"Eat, Rate, Love" – A Proposal for Modifying Yelp's Rating Systems
1/17/2016

1. Introduction

According to the Atlantic magazine, the very first review submitted to Yelp was four stars for Truly Mediterranean in 2004, consisting of the 4 words "dirt cheap, good falafels."[1] Today, those 4 words have grown to over 2.2 billion words that would fill 16,894 Zagat guides and take 42 years to read out loud. Over 142 million unique visitors use Yelp every month (79 million of them through their mobile phones), accessing over 90 million reviews of over 2.1 million businesses. Every minute, 26,380 reviews are posted. Every day, 24,000 new photos are uploaded.[2]

This large proliferation of information can lead to one of the few weaknesses of using Yelp to decide "where to go Friday night" – with so many restaurant reviews, it can be challenging to figure out where to go to if there are a lot of restaurants of the same type in the same area, all with comparable rankings. A quick check of Indian restaurants in the Schaumburg, Illinois area, for example, shows 14 Indian restaurants with a rating of 3 to 4 stars within a 3 mile radius. It would be useful if there were a way to help "cut through the clutter" and see if certain restaurants might stand out using different metrics. The goal of this analysis is to evaluate 2 different methods for doing this. These methods are examined for their viability, tried on existing Yelp data, and compared. Based on this comparison, a recommendation is made along with possible steps for taking the method further.

The first method examined is to create a new rating which gives more weight to those who have reviewed more restaurants of the same cuisine. Going back to the Schaumburg example, if someone has attended and reviewed all 12 restaurants, then their opinion should be given significantly more weight. Even someone who has been to 2 out of the 12 should be given more weight than someone who has just attended one.

The second method is to create an "immigrant" rating. Going back to the Indian restaurant example, one characteristic of Schaumburg is that it has a lot of immigrant Indian workers working there temporarily for various tech companies. On the theory that those workers would actively seek out restaurants that remind them closest of "home cooking" and also that they tend to seek out places offering the most value, one thing people might do is check the ratings given by those with clearly Indian names to see what they think. The proposal would be to check the user name in Yelp to guess at who might be an "immigrant," and create a different rating for a particular ethnic cuisine given specifically by those users. This method admittedly has some clear

[1] "Infographic: The Incredible Six-Year History of Yelp Reviews", Nicholas Jackson, The Atlantic, 7/20/11, http://www.theatlantic.com/technology/archive/2011/07/infographic-the-incredible-six-year-history-of-yelp-reviews/242072/

[2] Statistics all from "By the Numbers: 50 Amazing Yelp Statistics", http://expandedramblings.com/index.php/yelp-statistics/1/

deficiencies – it will ignore any “immigrants” who do not use their real names, and it will also mark as “immigrants” those who simply like an Indian name and choose to use it for their Yelp ID. The theory is that there might be enough information that cuts through the noise of those deficiencies to be able to provide useful information.

Data from the Yelp data challenge is used. This data consists of 1.6 million reviews from 366,000 users for 61,000 businesses in 10 cities: Edinburgh, Karlsruhe, Montreal, Waterloo, Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, and Madison.

2. The Yelp Dataset

The Yelp dataset consists of 5 files:

```
yelp_academic_dataset_business.json
yelp_academic_dataset_checkin.json
yelp_academic_dataset_review.json
yelp_academic_dataset_tip.json
yelp_academic_dataset_user.json
```

The main file is “review”. This consists of the text and star rating of each user review. The user and business are identified through a unique user_id and business_id. The date of the review as well as meta information on the # of votes the review got for being funny, useful, and cool, are also included here.

More details on the business can be found in the “business” file. Through the business_id, the full name of the business is given, along with its address, hours, coordinates, # of reviews, and average star rating (rounded to half a star). There is also a categories field for each business. It is this field that contains the cuisine (which can have multiple values) as well as other general categories such as “Kids Activities,” “Active Life”, and “Night Life”. Finally, there is also an attributes field which contains meta information such as whether Take-out is supported, whether it has WiFi, and whether it can be classified as “romantic”, “classy”, “hipster”, “divey”, “touristy” etc.

More details on each user can be found in the “user” file. Through user_id, the user name is given, along with how long they’ve been in Yelp, the total # of funny/useful/cool votes they’ve received, the total # reviews they’ve done, and a review_id for each review they have done.

The “tip” file contains the text for tips left by each user, linked by user_id and business_id.

The “checkin” file contains check-in time information for 45,000 of the businesses.

As can be seen by the file extensions, all of the data is in JSON format.

3. Initial Data Wrangling

The very first step after downloading and unzipping the dataset was to find a program to split the files so that the very large files (specifically the “review” file, which is 1.3GB unzipped) could be looked at with a text editor and examined. The program “hjsplit” was used in Windows for this.

After doing this and getting a sense of what was contained in the files, it was determined that the following information was needed:

-From the “review” file: business_id, user_id, stars
-From the “user” file: user_id, name
-From the “business” file: business_id, business_name, city, categories, review_count, stars

“categories” is needed to extract the cuisine. “review_count” and “stars” are useful as sanity checks on the data as it is being analyzed.

The “tips” and “check-in” files were not needed.

Given that a 1.3GB file is too large to read into R as-is (and would take a prohibitively long time even if it could), the first step was to preprocess the 3 files to extract only the fields needed. A python script was written to take care of this. Since python also has CSV support, conversion from JSON to CSV was done in this script as well.

In the python code there were only 2 slightly tricky things that were needed. The first was including “lineterminator = ‘\n’” in the outputWriter() command when writing CSV. This was needed to prevent outputWriter() from inserting a blank line after each CSV entry. The second is the use of “.encode” to take care of unicode characters that appear sometimes in the business name and city in the “business” file (used for French accents) and in the name in the “user” file. These special commands are needed to prevent an error from happening in the CSV “outputWriter.writerow” command.

Running the python script greatly reduced the file sizes (“review” went from 1.3GB to a much more manageable 75MB). They were then ready to read into R. Reading into R was done using the “read.csv” command:

```
reviews <- read.csv("yelp_academic_dataset_review.csv", header = FALSE)
users   <- read.csv("yelp_academic_dataset_user.csv", header = FALSE)
businesses <- read.csv("yelp_academic_dataset_business.csv", header = FALSE)
```

They were then combined into 1 data frame using the “inner_join” command so that business name, user name, city, cuisine, and rating are all in one data frame:

```
ru  <- inner_join(reviews, users)
rub <- inner_join(ru, businesses)
```

4. Analysis of Method 1: Giving More Weight to Multiple Reviewers of a Cuisine

The first step was to analyze the dataset to find the number of multiple reviewers there are for a cuisine – if there are very few, then adding weight to their opinions may ultimately have little impact on the overall rating. We also want to see how much the # of multiple reviews varies from cuisine to cuisine (our deep dive will later focus on Indian cuisine – we first want to make sure that usage patterns are similar enough across cuisines to be able to generalize).

Let us first look at Indian cuisine. We add an "is_indian" column to the table based on whether the word "Indian" appears in "categories", using grepl:

```
rub$is_indian <- grepl("Indian", rub$categories) == TRUE
```

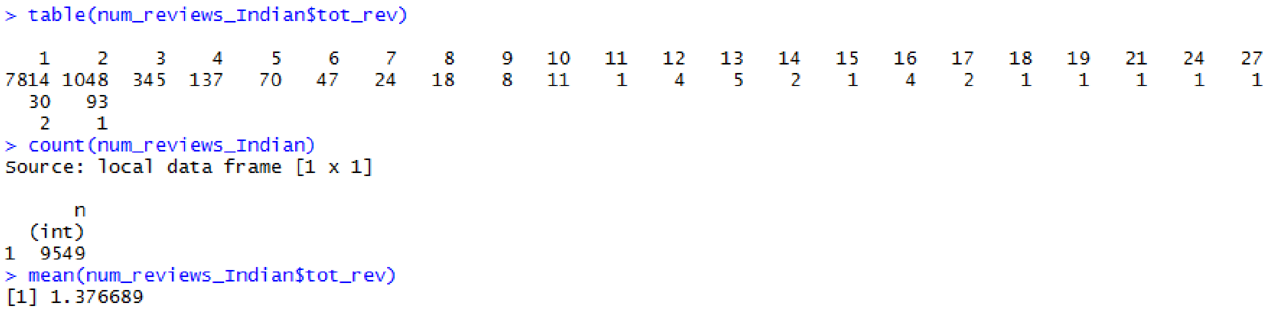
We then use subset to create a data frame of just reviews for Indian restaurants:

```
indian <- subset(rub, is_indian == TRUE)
```

Once we have this, we use the select, group_by, and summarise commands from dplyr to create a table of the # of reviews of Indian restaurants each user has done:

```
num_reviews_Indian <- indian %>% select(user_id, user_name, is_indian) %>%
     group_by(user_id) %>%
     summarise(tot_rev = sum(is_indian))
```

After doing this, we can use table, count, and mean to get review statistics:



```
> table(num_reviews_Indian$tot_rev)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   21   24   27 
7814 1048  345  137   70   47   24   18    8   11    1    4    5    2    1    4    2    1    1    1    1    1 
  30   93 
   2    1 
> count(num_reviews_Indian)
Source: local data frame [1 x 1]

      n
  (int)
1  9549
> mean(num_reviews_Indian$tot_rev)
[1] 1.376689
```

This yields a result of 9,549 total reviewers, with 7,814 doing just one review, 1,048 doing 2 reviews, and the rest doing 3 or more. The highest number of reviews is 93.

Roughly 10% of the users have done multiple reviews of Indian cuisine. This seems to imply there are enough users in general to justify trying a multiple review rating (and not so many that the added weights cancel each other out).

Using similar commands to the above, we can apply the same analysis to other cuisines. Let's use the top 10 "most-craved" ethnic cuisines in America, as ranked by Parade Magazine in an article on 5/15/2015[3]. Running similar commands on the Yelp dataset for these cuisines, we get the following result:

Cuisine	Total Reviewers	# > 1 Review	% > 1 Review	Avg Reviews per person	Max Reviews
(1) Chinese	33,359	7,733	23%	1.56	212
(2) Mexican	54,138	14,828	27%	1.77	145
(3) Italian	51,245	12,606	25%	1.63	90
(4) Japanese	44,849	11,688	26%	1.67	66
(5) Greek	10,820	1,799	17%	1.29	19
(6) French	19,127	3,413	18%	1.32	40
(7) Thai	20,699	4,032	20%	1.40	74
(8) Spanish	12,736	1,804	14%	1.25	27
(9) Indian	9,549	1,735	18%	1.38	93
(10) Mediterranean	19,400	3,534	18%	1.35	30

One can see that all cuisines had at least 10% of reviewers give multiple reviews. The average # of reviews also seems to go down as the # of total reviewers decreases. This makes sense – if there are fewer restaurants of a cuisine in a city (which we'll assume correlates with the # of reviewers of that cuisine in a city), then there will be less chance of multiple reviews happening.

An interesting side note is that the total # of reviewers doesn't seem to exactly match up with the Parade rankings. For example, even though Chinese food is rated #1 by Parade, there are more Yelp reviewers for Mexican, Italian, and Japanese food. Parade's definition of what Americans "crave" most appears to be slightly different from what the Yelp data indicates.

Given that all 10 cuisines have at least 10% multiple reviews (and in some cases as high as 27%), we conclude that it is worth proceeding further.

We then come to the real test -- modifying the rating using these weights and seeing what impact they have. Let's try this on Indian restaurants. We have the # of reviews for each user in num_reviews_Indian. If we join this back to our "indian" data frame containing all the individual ratings, we have a new table which has the rating the user gave as well as the # of Indian

[3] "Top 10 Ethnic Cuisines Americans Crave Most", Parade Magazine, 5/15/2015, http://parade.com/397203/parade/top-10-ethnic-cuisines-americans-crave-most/

restaurants they have reviewed. Let’s store this in a variable called cuisine-indian (or “cin” for short):

```
cin <- inner_join(indian, num_reviews_Indian)
```

Let’s create a new weighted rank which is simply the # of stars times the # of reviews:

```
cin$weighted_stars <- cin$stars * cin$tot_rev
```

Now, we can once again use the group_by and summarise commands in dplyr to generate both the original average star rating (as a sanity check) and the new, weighted star rating:

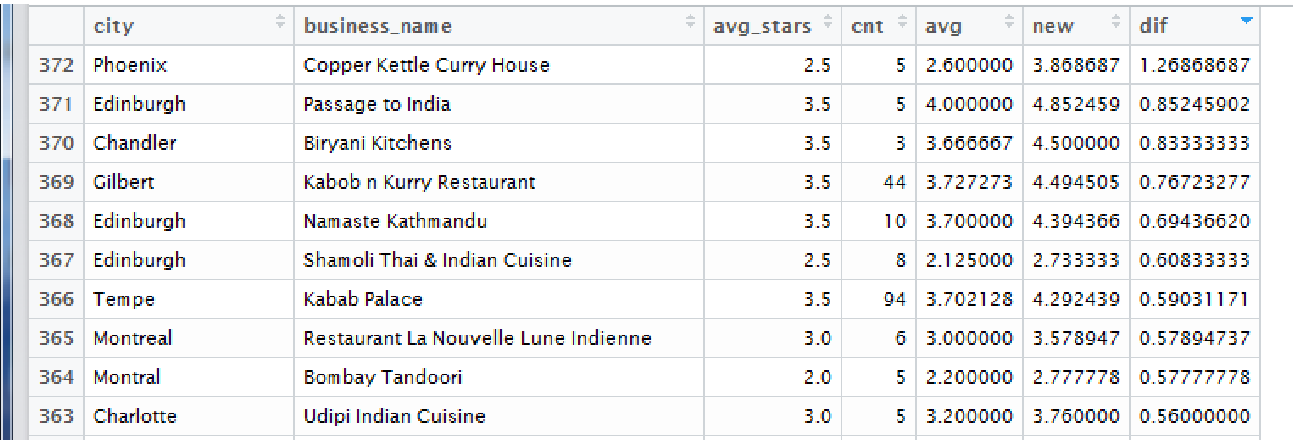
```
new_rating_Indian <- cin %>% select(city, business_name, avg_stars, stars,
                                    tot_rev, weighted_stars) %>%
                             group_by(city, business_name, avg_stars) %>%
                             summarise(cnt = n(),
                                       avg = sum(stars) / cnt,
                                       new = sum(weighted_stars)/sum(tot_rev),
                                       dif = new - avg)
```

After doing this, we can use summary to get a sense of the effect of the new rating:

```
> summary(new_rating_Indian$dif)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.38900 -0.23490 -0.06903 -0.07623  0.05728  1.26900
```

We see that the new weights can move the rating down by as many as 1.4 stars or up as high as 1.3 stars, with usually there being relatively little effect (mean = -0.08).

