WiFi
22	RestaurantsAttire	business	restaurant attire
23	Caters	business	caters
24	RestaurantsTakeOut	business	takeOut
25	RestaurantsDelivery	business	delivery
26	garage	business	garage
27	lot	business	lot

	id	dataset	description
28	street	business	street
29	valet	business	valet
30	validated	business	validated
31	casual	business	casual
32	classy	business	classy
33	divey	business	divey
34	hipster	business	hipster
35	intimate	business	intimate
36	romantic	business	romantic
37	touristy	business	touristy
38	trendy	business	trendy
39	upscale	business	upscale
40	breakfast	business	breakfast
41	brunch	business	brunch
42	dessert	business	dessert
43	dinner	business	dinner
44	latenight	business	latenight
45	lunch	business	lunch
46	checkin_count	reviews	checkin_count
47	months_open	reviews	number of months restaurant has been open
48	review_count	reviews	number of reviews for the business from reviews (duplicate?)
49	cool_review_count	reviews	number of reviews tagged as cool
50	funny_review_count	reviews	number of reviews tagged as funny
51	useful_review_count	reviews	number of reviews tagged as useful
52	popularity	reviews	response variable - overall measure of popularity by combining
53	cool_review_pm	reviews	number of cool reviews per month
54	funny_review_pm	reviews	number of funny reviews per month
55	useful_review_pm	reviews	number of useful reviews per month
56	cool_review_mean	reviews	mean number of cool reviews
57	funny_review_mean	reviews	mean number of funny reviews
58	stars_review_mean	reviews	mean number of review stars
59	useful_review_mean	reviews	mean number of useful reviews
60	stars_users_mean	users	mean number of average stars per user who reviewed
61	cool_users_mean	users	mean number of cool votes by users who reviewed
62	elite_users_sum	users	"number of users who have ever been elite
63	fans_users_sum	users	sum total number of fans of all those who reviewed
64	fans_users_mean	users	mean number of fans of all those who reviewed
65	funny_users_mean	users	mean number of funny votes by users who reviewed
66	review_users_sum	users	total number of reviews written by users who reviewed
67	review_users_mean	users	mean number of reviews written by users who reviewed
68	useful_users_sum	users	total number of useful votes by users who reviewed
69	useful_users_mean	users	mean number of useful votes by users who reviewed
70	friend_number_count	users	sum total number of friends of all those who reviewed
71	friend_number_mean	users	mean number of friends of all those who reviewed
72	reviews_users_pm_mean	users	mean number of reviews written per month by those who reviewed
73	fan_outlier_count	users	number of users who reviewed who are considered an outlier in terms of their number of fans
74	friend_outlier_count	users	number of users who reviewed who are considered an outlier in terms of their number of friends

1.2 - Full Data Distributions

X	address	business_id	city
Min. : 0	: 119	_-3aZXuecjOSjramI1IBkA: 1	Las Vegas:5573
1st Qu.:1393	5757 Wayne Newton Blvd: 52	_-DzOH5j-PbRrMf_Nm6w-Q: 1	NA
Median :2786	3799 Las Vegas Blvd S : 32	_-o4WsGS2yhFneiuMTNdSA: 1	NA
Mean :2786	3355 Las Vegas Blvd S : 27	_-QrdVYJbebaC2qschZ6iQ: 1	NA
3rd Qu.:4179	3570 Las Vegas Blvd S : 24	_0x7W6fizaPP76xNBxBLAQ: 1	NA
Max. :5572	3131 Las Vegas Blvd S : 23	_0ZIFTvfcA3UETO_S_JTNA: 1	NA
NA	(Other) :5296	(Other) :5567	NA

latitude	longitude	name	postal_code
Min. :35.92	Min. :-115.5	Starbucks : 138	Min. :87701
1st Qu.:36.10	1st Qu.:-115.2	Subway : 107	1st Qu.:89109
Median :36.12	Median :-115.2	7-Eleven : 83	Median :89118
Mean :36.13	Mean :-115.2	McDonald's : 77	Mean :89122
3rd Qu.:36.16	3rd Qu.:-115.1	Panda Express: 40	3rd Qu.:89135
Max. :36.38	Max. :-114.9	Taco Bell : 39	Max. :93013
NA	NA	(Other) :5089	NA's :10

review_count_business	stars_business	state	TopCategory
Min. : 3.0	Min. :1.000	NM: 1	American : 388
1st Qu.: 18.0	1st Qu.:3.000	NV:5572	Fast Food : 377
Median : 58.0	Median :3.500	NA	Coffee & Tea: 373
Mean : 193.8	Mean :3.526	NA	Mexican : 348
3rd Qu.: 192.0	3rd Qu.:4.000	NA	Bars : 255
Max. :8348.0	Max. :5.000	NA	Pizza : 244
NA	NA	NA	(Other) :3588

