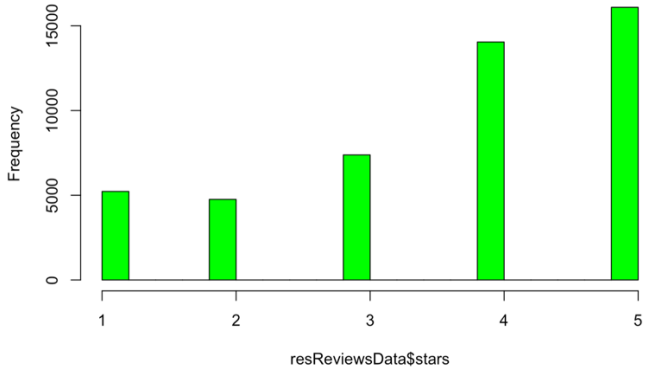
DATA MINING YELP DATA

Jaime Aranda

A. Explore the data. How are star ratings distributed? How will you use the star ratings to obtain a label indicating 'positive' or 'negative' – explain using the data, graphs, etc.? Do star ratings have any relation to 'funny', 'cool', 'useful'? (Is this what you expected?)

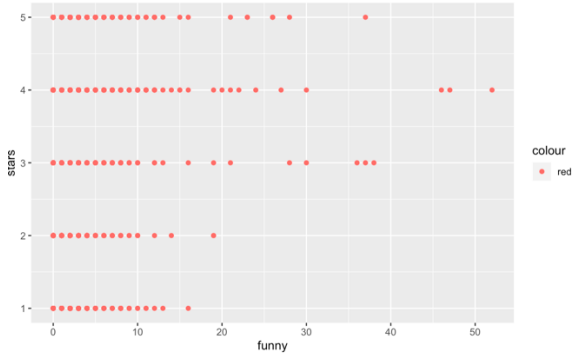
When plotted into a histogram we can see the star ratings are on a scale of 1-5. 1 being considered a "bad review" and 5 being considered a "good review". In our histogram the x axis represents the star review given and the y axis represents the frequency of occurrence per each star occurrence. 4- and 5-star ratings have the highest frequency of occurrences as we can see in our histogram below. 1- and 2-star ratings have a frequency of about 5000 occurrences each. On the other hand, 4- and 5-star ratings have a frequency of about 15,000 occurrences.



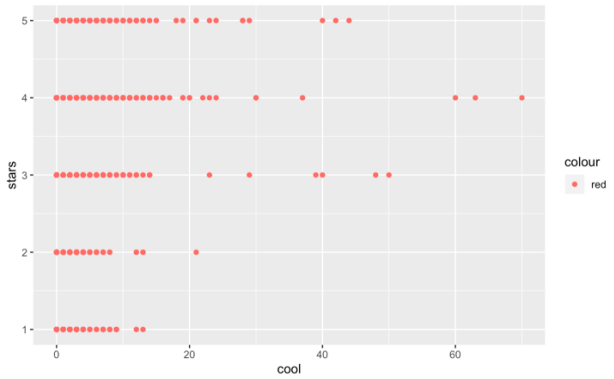
Our mean of star distribution is 3.65 with a standard deviation of 1.32.

In relation to "Funny", "Cool", "Useful" the following are my findings:

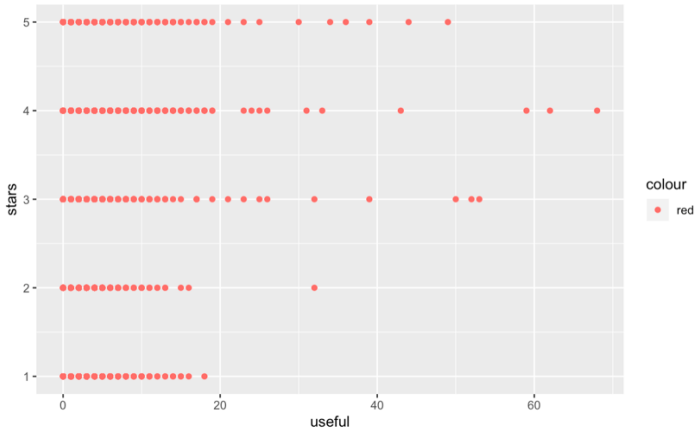
Funny:
The relationship here seems to be strongest at a 4-star rating with approximately 52 occurrences. We can also easily note that as the star rating goes down the number of occurrences does as well and vice versa.



