Yelp Data

STAT 333 2019 Fall

Pixu Shi, Department of Statistics, University of Wisconsin - Madison

---

Learning Objectives

1. We'll go through the Yelp data for your final project
2. I'll show you some R skills you might need for this project
3. I'll show you what the benchmark model is
4. I'll show you how to submit your results to Kaggle

A couple of disclosures

1. There is no "right" way to do this analysis. But, there are definitely "wrong" ways to analyze this data.
2. I am not an expert in natural language processing (NLP) and some of you may have a better understanding of NLP than I do.
3. I am limiting my statistical analysis with the tools that I taught you.

First Look at the Data and Data Clean up

```
# Read data and clean up some R formatting issues
yelp <- read.csv("Yelp_train.csv")
yelp_test <- read.csv("Yelp_test.csv")
yelp_validate <- read.csv("Yelp_validate.csv")
yelp_out <- rbind(yelp_test,yelp_validate)
str(yelp)
```

```
## 'data.frame':    55342 obs. of  114 variables:
##  $ X            : int  2677 4349 6109 6302 7224 13664 13743 16193 17312 19712 ...
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ stars        : int  4 4 5 5 5 4 2 1 2 3 ...
##  $ name         : Factor w/ 1361 levels "1847 At the Stamm House",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ text         : Factor w/ 55285 levels "- $9 pitchers of locally brewed beer\n- Gor
geous view of Lake Mendota\n- Best people watching in all of Madison"| __truncated__,..:
52288 12700 4861 50909 13741 11279 47234 1928 24348 26225 ...
##  $ date         : Factor w/ 55295 levels "2005-03-01 22:56:23",..: 38282 51025 28948
42322 45630 54597 44585 29598 27736 43657 ...
##  $ useful       : int  0 0 5 2 0 0 0 0 2 0 ...
##  $ funny        : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ cool         : int  0 1 1 0 0 0 0 0 0 0 ...
##  $ city         : Factor w/ 24 levels "Belleville","Black Earth",..: 13 13 13 13 13 1
3 13 13 13 13 ...
##  $ nchar        : int  604 132 1293 805 152 124 2764 768 468 2447 ...
##  $ nword        : int  113 26 238 142 31 25 497 143 92 473 ...
##  $ sentiment    : num  2.8 2 1.8 2 2.5 ...
##  $ categories   : Factor w/ 1288 levels "Acai Bowls, Salad, Sandwiches, Restaurants,
Juice Bars & Smoothies, Food",..: 1235 1235 1235 1235 1235 1235 1235 1235 1235 1235 ...
##  $ gem          : int  0 0 1 0 0 0 1 0 0 0 ...
##  $ incredible   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ perfection   : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ heaven       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ phenomenal   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ divine       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ die          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ deliciously  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ highly       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ heavenly     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ superb       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ amazing      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ favorites    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ delectable   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ perfect      : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ sourced      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ knowledgeable: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ wonderfully  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ deliciousness: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ knowledgable : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fantastic    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ adorable     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ wonderful    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fabulous     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ scrumptious  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hidden       : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ notch        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ favorite     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ disappoint   : int  0 0 0 0 0 0 1 1 0 0 ...
##  $ watering     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ delightful   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ yum          : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ outstanding  : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ awesome      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ everyday     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ relaxed      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dream        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ amazingly    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ delicious    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ affordable   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ scones       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ excellent    : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ rotating     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ paired       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ secret       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ beautifully  : int  0 0 0 0 0 0 1 1 0 0 ...
##  $ terrace      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ skeptical    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ perfectly    : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ delish       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ supposed     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mcdonald.s   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ undercooked  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ management   : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ sucks        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ flag         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ messed       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ unpleasant   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dirty        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mediocre     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ manager      : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ burned       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nope         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ upset        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zero         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ stale        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ screw        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ subpar       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ inattentive  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ edible       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ complained   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ rubbery      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lukewarm     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ garbage      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ contact      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ flavorless   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poor         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ apologize    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ gross        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ charged      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ receipt      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ worse        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ignored      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ response     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poorly       : int  0 0 0 0 0 0 0 0 0 0 ...
##   [list output truncated]
```

```
# Some basic data cleaning
# get rid of the first column, which is the original sample ID
yelp <- yelp[,-1]
yelp_out <- yelp_out[,-1]

# convert text into actual strings
yelp$text <- as.character(yelp$text)
yelp_out$text <- as.character(yelp_out$text)
yelp$categories <- as.character(yelp$categories)
yelp_out$categories <- as.character(yelp_out$categories)

# Refactorize yelp_out city and restaurant names after binding
yelp_out$name <- as.character(yelp_out$name)
yelp_out$city <- as.character(yelp_out$city)
yelp_out$city <- factor(yelp_out$city)

# Fix date variable into actual dates
yelp$date <- as.Date(yelp$date)
yelp_out$date <- as.Date(yelp_out$date)
```

```
view_vars <- c("Id","stars","name","nchar","nword","text")
# The first 3 reviews
yelp[1:3,view_vars]
```