BikeParking	RestaurantsPriceRange2	HasTV	RestaurantsGoodForGroups
: 888	Min. :1.000	:1568	:1357
False:1626	1st Qu.:1.000	False:1880	False: 325
True :3059	Median :2.000	True :2125	True :3891
NA	Mean :1.605	NA	NA
NA	3rd Qu.:2.000	NA	NA
NA	Max. :4.000	NA	NA
NA	NA's :463	NA	NA

OutdoorSeating	RestaurantsReservations	GoodForKids	Alcohol
:1229	:1434	:1676	:1601
False:3111	False:2975	False: 695	beer_and_wine: 580
True :1233	True :1164	True :3202	full_bar :1442
NA	NA	NA	none :1950
NA	NA	NA	NA
NA	NA	NA	NA

OutdoorSeating	RestaurantsReservations	GoodForKids	Alcohol
NA	NA	NA	NA

NoiseLevel	BusinessAcceptsCreditCards	WiFi	RestaurantsAttire
:2053	: 281	:1424	:1947
average :2569	False: 91	free:1979	casual:3503
loud : 296	True :5201	no :2113	dressy: 122
quiet : 535	NA	paid: 57	formal: 1
very_loud: 120	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

Caters	RestaurantsTakeOut	RestaurantsDelivery	garage
:1753	: 896	:1592	: 873
False:2110	False: 447	False:3067	False:3902
True :1710	True :4230	True : 914	True : 798
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

lot	street	valet	validated
: 873	: 873	: 873	: 873
False:2369	False:4377	False:4365	False:4668
True :2331	True : 323	True : 335	True : 32
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

casual	classy	divey	hipster
:1606	:1606	:1605	:1606
False:1485	False:3778	False:3872	False:3876
True :2482	True : 189	True : 96	True : 91
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

intimate	romantic	touristy	trendy
:1606	:1606	:1606	:1606
False:3884	False:3902	False:3854	False:3611
True : 83	True : 65	True : 113	True : 356
NA	NA	NA	NA

intimate	romantic	touristy	trendy
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

upscale	breakfast	brunch	dessert
:1607	:2341	:2341	:2341
False:3852	False:2661	False:2632	False:2792
True : 114	True : 571	True : 600	True : 440
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA
NA	NA	NA	NA

dinner	latenight	lunch	checkin_count
:2341	:2341	:2341	Min. : 0.0
False:1318	False:2712	False:1061	1st Qu.: 44.0
True :1914	True : 520	True :2171	Median : 182.0
NA	NA	NA	Mean : 695.1
NA	NA	NA	3rd Qu.: 654.0
NA	NA	NA	Max. :38277.0
NA	NA	NA	NA

months_open	review_count	cool_review_count	funny_review_count
Min. : 0.03	Min. : 2	Min. : 0	Min. : 0.0
1st Qu.: 33.35	1st Qu.: 18	1st Qu.: 9	1st Qu.: 7.0
Median : 73.52	Median : 59	Median : 36	Median : 29.0
Mean : 72.14	Mean : 197	Mean : 137	Mean : 107.4
3rd Qu.:103.55	3rd Qu.: 196	3rd Qu.: 127	3rd Qu.: 95.0
Max. :165.52	Max. :8570	Max. :6852	Max. :5935.0
NA	NA	NA	NA

useful_review_count	popularity	cool_review_pm	funny_review_pm
Min. : 0.0	Min. : 0.0337	Min. : 0.0000	Min. : 0.0000
1st Qu.: 17.0	1st Qu.: 1.3124	1st Qu.: 0.1475	1st Qu.: 0.1317
Median : 64.0	Median : 4.4023	Median : 0.5982	Median : 0.4921
Mean : 227.8	Mean : 14.9258	Mean : 2.9120	Mean : 2.0045
3rd Qu.: 214.0	3rd Qu.: 14.8738	3rd Qu.: 2.6119	3rd Qu.: 1.8322
Max. :12701.0	Max. :754.5333	Max. :165.3846	Max. :96.1539
NA	NA	NA	NA

