Yelp: Exploratory Analysis

Ryan Kobler

11/20/2019

Packages:

```
library(jsonlite)
library(tidytext)
library(stringr)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(ggplot2)
```

Load in the data:

```
#path <- "/Users/ryankobler/Downloads/yelp_review.csv"
#yelp <- read.csv(path)

# Take sample of the data
yelp %>% sample_frac(0.01)
# Save sample for easy access
write.csv(df, "yelp-train.csv")

# Take sample of the data
yelp_sample <- yelp_review %>% sample_frac(0.01)
# Save sample for easy access
write.csv(yelp_sample, "yelp-train.csv")

# Load training data
yelp_train <- read.csv("yelp-train.csv")
yelp_sample <- yelp_train
```

Clean

Count total number of words

```r
#count the number of words in the review
yelp_sample <- yelp_train
yelp_sample <- mutate(yelp_sample, numwords = str_count(yelp_sample$text, " "))
#univariate analysis of size
ggplot(data = yelp_sample, aes(x = numwords)) + geom_bar()
#analysis of size vs star rating with locally weighted polynomial
ggplot(data = yelp_sample, aes(x = numwords, y = stars)) + geom_jitter(size = 0.25) + geom_smooth()
```

Define functions to extract features:

This method of feature analysis draws from the tidytext package and resource: https://www.tidytextmining.com/sentiment.html

```r
# Note: this function requires the tidytext package & drops all
# words that do not convey sentiment
dropStopwords <- function(string){
  # Remove all punctuation except apostrophes & replace with " "
  noPunc <- gsub("[^[:alnum:][:space:]']", " ", string)
  # Split the larger string by space using strsplit()
  splitBySpace <- unlist(strsplit(noPunc, split = " "))
  # Remove missing chunks
  splitBySpace <- splitBySpace[splitBySpace != ""]
  todrop <- get_stopwords() # query dictionary of stopwords
  todrop <- todrop[[1]]

  # remove stop words and wrap as lowercase
  tolower(splitBySpace[!splitBySpace %in% todrop])

}

# Function below takes sentiment string in nrc and a sequence of
# pruned words associated with 1 review
nrcSentimentCount <- function(senti, text){
  sentiment <- nrc %>%
  filter(sentiment == senti) %>%
  select(word)

  # outputs a count of the number of "trues"
  sum(unlist(text) %in% unlist(sentiment))
}

# Remove the stop words and save in new column
#yelp_train$prunedtext <- lapply(yelp_train$text, FUN = dropStopwords)


get_sentiments("nrc") # we chose this one for now, can consider adding
```

```
## # A tibble: 13,901 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # ... with 13,891 more rows
```

```
# other sentiment lexicons to increase our # of predictors

get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

```
get_sentiments("loughran")
```

```
## # A tibble: 4,150 x 2
##    word         sentiment
##    <chr>        <chr>
##  1 abandon      negative
##  2 abandoned    negative
##  3 abandoning   negative
##  4 abandonment  negative
##  5 abandonments negative
##  6 abandons     negative
##  7 abdicated    negative
##  8 abdicates    negative
##  9 abdicating   negative
## 10 abdication   negative
## # ... with 4,140 more rows
```

Testing on tiny data set

```r
yelp.small <- yelp_train[1:3,]
yelp.small$prunedtext <- lapply(yelp.small$text, FUN = dropStopwords)

yelp.small$prunedtext

# Matching sentiments from the NRC dictionary
nrc$sentiment <- as.factor(nrc$sentiment)

yelp.small %>%
  mutate(joy = nrcSentimentCount("fear", prunedtext))

# Grab all of the sentiments from for testing
sentiments <- levels(as.factor(nrc$sentiment))

# Generate new columns that count the number of times each sentiment word appears
yelp.small$joy <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "joy")
yelp.small$anger <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "anger")
yelp.small$fear <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "fear")
yelp.small$positive <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "positive")
yelp.small$negative <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "negative")
yelp.small$surprise <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "surprise")
yelp.small$disgust <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "disgust")
yelp.small$trust <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "trust")
yelp.small$anticipation <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "anticipation")
yelp.small$anger <- sapply(yelp.small$prunedtext, nrcSentimentCount, senti = "anger")
```

Apply nrcSentimentCount function to the full training data set

Q: Is there an easier/cleaner way to create all of these columns?

```r
# Generate new column called prunedtext
yelp_train$prunedtext <- lapply(yelp_train$text, FUN = dropStopwords)
# This column gives us a little trickiness if we try to export as .csv
class(yelp_train$prunedtext) # note that this column is list of lists

yelp_train$joy <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "joy")
yelp_train$anger <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "anger")
yelp_train$fear <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "fear")
yelp_train$positive <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "positive")
yelp_train$negative <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "negative")
yelp_train$surprise <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "surprise")
yelp_train$disgust <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "disgust")
yelp_train$trust <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "trust")
yelp_train$anticipation <- sapply(yelp_train$prunedtext, nrcSentimentCount, senti = "anticipation")
```

Convert columns to numbers and normalize by length of review

Note that as.numeric is no longer explicitly necessary because we've used sapply instead of lapply to generate the sentiment counts. But it is not harmful.