Cool:
Like "Funny" there is a similar relationship between "Cool" and the star rating frequency. A 4-star rating has the most frequency with the word cool. "Cool" is also less attributed to what we would consider as bad ratings (2 stars and 1 star ratings).



Useful: the usefulness or a rating seems to correlate with higher with highly rated review. That is, higher rated reviews seem to be looked at as more useful. It was my belief that bad reviews would be considered more useful since they can be deciding factors if a person decides to eat at a certain place or not.



Star rating is a very useful method for analyzing how positive or negative a rating is. We can definitely use 4 and 5 star ratings and attribute them to “positive” and take 1 and 2 star ratings and attribute them to “negative”. Based on further analysis we will decide later on what to do with the 3 star ratings between making them positive, negative, or dumping them out of the analysis altogether.

B. What are some words indicative of positive and negative sentiment? (One approach is to determine the average star rating for a word based on star ratings of documents where the word occurs). Do these ‘positive’ and ‘negative’ words make sense in the context of user reviews? (For this, since we wish to get a general sense of positive/negative terms, you may like to consider a pruned set of terms -- say, those which occur in a certain minimum and maximum number of documents).

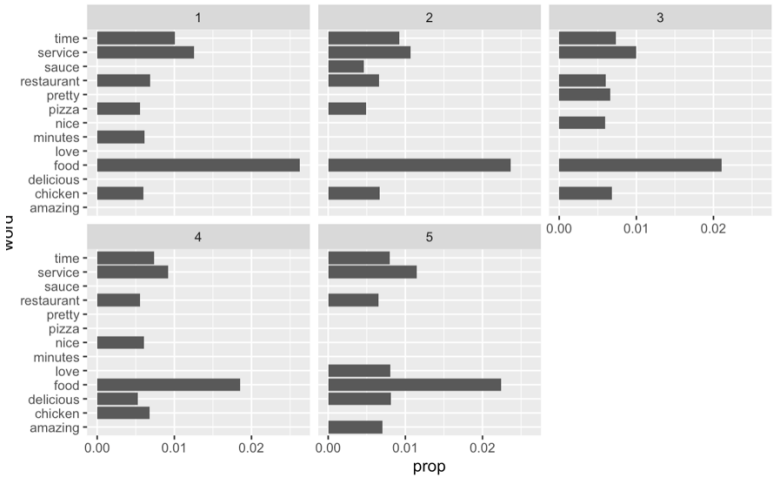
Our first tactic is to view the most frequent words in the document. When we look at this we notice very few of the top 10 words have any sentiment. As can be seen in the table below only 3 of the top 10 words have sentiment.

word <chr>	n <int>
food	25240
service	12380
time	9419
chicken	7700
restaurant	7245
nice	5969
pizza	5800
delicious	5571
love	5526
menu	5352

If we sort by star ratings and look at the top 15 words based on this, we begin to see more positive sentiment attributed to higher star rating and lower star rating attributed with more negative words. An example of this with a 5-star rating is shown below. The top 15 most common words with 5-star ratings contain no negative sentiment words and 7 positive sentiment words which make up 46% of the words. The rest of the words (8) contain no sentiment. The top 15 most frequent one-star rating words contain no positive sentiment words and 3 negative sentiment words. The rest of the words show no sentiment.

	stars	word	n		stars	word	n
1	5	food	6084	26674	1	food	2048
2	5	service	3931	26675	1	service	1254
3	5	love	3083	26676	1	time	1106
4	5	time	3006	26677	1	bad	1064
5	5	delicious	2617	26678	1	eat	865
6	5	restaurant	2295	26679	1	restaurant	723
7	5	amaze	2263	26680	1	wait	706
8	5	friendly	2152	26681	1	taste	629
9	5	eat	2151	26682	1	minute	589
10	5	chicken	1688	26683	1	table	490
11	5	staff	1686	26684	1	chicken	477
12	5	price	1681	26685	1	tell	467
13	5	fresh	1666	26686	1	customer	465
14	5	nice	1649	26687	1	leave	461
15	5	recommend	1548	26688	1	drink	451

We can graph the most common words by the probability of them appearing per star rating. Below, we can see that of the top 13 words, 4 carry sentiment love, amazing, nice, and delicious. We can also see that the probability of these words appearing is only with 4- or 5-star ratings which, makes sense since they are positive sentiment words.