	Id	stars	name	nchar	nword
	<int>	<int>	<fctr>	<int>	<int>
1	1	4	1847 At the Stamm House	604	113
2	2	4	1847 At the Stamm House	132	26
3	3	5	1847 At the Stamm House	1293	238

3 rows | 1-6 of 7 columns

```
# randomly view 3 one star reviews
yelp[sample(which(yelp$stars == 1),3), view_vars]
```

	Id	stars	name	nchar	nword
	<int>	<int>	<fctr>	<int>	<int>
41361	41361	1	Salads UP	603	110
39948	39948	1	Restaurant Muramoto	351	63
43540	43540	1	Stalzy's Deli	476	91

3 rows | 1-6 of 7 columns

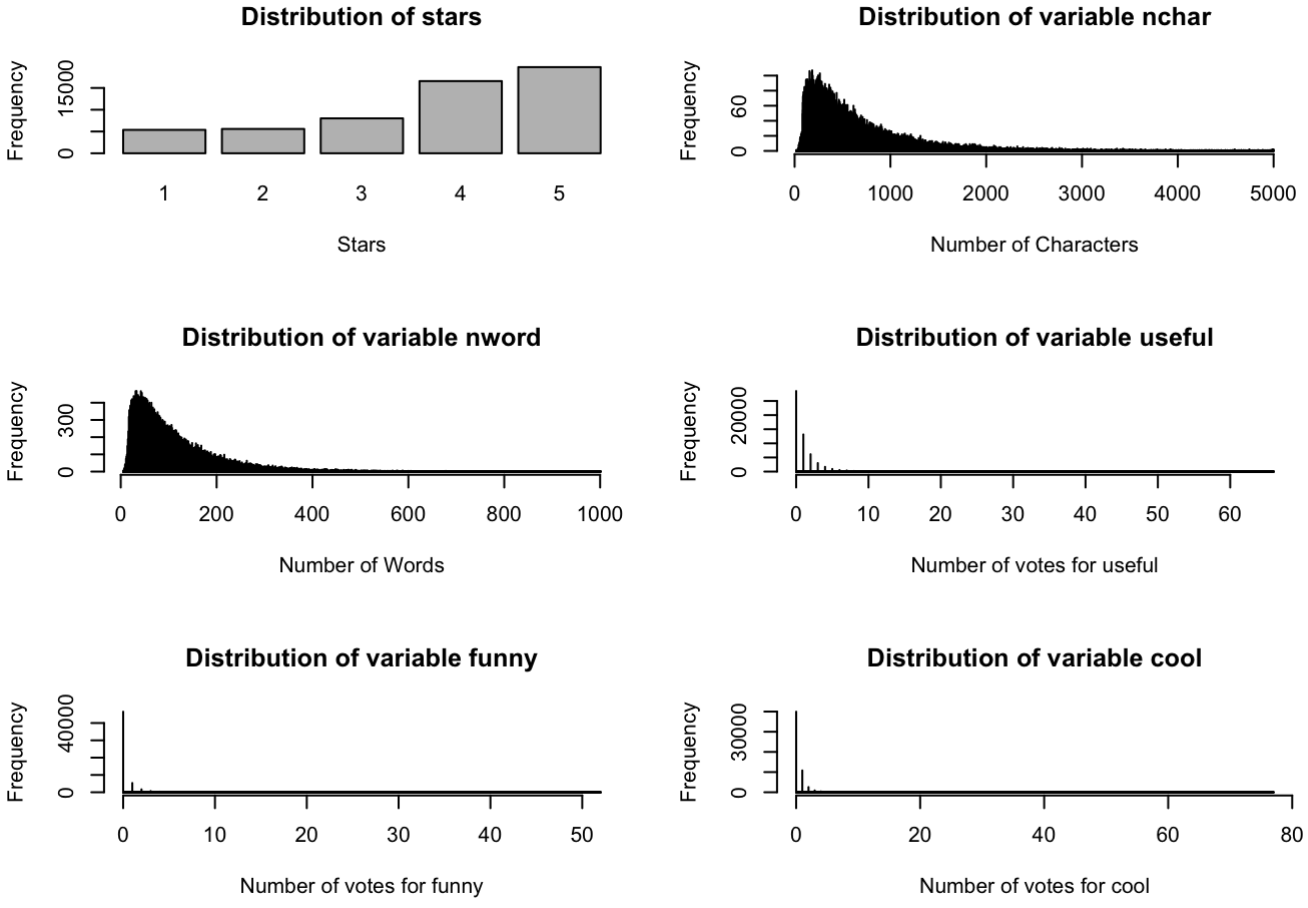
```
# randomly view 3 Five star reviews
yelp[sample(which(yelp$stars == 5),3), view_vars]
```