```r
# Add column that counts the number of total words
yelp_train$nwords <- str_count(yelp_train$text, " ")

# Generate the ratios
yelp_train$joyratio <- as.numeric(yelp_train$joy)/(yelp_train$nwords)
yelp_train$angerratio <-  as.numeric(yelp_train$anger)/(yelp_train$nwords)
yelp_train$fearratio <-  as.numeric(yelp_train$fear)/(yelp_train$nwords)
yelp_train$positiveratio <-  as.numeric(yelp_train$positive)/(yelp_train$nwords)
yelp_train$negativeratio <-  as.numeric(yelp_train$negative)/(yelp_train$nwords)
yelp_train$surpriseratio <-  as.numeric(yelp_train$surprise)/(yelp_train$nwords)
yelp_train$disgustratio <-  as.numeric(yelp_train$disgust)/(yelp_train$nwords)
yelp_train$trustratio <-  as.numeric(yelp_train$trust)/(yelp_train$nwords)
yelp_train$anticipationratio <-  as.numeric(yelp_train$anticipation)/(yelp_train$nwords)
```

Missingness:

In this stage of the process, we found that some of our data was missing in that some reviews had no words detected. The problem came down to review language, so we use the `textcat` package to identify the language of the review and dplyr to filter out the non-English reviews.

This can be thought of as missingness in that our results only represent Yelp reviewers writing in English.

```r
yelp_train %>%
  filter(nwords==0)

# This will take a while to run --> do overnight
# partition the data to see where the runtime issue is... the 3000 observation
# samples run in under a minute.
n <- nrow(yelp_train)
yelp_train$language <- NA
yelp_train[1:1000,]$language <- textcat(yelp_train[1:1000,]$text)
yelp_train[1000:3000,]$language <- textcat(yelp_train[1000:3000,]$text)
yelp_train[3000:6000,]$language <- textcat(yelp_train[3000:6000,]$text)
yelp_train[6001:9000,]$language <- textcat(yelp_train[6001:9000,]$text)
yelp_train[9000:40000,]$language <- textcat(yelp_train[9000:40000,]$text)
yelp_train[40000:n,]$language <- textcat(yelp_train[40000:n,]$text)

# save languages in another csv
langs <- yelp_train %>%
  select(X1, review_id, user_id, business_id, stars, language)

# writing only id vars and languages
write.csv(langs, "DATA/withlanguage.csv")

# Remove non-English reviews
yelp_train_en <- yelp_train %>%
  filter(language == "english" | language == "scots")

# Dropped 671 observations
nrow(yelp_train)
nrow(yelp_train_en)
```

Write yelp_train data frame to a csv

Save in /DATA so we don't have to run everything again. We remove the `prunedtext` column because we're guessing that the csv filetype cannot handle list type entries.

```r
write.csv(yelp_train %>% select(-prunedtext), "DATA/withlanguage.csv")
write.csv(yelp_train[, -12], "yelp-train3.csv")
```

For knitting to pdf

Because it would take forever for this to knit, we import the already clean data set now and visualize below!

```r
yelp_train <- read.csv("DATA/withlanguage.csv")
yelp_train$prunedtext <- lapply(yelp_train$text, FUN = dropStopwords)

yelp_train <- yelp_train %>%
  filter(language == "english" | language == "scots")
```

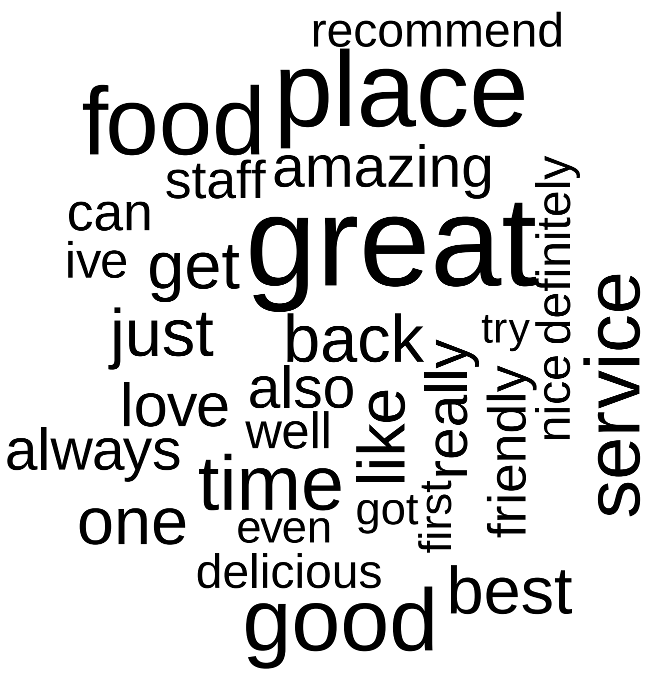
Generate word clouds for each rating level