useful_review_pm	cool_review_mean	funny_review_mean	stars_review_mean
Min. : 0.0000	Min. : 0.0000	Min. : 0.0000	Min. :1.000
1st Qu.: 0.3019	1st Qu.: 0.3214	1st Qu.: 0.2604	1st Qu.:2.974

useful_review_pm	cool_review_mean	funny_review_mean	stars_review_mean
Median : 1.1065	Median : 0.5455	Median : 0.4343	Median :3.636
Mean : 4.5917	Mean : 0.7546	Mean : 0.5907	Mean :3.523
3rd Qu.: 4.3497	3rd Qu.: 0.8837	3rd Qu.: 0.6881	3rd Qu.:4.162
Max. :196.1539	Max. :28.3333	Max. :19.3333	Max. :5.000
NA	NA	NA	NA

useful_review_mean	stars_users_mean	cool_users_mean	elite_users_sum
Min. : 0.0000	Min. :1.000	Min. : 0.000	Min. : 0.00
1st Qu.: 0.6667	1st Qu.:3.452	1st Qu.: 4.765	1st Qu.: 0.00
Median : 0.9875	Median :3.729	Median : 14.493	Median : 2.00
Mean : 1.2342	Mean :3.668	Mean : 108.071	Mean : 16.83
3rd Qu.: 1.4500	3rd Qu.:3.933	3rd Qu.: 69.775	3rd Qu.: 9.00
Max. :36.3333	Max. :5.000	Max. :13323.000	Max. :1578.00
NA	NA's :26	NA's :26	NA's :26

fans_users_sum	fans_users_mean	funny_users_mean	review_users_sum
Min. : 0.0	Min. : 0.0000	Min. : 0.000	Min. : 1
1st Qu.: 10.0	1st Qu.: 0.5366	1st Qu.: 4.333	1st Qu.: 314
Median : 67.0	Median : 1.4052	Median : 11.750	Median : 1313
Mean : 701.3	Mean : 4.2895	Mean : 77.621	Mean : 8512
3rd Qu.: 335.5	3rd Qu.: 4.1723	3rd Qu.: 55.068	3rd Qu.: 4942
Max. :63081.0	Max. :694.1429	Max. :10396.357	Max. :709651
NA's :26	NA's :26	NA's :26	NA's :26

review_users_mean	useful_users_sum	useful_users_mean	friend_number_count
Min. : 1.00	Min. : 0	Min. : 0.00	Min. : 1
1st Qu.: 17.92	1st Qu.: 291	1st Qu.: 15.78	1st Qu.: 666
Median : 28.62	Median : 1753	Median : 34.84	Median : 2584
Mean : 50.10	Mean : 20293	Mean : 151.41	Mean : 11788
3rd Qu.: 59.50	3rd Qu.: 11499	3rd Qu.: 123.81	3rd Qu.: 8920
Max. :1650.00	Max. :1773369	Max. :14771.00	Max. :844347
NA's :26	NA's :26	NA's :26	NA's :26

friend_number_mean	reviews_users_pm_mean	fan_outlier_count	friend_outlier_count	popularity_log
Min. : 1.00	Min. : 0.01613	Min. : 0.00	Min. : 0.00	Min. :-3.3893
1st Qu.: 40.06	1st Qu.: 0.38555	1st Qu.: 1.00	1st Qu.: 1.00	1st Qu.: 0.2719
Median : 62.00	Median : 0.57934	Median : 4.00	Median : 5.00	Median : 1.4821
Mean : 80.11	Mean : 0.88969	Mean : 27.41	Mean : 23.33	Mean : 1.4954
3rd Qu.: 93.86	3rd Qu.: 0.99759	3rd Qu.: 16.00	3rd Qu.: 18.00	3rd Qu.: 2.6996
Max. :2231.00	Max. :47.75000	Max. :2289.00	Max. :1701.00	Max. : 6.6261
NA's :26	NA's :26	NA's :26	NA's :26	NA