	Id <int>	stars <int>	name <fctr>	nchar <int>	nword <int>
31550	31550	5	Monty's Blue Plate Diner	442	85
43943	43943	5	Stella's Bakery	228	44
54754	54754	5	Wilson's Bar	641	115

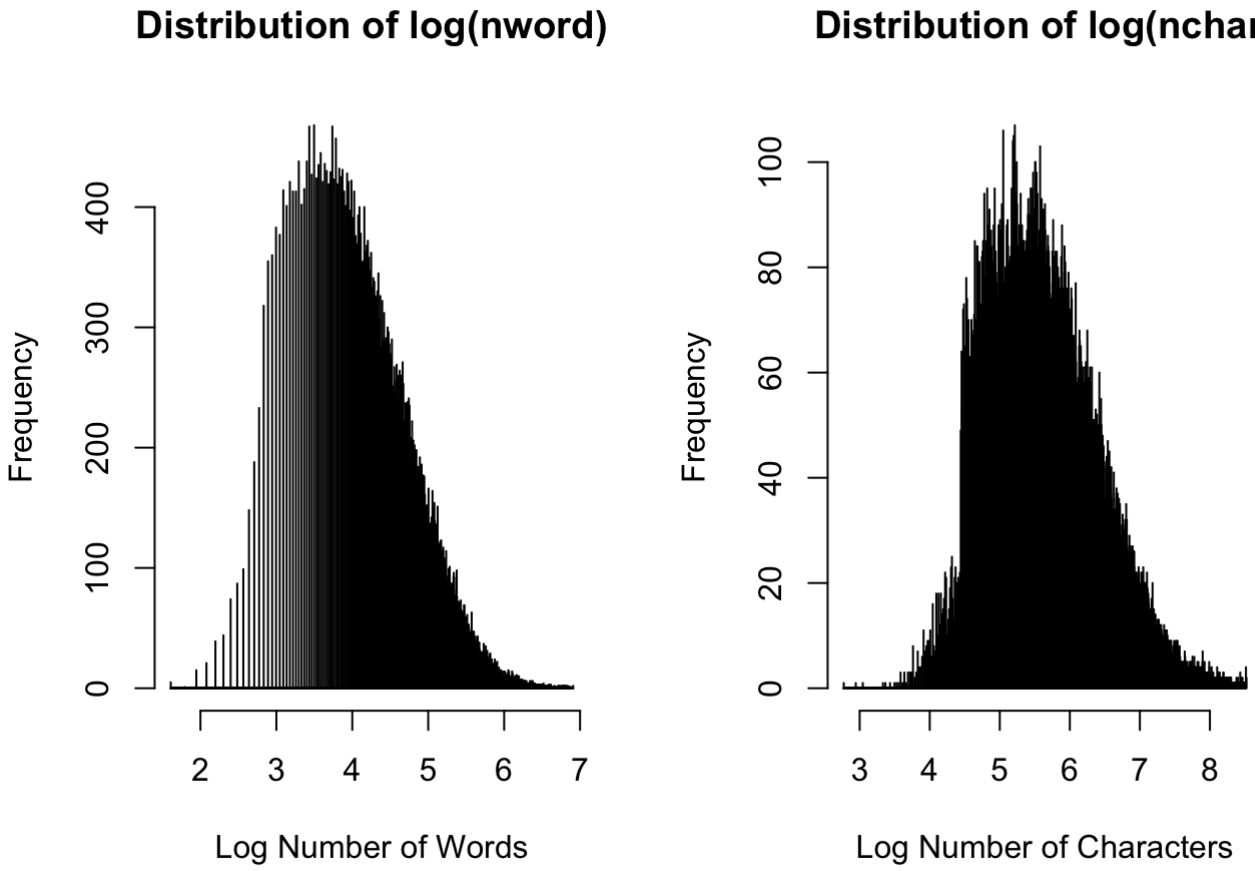
3 rows | 1-6 of 7 columns

Visualization

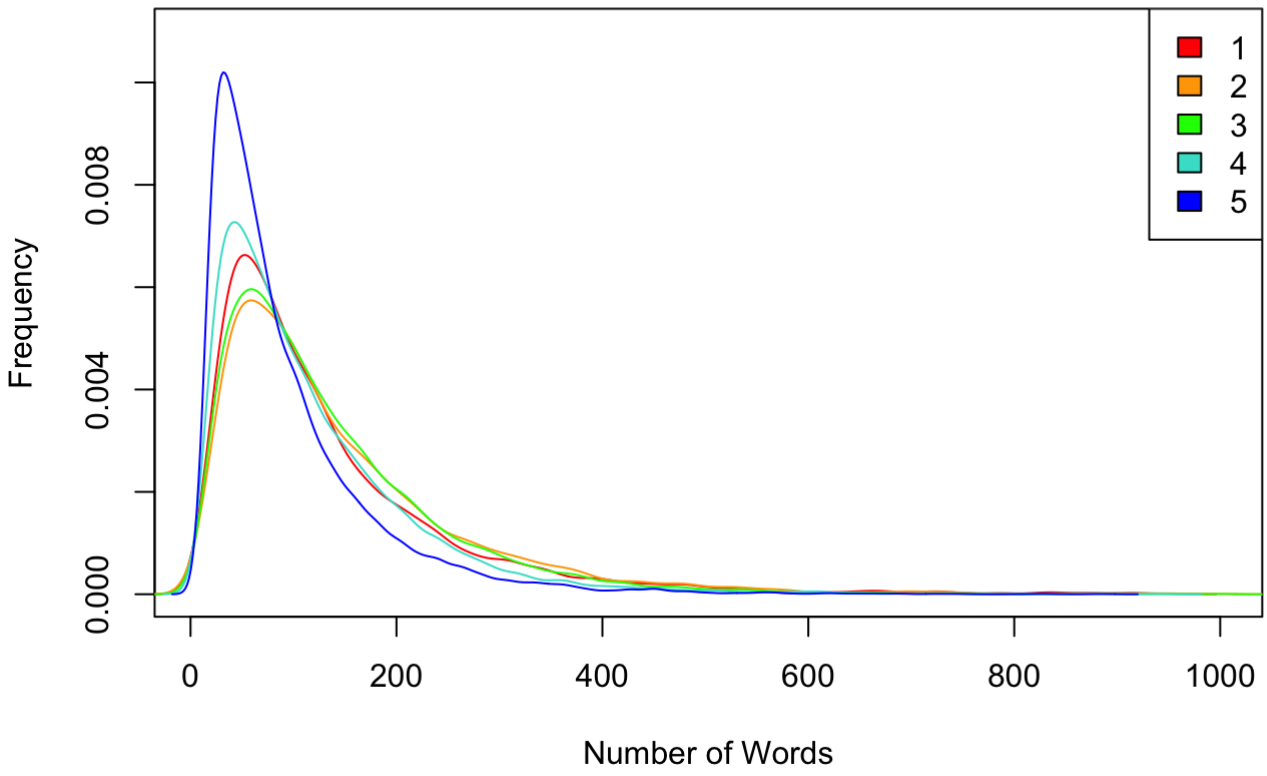
```
par(mfrow = c(3,2))
barplot(table(yelp$stars),main="Distribution of stars",xlab="Stars",ylab="Frequency")
hist(yelp$nchar, breaks=10000,main="Distribution of variable nchar",xlab="Number of Char
acters",ylab="Frequency")
hist(yelp$nword, breaks=10000,main="Distribution of variable nword",xlab="Number of Word
s",ylab="Frequency")
hist(yelp$useful, breaks=10000,main="Distribution of variable useful",xlab="Number of vo
tes for useful",ylab="Frequency")
hist(yelp$funny, breaks=10000,main="Distribution of variable funny",xlab="Number of vote
s for funny",ylab="Frequency")
hist(yelp$cool, breaks=10000,main="Distribution of variable cool",xlab="Number of votes
 for cool",ylab="Frequency")
```