```r
# Save individual words from the X-star reviews as the vector wordsX
words4 <- yelp_train %>%
  filter(stars == 4) %>%
  pull(prunedtext)

words5 <- yelp_train %>%
  filter(stars == 5) %>%
  pull(prunedtext)

words1 <- yelp_train %>%
  filter(stars == 1) %>%
  pull(prunedtext)

# we unlist() the above vectors to get big pile o'
# words across all reviews since wordcloud() takes character vectors as input,
# not lists
wordcloud(unlist(words5), min.freq = 10, max.words = 30)
```

```
## Loading required namespace: tm
```

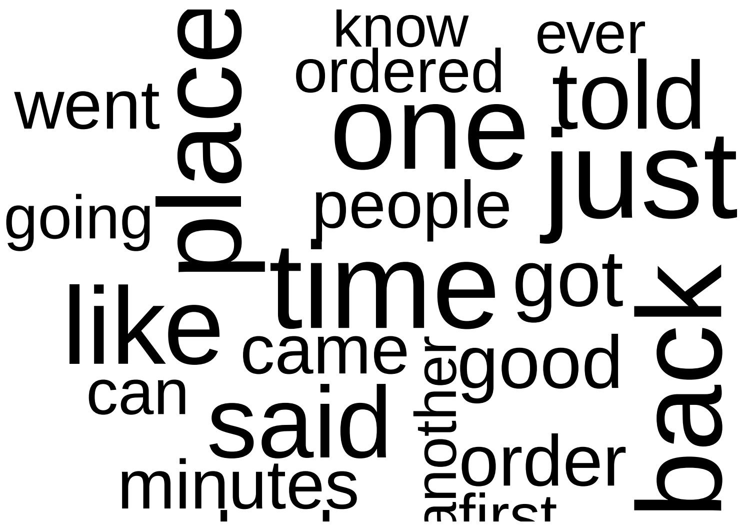
```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
## transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```



```
wordcloud(unlist(words1), min.freq = 10, max.words = 30)
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
## transformation drops documents

## Warning in wordcloud(unlist(words1), min.freq = 10, max.words = 30): food
## could not be fit on page. It will not be plotted.

## Warning in wordcloud(unlist(words1), min.freq = 10, max.words = 30):
## service could not be fit on page. It will not be plotted.
```