We really begin to see sentiment settle in with the star ratings when calculate the average star rating with each word. This calculation was performed on the top 25 positive and negative words attributed to star rating. Below we show the top 10 for positive and negative.

word <chr>	totWS <dbl>
alan	1.111019e-04
bullshit	1.031752e-04
coffe	1.067712e-04
disgust	9.535263e-05
disrespectful	9.534686e-05
dominoes	1.081424e-04
evidently	1.103690e-04
fax	1.055001e-04
neven	1.045465e-04
nnwas	1.083632e-04

word <chr>	totWS <dbl>
amazing	0.05152061
bar	0.04714207
cheese	0.05870116
chicken	0.09918876
delicious	0.07211654
eat	0.05590389
food	0.32299831
fresh	0.05544465
friendly	0.06158853
love	0.07185574

C.

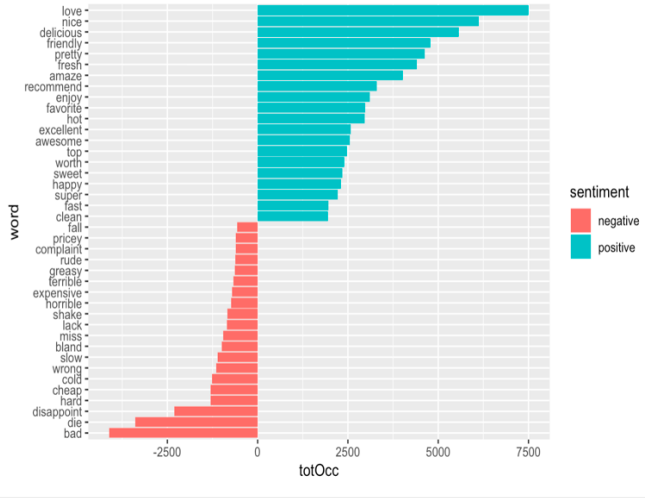
We will consider three dictionaries, available through the tidytext package – the NRC dictionary of terms denoting different sentiments, the extended sentiment lexicon developed by Prof Bing Liu, and the AFINN dictionary which includes words commonly used in user-generated content in the web. The first provides lists of words denoting different sentiment (for eg., positive, negative, joy, fear, anticipation, …), the second specifies lists of positive and negative words, while the third gives a list of words with each word being associated with a positivity score from -5 to +5. How many matching terms are there for each of the dictionaries? Consider using the dictionary based positive and negative terms to predict sentiment (positive or negative based on star rating) of a movie. One approach for this is: using each dictionary, obtain an aggregated positiveScore and a negativeScore for each review; for the AFINN dictionary, an aggregate positivity score can be obtained for each review. Are you able to predict review sentiment based on these aggregated scores, and how do they perform? Does any dictionary perform better?

The 3 dictionaries implemented for this assignment were bing, nrc, and afinn. The below table shows the amount of matching terms that each dictionary had. The right most column shows the words that matched with each dictionary and have positive or negative sentiment attributed to them.