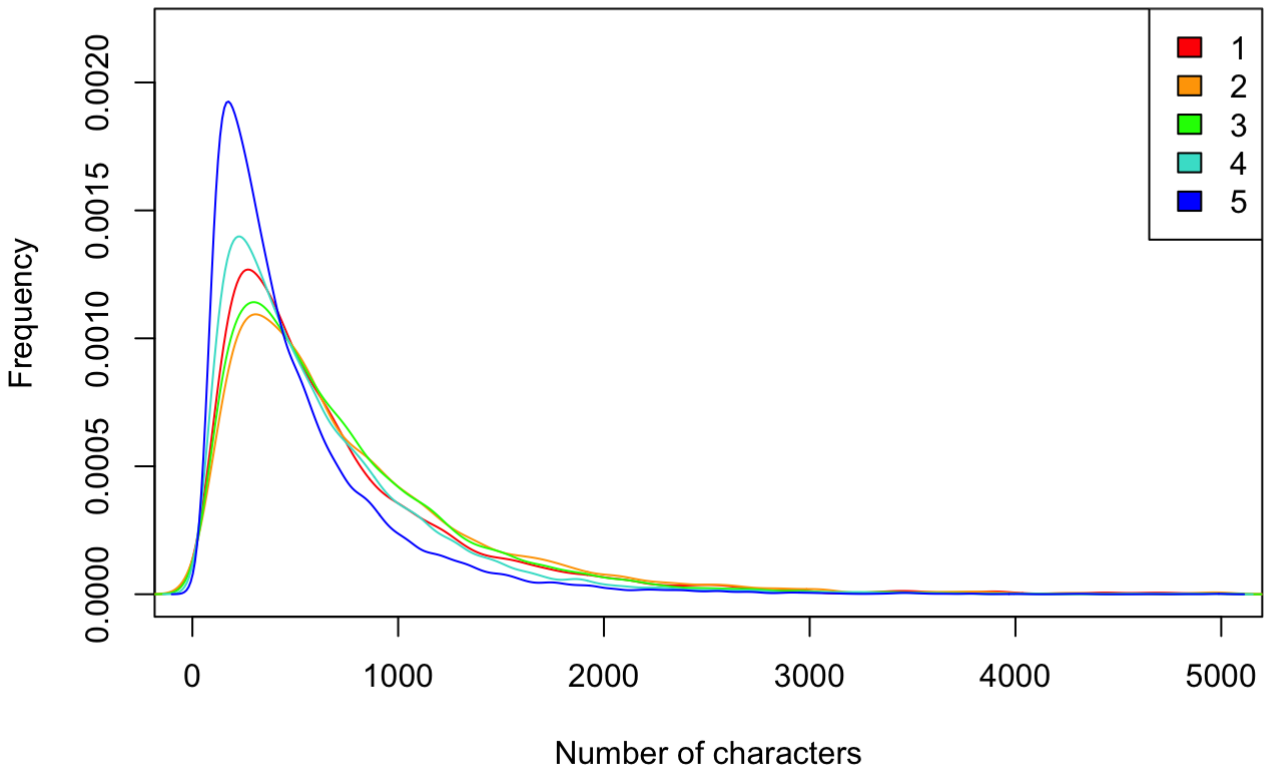
```
par(mfrow=c(1,2))
hist(log(yelp$nword),breaks=10000,main="Distribution of log(nword)",xlab="Log Number of
 Words",ylab="Frequency")
hist(log(yelp$nchar),breaks=10000,main="Distribution of log(nchar)",xlab="Log Number of
 Characters",ylab="Frequency")
```