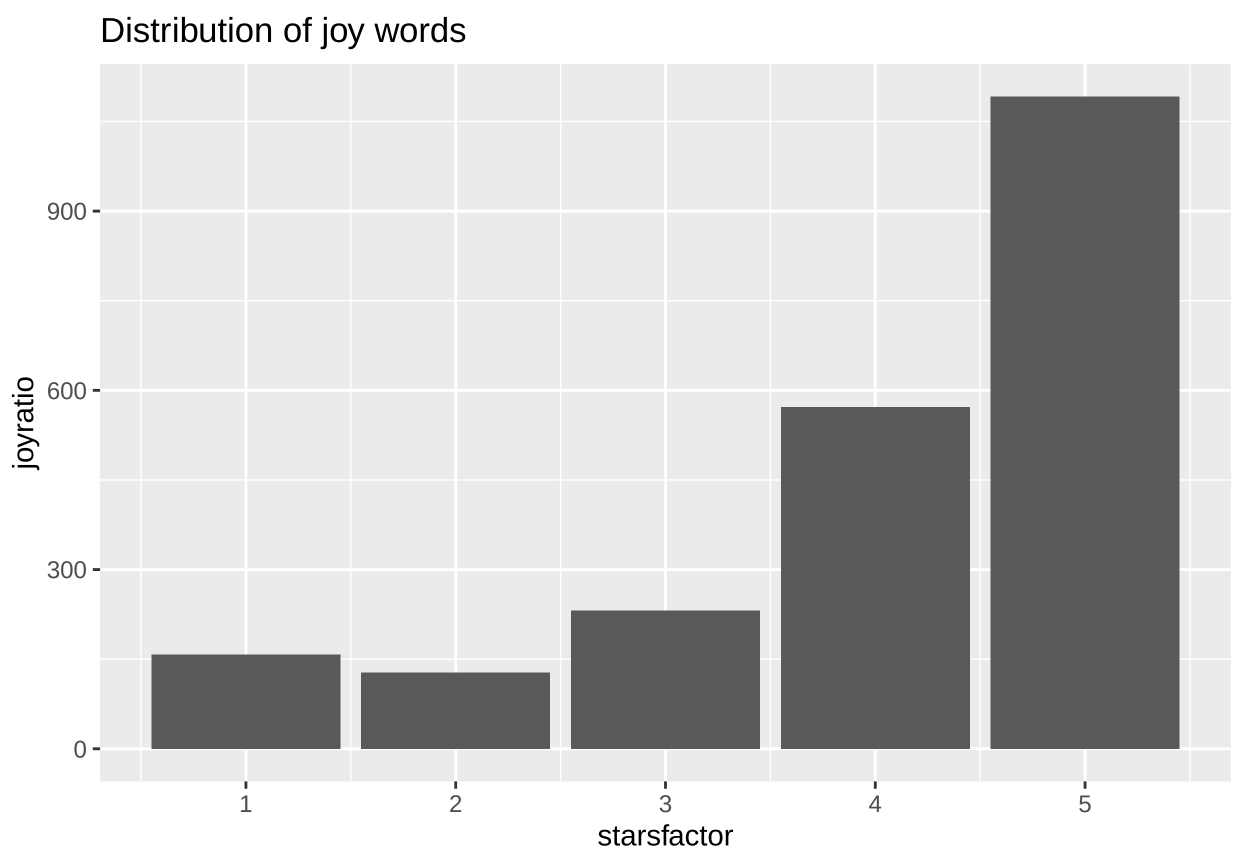


Idea: could think about coloring the words by sentiment.

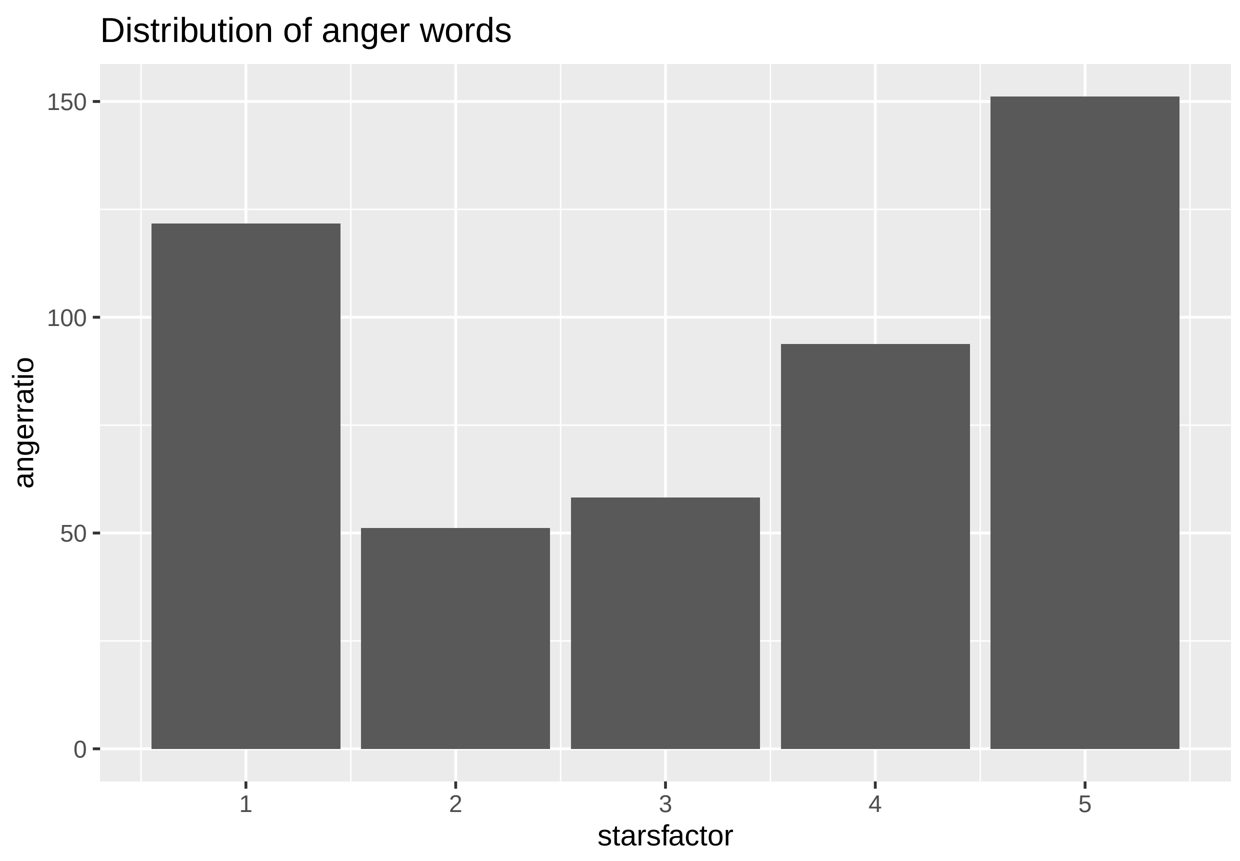
Histograms of Sentiments

Bar chart of ratio of 'joy' to total number of words.