If we use view(new_rating_Indian) and sort based on the difference to see the restaurants that benefited the most, we see that quite a few of them have relatively few reviews:



	city	business_name	avg_stars	cnt	avg	new	dif
372	Phoenix	Copper Kettle Curry House	2.5	5	2.600000	3.868687	1.26868687
371	Edinburgh	Passage to India	3.5	5	4.000000	4.852459	0.85245902
370	Chandler	Biryani Kitchens	3.5	3	3.666667	4.500000	0.83333333
369	Gilbert	Kabob n Kurry Restaurant	3.5	44	3.727273	4.494505	0.76723277
368	Edinburgh	Namaste Kathmandu	3.5	10	3.700000	4.394366	0.69436620
367	Edinburgh	Shamoli Thai & Indian Cuisine	2.5	8	2.125000	2.733333	0.60833333
366	Tempe	Kabab Palace	3.5	94	3.702128	4.292439	0.59031171
365	Montreal	Restaurant La Nouvelle Lune Indienne	3.0	6	3.000000	3.578947	0.57894737
364	Montral	Bombay Tandoori	2.0	5	2.200000	2.777778	0.57777778
363	Charlotte	Udipi Indian Cuisine	3.0	5	3.200000	3.760000	0.56000000

The top 3 beneficiaries, for example, all had 5 or fewer reviews. Our proposed tweak would have a disproportionately large effect in those cases. To remove those cases, let’s only look at restaurants with more than 5 reviews:

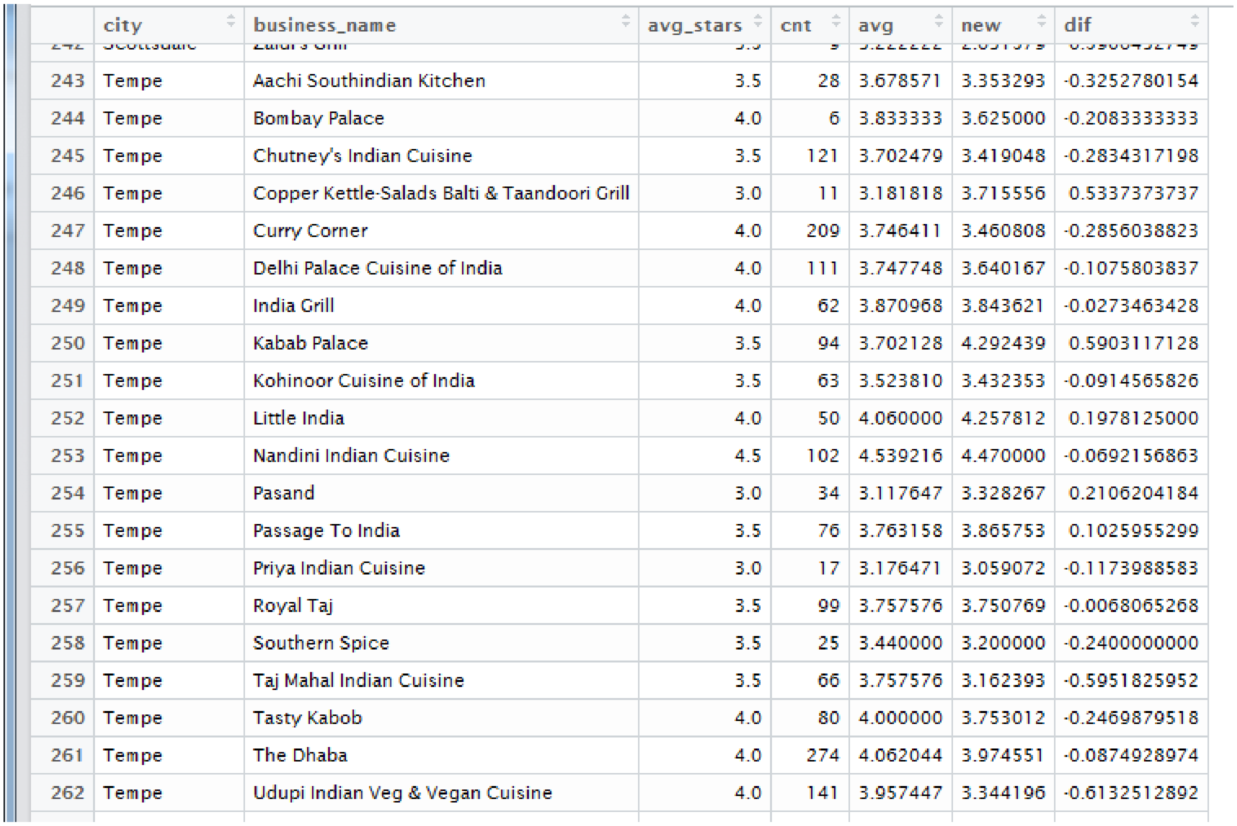
```
nri5 <- subset(new_rating_Indian, cnt > 5)
```

Here if we repeat summary, we get a new smaller range:

```
> summary(nri5$dif)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
-1.14000 -0.24060 -0.09318 -0.09156  0.05591  0.76720 
```

We can see that the impact is an increase of up to 0.77 stars and a decrease of as much as 1.14 stars. The number of restaurants goes down from 372 to 270.

The most important question, though, is would this difference potentially help a user get better results from Yelp? Let’s look at the sample case of Indian restaurants in Tempe, Arizona. The results for Indian restaurants with more than 5 reviews are as follows:



	city	business_name	avg_stars	cnt	avg	new	dif
243	Tempe	Aachi Southindian Kitchen	3.5	28	3.678571	3.353293	-0.3252780154
244	Tempe	Bombay Palace	4.0	6	3.833333	3.625000	-0.2083333333
245	Tempe	Chutney's Indian Cuisine	3.5	121	3.702479	3.419048	-0.2834317198
246	Tempe	Copper Kettle-Salads Balti & Taandoori Grill	3.0	11	3.181818	3.715556	0.5337373737
247	Tempe	Curry Corner	4.0	209	3.746411	3.460808	-0.2856038823
248	Tempe	Delhi Palace Cuisine of India	4.0	111	3.747748	3.640167	-0.1075803837
249	Tempe	India Grill	4.0	62	3.870968	3.843621	-0.0273463428
250	Tempe	Kabab Palace	3.5	94	3.702128	4.292439	0.5903117128
251	Tempe	Kohinoor Cuisine of India	3.5	63	3.523810	3.432353	-0.0914565826
252	Tempe	Little India	4.0	50	4.060000	4.257812	0.1978125000
253	Tempe	Nandini Indian Cuisine	4.5	102	4.539216	4.470000	-0.0692156863
254	Tempe	Pasand	3.0	34	3.117647	3.328267	0.2106204184
255	Tempe	Passage To India	3.5	76	3.763158	3.865753	0.1025955299
256	Tempe	Priya Indian Cuisine	3.0	17	3.176471	3.059072	-0.1173988583
257	Tempe	Royal Taj	3.5	99	3.757576	3.750769	-0.0068065268
258	Tempe	Southern Spice	3.5	25	3.440000	3.200000	-0.2400000000
259	Tempe	Taj Mahal Indian Cuisine	3.5	66	3.757576	3.162393	-0.5951825952
260	Tempe	Tasty Kabob	4.0	80	4.000000	3.753012	-0.2469879518
261	Tempe	The Dhaba	4.0	274	4.062044	3.974551	-0.0874928974
262	Tempe	Udupi Indian Veg & Vegan Cuisine	4.0	141	3.957447	3.344196	-0.6132512892

From the ratings, one can see 8 restaurants with an official rating of 4 (Bombay Palace, Curry Corner, Delhi Palace, India Grill, Little India, Tasty Kabob, The Dhaba, and Udupi Indian) and 1 with a 4.5 (Nandini). Looking at the new weighted rating we get:

	Official	Old	New
Bombay Palace	4	3.83	3.63
Curry Corner	4	3.75	3.46
Delhi Palace	4	3.75	3.64
India Grill	4	3.87	3.84
Little India	4	4.06	4.26
Tasty Kabob	4	4.00	3.75
The Dhaba	4	4.06	3.97
Udupi Indian	4	3.96	3.34
Nandini	4.5	4.54	4.47

The first thing we can see is that Nandini maintains its rating of 4.5 with the new rating, so that would be the clear first choice. Among the 8 restaurants with 4 stars, we see that 4 restaurants drop to a rounded value of 3.5 (Bombay Palace, Curry Corner, Delhi Palace, and Udupi Indian), 3 maintain a rating of 4 (India Grill, Tasty Kabob, and The Dhaba), and one increases to 4.5 when rounded (Little India). Therefore, if the weighted system is used, one can see that 1 recommendation comes through from the group of 8 (Little India).

In this section, we saw that for the most popular cuisines there appear to be enough people who have reviewed other restaurants of the same cuisine to make this method meaningful. We also saw that if we applied this weight on the Yelp dataset (and imposed a limit of at least 5 reviews), the impact ranged from -1.14 to +0.77 stars. Finally, we saw when we looked at the specific case of Tempe, Arizona, that this rating can help a user pick from many different choices.

5. Analysis of Method 2: Adding an "Immigrant Rating"

The second proposed method is to add an "Immigrant Rating." This section tries this on the Yelp database for Indian cuisine and analyzes the results.

Since the Yelp database does not include ethnicity in its user database, the only way to try to guess ethnicity is through the user name. The first step was to generate a list of names that would qualify as being uniquely Indian. To do this, the user names in the Yelp "user" dataset were examined for potential candidates. When a name looked like a potential candidate, the question "Is ___ an Indian name" was googled to see if sites such as www.indiaparenting.com, www.modernindianbabynames.com, and www.indiachildnames.com would show up in the top of the search result. Anything that did so was included in the list. Any popular names in America were left out ("Daniel", for example, appears in the indiaparenting.com database but is left out). This resulted in 608 names, ranging from "Aayush" and "Abhijeet" to "Yogesh" and "Yuvaraj". The list of names used is included in Appendix A of this document.

This list of names was read into R using the scan command:

```
inames <- scan("indian_names.txt", what = character())
```

Taking the "indian" dataset from Method 1 that contains the business name, reviewer name, and star rating for all Indian restaurants in the dataset, we add a "reviewer_indian_name" field by using %in%:

```
indian$reviewer_indian_name <- indian$user_name %in% inames
```

Also, to simplify calculation later, we add an "istars" field which is simply the number of stars the reviewer gave the restaurant if that person has a uniquely Indian name, and a 0 otherwise:

```
indian$istars <- indian$stars * indian$reviewer_indian_name
```

After doing this, we can find out how many reviews fall under the "immigrant" category by using the "table command":

```
> table(indian$reviewer_indian_name)

FALSE  TRUE
11872  1274
```

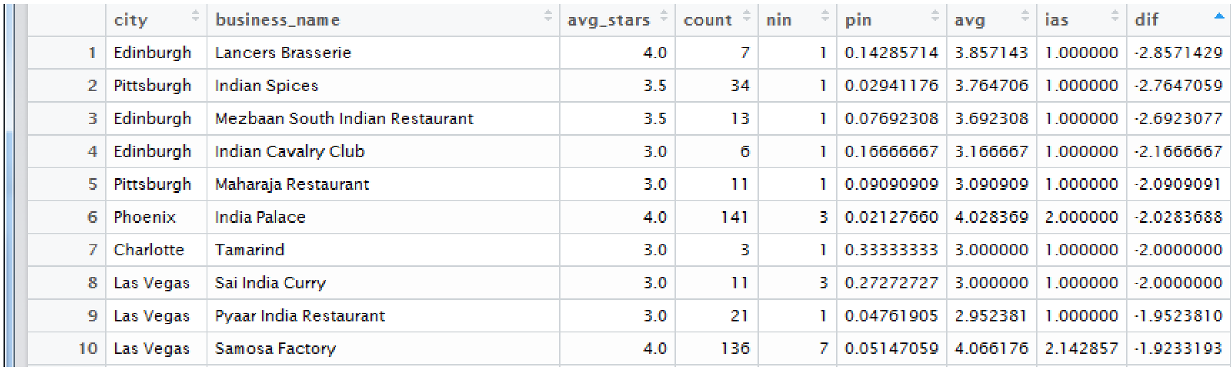
We can see that, out of the 13,146 reviews, 1,274, or a little under 10%, of the reviews are eligible for the "Immigrant Rating." This seems like a substantial enough amount to go forward.