1.3 - Data Processing Code

1.3.1 - Read and Chunk Data

Python code to read in the raw data and deal with the large JSON files.

```
import pandas as pd
import numpy as np

# Constants
city_name = 'Las Vegas'
output_review_name = 'data/LV_reviews.csv'
output_users_name = 'data/LV_users.csv'

# Get business IDS
business = pd.read_json('data/business.json', lines=True)
subset_business_ids = set(business[business['city']==city_name]['business_id'])

# Get chunks from large datasets
user_chunks = pd.read_json('data/user.json', lines=True, chunksize=100000)
review_chunks = pd.read_json('data/review.json', lines=True, chunksize=100000)

# Set up empty df columns
for chunk in user_chunks:
    columns_users = chunk.columns
    break

for chunk in review_chunks:
    columns_reviews = chunk.columns
    break

# Get reviews from the business subet
reviews_df = pd.DataFrame(columns=columns_reviews)
for chunk in review_chunks:
    subset = chunk[chunk['business_id'].isin(subset_business_ids)]
    reviews_df = reviews_df.append(subset, ignore_index = True)

# Subset Users by Reviews
subset_user_ids = set(reviews_df['user_id'])

# Get users from reviews
users_df = pd.DataFrame(columns=columns_users)
for chunk in user_chunks:
    subset = chunk[chunk['user_id'].isin(subset_user_ids)]
    users_df = users_df.append(subset, ignore_index = True)

# Write files
users_df.to_csv(output_users_name)
reviews_df.to_csv(output_review_name)
```

1.3.2 - Clean User Data

Python code to take the user data, calculate statistics and

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import datetime

# Constants
city_name = 'Las Vegas'

# ------------------------- #
#    READ AND CLEAN DATA
# ------------------------- #
# Read data
users = pd.read_csv('data/LV_users.csv', low_memory=False)
users = users.drop('Unnamed: 0', axis=1)

reviews = pd.read_csv('data/LV_reviews.csv', low_memory=False)
reviews = reviews.drop('Unnamed: 0', axis=1)
# Clean NAs (only 1 for vegas)
reviews = reviews.dropna()

business = pd.read_csv('data/business.csv', low_memory=False)
business = business.drop('Unnamed: 0', axis=1)

business = business[business['city'] == 'Las Vegas']


# Clean Reviews
# Get the last review to represent the end of the dataset
latest_review = sorted(reviews['date'], reverse=True)[0]

# Convert dates
latest_review = datetime.datetime.strptime(latest_review, '%Y-%m-%d %H:%M:%S')
users['yelping_since_date'] = [datetime.datetime.strptime(x[-1], '%Y-%m-%d %H:%M:%S') for
                               x in users.itertuples()]

# average days per month
av_day_per_month = 30.44

# ------------------------- #
#   CALCULATE USER STATS
# ------------------------- #
users['count_friends'] = [len(row[-8].split(',')) for row in users.itertuples()]
users['months_of_activity'] = latest_review - users['yelping_since_date']
users['months_of_activity'] = [x.days for x in users['months_of_activity']]
users['months_of_activity'] = round(users['months_of_activity']/av_day_per_month)
users['reviews_per_month'] = users['review_count'] / users['months_of_activity']
users['elite_binary'] = np.where(pd.isna(users['elite']), 0, 1)

# Friends
Q1, Q3 = np.percentile(users['count_friends'], [25 ,75])
IQR = Q3 - Q1
```

```
influencer_friend_threshold = Q3 + (IQR*1.5)

# Fans
Q1, Q3 = np.percentile(users['fans'], [25 ,75])
IQR = Q3 - Q1
influencer_fan_threshold = Q3 + (IQR*1.5)

users['fan_outlier'] = np.where(users['fans']>influencer_fan_threshold, 1, 0)
users['friend_outlier'] = np.where(users['count_friends']>influencer_friend_threshold, 1, 0)

# ------------------------ #
#          JOINING
# ------------------------ #
# Ensure all ids are string
reviews.loc[:,'user_id'] = reviews['user_id'].astype(str)
reviews.loc[:,'business_id'] = reviews['business_id'].astype(str)
users.loc[:,'user_id'] = users['user_id'].astype(str)
business.loc[:,'business_id'] = business['business_id'].astype(str)

# Ensure there are no leading or trailing spaces
reviews['user_id'] = [string.strip(" ") for string in reviews['user_id']]
reviews['business_id'] = [string.strip(" ") for string in reviews['business_id']]
users['user_id'] = [string.strip(" ") for string in users['user_id']]
business['business_id'] = [string.strip(" ") for string in business['business_id']]