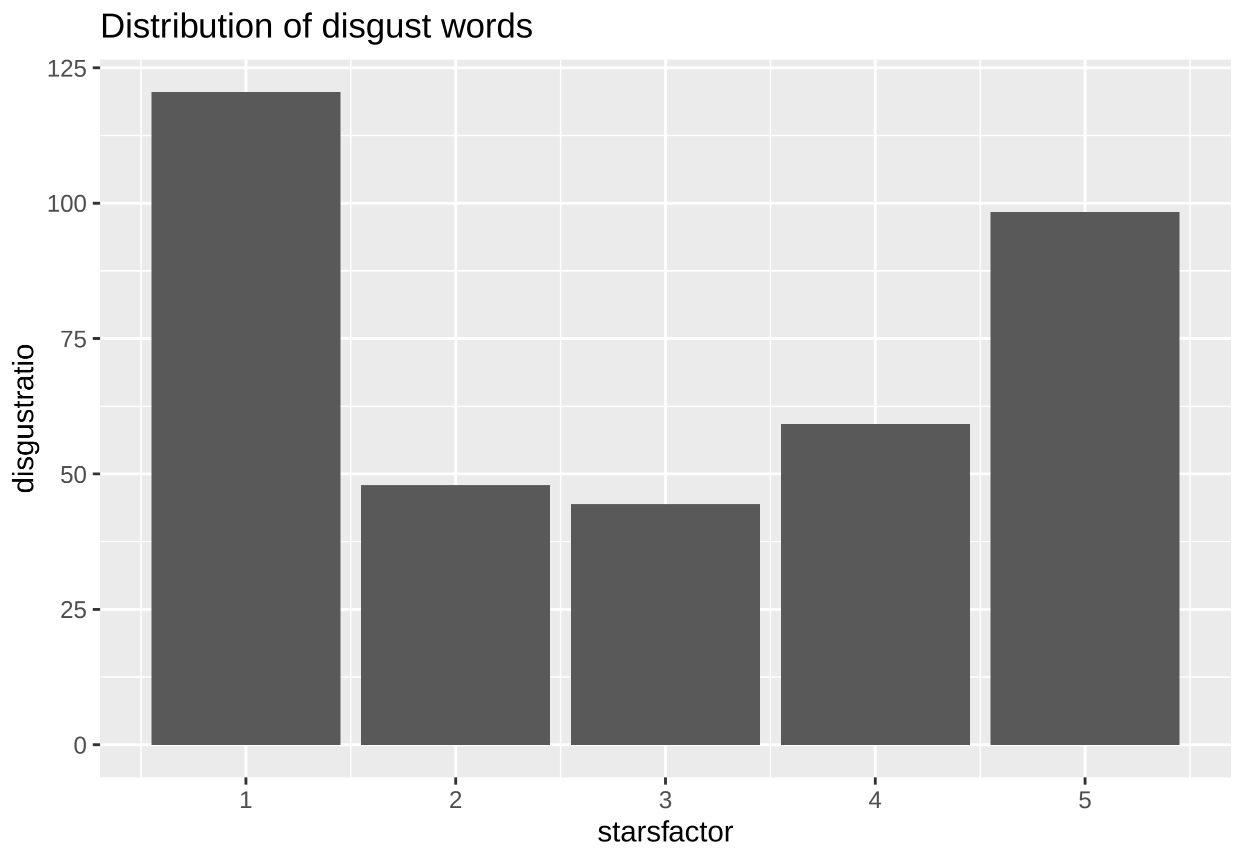
```
# Joy
ggplot(yelp_train, aes(starsfactor, joyratio)) +
  geom_bar(stat = "identity") +
  ggtitle("Distribution of joy words")
```