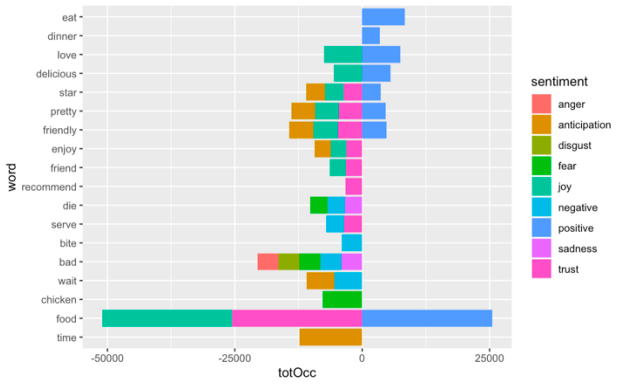
Dictionary	Words matched	Words matched with sentiment
Bing	170775	960
NRC	2902	2902
AFINN	136444	527

If we look into the sentiment words pulled with each dictionary, we will see that we finally begin to obtain more sentiment and it is separated by positive and negative words. Below are some of the top results obtained by each dictionary.

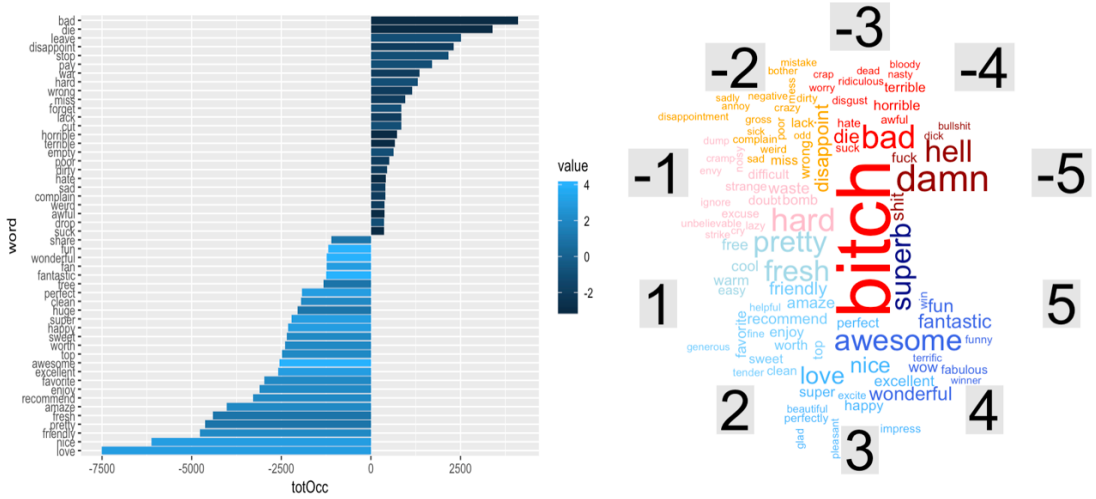
Bing:



NRC:



AFINN:



After ensuring that we have successfully split the positively and negatively attributed words we can begin to obtain sentiment scores attributed to star ratings.

One way to achieve positive or negative sentiment predictions based on the star rating is to take the average sentiment score based off of the stars received. For each of the tree dictionaries we have taken the average number of positive sentiment words (avgPos), negative sentiment words (avgNeg), and the average sentiment score (avgSentiSc) which is a combination of (average positive words / total number of words) – (average negative words/ total number of words). The following were our results per dictionary:

Bing:

With Bing we get 70.9% of 1-star sentiment resulting in negative sentiment. On the other end we get 81% of words as having positive sentiment for 5-star ratings. We can also note our avgSentiSc changes steadily from -.41 at 1 star to .62 with 5 stars.

stars <int>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.2907672	0.7092328	-0.4184657
2	0.4470575	0.5529425	-0.1058849
3	0.5963372	0.4036628	0.1926743
4	0.7365907	0.2634093	0.4731814
5	0.8106257	0.1893743	0.6212514

NRC:

As we can see below, NRC gives us very different results from bing. Nrc detected more positive words (20%) then negative words (17%) with one-star ratings. While in comparison we can see 5-star ratings had an average positive score of 31%. These significantly lower scores leave our avgSentiSc between 2% (at 1 stars) to 23% (at 5 stars) so as we can see the margins are not as large as with bing.

stars <int>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.2000620	0.17622989	0.02383213
2	0.2418033	0.15319448	0.08860885
3	0.2691100	0.12546248	0.14364756
4	0.2997070	0.09630592	0.20340105
5	0.3107632	0.07830052	0.23246266

AFINN:

Due to the scale used with AFINN in comparison to bing and NRC we took a different approach. Our avgSenti column represents the average score that each star rating got on the numerical scale AFINN runs on. As we can see, the lower the star rating the lower the avgSenti and the higher the star rating the higher the avgSenti.

stars <int>	avgLen <dbl>	avgSenti <dbl>
1	3.990048	-2.3485745
2	4.296949	0.7093426
3	4.323694	3.1575313
4	4.240190	5.5405347
5	3.968063	6.4632035

We also ran our models through a confusion matrix to measure the performance and obtained the following results.