Using "group_by" and "summarise" from dplyr, we can regenerate the average rating for a restaurant (as a sanity check) and also the "Immigrant Rating," which would simply be the sum of the newly generated “istars” for each restaurant divided by the # of “immigrants” who reviewed that restaurant (found by “sum(reviewer_indian_name)” for each group):

```
avg_rating_Indian <- indian %>% select(business_id, business_name, city, stars,
                                       avg_stars, reviewer_indian_name,
                                       is_indian, istars) %>%
                                group_by(city, business_name, avg_stars) %>%
                                summarise(count = n(),
                                          nin = sum(reviewer_indian_name),
                                          pin = sum(reviewer_indian_name) / n(),
                                          avg = sum(stars) / count,
                                          ias = sum(istars) / nin,
                                          dif = ias - avg)
```

Here, “nin” stands for “number of Indian names”, “pin” stands for “percentage with Indian names”, “ias” stands for “Indian (rating) average stars [the new rating]”, and “dif” stands for the difference between the new and the original rating. “avg_stars” from the Yelp database is the official average stars rating recorded for that business.

After doing this and sorting the results by the difference between the new rating and the original rating, we see that there are dramatic differences, but in many of those cases the difference is due to there being only 1 “immigrant”:

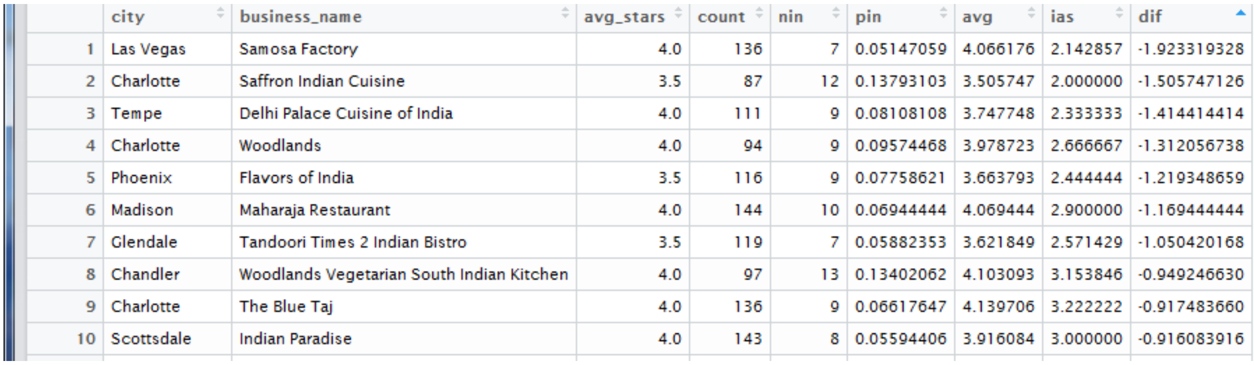


	city	business_name	avg_stars	count	nin	pin	avg	ias	dif
1	Edinburgh	Lancers Brasserie	4.0	7	1	0.14285714	3.857143	1.000000	-2.8571429
2	Pittsburgh	Indian Spices	3.5	34	1	0.02941176	3.764706	1.000000	-2.7647059
3	Edinburgh	Mezbaan South Indian Restaurant	3.5	13	1	0.07692308	3.692308	1.000000	-2.6923077
4	Edinburgh	Indian Cavalry Club	3.0	6	1	0.16666667	3.166667	1.000000	-2.1666667
5	Pittsburgh	Maharaja Restaurant	3.0	11	1	0.09090909	3.090909	1.000000	-2.0909091
6	Phoenix	India Palace	4.0	141	3	0.02127660	4.028369	2.000000	-2.0283688
7	Charlotte	Tamarind	3.0	3	1	0.33333333	3.000000	1.000000	-2.0000000
8	Las Vegas	Sai India Curry	3.0	11	3	0.27272727	3.000000	1.000000	-2.0000000
9	Las Vegas	Pyaar India Restaurant	3.0	21	1	0.04761905	2.952381	1.000000	-1.9523810
10	Las Vegas	Samosa Factory	4.0	136	7	0.05147059	4.066176	2.142857	-1.9233193

Let’s set an arbitrary value of needing at least 5 “immigrants” to be able to generate a useful “immigrant rating”. We use “subset” to screen for this:

```
ari5 <- subset (avg_rating_Indian, nin > 5)
```

When we do this and look at the values with the greatest difference, the max difference decreases from -2.85 to -1.9:



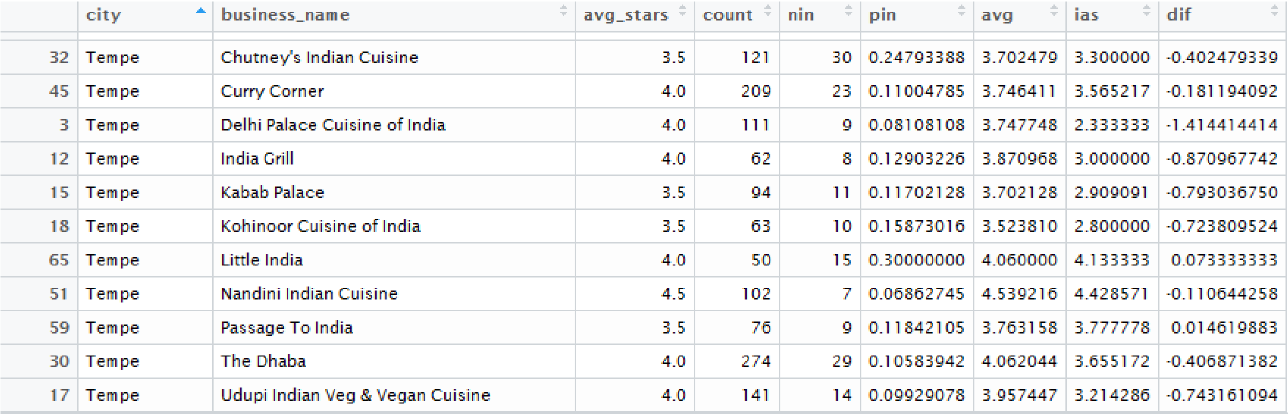
	city	business_name	avg_stars	count	nin	pin	avg	ias	dif
1	Las Vegas	Samosa Factory	4.0	136	7	0.05147059	4.066176	2.142857	-1.923319328
2	Charlotte	Saffron Indian Cuisine	3.5	87	12	0.13793103	3.505747	2.000000	-1.505747126
3	Tempe	Delhi Palace Cuisine of India	4.0	111	9	0.08108108	3.747748	2.333333	-1.414414414
4	Charlotte	Woodlands	4.0	94	9	0.09574468	3.978723	2.666667	-1.312056738
5	Phoenix	Flavors of India	3.5	116	9	0.07758621	3.663793	2.444444	-1.219348659
6	Madison	Maharaja Restaurant	4.0	144	10	0.06944444	4.069444	2.900000	-1.169444444
7	Glendale	Tandoori Times 2 Indian Bistro	3.5	119	7	0.05882353	3.621849	2.571429	-1.050420168
8	Chandler	Woodlands Vegetarian South Indian Kitchen	4.0	97	13	0.13402062	4.103093	3.153846	-0.949246630
9	Charlotte	The Blue Taj	4.0	136	9	0.06617647	4.139706	3.222222	-0.917483660
10	Scottsdale	Indian Paradise	4.0	143	8	0.05594406	3.916084	3.000000	-0.916083916

Using summary we see that the potential difference ranges from -1.9 to an increase of .31, with the mean value being -0.4 (this is interesting – perhaps reviewers from India or of Indian heritage tend to be harder on Indian restaurants than reviewers from America …)

```
> summary(ari5$dif)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
-1.92300 -0.72230 -0.34500 -0.43600 -0.09828  0.30980 
```

The other item to note is that, when imposing the requirement that there be at least 5 immigrant ratings, the number of restaurants with an “Immigrant Rating” decreases from 392 to 70.

Once again, as was the case when looking at Method 1, the really important question is whether this rating has a potential to improve the user experience. Let’s again look at the test case of Indian restaurants in Tempe, Arizona. Sorting by city and looking at Tempe, we get:



	city	business_name	avg_stars	count	nin	pin	avg	ias	dif
32	Tempe	Chutney's Indian Cuisine	3.5	121	30	0.24793388	3.702479	3.300000	-0.402479339
45	Tempe	Curry Corner	4.0	209	23	0.11004785	3.746411	3.565217	-0.181194092
3	Tempe	Delhi Palace Cuisine of India	4.0	111	9	0.08108108	3.747748	2.333333	-1.414414414
12	Tempe	India Grill	4.0	62	8	0.12903226	3.870968	3.000000	-0.870967742
15	Tempe	Kabab Palace	3.5	94	11	0.11702128	3.702128	2.909091	-0.793036750
18	Tempe	Kohinoor Cuisine of India	3.5	63	10	0.15873016	3.523810	2.800000	-0.723809524
65	Tempe	Little India	4.0	50	15	0.30000000	4.060000	4.133333	0.073333333
51	Tempe	Nandini Indian Cuisine	4.5	102	7	0.06862745	4.539216	4.428571	-0.110644258
59	Tempe	Passage To India	3.5	76	9	0.11842105	3.763158	3.777778	0.014619883
30	Tempe	The Dhaba	4.0	274	29	0.10583942	4.062044	3.655172	-0.406871382
17	Tempe	Udupi Indian Veg & Vegan Cuisine	4.0	141	14	0.09929078	3.957447	3.214286	-0.743161094

There are 6 restaurants with an official rating of 4.0 and 1 with a rating of 4.5. Let’s see what the new rating gives:

	Official	Old	New
Curry Corner	4	3.75	3.57
Delhi Palace	4	3.75	2.33
India Grill	4	3.87	3.00
Little India	4	4.06	4.13
The Dhaba	4	4.06	3.66
Udupi Indian	4	3.96	3.21
Nandini	4.5	4.54	4.43

With the “Immigrant Rating” it becomes easier to differentiate between the 7 choices. The first thing we see is that “Nandini” maintains its high 4.5 rating. That would be the clear first choice. Let’s say that you’ve already tried Nandini and want to choose between the six 4 star choices. Here with the new rating, we see dramatic drops for “Delhi Palace”, “India Grill”, and “Udupi Indian.” Of the 3 remaining choices, we see that “Little India” is the only choice that actually increased in value compared to the original average (and would be the only one to round to 4 stars). In this case the new rating would result in a clear recommendation of Little India.

It is interesting to note that the recommended choice of Little India is the same one recommended by using Method 1. In the next section we will analyze whether this also occurs in other cities.

It is also interesting to note that “Bombay Palace” and “Tasty Kabob”, which showed up among the Method 1 finalists, did not have enough “immigrant” reviews to qualify them for a rating with Method 2.

Using Tempe as an example, we see that Method 2 could be a useful data point in selecting between different places that would otherwise have the same rating.

6. Comparing the 2 Methods

In the previous sections we established that both methods seem to have enough data to provide useful information and that both can be helpful in at least one city (Tempe). How about for other cities in the Yelp database? In this section we look at this question and see if we can draw a conclusion about which method might be most useful for the problem we are trying to solve of “cutting through the clutter.” Let’s look first at Method 1. A summary of the results using Method 1 is shown below (details on how these results were derived can be found in Appendix B):

City	**# Restaurants w/ 4 stars**	**Result**	**Recommended Restaurant(s)**
Champaign	1	No need for additional rating	-----
Chandler	3	3 reduced to 2	Woodlands / Indus
Charlotte	8	8 reduced to 1	Aroma
Edinburgh	12 (*)	12 reduced to 1	Noor
Karlsruhe	1	No need for additional rating	-----
Las Vegas	9	9 reduced to 2	Mint / Saffron
Madison	9	9 reduced to 1	Maharaja
Mesa	2	2 reduced to 1	Guru Palace
Montreal	10 (*)	10 reduced to 1	Restaurant Tibetan
Phoenix	7	7 reduced to 4	Star/ Khyber / Garden / Saffron
Pittsburgh	7	7 reduced to 2	Tamarind / India on Wheels
Scottsdale	3	3 reduced to 2	Indian Paradise / Jewel of Crown
Tempe	6	7 reduced to 1	Little India
Waterloo	1	No need for additional rating	-----

Outcome: 1 qualifier (3)	Outcome: Partial Reduce (5)	Outcome: Clear choice found (6)
Champaign Karlsruhe Waterloo	Chandler (3 to 2) Las Vegas (9 to 2) Phoenix (7 to 4) Pittsburgh (7 to 2) Scottsdale (3 to 2)	Charlotte (8 to 1) Edinburgh (12 to 1) Madison (9 to 1) Mesa (2 to 1) Montreal (10 to 1) Tempe (7 to 1)

* = Use 4.5 star restaurants instead of 4 in this city because there are many 4.5 star restaurants.