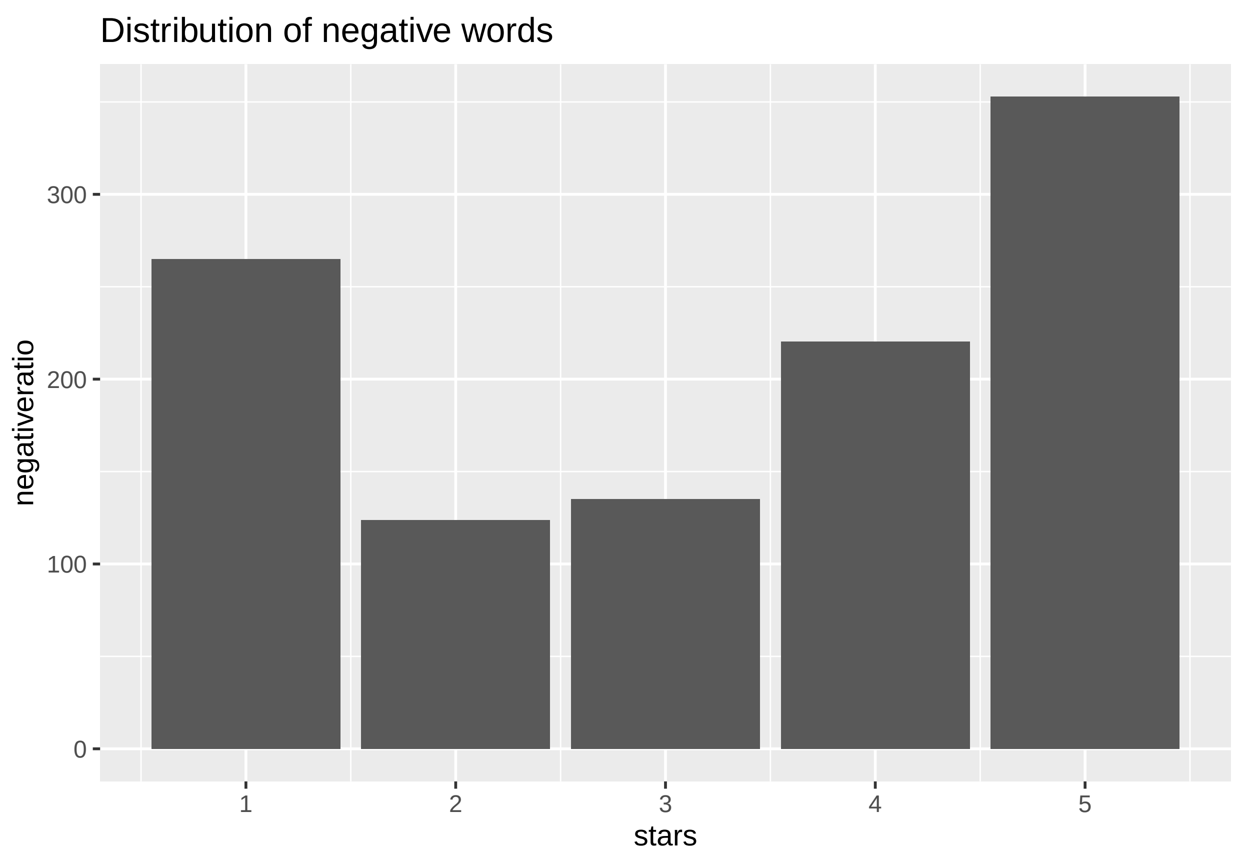
```r
# Anger
ggplot(yelp_train, aes(starsfactor, angerratio)) +
  geom_bar(stat = "identity") +
  ggtitle("Distribution of anger words")
```



```
# Disgust
ggplot(yelp_train, aes(starsfactor, disgustratio)) +
  geom_bar(stat = "identity") +
  ggtitle("Distribution of disgust words")
```



```r
# Negativity
ggplot(yelp_train, aes(stars, negativeratio)) +
  geom_bar(stat = "identity") +
  ggtitle("Distribution of negative words")
```



Data Description

The training data set, which is a 1% random sample of the entire yelp_review universe, is made up of 52,616 observations and 19 variables. Each observation corresponds to a review from a random business (businesses sampled with replacement). We use the NRC lexicon/dictionary to categorize all the non-stopwords in the reviews as anger, anticipation, disgust, fear, negative, positive, sadness, surprise, and trust. This was an arbitrary choice, for there are several other lexicons such as "loughran" and "afinn" both of which rank and categorize the sentiment of words differently.

The useful predictors include: - `joy`: number of joy-categorized words that appear in the body of the review (after removing stop words). And each of our variables named in this way follow the example given above.

We also include ratios for each, so `joyratio` is the total number of joy words divided by the total number of non-stop words to avoid priviledging length.

```
# Glimpse the data set to examine the predictors
glimpse(yelp_train)
```

```
## Observations: 51,945
## Variables: 33
## $ X                 <int> 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
## $ X1                <int> 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
## $ review_id         <fct> lOc-gTpyEHIBeH6qG7iekw, 864AWE6eqipNE7SPVYSG...
## $ user_id           <fct> hyMa8iiLndcfqF3qJdN67w, cMEtAiW60I5wE_vLfTxo...
## $ business_id       <fct> grZEbAsZwWA3yJMwDRlONw, LR1JNvpNrx2N4HNAyhHv...
## $ stars             <int> 4, 5, 2, 1, 2, 4, 3, 2, 5, 5, 4, 5, 3, 5, 5,...
## $ date              <fct> 2011-02-20, 2014-08-11, 2016-12-14, 2017-03-...
## $ text              <fct> "I only came here for lunch so I can only sp...
## $ useful            <int> 3, 3, 0, 0, 0, 4, 2, 0, 0, 2, 2, 1, 0, 22, 0...
## $ funny             <int> 0, 2, 0, 0, 2, 5, 0, 0, 0, 0, 1, 0, 0, 14, 0...
## $ cool              <int> 2, 2, 0, 0, 0, 6, 0, 0, 0, 0, 1, 1, 0, 23, 0...
## $ joy               <int> 4, 6, 2, 3, 5, 6, 6, 8, 4, 11, 4, 9, 5, 3, 2...
## $ starsfactor       <int> 4, 5, 2, 1, 2, 4, 3, 2, 5, 5, 4, 5, 3, 5, 5,...
## $ anger             <int> 0, 1, 2, 1, 5, 1, 5, 2, 0, 2, 0, 1, 1, 0, 0,...
## $ nwords            <int> 75, 94, 50, 412, 231, 80, 192, 248, 18, 340,...
## $ joyratio          <dbl> 0.053333333, 0.063829787, 0.040000000, 0.007...
## $ disgust           <int> 0, 1, 2, 1, 3, 2, 4, 1, 0, 3, 0, 0, 3, 0, 0,...
## $ negative          <int> 0, 2, 3, 14, 6, 1, 5, 4, 1, 5, 2, 2, 4, 0, 2...
## $ positive          <int> 5, 12, 2, 18, 10, 8, 8, 12, 5, 20, 7, 10, 9,...
## $ angerratio        <dbl> 0.000000000, 0.010638298, 0.040000000, 0.002...
## $ language          <fct> english, english, english, english, english,...
## $ fear              <int> 0, 1, 1, 2, 3, 1, 1, 2, 1, 3, 1, 2, 4, 0, 2,...
## $ surprise          <int> 2, 2, 0, 1, 2, 4, 4, 6, 1, 7, 1, 2, 3, 0, 2,...
## $ trust             <int> 4, 4, 2, 13, 5, 6, 6, 8, 5, 15, 3, 6, 4, 4, ...
## $ anticipation      <int> 1, 5, 1, 17, 5, 5, 7, 5, 1, 14, 3, 3, 5, 3, ...
## $ fearratio         <dbl> 0.000000000, 0.010638298, 0.020000000, 0.004...
## $ positiveratio     <dbl> 0.06666667, 0.12765957, 0.04000000, 0.043689...
## $ negativeratio     <dbl> 0.000000000, 0.021276596, 0.060000000, 0.033...
## $ surpriseratio     <dbl> 0.026666667, 0.021276596, 0.000000000, 0.002...
## $ disgustratio      <dbl> 0.000000000, 0.010638298, 0.040000000, 0.002...
## $ trustratio        <dbl> 0.053333333, 0.042553191, 0.040000000, 0.031...
## $ anticipationratio <dbl> 0.013333333, 0.053191489, 0.020000000, 0.041...
## $ prunedtext        <list> [<"i", "came", "lunch", "i", "can", "speak"...
```