Bing:

```
       predicted
actual    -1     1
    -1  5384  1654
    1   3758 18211
```

NRC:

```
       predicted
actual    -1     1
    -1  3165  4164
    1   2914 19640
```

AFINN:

```
      predicted
actual    -1     1
    -1  4422  2475
    1   2686 18833
```

Dictionary	Accuracy
Bing	.813
NRC	.763
AFINN	.818

Thus, AFINN is our most accurate model with an accuracy of approximately 82%.

D. Develop models to predict review sentiment. For this, split the data randomly into training and test sets. To make run times manageable, you may take a smaller sample of reviews (minimum should be 10,000). One may seek a model built using only the terms matching any or all of the sentiment dictionaries, or by using a broader list of terms (the idea here being, maybe words other than only the dictionary terms can be useful). You should develop at least three different types of models (Naïve Bayes, and two others of your choice ….Lasso logistic regression (why Lasso?), SVM, random forest,…?)

(i) Develop models using only the sentiment dictionary terms – try the three different dictionaries; how do the dictionaries compare in terms of predictive performance for rating ? Then with a combination of the three dictionaries, ie. combine all dictionary terms. Do you use term frequency, tfidf, or other measures, and why? What is the size of the documentterm matrix? Should you use stemming when using the dictionaries?

Stemming is a great tool to have with or without dictionaries. This is because it cuts the words down to the roots. This way words with similar meanings/ definitions will be grouped together. An example of this in terms of sentiment for the purposes of this project would be “happier” & “happiest” just being cut down and grouped with the word “happy”.

Below are our results from running the model through all 3 of our original dictionaries (Bing, NRC, AFINN).

Dictionary	Model	AUC
Bing	Random Forest	.91
	Ridge Regression	.95
	Naïve Bayes	.729
NRC	Random Forest	.809
	Ridge Regression	.79
	Naïve Bayes	1.07
AFINN	Random Forest	.896
	Ridge Regression	.91
	Naïve Bayes	.737

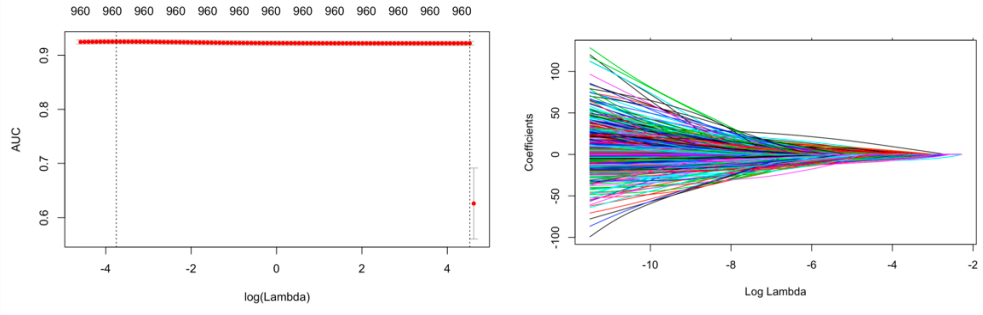
As we can see our best model is ran through the bing dictionary using the ridge regression. This model gives us an AUC of .95. The model was ran on tf-idf because it is an excellent measure for reflecting how important the words are to the model.