# JOIN
reviews_join = reviews.copy()
reviews_join= reviews_join.rename(columns={'cool':'cool_review', 'funny':'funny_review',
  'useful':'useful_review'})
reviews_join = reviews_join.merge(users, how='left', left_on='user_id', right_on='user_id')

print("Number of NA users in all business reviews: {}".format(len(reviews_join[pd.isna(
  reviews_join['count_friends'])])))
print('Totals rows in all business reviews: {}'.format(len(reviews_join)))

# Keep only relevant columns for now
reviews_join_user = reviews_join.loc[:,['business_id', 'cool_review', 'funny_review',
       'stars', 'useful_review', 'average_stars', 'cool', 'elite_binary', 'fans',
       'funny', 'review_count', 'useful', 'count_friends', 'reviews_per_month',
       'fan_outlier', 'friend_outlier']]

# Subset to just food businesses
unique_food_business = set(business['business_id'])

# Subset reviews to just food
reviews_join_user_food = reviews_join_user[reviews_join_user['business_id'].isin(
  unique_food_business)]

print("Number of NA users in all business reviews: {}".format(len(reviews_join_user_food[pd.isna(reviews
print('Totals rows in all business reviews: {}'.format(len(reviews_join_user_food)))

# Correct column dtypes
reviews_join_user_food['cool_review'] = reviews_join_user_food['cool_review'].astype(int)
```

```
# Aggregate to business level
reviews_join_user_food = reviews_join_user_food.groupby('business_id').agg(
  {'cool_review':np.mean, 'funny_review':np.mean,
    'stars':np.mean, 'useful_review':np.mean,
    'average_stars':np.mean, 'cool':np.mean, 'elite_binary':sum,
    'fans':[sum, np.mean],
    'funny':np.mean, 'review_count':[sum, np.mean],
    'useful':[sum, np.mean], 'count_friends':[sum, np.mean],
    'reviews_per_month':np.mean, 'fan_outlier':sum,
    'friend_outlier':sum})

print("Number of NAs in output: {}".format(np.count_nonzero(np.isnan(
  reviews_join_user_food['average_stars']['mean'].values))))
print("Number of Total in output: {}".format(len(reviews_join_user_food)))

# Ensure sums are NA where the mean values are NA (aka fake zeros)
reviews_join_user_food['elite_binary']['sum'] = np.where(pd.isna(
  reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['elite_binary']['sum'])
reviews_join_user_food['fans']['sum'] = np.where(pd.isna(
  reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['fans']['sum'])
reviews_join_user_food['review_count']['sum'] = np.where(
  pd.isna(reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['review_count']['sum'])
reviews_join_user_food['useful']['sum'] = np.where(
  pd.isna(reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['useful']['sum'])
reviews_join_user_food['count_friends']['sum'] = np.where(
  pd.isna(reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['count_friends']['sum'])
reviews_join_user_food['fan_outlier']['sum'] = np.where(
  pd.isna(reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['fan_outlier']['sum'])
reviews_join_user_food['friend_outlier']['sum'] = np.where(
  pd.isna(reviews_join_user_food['average_stars']['mean']),
  np.nan,
  reviews_join_user_food['friend_outlier']['sum'])

# Reindex multiindex columns
new_flat_cols = ['_'.join(col).strip() for col in reviews_join_user_food.columns.values]
reviews_join_user_food.columns = new_flat_cols
reviews_join_user_food.reset_index(inplace=True)