In most cases, restaurants with 4 stars were looked at because there are more choices in 4 stars than 4.5 stars. In the 11 cities where there was a choice, Method 1 was able to help reduce the number of choices in every city. In 6 out of those 11 (55%), it was able to identify one clear choice.

Now let's look at the results for Method 2 (the "Immigrant Ratings"). The table below summarizes the results (once again details are available in Appendix B):

<table>
<tr><th>City</th><th># Restaurants w/ 4 stars</th><th>Result</th><th>Recommended Restaurant</th></tr>
<tr><td>Champaign</td><td>1</td><td>No need for additional rating</td><td>-----</td></tr>
<tr><td>Chandler</td><td>2</td><td>2 reduced to 1</td><td>NASHA</td></tr>
<tr><td>Charlotte</td><td>6</td><td>6 reduced to 1</td><td>Aroma</td></tr>
<tr><td>Las Vegas</td><td>6</td><td>6 reduced to 1</td><td>Taj Palace</td></tr>
<tr><td>Madison</td><td>2</td><td>2 reduced to 1</td><td>Maharani</td></tr>
<tr><td>Mesa</td><td>1</td><td>No need for additional rating</td><td>-----</td></tr>
<tr><td>Montreal</td><td>1</td><td>No need for additional rating</td><td>-----</td></tr>
<tr><td>Phoenix</td><td>2</td><td>2 reduced to 1</td><td>Star of India</td></tr>
<tr><td>Pittsburgh</td><td>3</td><td>3 reduced to 1</td><td>Tamarind Flavor</td></tr>
<tr><td>Scottsdale</td><td>2</td><td>2 reduced to 1</td><td>Jewel of the Crown</td></tr>
<tr><td>Tempe</td><td>6</td><td>6 reduced to 1</td><td>Little India</td></tr>
<tr><td colspan="2">Outcome: No qualifiers (3)
Edinburgh
Karlsruhe
Waterloo</td><td>Outcome: 1 qualifier (3)
Champaign
Mesa
Montreal</td><td>Outcome: Clear choice found (8)
Chandler (2 -> 1)
Charlotte (6 -> 1)
Las Vegas (6 -> 1)
Madison (2 -> 1)
Phoenix (2 -> 1)
Pittsburgh (3 -> 1)
Scottsdale (2 -> 1)
Tempe (6-> 1)</td></tr>
</table>

Here we see an interesting result – the "Immigrant Rating" seems to converge on one recommendation much more often (in the 8 cities in the Yelp database where there was a choice, it identified a clear choice in all 8 cities!). It would be interesting to see in how many other cities this applies.

This convergence comes with a tradeoff of less inclusiveness. We see that we "lose" 3 cities with Method 2 – Edinburgh, Karlsruhe, and Waterloo all did not have any Indian restaurants with 4 stars that had at least 5 "immigrant" reviews.

Are the final recommendations the same for the 2 methods as they were for Tempe? Looking at the final results, we see that the same recommendation was made in 2 out of the 8 cities (Charlotte and Tempe). In 3 out of the 8 (Phoenix, Pittsburgh, and Scottsdale), Method 1 includes the restaurant recommended by Method 2 plus additional ones. In the remaining 3 cities

(Chandler, Las Vegas, Madison) the recommendations are completely different – so we can conclude that the methods yield different results, though at times there is some overlap.

One other item to note is there are a few restaurants recommended by Method 1 that, based on the name, might not be Indian in focus ("Restaurant Tibetan" in Montreal and "Khyber Halal" in Phoenix). These choices are not found in Method 2.

Which method seems better for "cutting through the clutter"? Based on the results, it appears Method 2 would be the better choice because it yields a clear choice more often (100% of the time with the Yelp Academic Dataset vs 55% for Method 1). Based on this we recommend **Method 2 – the "Immigrant Rating**", be used. This method will be used for the rest of this analysis.

For the next step we need to look at a few implementation questions: How would this rating actually be used in the product? What are the actual benefits? These are covered in the next 2 sections.

7. Recommended Use of the New Rating

Now that we've made a choice of new rating system, the next question is how would this be used in the Yelp app itself? This question can be broken down into several subquestions.

The first is would the "Immigrant Rating" replace the current rating system, or be a secondary rating? We recommend that it be considered as a secondary ranking, for 2 reasons:

(a) The current average ranking has already been in use for many years and users have gotten used to it. Any change to this main system should only be done incrementally, if at all.

(b) The data shows that the # of restaurants with Immigrant Ratings was 70 (out of 392). Because of the low coverage, this would leave many restaurants without a rating if used as a primary rating.

The second question is whether to use a raw rating score, or some kind of badge if certain criteria are met. For example, in the data we looked at, we could implement an "authenticity" badge that a restaurant gets if it has an "Immigrant Rating" of 4 stars or more. This would have 2 advantages:

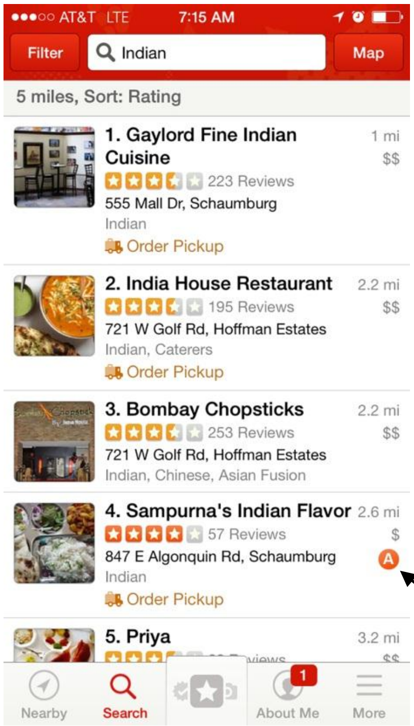
(a) It is less confusing – rather than having a number that users might not know how to interpret and might confuse with the main rating, they just see a simple icon they could click on to get details about.

(b) It is more likely to be accepted by the restaurants. The data shows that the “Immigrant Rating” showed dramatically lower scores for a lot of places. Rather than publish these scores and invite a lot of blowback from businesses that got dinged with a lower rating, it may be better to keep the award positive – for those that achieve a certain level, award them with a badge showing that they got a higher score.

The main disadvantage of using a “badge” is information would be lost. For example, if we used a score of 4 or more as criteria for this badge, the data shows that there would be 3 cities out of the 8 that would have no restaurants with a badge (Madison, Phoenix, and Scottsdale [this can be seen in the data in Appendix B]).

The effect on the UI alone is a strong argument for using a badge instead of a raw number. However, the loss of information is a key consideration (since we decided on this approach because it was best at “cutting through the clutter”). To get around this, we recommend a badge with a modified selection criteria – it can be earned with an “Immigrant Rating” of 4 stars or more or, if there are no restaurants in the same area with an “Immigrant Rating” of 4, a 3.5 is sufficient for a badge. Looking at Madison, Phoenix, and Scottsdale, this would cause the information that was lost with a badge to be retained.

Using this approach, information can be added in a relatively unobtrusive way and the selective power of the “Immigrant Rating” can be leveraged fully. The figures below show an example of what the “Authenticity Badge” could potentially look like in the Yelp app:



●●●○○ AT&T LTE 7:15 AM

Filter | Indian | Map

5 miles, Sort: Rating

1. Gaylord Fine Indian Cuisine 1 mi $$
223 Reviews
555 Mall Dr, Schaumburg
Indian
Order Pickup

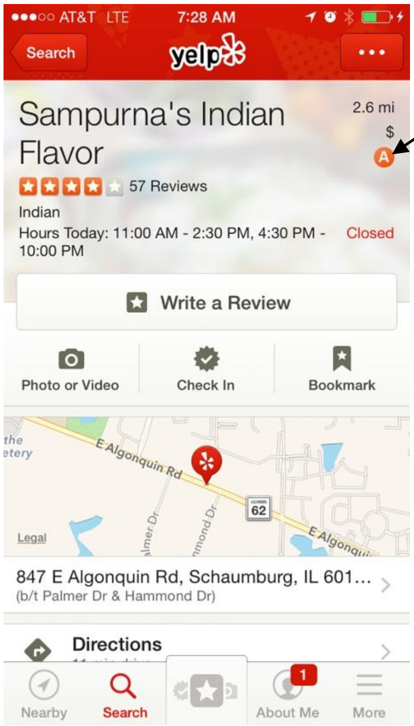
2. India House Restaurant 2.2 mi $$
195 Reviews
721 W Golf Rd, Hoffman Estates
Indian, Caterers
Order Pickup

3. Bombay Chopsticks 2.2 mi $$
253 Reviews
721 W Golf Rd, Hoffman Estates
Indian, Chinese, Asian Fusion

4. Sampurna's Indian Flavor 2.6 mi $
57 Reviews
847 E Algonquin Rd, Schaumburg A
Indian
Order Pickup

5. Priya 3.2 mi

Authenticity Badge



●●●○○ AT&T LTE 7:28 AM

Search | yelp | •••

Sampurna's Indian Flavor 2.6 mi $ A
57 Reviews
Indian
Hours Today: 11:00 AM - 2:30 PM, 4:30 PM - 10:00 PM Closed

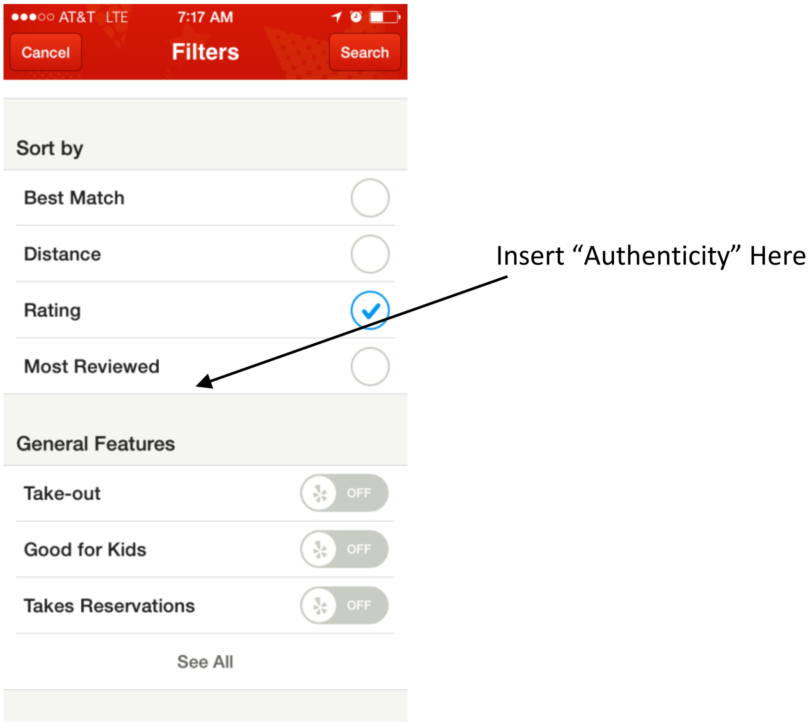
Write a Review

847 E Algonquin Rd, Schaumburg, IL 601…
(b/t Palmer Dr & Hammond Dr)

Directions

Authenticity Badge

Authenticity could also be used as a selection criteria after “Most Reviewed” for filtering:



Sort by

Best Match ○

Distance ○

Rating ✓

Most Reviewed ○

General Features

Take-out OFF

Good for Kids OFF

Takes Reservations OFF

See All

Insert “Authenticity” Here

This would allow any restaurants with the Authenticity Badge, if any, to be listed first. Raw ratings could also potentially be used here since the raw scores would not appear but could be inferred from the sort order.

8. Advantages of the New Rating

Since implementing this would require time and resources, it is important to consider the question of what advantages this new rating would bring. Besides the general advantage of providing a better user experience, we can think of the following specific advantages:

(1) Potential engine for increasing ad revenue

-The main driver for Yelp’s revenues is local ads. Checking for who uses the “authenticity” filter repeatedly or who consistently goes to “authentic” restaurants could be an avenue for charging premium advertising dollars to those who want to target Indian nationals living in the US or those with a strong interest in Indian culture (or perhaps foodies in general). This could be applied to other ethnic groups (rice cooker companies targeting the Chinese and Japanese market, for example …).