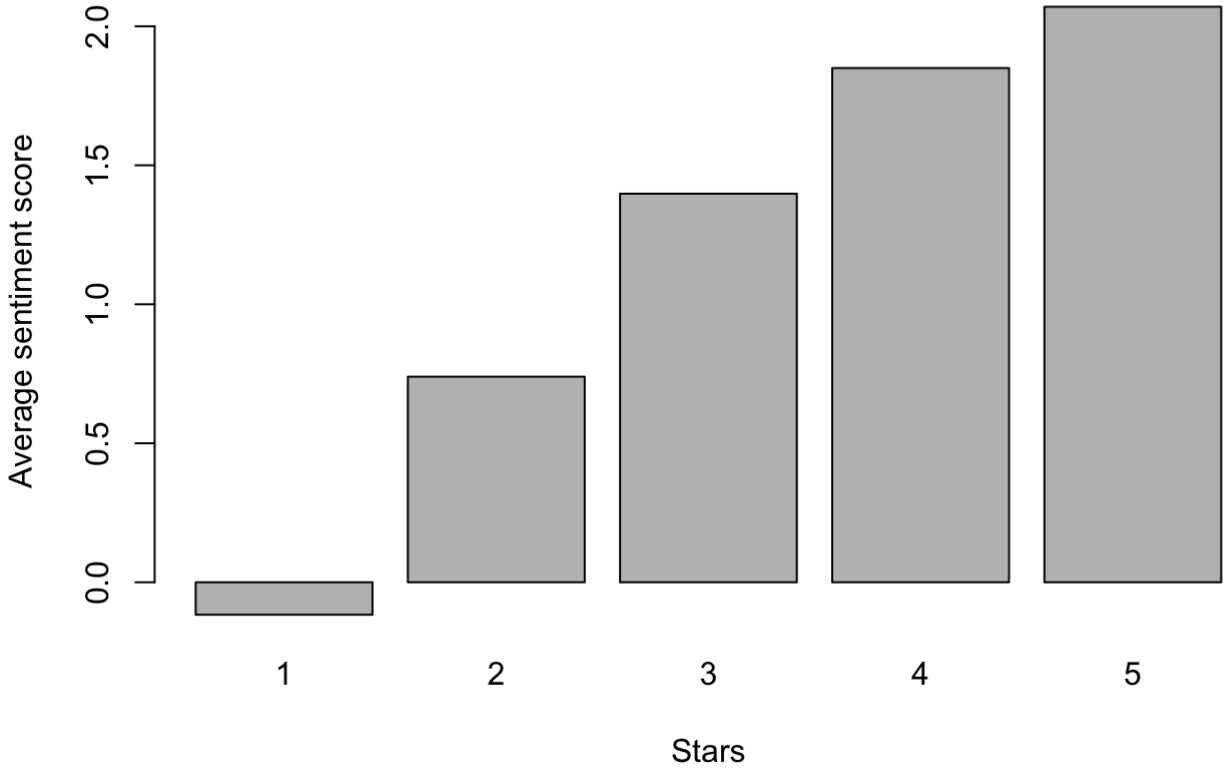

```
# Distribution of nword by star rating
plot(range(yelp$nword), c(0,0.011), main="Distribution of nword", xlab="Number of Words"
, ylab="Frequency", type='n')
colpalette <- c("red","orange","green","turquoise","blue")
for (i in 1:5){
  subsamples <- yelp$stars==i
  d <- density(yelp$nword[subsamples])
  lines(d, col=colpalette[i])
}
legend("topright",legend=levels(factor(yelp$stars)), fill=c("red","orange","green","turq
uoise","blue"))
```




```
# Distribution of nchar by star rating
plot(range(yelp$nchar), c(0,0.0022), main="Distribution of nchar", xlab="Number of chara
cters", ylab="Frequency", type='n')
colpalette <- c("red","orange","green","turquoise","blue")
for (i in 1:5){
  subsamples <- yelp$stars==i
  d <- density(yelp$nchar[subsamples])
  lines(d, col=colpalette[i])
}
legend("topright",legend=levels(factor(yelp$stars)), fill=c("red","orange","green","turq
uoise","blue"))
```




```
# Relationship between sentiment score and star rating
meanscore <- rep(0,5)
names(meanscore) <- 1:5
for (i in 1:5) meanscore[i] <- mean(yelp$sentiment[yelp$stars==i])
barplot(meanscore, xlab='Stars', ylab="Average sentiment score")
```