reviews_join_user_food.to_csv('data/vegas_business_user_info.csv', index=False)
```

1.3.3 - Clean Business Tables and EDA

Python code to clean and process business attributes

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from collections import Counter
pd.set_option('display.max_rows', 500)
import json


bus= pd.read_csv('business.csv')

#Get city
my_city = bus[bus['city']=='Las Vegas']
my_city = my_city[my_city['is_open']==1]

# Initial category counts
# Loop through all categories, and count each unique one
All_categories = my_city['categories'].values
All_categories_list = []
for i in All_categories:
    try:
        curr = i.split(',')
        for j in curr:
            All_categories_list.append(j.strip(" "))
    except:
        continue

All_categories_count = Counter(All_categories_list)
categories_df = pd.DataFrame.from_dict(All_categories_count, orient='index')
categories_df.sort_values(0,ascending=False,inplace=True)
categories_df.rename(columns={0:'Count'},inplace=True)


# Extract food related category list, adding on Bars as additional
food_cat_list=['American (New)','American (Traditional)','Canadian (New)','Absinthe Bars ','Beach Bars
           'Hookah Bars ','Hotel bar ','Irish Pub ','Pubs ','Pulquerias ','Sake Bars ','Speakeasies ','S
           'Vermouth Bars ','Whiskey Bars ','Wine Bars ','Beer Gardens ','Club Crawl ','Coffeeshops ','A
           'South African ','American ','American ','Andalusian ','Arabian ','Arab Pizza ','Argentine '
           'Australian ','Austrian ','Baguettes ','Bangladeshi ','Barbeque ','Basque ','Bavarian ','Beer
           'Flemish ','Bistros ','Black Sea ','Brasseries ','Brazilian ','Brazilian Empanadas ','Central
           'Northern Brazilian ','Rodizios ','Breakfast & Brunch ','Pancakes ','British ','Buffets ','Bu
           'Themed Cafes ','Cafeteria ','Cajun/Creole ','Cambodian ','Canadian ','Canteen ','Caribbean
           'Trinidadian ','Catalan ','Cheesesteaks ','Chicken Shop ','Chicken Wings ','Chilean ','Chines
           'Fuzhou ','Hainan ','Hakka ','Henghwa ','Hokkien ','Hunan ','Pekinese ','Shanghainese ','Szec
           'Corsican ','Creperies ','Cuban ','Curry Sausage ','Cypriot ','Czech ','Czech/Slovakian ','Da
           'Dinner Theater ','Dumplings ','Eastern European ','Eritrean ','Ethiopian ','Fast Food ','Fil
           'Flatbread ','Fondue ','Food Court ','Freiduria ','French ','Alsatian ','Auvergnat ','Berric
           'Mauritius ','Nicoise ','Provencal ','Reunion ','French Southwest ','Galician ','Game Meat '
           'Baden ','Eastern German ','Franconian ','Hessian ','Northern German ','Palatine ','Rhinelan
           'Guamanian ','Halal ','Hawaiian ','Heuriger ','Himalayan/Nepalese ','Honduran ','Hong Kong St
```

```
            'Hungarian ','Iberian ','Indian ','Indonesian ','International ','Irish ','Island Pub ','Isra
            'Altoatesine ','Apulian ','Calabrian ','Cucina campana ','Emilian ','Friulan ','Ligurian ','
            'Roman ','Sardinian ','Sicilian ','Tuscan ','Venetian ','Japanese ','Blowfish ','Conveyor Be
            'Oyakodon ','Hand Rolls ','Horumon ','Izakaya ','Japanese Curry ','Kaiseki ','Kushikatsu ','
            'Onigiri ','Ramen ','Robatayaki ','Soba ','Sukiyaki ','Takoyaki ','Tempura ','Teppanyaki ','
            'Western Style Japanese Food ','Yakiniku ','Yakitori ','Jewish ','Kebab ','Kopitiam ','Korea
            'Laotian ','Latin American ','Colombian ','Salvadoran ','Venezuelan ','Live/Raw Food ','Lyon
            'Meatballs ','Mediterranean ','Falafel ','Mexican ','Eastern Mexican ','Jaliscan ','Northern
            'Tamales ','Yucatan ','Middle Eastern ','Egyptian ','Lebanese ','Milk Bars ','Modern Australi
            'Moroccan ','New Mexican Cuisine ','New Zealand ','Nicaraguan ','Night Food ','Nikkei ','Noo
            'Oriental ','PF/Comercial ','Pakistani ','Pan Asian ','Parent Cafes ','Parma ','Persian/Iran
            'Polish ','Pierogis ','Polynesian ','Pop-Up Restaurants ','Portuguese ','Alentejo ','Algarve
            'Madeira ','Minho ','Ribatejo ','Tras-os-Montes ','Potatoes ','Poutineries ','Pub Food ','Ri
            'Russian ','Salad ','Sandwiches ','Scandinavian ','Schnitzel ','Scottish ','Seafood ','Serbo
            'Singaporean ','Slovakian ','Somali ','Soul Food ','Soup ','Southern ','Spanish ','Arroceria
            'Supper Clubs ','Sushi Bars ','Swabian ','Swedish ','Swiss Food ','Syrian ','Tabernas ','Tai
            'Tavola Calda ','Tex-Mex ','Thai ','Traditional Norwegian ','Traditional Swedish ','Trattori
            'Homemade Food ','Lahmacun ','Ottoman Cuisine ','Turkish Ravioli ','Ukrainian ','Uzbek ','Veg
            'Waffles ','Wok ','Yugoslav ','Acai Bowls ','Backshop ','Bagels ','Bakeries ','Beer','Bento
            'Beverage Store ','Breweries ','Brewpubs ','Bubble Tea ','Butcher ','CSA ','Chimney Cakes ',
            'Coffee & Tea Supplies ','Coffee Roasteries ','Cupcakes ','Custom Cakes ','Delicatessen ','D
            'Distilleries ','Do-It-Yourself Food ','Donairs ','Donuts ','Empanadas ',
            'Food Delivery Services ','Food Trucks ','Friterie ','Gelato ','Hawker Centre ','Honey ','Ic
            'Imported Food ','Internet Cafes ','Japanese Sweets ','Taiyaki ','Juice Bars & Smoothies ','