Bing:

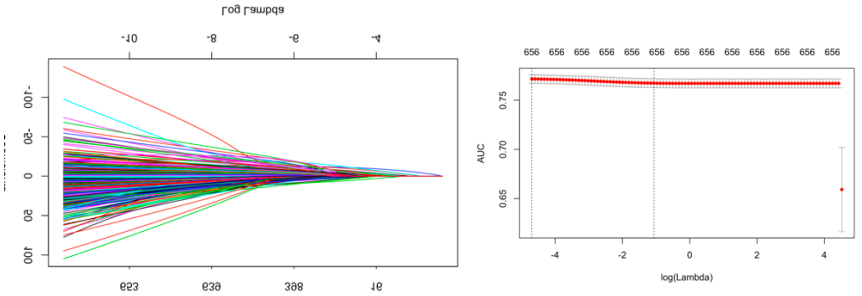
```
Call:
 ranger(dependent.variable.name = "hiLo", data = revDTM_sentiBing_trn %>%      select(-review_id), num.trees =
500, importance = "permutation",      probability = TRUE)

Type:                             Probability estimation
Number of trees:                  500
Sample size:                      14504
Number of independent variables:  960
Mtry:                             30
Target node size:                 10
Variable importance mode:         permutation
Splitrule:                        gini
OOB prediction error (Brier s.):  0.09080432
```

```
Area under the curve: 0.9887
Area under the curve: 0.9188
      preds
actual FALSE  TRUE
    -1  3046   461
    1    136 10861
      preds
actual FALSE  TRUE
    -1  2325  1206
    1    502 10470
```



NRC:



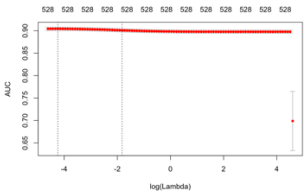

```
Call:
 ranger(dependent.variable.name = "hiLo", data = revDTM_sentiNRC_trn %>%
500, importance = "permutation",      probability = TRUE)

Type:                             Probability estimation
Number of trees:                  500
Sample size:                      14942
Number of independent variables:  806
Mtry:                             28
Target node size:                 10
Variable importance mode:         permutation
Splitrule:                        gini
OOB prediction error (Brier s.):  0.1476163
```

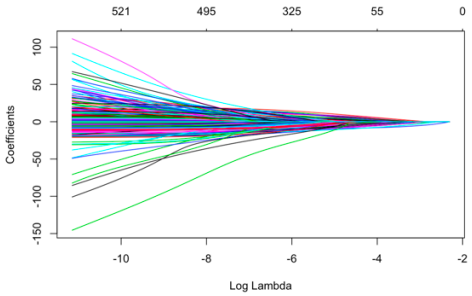
```
[1] 1.068529
[1] 1.079719
```

AFINN:

```
Setting levels: control = -1, case = 1
Setting direction: controls < cases
Area under the curve: 0.733
Setting levels: control = -1, case = 1
Setting direction: controls < cases
Area under the curve: 0.7379
```




```
Setting levels: control = -1, case = 1
Setting direction: controls > cases
Area under the curve: 0.9814
Setting levels: control = -1, case = 1
Setting direction: controls > cases
Area under the curve: 0.8963
      preds
actual FALSE  TRUE
    -1   593  2849
    1  10579   187
      preds
actual FALSE  TRUE
    -1  1372  2083
    1  10163   590
```



(ii)

Develop models using a broader list of terms – how do you obtain these terms? Will you use stemming here? Report on performance of the models. Compare performance with that in part (c) above.

For the final model we began by removing words that occur in more than 90% of reviews and less then 30% as well. We did a pivot_wider on this data and got a table with the dimensions of 35310, 3362. We also removed all reviews with a 3-star rating. Lastly, all NA's were replaced with 0 and a random forest model was ran. This run of data gave an AUC of 1.33 on the testing data. This of course is outside of the given AUC score parameters so we can compare our models in C to this by using the confusion matrix accuracy. From running the entire data set we got an accuracy of .889

```
[1] 1.344737
[1] 1.331534
      preds
actual FALSE  TRUE
    -1  3600    32
    1      7 11527
      preds
actual FALSE  TRUE
    -1  2381  1409
    1    289 11087
```

In comparison with part C we can say that running the model on the entire data set is worth it in terms of cost because it improved the accuracy of the model by .071.