(2) Way to help Yelp Daily Deals program

-Seeing who uses the Authenticity filter or repeatedly selects “authentic” restaurants can be used as a way to segment the user population and provide another target market for Daily Deals customers. A travel company advertising special airfare rates to Japan, for example, may want to do a Daily Deal targeted to those interested in “Authentic” Japanese cuisine.

(3) Way to improve Yelp’s recommendations

-This same segmentation can be used to increase the chances of a successful restaurant recommendation. Someone’s established interest in “authentic” food could be used to help determine the content of the e-mail blasts Yelp sends to users to encourage them to continue using Yelp / patronize Yelp-friendly businesses.

(4) Way to increase number of reviews / user engagement

-If foreign born nationals learn their reviews have more weight they might be encouraged to do more reviews. This will result in more information for Yelp and higher usage rates. It could also potentially give a sense of empowerment that they have useful information to offer and increase the level of engagement with Yelp in general with these audiences.

9. Potential Next Steps

There are many different directions further work on this could go, following 3 main paths:

(1) Verify the “authenticity” of the “authenticity rating”:

(a) Hire human panels to sanity check the recommendations made in the 8 cities.

(b) Solicit input from newspaper food critics in the respective cities on its accuracy.

(c) Have a contest – solicit Yelper’s opinions on the most authentic restaurant of various cuisines in their city where all entries are eligible for a prize. Compare results with “Immigrant Rating” results.

(2) Improve the rating with more inputs / tweaks to the algorithm:

(a) Consider adding other inputs to “authenticity” to increase restaurant coverage

(i) Analyze text reviews to see if “authenticity” information can be extracted.

(ii) Explore the idea of using percentage of “immigrant” reviewers as a way to determine authenticity.

(b) Consider using the Weighted Rating from Method 1 as an input to “Authenticity”

(c) Consider incorporating information such as whether a review is “useful”, whether those reviewers have also reviewed ethnic grocery stores, or whether a restaurant is marked as “touristy” to “authenticity.”

(3) Extend the rating further / Try a rollout

(a) Find a way to automate generation of the name lists. For example, generate it by scraping an Indian baby names site and then scrubbing it to remove any common American names.

(b) Try other ethnicities to see if there are enough ethnic names to use – Chinese, Vietnamese, and Middle Eastern might be good ones to try next.

(c) Consider trying a test run of the new ranking in areas that have a large number of the same type of restaurant. For example:

-Chinese restaurants in the Bay Area
-Pho restaurants in the LA / San Diego Area
-Indian restaurants in the Chicago area.

Provide a means for feedback of this new feature (do you think it made a good selection?). See what happens. Does the system “take” and a lot of people start using it, or is there resistance / criticism of the idea?

10. Conclusions / Summary

-In this analysis, we looked at 2 potential ways for providing secondary ratings to make it easier for Yelp users to select between many different restaurants of the same cuisine – a rating that gives more weight to reviewers who have reviewed other restaurants of the same cuisine, and an "Immigrant Rating" of those with Yelp user names that suggest they might have ethnic expertise with that cuisine.

-Both methods seemed to have enough users in the Yelp database to provide meaningful rankings (at least 10% and as many as 27% of reviewers in the 10 most popular cuisines have reviewed other restaurants of that cuisine, roughly 10% of the reviewers of Indian food have names that would qualify for the "Immigrant Rating").

-Both methods showed they could be useful for selecting between many different restaurants of a cuisine (both successfully identified 1 candidate from at least 6 identically rated restaurants in Tempe, Arizona).

-The 2 methods yield different results with some overlap in some cities. In general, the "Immigrant Rating" had more of an impact and tended to be more negative. The range of the "Weighted Rating" was -1.14 to 0.77 stars with a mean/median of around 0. The range of the "Immigrant Rating" was -1.92 to 0.31 stars with a mean/median of around -0.4.

-Comparing the 2 methods with all of the cities in the Yelp database, the "Immigrant Rating" provided more direct information (in 8 cities where there was a choice, it honed in on one recommendation all 8 times, versus this happening 55% of the time with the "Weighted Rating"). However, there is a tradeoff of a loss of inclusivity (there were 3 cities that didn't have enough 4-star restaurants to qualify for the "Immigrant Rating"). The "Immigrant Rating" was recommended as the best way to "cut throught the clutter."

-If the "Immigrant Rating" is used in Yelp in the form of a badge that a restaurant gets if it has an immigrant rating of 4 or higher, information is lost in 3 cities (we no longer have a clear top choice). This loss of information can be countered by tweaking the rating so that it accepts a rating of 3.5 in cities with no "Immigrant Ratings" of 4. Based on this, we recommend exploring using the "Immigrant Rating" in the form of an "authenticity badge" using that tweak.

As we look at the over 2.2 billion words and counting that are in Yelp's database reviews, one key question will be how to distill all that information to provide the most relevant pieces possible to each user based on what information it has on that user. The "authenticity" badge could be one tool in helping Yelp to use data to achieve that goal.

Appendix A: Indian Names Used

Aayush
Abhi
Abhijeet
Abhijit
Abhilash
Abhinandan
Abhinav
Abhinay
Abhisek
Abhishek
Abilash
Achyuthan
Aditi
Aditya
Adnan
Ahana
Aisha
Aishaa
Ajay
Ajinkya
Akash
Akhil
Akhilesh
Akshada
Akshay
Akshaya
Alina
Alok
Amal
Amalee
Amalia
Aman
Amandeep
Amar
Amaris
Amarnath
Ambika
Ami
Amil
Amir
Amirah
Amisha
Amit
Amogha
Amrita
Anagha
anand
Anand Kumar
Ananth
Ananya
Aneesh
Ani
Aniket
Anil
Anindya
Anirudh
Anish
Anju
Ankan
Ankit
Ankita
Ankur
Ankush
Anoop
Anshul
Anu
Anubhuti
Anuj
Anum
Anup
Anupama
AnuPriya
Anurag
AnurAg
Anush
Anusha
Apara
Aparna
Apeksha
Aravind
Aravind
Baalaaji
Archana
Arin
Arindam
Arjun
Arpita
Arshad
Arul
Arun
Arunkumar
Arvind
Asha
Ashank
Ashish
Ashmita
Ashok
Ashwin
Asish
Asmita
Asodha
Atreyee
Atul
Atulya
Avinash
Kumar
Avishek
Avishekh
Ayesha
Baban
Babjee
Bala
Balaji
Bharadwaj
Bharat
Bharath
Bhavana
Bhavik
Bhavin
Bhavisha
Bhavya
Bhrata
Bhujang
Bhumi
Bhumika
Bibek
Bijal
Brij
Brijen
Brijesh
Brinda
Chaitanya
Chaitra
Chandan
Chandini
Chandra
Chandrasekar
Chaundra
Chetan
Chetu
Chhayakanta
Chiku
Chintan
Chinu
Chirag
Chitra
Chitta
Cintya
Dakshina
Debashri
Deedar
Deeksha
Deepak
Deepali
Deepanjali
Deepesh
Deepika
Deepthy
Deepti
Deiva
Dev
Devang
Devesh
Dhanasekar
Dheeraj
Dhinakaran
Dhruv
Digvijay
Dinesh
Diplekha
Disha
Divyansh
Gaggandeep
Ganesh
Gaurang
Gaurav
Gautam
Gayatri
Geetha
Ghena
Girish
Gita
Gopi
Gurpreet
Gurpreet
Singh
Hardeep
Haresh
Hariharan
Harinath
Harini
Harish
Harjit
Harpreet
Harsh
Harsha
Harshit
Himanshu
Hitesh
Humza
Jagadeesh
Jahan
Jai Veer Singh
Jaina
Jaswant A.
Jateen
Jatin
jaya
jayashri
jayatheerth
jaydei
JAYESH
jayshree
jeetendar
jharna
jimeet
jinesh
Kalpesh
Kalyan
Kamesh
Kanishk
Kannan
Karan
Karthik
Karthiknathan
Kartik
Karuna
Kashyap
Kavita
Kavitha

Appendix A: Indian Names Used

Kedar
Keerthi
Keren
Ketan
Kewal
Keya
Khushbu
Kira
Kiran
Kirra
Kirti
Kishore
Krishan
Krishna
KrishNa
Krishnasri
Kriti
Kritika
Krupesh
Kumar
Kunal
Kyara
Laks
Lakshmi
Lanka
Ksheera
Sagar
Laxmi
Madhu
Madhur
Madhuri
Mahadeva
Mahanth
Mahathi
Mahawish
Mahesh
Maithreyi
makku
Mala
Manav
Mandi
Mandisha
Mangala
Manikandan
Manish
Manjeera

Manveer
Mayur
Meena
Meera
meha
Mehta
Mehul
Mihir
Milind
Misbah
Mohun
Monal
Monali
Monish
Mukesh
Mukund
Munir
Nabeel
Nachiket
Nadeem
Naga
Nagdeep
Nahja
Najwan
Nakul
Namita
Namrata
Nanda
Nandini
Narayan
Naren
Naresh
Naveen
Navneet
Navpreet
Navya
Nayan
Neel
Neelam
Neelesh
Neeraj
Neha
Nidhi
Nikhil
Nilanjan
Nilesh

Nimish
Niraj
Niranjana
Nirmal
Nisarg
Nishant
Nishitaa
Nithesh
Nitin
Nitya
Nivedhitha
Padmaja
Padmashree
Palak
Pancham
Pankaj
Parikshith
Parth
Parthiv
Pavan
Pavi
Pavitha
Pavithra
Pavneet
Peeyush
Pinak
Piyoosh
Piyush
Pooja
Poorna
Poornima
Prbhakar
Prabhjot
Prabhu
Prabin
Pracheta
Prachi
Pragna
Prajod
Pranab
Pranathi
Pranav
Pranay
Praneeth
Praphul
Prasad

Prasanna
Prasen
Prashant
Prashanth
Prateek
Pratiba
Pratik
Pratiti
Praveen
Prayag
Preet
Preeti
Prem
Premsankar
Prerana
Priya
Priyam
Priyank
Prudhvi
Puneet
Punita
Purnima
Rachit
Raghav
Raghava
Raghu
Raghuram
Rahul
Rai
Raj
Raja
Rajat
Rajeev
Rajendra
Rjean K. C.
Rajesh
Rajiv
Rajni
Raju
Rakesh
Raki
Ram
Ramah
Raman
Ramandeep
Ramesh

Rasesh
Rashmi
Rav
Raveendran
Ravi
Ravi Krishna
Raviteja
Risha
Rishi
Rishik
Rishoo
Ritesh
Rohan
Rohit
Ronak
Roshan
Roshani
Roshni
Ruchi
Rupa
Rupu
Rushabh
Rushikesh
Sachin
Sadhu
Sagar
Sai
Sairam
Sakshi
Samara
Sambhav
Sameer
Samir
Sampad
Sandeep
Sandhya
Sangeetha
Sanjay
Sanjib
Sanjith
Sanjiv
Santosh
Sarang
Sarnnya
Saravana
Saravanan