            'Kombucha ','Meaderies ','Milkshake Bars ','Mulled Wine ','Nasi Lemak ','Organic Stores ','Pa
            'Patisserie/Cake Shop ','Piadina ','Poke ','Pretzels ','Salumerie ','Shaved Ice ','Shaved Sn
            'Candy Stores ','Cheese Shops ','Chocolatiers & Shops ','Dagashi ','Dried Fruit ','Frozen Fo
            'Herbs & Spices ','Macarons ','Meat Shops ','Olive Oil ','Pasta Shops ','Popcorn Shops ','Se
            'Street Vendors ','Sugar Shacks ','Tea Rooms ','Torshi ','Tortillas ','Water Stores ','Winer

food_cat_list = [x.strip() for x in food_cat_list]
food_cat_list_plus = food_cat_list.copy()
food_cat_list_plus.append('Bars')

# Number of unique food categories
len(food_cat_list)


#getting rows that contain any of the food categories above for subsetting
count=0
indices = []
for i in np.arange(len(my_city)):
    cat = my_city['categories'].iloc[i]
    try:
        cats=cat.split(',')
        cats = [i.strip() for i in cats]
        if any(word in cats for word in food_cat_list_plus):
            count +=1
            indices.append(i)

    except:
        continue
```

```
FoodPlaces = my_city.copy(deep=True)
FoodPlaces = FoodPlaces.iloc[indices,:]
print(FoodPlaces.shape)


# Return top category that isn't restaurant/food. if nothing else, return the generic
def get_type(x):
    cats = x.split(',')
    cats = [i.strip() for i in cats]
    for cat in cats:
        if cat in food_cat_list:
            if(cat=='American (Traditional)' or cat=='American (New)'):
                return 'American'
            else:
                return cat
        else:
            continue
    if 'Bars' in cats:
        return 'Bars'


FoodPlaces['TopCategory'] = FoodPlaces['categories'].apply(get_type)

# How many unique top categories are there?
types = set(FoodPlaces['TopCategory'])
len(types)

#Plot top 25 categories
plt.figure(figsize=(50,20))
plt.tight_layout()
sns.set(font_scale=3)
sns.countplot(x='TopCategory',data=FoodPlaces,orient="v",order = FoodPlaces['TopCategory'].value_counts
plt.title('Top 25 Categories Food Categories in Las Vegas')
plt.xticks(rotation=90)
plt.show()

#Fixing JSON formatting in attribute column
def fixattrs(x):
    if isinstance(x,str):
        newx = x.replace('}\"','}').replace('\"{','{').replace('\'{','{').replace(" True","\"True\"").re
        return json.loads(newx)


FoodPlaces['attribute_dict'] = FoodPlaces['attributes'].apply(fixattrs)

#Getting list of attrs and getting counts
attrs = []
for row in FoodPlaces['attribute_dict']:
    if isinstance(row,dict):
        for i in list(row.keys()):
            attrs.append(i)

attr_counts = Counter(attrs)
```

```
attr_counts_dict = dict(attr_counts)
counts = list(attr_counts.values())
newList = [x / len(FoodPlaces) for x in counts]


#Plotting top attributes
attrcount_df = pd.DataFrame({'attribute':list(attr_counts_dict.keys()),'percent':newList})
attrcount_df.reset_index(inplace=True)
attrcount_df.sort_values(by='percent',inplace=True,ascending=False)
plt.figure(figsize=(50,20))
sns.barplot(x="attribute", y="percent", data=attrcount_df[0:18],palette="Blues_d")
plt.title('Top 18 Attributes for Food Categories in Las Vegas')
plt.xticks(rotation=90)
plt.show()

top_attrs = list(attrcount_df.iloc[0:18,1])


# Function cleaning the attributes, extracting only top ones
def clean_attrs(mdict):
    if isinstance(mdict,dict):
        dictcopy = mdict.copy()
        attrs = list(mdict.keys())
        for attr in attrs:
            if attr not in top_attrs:
                del dictcopy[attr]
        return dictcopy
    else:
        return



FoodPlaces['TopAttrs'] = FoodPlaces['attribute_dict'].apply(clean_attrs)



#check that it filtered some out
FoodPlaces[FoodPlaces['TopAttrs']!=FoodPlaces['attribute_dict']].head()



#one hot encoding sub attributes
attributes = FoodPlaces['TopAttrs'].apply(pd.Series)
parking = attributes['BusinessParking'].apply(pd.Series)
parking.drop(0,axis=1,inplace=True)

Ambience = attributes['Ambience'].apply(pd.Series)
Ambience.drop(0,axis=1,inplace=True)

Ambience = attributes['Ambience'].apply(pd.Series)
Ambience.drop(0,axis=1,inplace=True)

attributes.drop('Ambience',axis=1,inplace=True)
```

```
attributes.drop('BusinessParking',axis=1,inplace=True)
attributes.drop('GoodForMeal',axis=1,inplace=True)


HotAttrs = pd.concat([attributes,parking,Ambience,Meal],axis=1)


FinalFood = pd.concat([FoodPlaces,HotAttrs],axis=1)
FinalFood.drop(['Unnamed: 0','attributes','is_open','hours','TopAttrs','attribute_dict','categories'],a


# investigating nan None problem
list(FinalFood['OutdoorSeating'].unique())

# Fixing missing values
FinalFood['RestaurantsGoodForGroups'].replace('None', None, inplace=True)
FinalFood['BikeParking'].replace('None', None, inplace=True)
FinalFood['RestaurantsPriceRange2'].replace('None', None, inplace=True)
FinalFood['HasTV'].replace('None', None, inplace=True)
FinalFood['RestaurantsGoodForGroups'].replace('None', None, inplace=True)
FinalFood['OutdoorSeating'].replace('None', None, inplace=True)

FinalFood = FinalFood.replace('None', None)

#Verifying it worked
list(FinalFood['WiFi'].unique())

FinalFood.to_csv('vegas_food_businesses_clean.csv')
```