Source: https://www.kaggle.com/yelp-dataset/yelp-dataset/version/6#yelp_review.csv

Univariate analysis of the response

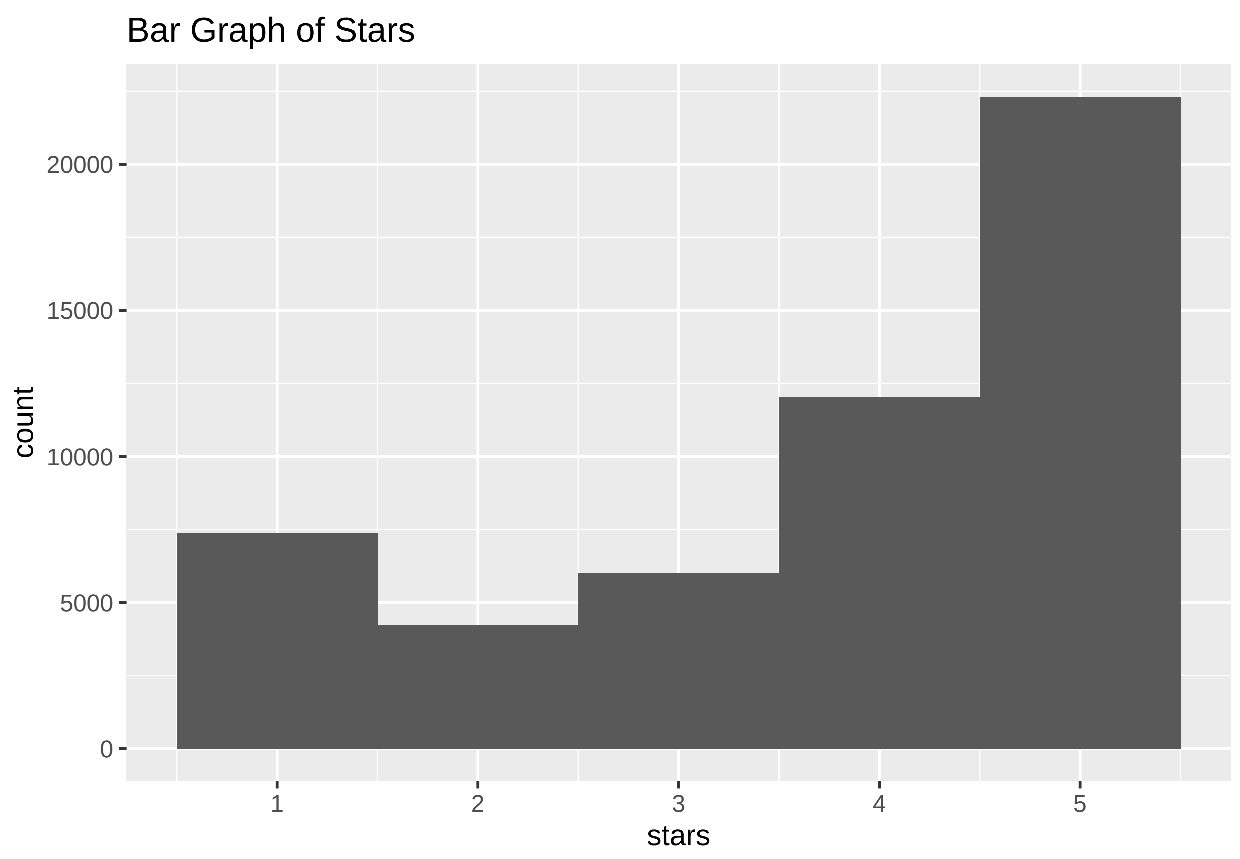
```r
mean(yelp_train$stars)
```

```
## [1] 3.725575
```

```r
var(yelp_train$stars)
```

```
## [1] 2.072406
```