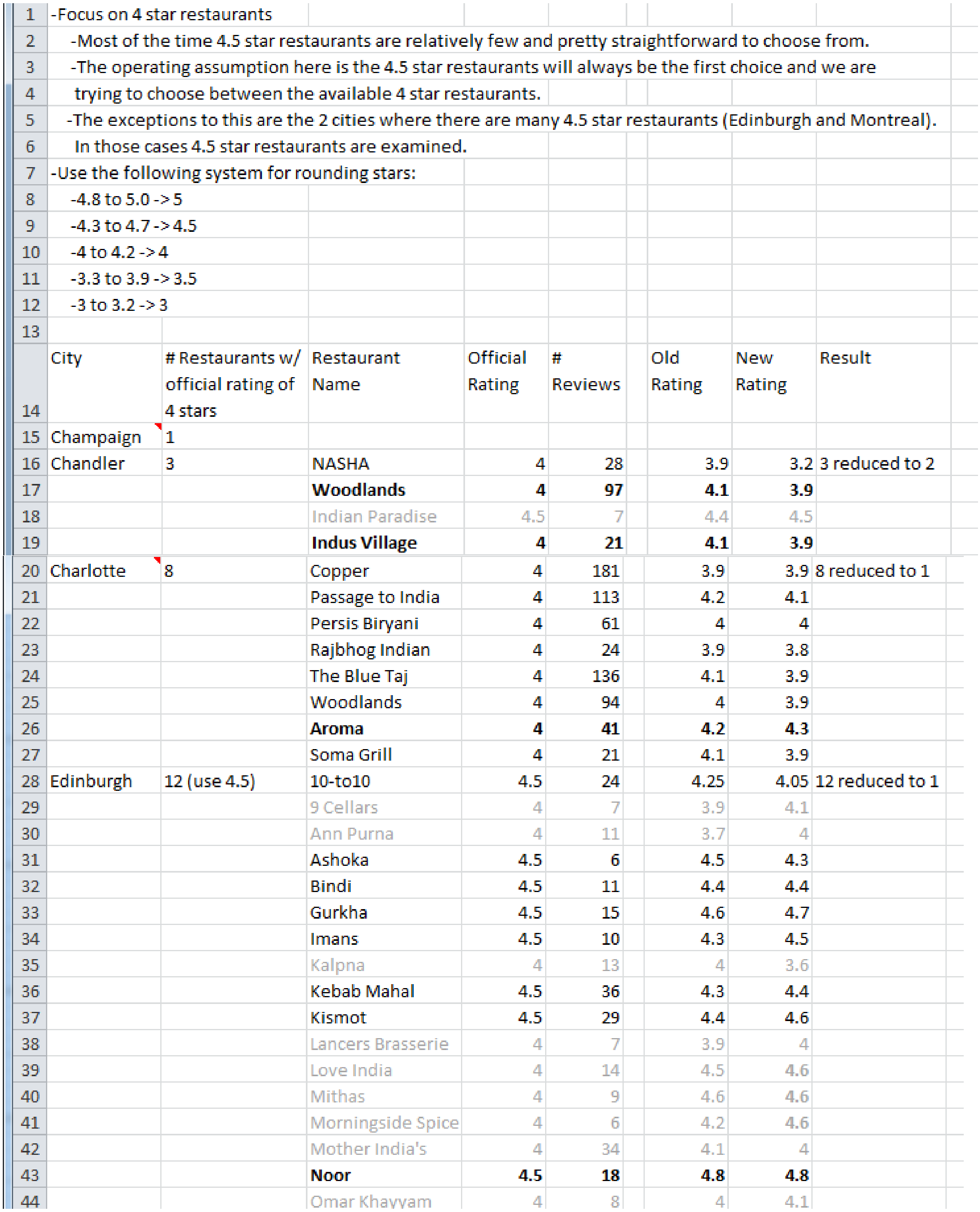
Appendix A: Indian Names Used

Saroj
Sarun
Sashidhar
Satender
Sathya
Satya
Satyajit
Satyaswaroop
Satyin
Saumav
Saumya
Saurabh
Savinay
Sayed
Sayjul
Seema
Seema and
Amit
Senthilkumar
Shailesh
Shaji
Shakerul
Shakil
Shalini
Shalu
Shamanth
Shanda
Shankar
Shantanu
Sharath
Shashank
Sheema
Sheetal
Shikha
Shilpa
Shilpashree
Shilpi
Shiraz
Shiv
Shiva
Shivani
Shivanshu
Shreejay
Shrikant
Shrimant
Shriya
Shruti
Siddharth
Sidhartha
Sidhu
Sirish
Sneha
Snigdha
Soham
Someshwar
Somnath
Sonal
Sonel
Soroush
Soujanya
Sourav
Souvir
Sowmiya
Spandana
Sree
Sri
Sri Balaji
Sridhar
Srikanth
Srikumar
Srini
Srinivas
Sriram
Srishti
Srivatsava
Subha
Subhash
Sdatta
Sudha
Sudhakar
Sudhanwa
Sudheer
Sudip
Sujith
Sukh
Suma
Sumaira
Sumaithri
Suman
Sumit
Sundar
Sunil
Supreeth
Surajit
Suraya
Surranna
Surya
Sushant
Suvir
Swapnil
Swati
Swikrit
Syed
Tanweer
Tapan
Tarun
Tripti
Tuhin
Tushaar
Tushar
Tushara
Umesh
Urvi
Urvish
Vaivelan
Vaibhav
Vaishali
Vaishnav
Vamsee
Vamsi
Varinder
Varun
Varuni
Vasista
Veena
Vemana
Venkat
Venkata
Venkesh
Vibhor
Vibhu
Vignesh
Vijay
Vijay Kumar
Vijaykumarty
Vijith
Vikas
Vikash
Vikram
Vinay
Vinayak
Vineet
Vineeth
Vinit
Vinod
Vinoth
Vinuth
Vipin
Vipin Das
Vipul
Vishal
Vishnu
Vishwa
Vivek
Yogesh
Yuvaraj

Appendix B: Details of City-by-city Results for Methods 1 and 2

Method 1: “Weighted” Ratings Results Detail:

Note: Process followed described in detail at the beginning. Recommended restaurant is shown in bold.



1 -Focus on 4 star restaurants

2 -Most of the time 4.5 star restaurants are relatively few and pretty straightforward to choose from.

3 -The operating assumption here is the 4.5 star restaurants will always be the first choice and we are

4 trying to choose between the available 4 star restaurants.

5 -The exceptions to this are the 2 cities where there are many 4.5 star restaurants (Edinburgh and Montreal).

6 In those cases 4.5 star restaurants are examined.

7 -Use the following system for rounding stars:

8 -4.8 to 5.0 -> 5

9 -4.3 to 4.7 -> 4.5

10 -4 to 4.2 -> 4

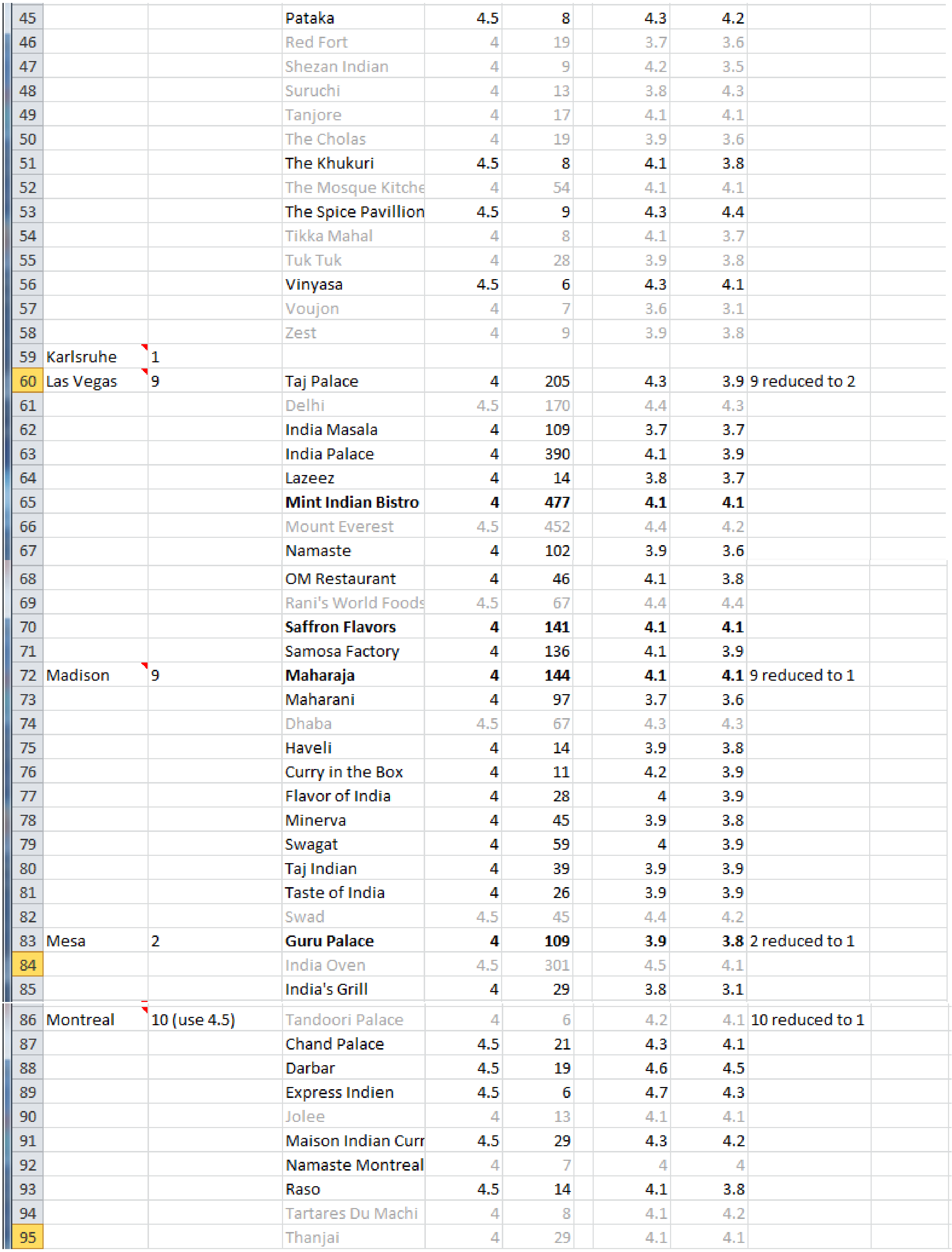
11 -3.3 to 3.9 -> 3.5

12 -3 to 3.2 -> 3

13

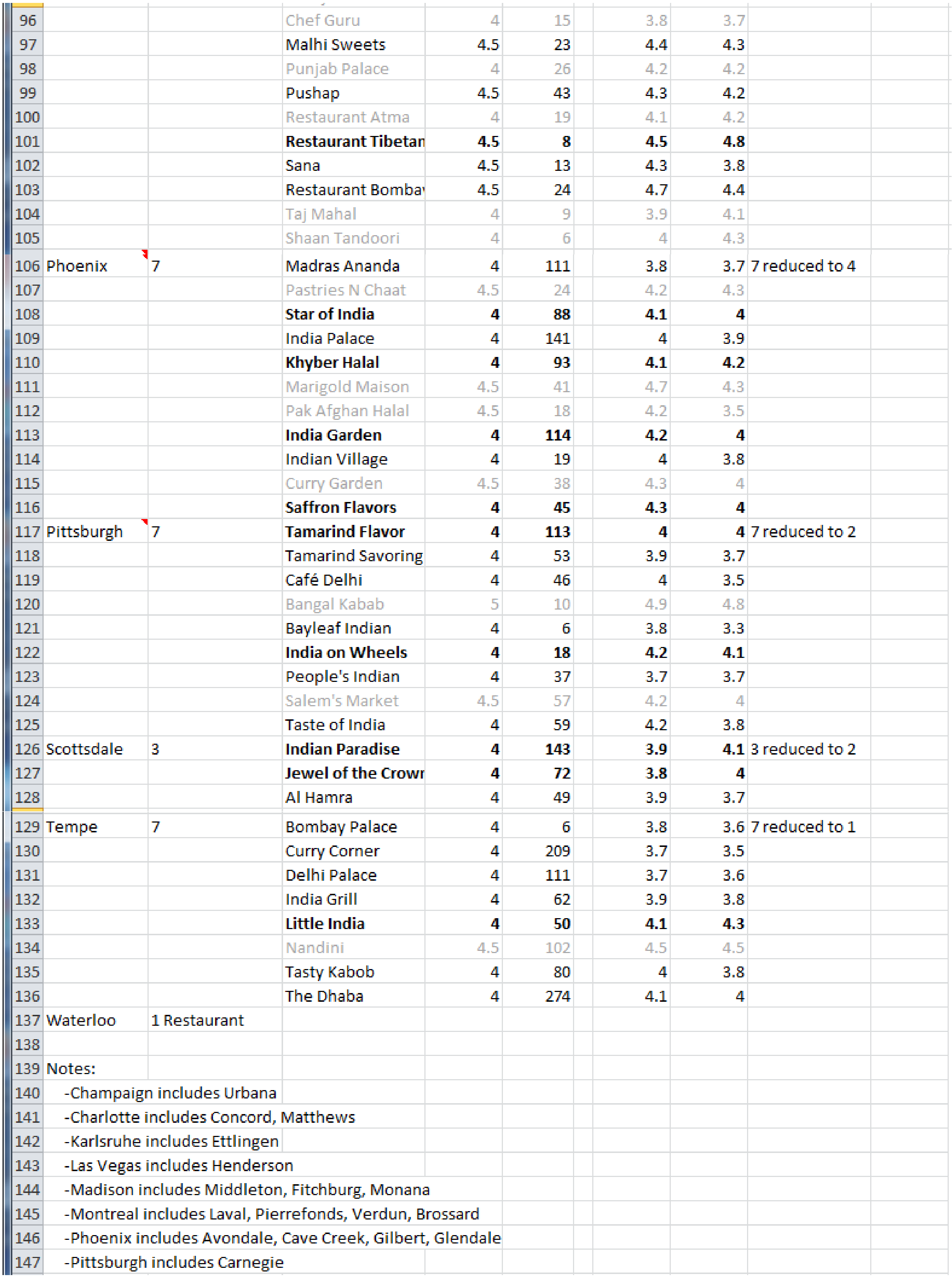
14	City	# Restaurants w/ official rating of 4 stars	Restaurant Name	Official Rating	# Reviews	Old Rating	New Rating	Result
15	Champaign	1						
16	Chandler	3	NASHA	4	28	3.9	3.2	3 reduced to 2
17			**Woodlands**	**4**	**97**	**4.1**	**3.9**	
18			Indian Paradise	4.5	7	4.4	4.5	
19			**Indus Village**	**4**	**21**	**4.1**	**3.9**	
20	Charlotte	8	Copper	4	181	3.9	3.9	8 reduced to 1
21			Passage to India	4	113	4.2	4.1	
22			Persis Biryani	4	61	4	4	
23			Rajbhog Indian	4	24	3.9	3.8	
24			The Blue Taj	4	136	4.1	3.9	
25			Woodlands	4	94	4	3.9	
26			**Aroma**	**4**	**41**	**4.2**	**4.3**	
27			Soma Grill	4	21	4.1	3.9	
28	Edinburgh	12 (use 4.5)	10-to10	4.5	24	4.25	4.05	12 reduced to 1
29			9 Cellars	4	7	3.9	4.1	
30			Ann Purna	4	11	3.7	4	
31			Ashoka	4.5	6	4.5	4.3	
32			Bindi	4.5	11	4.4	4.4	
33			Gurkha	4.5	15	4.6	4.7	
34			Imans	4.5	10	4.3	4.5	
35			Kalpna	4	13	4	3.6	
36			Kebab Mahal	4.5	36	4.3	4.4	
37			Kismot	4.5	29	4.4	4.6	
38			Lancers Brasserie	4	7	3.9	4	
39			Love India	4	14	4.5	4.6	
40			Mithas	4	9	4.6	4.6	
41			Morningside Spice	4	6	4.2	4.6	
42			Mother India's	4	34	4.1	4	
43			**Noor**	**4.5**	**18**	**4.8**	**4.8**	
44			Omar Khayyam	4	8	4	4.1	