1.3.4 - Join Tables into Final Dataset

Python code to join all the tables together into one clean dataset

```
import numpy as np
import pandas as pd

clean_business = pd.read_csv('data/vegas_food_businesses_cleaned.csv')
clean_reviews = pd.read_csv('data/review_checkin_final.csv')
clean_users = pd.read_csv('data/vegas_business_user_info.csv')

clean_business.drop('Unnamed: 0', axis=1, inplace=True)

# Rename review dataset for clarity
clean_reviews = clean_reviews.rename(columns={'cool_stand':'cool_review_pm', 'funny_stand':'funny_revie
                                              'useful_stand':'useful_review_pm', 'useful':'useful_revie
                                              'funny':'funny_review_count', 'cool':'cool_review_count',
                                              'review':'review_count', 'checkin':'checkin_count'})

clean_users = clean_users.rename(columns={'stars_mean':'stars_review_mean', 'average_stars_mean':'stars_
                                          'cool_mean':'cool_users_mean', 'elite_binary_sum':'elite_user
                                          'fans_sum':'fans_users_sum', 'fans_mean':'fans_users_mean',
                                          'funny_mean':'funny_users_mean', 'review_count_sum':'review_u
                                          'review_count_mean':'review_users_mean', 'useful_sum':'useful_
```

```
                                        'useful_mean':'useful_users_mean', 'count_friends_sum':'friend
                                        'count_friends_mean':'friend_number_mean', 'reviews_per_month
                                        'fan_outlier_sum':'fan_outlier_count','friend_outlier_sum':'fr

clean_business = clean_business.rename(columns={'review_count':'review_count_business',
  'stars':'stars_business'})

vegas_yelp_dataset = pd.merge(clean_business, clean_reviews, how='left', left_on='business_id',
                              right_on='business_id')
vegas_yelp_dataset = pd.merge(vegas_yelp_dataset, clean_users, how='left', left_on='business_id',
                              right_on='business_id')
vegas_yelp_dataset.to_csv('data/vegas_yelp_dataset.csv')

print('Overall columns:')
print(vegas_yelp_dataset.columns)
```