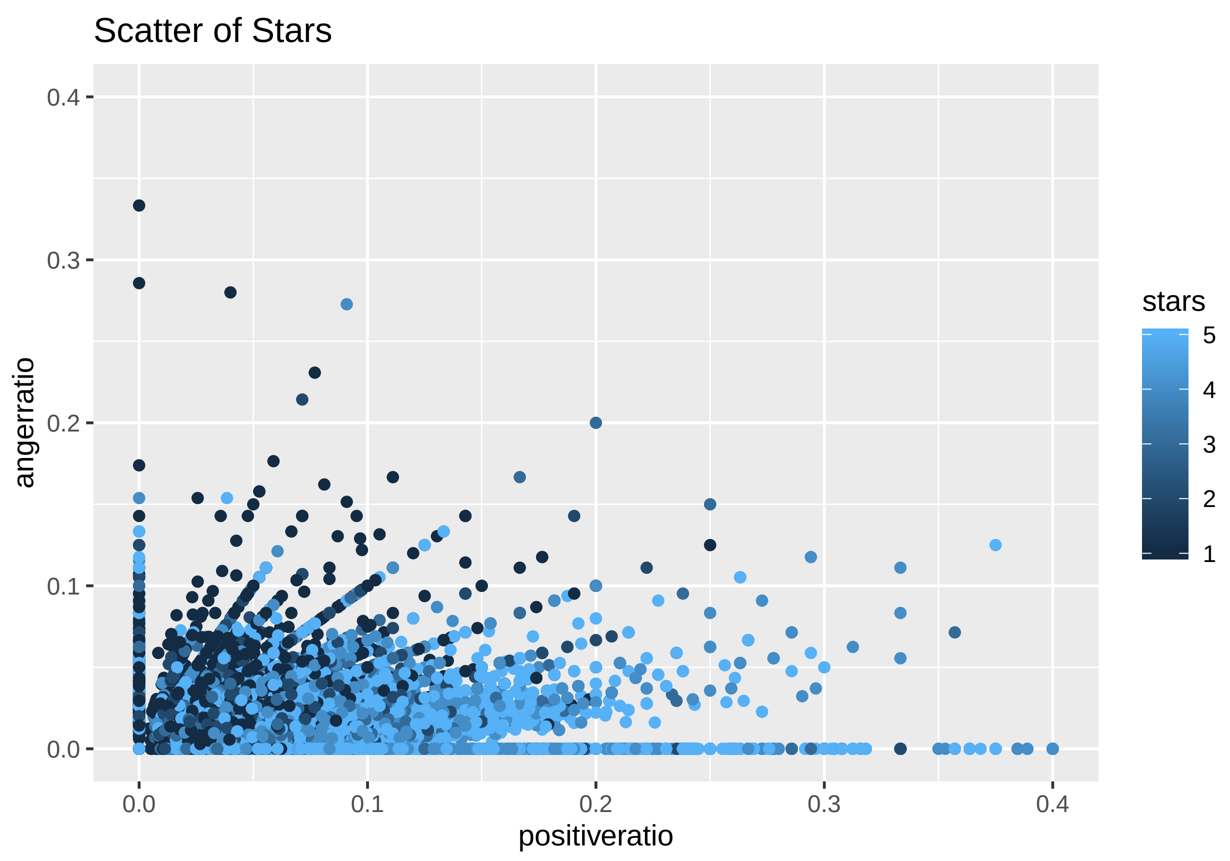
```r
ggplot(yelp_train, aes(x=stars)) +
  geom_histogram(binwidth=1) +
  ggtitle("Bar Graph of Stars")
```



The mean number of stars in our sample is around 3.726 and the variance is around 2.067. We can see that the distribution of stars is somewhat oddly shaped with lots of five and four star review and a fair number of 1 star reviews.

```
ggplot(yelp_train, aes(x=positiveratio, y = angerratio)) +
  geom_point(aes(color = stars)) +
  ggtitle("Scatter of Stars") + xlim(x = 0,.4) + ylim(y = 0, .4)
```

```
## Warning: Removed 29 rows containing missing values (geom_point).
```



It appears from the

Bivariate/Trivariate Graphs