Appendix B: Details of City-by-city Results for Methods 1 and 2



45			Pataka	4.5	8	4.3	4.2	
46			Red Fort	4	19	3.7	3.6	
47			Shezan Indian	4	9	4.2	3.5	
48			Suruchi	4	13	3.8	4.3	
49			Tanjore	4	17	4.1	4.1	
50			The Cholas	4	19	3.9	3.6	
51			The Khukuri	4.5	8	4.1	3.8	
52			The Mosque Kitche	4	54	4.1	4.1	
53			The Spice Pavillion	4.5	9	4.3	4.4	
54			Tikka Mahal	4	8	4.1	3.7	
55			Tuk Tuk	4	28	3.9	3.8	
56			Vinyasa	4.5	6	4.3	4.1	
57			Voujon	4	7	3.6	3.1	
58			Zest	4	9	3.9	3.8	
59	Karlsruhe	1						
60	Las Vegas	9	Taj Palace	4	205	4.3	3.9	9 reduced to 2
61			Delhi	4.5	170	4.4	4.3	
62			India Masala	4	109	3.7	3.7	
63			India Palace	4	390	4.1	3.9	
64			Lazeez	4	14	3.8	3.7	
65			**Mint Indian Bistro**	**4**	**477**	**4.1**	**4.1**	
66			Mount Everest	4.5	452	4.4	4.2	
67			Namaste	4	102	3.9	3.6	
68			OM Restaurant	4	46	4.1	3.8	
69			Rani's World Foods	4.5	67	4.4	4.4	
70			**Saffron Flavors**	**4**	**141**	**4.1**	**4.1**	
71			Samosa Factory	4	136	4.1	3.9	
72	Madison	9	**Maharaja**	**4**	**144**	**4.1**	**4.1**	9 reduced to 1
73			Maharani	4	97	3.7	3.6	
74			Dhaba	4.5	67	4.3	4.3	
75			Haveli	4	14	3.9	3.8	
76			Curry in the Box	4	11	4.2	3.9	
77			Flavor of India	4	28	4	3.9	
78			Minerva	4	45	3.9	3.8	
79			Swagat	4	59	4	3.9	
80			Taj Indian	4	39	3.9	3.9	
81			Taste of India	4	26	3.9	3.9	
82			Swad	4.5	45	4.4	4.2	
83	Mesa	2	**Guru Palace**	**4**	**109**	**3.9**	**3.8**	2 reduced to 1
84			India Oven	4.5	301	4.5	4.1	
85			India's Grill	4	29	3.8	3.1	
86	Montreal	10 (use 4.5)	Tandoori Palace	4	6	4.2	4.1	10 reduced to 1
87			Chand Palace	4.5	21	4.3	4.1	
88			Darbar	4.5	19	4.6	4.5	
89			Express Indien	4.5	6	4.7	4.3	
90			Jolee	4	13	4.1	4.1	
91			Maison Indian Curr	4.5	29	4.3	4.2	
92			Namaste Montreal	4	7	4	4	
93			Raso	4.5	14	4.1	3.8	
94			Tartares Du Machi	4	8	4.1	4.2	
95			Thanjai	4	29	4.1	4.1	

Appendix B: Details of City-by-city Results for Methods 1 and 2

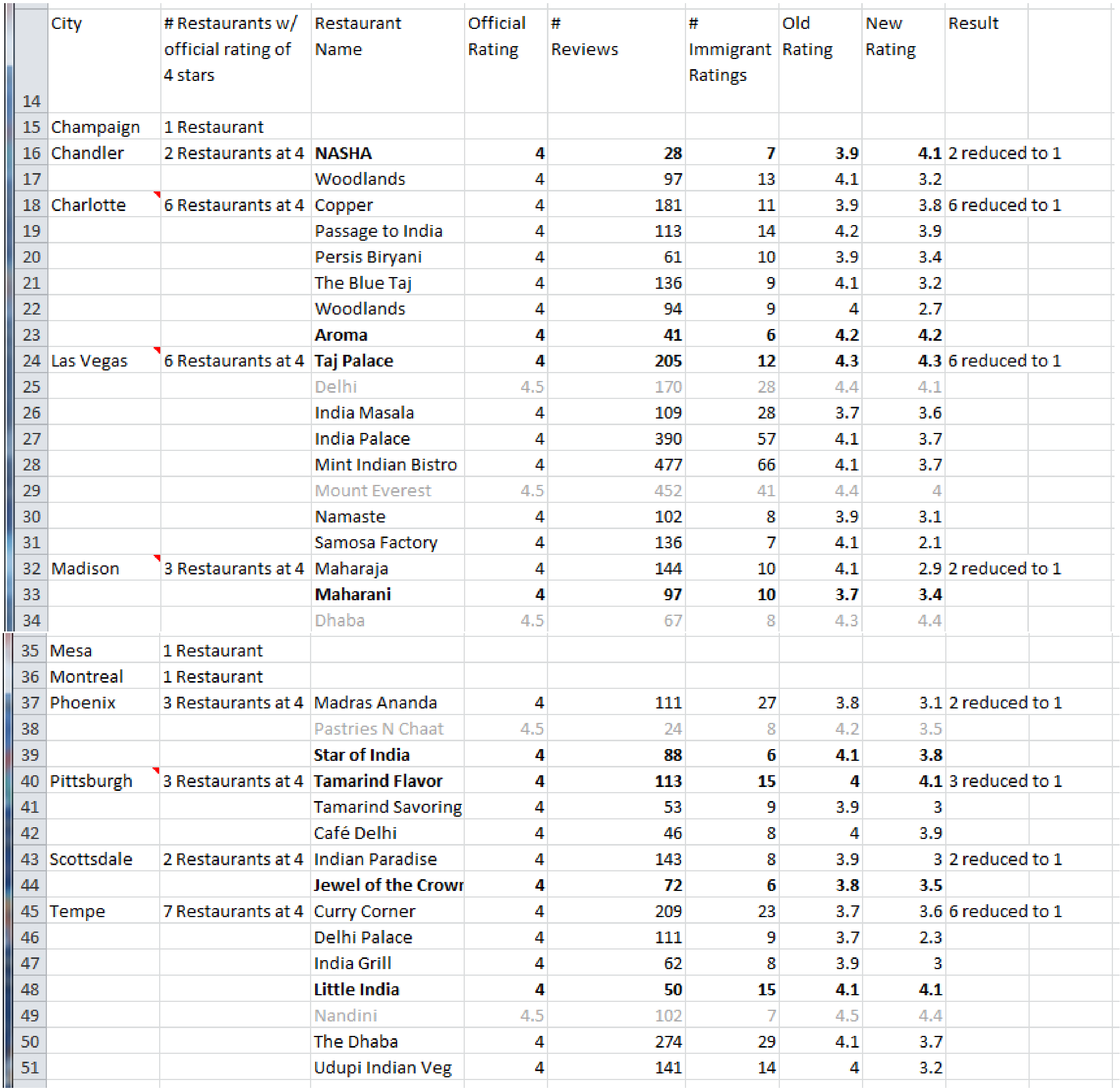


96			Chef Guru	4	15	3.8	3.7	
97			Malhi Sweets	4.5	23	4.4	4.3	
98			Punjab Palace	4	26	4.2	4.2	
99			Pushap	4.5	43	4.3	4.2	
100			Restaurant Atma	4	19	4.1	4.2	
101			**Restaurant Tibetan**	**4.5**	**8**	**4.5**	**4.8**	
102			Sana	4.5	13	4.3	3.8	
103			Restaurant Bombay	4.5	24	4.7	4.4	
104			Taj Mahal	4	9	3.9	4.1	
105			Shaan Tandoori	4	6	4	4.3	
106	Phoenix	7	Madras Ananda	4	111	3.8	3.7	7 reduced to 4
107			Pastries N Chaat	4.5	24	4.2	4.3	
108			**Star of India**	**4**	**88**	**4.1**	**4**	
109			India Palace	4	141	4	3.9	
110			**Khyber Halal**	**4**	**93**	**4.1**	**4.2**	
111			Marigold Maison	4.5	41	4.7	4.3	
112			Pak Afghan Halal	4.5	18	4.2	3.5	
113			**India Garden**	**4**	**114**	**4.2**	**4**	
114			Indian Village	4	19	4	3.8	
115			Curry Garden	4.5	38	4.3	4	
116			**Saffron Flavors**	**4**	**45**	**4.3**	**4**	
117	Pittsburgh	7	**Tamarind Flavor**	**4**	**113**	**4**	**4**	7 reduced to 2
118			Tamarind Savoring	4	53	3.9	3.7	
119			Café Delhi	4	46	4	3.5	
120			Bangal Kabab	5	10	4.9	4.8	
121			Bayleaf Indian	4	6	3.8	3.3	
122			**India on Wheels**	**4**	**18**	**4.2**	**4.1**	
123			People's Indian	4	37	3.7	3.7	
124			Salem's Market	4.5	57	4.2	4	
125			Taste of India	4	59	4.2	3.8	
126	Scottsdale	3	**Indian Paradise**	**4**	**143**	**3.9**	**4.1**	3 reduced to 2
127			**Jewel of the Crown**	**4**	**72**	**3.8**	**4**	
128			Al Hamra	4	49	3.9	3.7	
129	Tempe	7	Bombay Palace	4	6	3.8	3.6	7 reduced to 1
130			Curry Corner	4	209	3.7	3.5	
131			Delhi Palace	4	111	3.7	3.6	
132			India Grill	4	62	3.9	3.8	
133			**Little India**	**4**	**50**	**4.1**	**4.3**	
134			Nandini	4.5	102	4.5	4.5	
135			Tasty Kabob	4	80	4	3.8	
136			The Dhaba	4	274	4.1	4	
137	Waterloo	1 Restaurant						
138								
139	Notes:							
140	-Champaign includes Urbana							
141	-Charlotte includes Concord, Matthews							
142	-Karlsruhe includes Ettlingen							
143	-Las Vegas includes Henderson							
144	-Madison includes Middleton, Fitchburg, Monana							
145	-Montreal includes Laval, Pierrefonds, Verdun, Brossard							
146	-Phoenix includes Avondale, Cave Creek, Gilbert, Glendale							
147	-Pittsburgh includes Carnegie							

Appendix B: Details of City-by-city Results for Methods 1 and 2

Method 2: "Immigrant" Ratings Results Detail:

Note: The same notes at the beginning and end of Method 1 about the process followed and what city names were combined apply here as well. Recommended restaurant is shown in bold.



14	City	# Restaurants w/ official rating of 4 stars	Restaurant Name	Official Rating	# Reviews	# Immigrant Ratings	Old Rating	New Rating	Result
15	Champaign	1 Restaurant							
16	Chandler	2 Restaurants at 4	**NASHA**	**4**	**28**	**7**	**3.9**	**4.1**	2 reduced to 1
17			Woodlands	4	97	13	4.1	3.2	
18	Charlotte	6 Restaurants at 4	Copper	4	181	11	3.9	3.8	6 reduced to 1
19			Passage to India	4	113	14	4.2	3.9	
20			Persis Biryani	4	61	10	3.9	3.4	
21			The Blue Taj	4	136	9	4.1	3.2	
22			Woodlands	4	94	9	4	2.7	
23			**Aroma**	**4**	**41**	**6**	**4.2**	**4.2**	
24	Las Vegas	6 Restaurants at 4	**Taj Palace**	**4**	**205**	**12**	**4.3**	**4.3**	6 reduced to 1
25			Delhi	4.5	170	28	4.4	4.1	
26			India Masala	4	109	28	3.7	3.6	
27			India Palace	4	390	57	4.1	3.7	
28			Mint Indian Bistro	4	477	66	4.1	3.7	
29			Mount Everest	4.5	452	41	4.4	4	
30			Namaste	4	102	8	3.9	3.1	
31			Samosa Factory	4	136	7	4.1	2.1	
32	Madison	3 Restaurants at 4	Maharaja	4	144	10	4.1	2.9	2 reduced to 1
33			**Maharani**	**4**	**97**	**10**	**3.7**	**3.4**	
34			Dhaba	4.5	67	8	4.3	4.4	
35	Mesa	1 Restaurant							
36	Montreal	1 Restaurant							
37	Phoenix	3 Restaurants at 4	Madras Ananda	4	111	27	3.8	3.1	2 reduced to 1
38			Pastries N Chaat	4.5	24	8	4.2	3.5	
39			**Star of India**	**4**	**88**	**6**	**4.1**	**3.8**	
40	Pittsburgh	3 Restaurants at 4	**Tamarind Flavor**	**4**	**113**	**15**	**4**	**4.1**	3 reduced to 1
41			Tamarind Savoring	4	53	9	3.9	3	
42			Café Delhi	4	46	8	4	3.9	
43	Scottsdale	2 Restaurants at 4	Indian Paradise	4	143	8	3.9	3	2 reduced to 1
44			**Jewel of the Crowr**	**4**	**72**	**6**	**3.8**	**3.5**	
45	Tempe	7 Restaurants at 4	Curry Corner	4	209	23	3.7	3.6	6 reduced to 1
46			Delhi Palace	4	111	9	3.7	2.3	
47			India Grill	4	62	8	3.9	3	
48			**Little India**	**4**	**50**	**15**	**4.1**	**4.1**	
49			Nandini	4.5	102	7	4.5	4.4	
50			The Dhaba	4	274	29	4.1	3.7	
51			Udupi Indian Veg	4	141	14	4	3.2	

Appendix C: Python Preprocessing Script

```
#!/usr/bin/python

import sys
import json
import csv
import io

"""ConvertYelp

Written by:  Robert Chen
Date:        1/10/2016

Processes one of 3 Yelp JSON files.  It reads in the JSON data, extracts the fields
needed from that file, and writes the result to a CSV file.

The "json" and "csv" libraries are used for handling the input and output.  The encode
method is used to handle unicode characters that appear in "name" and "city" in the Yelp
"business" data file and in "name" in the Yelp "user" data file.

To use, type the following commands:

   python ConvertYelp.py yelp_academic_dataset_review.json
   python ConvertYelp.py yelp_academic_dataset_user.json
   python ConvertYelp.py yelp_academic_dataset_business.json

   A corresponding .csv file is generated for each run.

"""

def filter_and_convert_to_csv(filename):
  """
  Open the specified file.  The file is either the "review", "user", or "business" file.
  Depending on the type of file it is, read in certain fields and then save the result
  in CSV format.
  """

  if "review" in filename:

     outputFile = open('yelp_academic_dataset_review.csv', 'w')
     fields= ['user_id', 'business_id', 'stars']

     # The "lineterminator='\n' is needed to prevent an extra blank line between each line.
     outputWriter = csv.DictWriter(outputFile, fieldnames = fields, lineterminator='\n')
     print "Converting " + filename + " to " + "yelp_academic_dataset_review.csv ..."

     # Read line by line, write user_id, business_id, and stars to CSV file
     for line in open(filename, 'r'):
        r = json.loads(line)
        outputWriter.writerow({'user_id': r['user_id'], 'business_id': r['business_id'],
                              'stars': r['stars']})

  elif "user" in filename:

     outputFile = open('yelp_academic_dataset_user.csv', 'w')
```

Appendix C: Python Preprocessing Script

```
     fields= ['user_id', 'name']

     # The "lineterminator='\n' is needed to prevent an extra blank line between each line.
     outputWriter = csv.DictWriter(outputFile, fieldnames = fields, lineterminator='\n')
     print "Converting " + filename + " to " + "yelp_academic_user_review.csv ..."

     # Read line by line, write user_id and name to CSV file
     for line in open(filename, 'r'):
        r = json.loads(line)

        # To handle name values with unicode, call "encode" to remove the unicode character.
        # This presents a problem from occurring in "writerow" (which cannot handle unicode
         # well)

        n = r['name']
        n1 = n.encode('ascii', 'ignore')
        outputWriter.writerow({'user_id': r['user_id'], 'name': n1})

  elif "business" in filename:

     outputFile = open('yelp_academic_dataset_business.csv', 'w')
     fields= ['business_id', 'city', 'name', 'categories', 'review_count', 'stars']
     outputWriter = csv.DictWriter(outputFile, fieldnames = fields, lineterminator='\n')
     print "Converting " + filename + " to " + "yelp_academic_dataset_review.csv ..."

     # Read line by line, write relevant fields if the business is a restaurant
     for line in open(filename, 'r'):
        r = json.loads(line)
        categories = str(r['categories'])
        if "Restaurants" in categories:
           # To handle name values with unicode, call "encode" to remove the unicode character.
           # This presents a problem from occurring in "writerow" (which cannot handle unicode
           # well)

           n = r['name']
           n1 = n.encode('ascii', 'ignore')
           c = r['city']
           c1 = c.encode('ascii', 'ignore')

           # Now write the result to a CSV file
           outputWriter.writerow({'business_id': r['business_id'], 'city': c1, 'name': n1,
                                  'categories': r['categories'],
                                  'review_count': r['review_count'], 'stars': r['stars']})

         else:

           print "Error!  Unexpected filename used."
           exit()

  outputFile.close

def main():
  # This command-line parsing code is provided.
  # Make a list of command line arguments, omitting the [0] element
```

Appendix C: Python Preprocessing Script

```
  # which is the script itself.
  args = sys.argv[1:]

  if not args:
    print 'usage: file'
    sys.exit(1)

  filter_and_convert_to_csv(sys.argv[1])

if __name__ == '__main__':
  main()
```

Appendix D: R Processing Script

```
###########################################
# R commands to process the Yelp database #
###########################################

#############################################
# Part 1:  Setup and initial data wrangling #
#############################################

# Load library
library(dplyr)

# Read in csv files
reviews    <- read.csv("yelp_academic_dataset_review.csv",   header = FALSE)
users      <- read.csv("yelp_academic_dataset_user.csv",     header = FALSE)
businesses <- read.csv("yelp_academic_dataset_business.csv", header = FALSE)

# Add names to the fields
colnames(reviews)[1] = "user_id"
colnames(reviews)[2] = "business_id"
colnames(reviews)[3] = "stars"
colnames(users)[1] = "user_id"
colnames(users)[2] = "user_name"
colnames(businesses)[1] = "business_id"
colnames(businesses)[2] = "city"
colnames(businesses)[3] = "business_name"
colnames(businesses)[4] = "categories"
colnames(businesses)[5] = "review_count"
colnames(businesses)[6] = "avg_stars"

# Join the files
ru  <- inner_join(reviews, users)
rub <- inner_join(ru, businesses)

######################################################
# Part 2a:  Analysis of Method 1 -- Initial Analysis #
######################################################

# Add "is_indian" field for any review that has "Indian" in "categories"
rub$is_indian <- grepl("Indian", rub$categories) == TRUE

# Make a dataframe of just reviews of Indian restaurants
indian <- subset(rub, is_indian == TRUE)

# Generate a summary of # of reviews of that cuisine done by each reviewer
num_reviews_Indian <- indian %>% select(user_id, user_name, is_indian) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_indian))

# Print the table, show the total # of entries, and find the avg # of reviews per user
table(num_reviews_Indian$tot_rev)
count(num_reviews_Indian)
mean(num_reviews_Indian$tot_rev)

#################################################################
```

Appendix D: R Processing Script

```
# Part 2b:  Analysis of Method 1 -- Extension to Other Cuisines #
#################################################################

rub$is_chinese <- grepl("Chinese", rub$categories) == TRUE
chinese <- subset(rub, is_chinese == TRUE)
num_reviews_Chinese <- chinese %>% select(user_id, user_name, is_chinese) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_chinese))
table(num_reviews_Chinese$tot_rev)
count(num_reviews_Chinese)
mean(num_reviews_Chinese$tot_rev)

rub$is_mexican <- grepl("Mexican", rub$categories) == TRUE
mexican <- subset(rub, is_mexican == TRUE)
num_reviews_Mexican <- mexican %>% select(user_id, user_name, is_mexican) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_mexican))
table(num_reviews_Mexican$tot_rev)
count(num_reviews_Mexican)
mean(num_reviews_Mexican$tot_rev)

rub$is_italian <- grepl("Italian", rub$categories) == TRUE
italian <- subset(rub, is_italian == TRUE)
num_reviews_Italian <- italian %>% select(user_id, user_name, is_italian) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_italian))
table(num_reviews_Italian$tot_rev)
count(num_reviews_Italian)
mean(num_reviews_Italian$tot_rev)

# For Japanese, look for "Japanese" or "Sushi"
rub$is_japanese <- (grepl("Japanese", rub$categories) == TRUE) |
                   (grepl("Sushi",    rub$categories) == TRUE)
japanese <- subset(rub, is_japanese == TRUE)
num_reviews_Japanese <- japanese %>% select(user_id, user_name, is_japanese) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_japanese))
table(num_reviews_Japanese$tot_rev)
count(num_reviews_Japanese)
mean(num_reviews_Japanese$tot_rev)

rub$is_greek <- grepl("Greek", rub$categories) == TRUE
greek <- subset(rub, is_greek == TRUE)
num_reviews_Greek <- greek %>% select(user_id, user_name, is_greek) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_greek))
table(num_reviews_Greek$tot_rev)
count(num_reviews_Greek)
mean(num_reviews_Greek$tot_rev)

rub$is_french <- grepl("French", rub$categories) == TRUE
french <- subset(rub, is_french == TRUE)
num_reviews_French <- french %>% select(user_id, user_name, is_french) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_french))
table(num_reviews_French$tot_rev)
count(num_reviews_French)
mean(num_reviews_French$tot_rev)

rub$is_thai <- grepl("Thai", rub$categories) == TRUE
thai <- subset(rub, is_thai == TRUE)
num_reviews_Thai <- thai %>% select(user_id, user_name, is_thai) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_thai))
table(num_reviews_Thai$tot_rev)
count(num_reviews_Thai)
mean(num_reviews_Thai$tot_rev)

rub$is_spanish <- (grepl("Spanish", rub$categories) == TRUE) |
                  (grepl("Tapas",   rub$categories) == TRUE)
spanish <- subset(rub, is_spanish == TRUE)
num_reviews_Spanish <- spanish %>% select(user_id, user_name, is_spanish) %>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_spanish))
table(num_reviews_Spanish$tot_rev)
count(num_reviews_Spanish)
mean(num_reviews_Spanish$tot_rev)

rub$is_mediterranean <- grepl("Mediterranean", rub$categories) == TRUE
mediterranean <- subset(rub, is_mediterranean == TRUE)
num_reviews_Mediterranean <- mediterranean %>% select(user_id, user_name, is_mediterranean)
%>%
  group_by(user_id) %>%
  summarise(tot_rev = sum(is_mediterranean))
table(num_reviews_Mediterranean$tot_rev)
count(num_reviews_Mediterranean)
mean(num_reviews_Mediterranean$tot_rev)

####################################################################
# Part 2c:  Analysis of Method 1 -- Apply new weight and see effect #
####################################################################

# Combine num_reviews information with original data frame of indian restaurant reviews
cin <- inner_join(indian, num_reviews_Indian)

# Generate "weighted_stars" for later calculation
cin$weighted_stars <- cin$stars * cin$tot_rev

# Use "summarise" to generate a new rating for each restaurant
new_rating_Indian <- cin %>% select(city, business_name, avg_stars, stars,
                                    tot_rev, weighted_stars) %>%
  group_by(city, business_name, avg_stars) %>%
  summarise(cnt = n(),
            avg = sum(stars) / cnt,
            new = sum(weighted_stars) / sum(tot_rev),
            dif = new - avg)

# Print summary data of the effect this new rating has
```

Appendix D: R Processing Script

```
summary(new_rating_Indian$dif)

# Limit to those with at least 5 ratings and redo summary
nri5 <- subset(new_rating_Indian, cnt > 5)
summary(nri5$dif)


################################################################
# Part 3:  Analysis of Method 2 -- Generate "immigrant" rating #
################################################################

# Read Indian names into a list
inames <- scan("indian_names.txt", what = character())

# Add field "reviewer_indian_name" to indian reviews if user name is in the list
indian$reviewer_indian_name <- indian$user_name %in% inames

# Generate "istars" for internal calculation later
indian$istars <- indian$stars * indian$reviewer_indian_name

# Find out # of reviewers with a uniquely Indian name
table(indian$reviewer_indian_name)
1274/(1274 + 11872)    # .096

# Generate new "immigrant" rating
avg_rating_Indian <- indian %>% select(business_id, business_name, city, stars,
                                       avg_stars, reviewer_indian_name,
                                       is_indian, istars) %>%
                                group_by(city, business_name, avg_stars) %>%
                                summarise(count = n(),
                                          nin = sum(reviewer_indian_name),
                                          pin = sum(reviewer_indian_name) / n(),
                                          avg = sum(stars) / count,
                                          ias = sum(istars) / nin,
                                          dif = ias - avg)

# Find out extent of effect of new rating
summary(avg_rating_Indian$dif)

# Limit to those restaurants with at least 5 "immigrant" reviews and look at effect again
ari5 <- subset (avg_rating_Indian, nin > 5)
summary(ari5$dif)
```