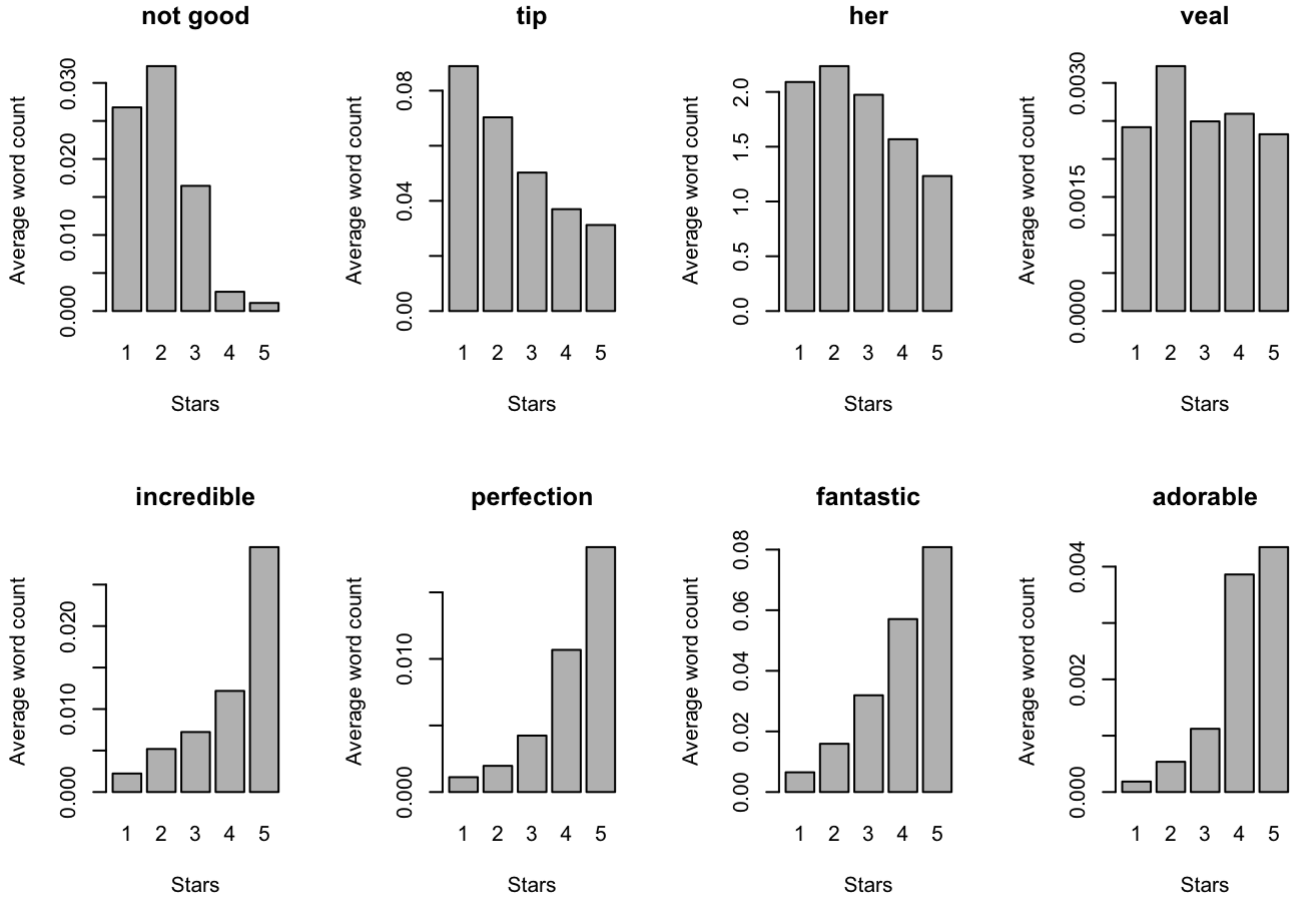
Exploring the Review Text

To make them easier to understand ,these codes are not written in the most efficient way. You are welcome to explore different coding tricks, different R packages on string operations, text mining, and natual language processing.

```
# generate some new predictors
library(stringr)
new_words <- c("not good", "tip", "her", "veal")
new_X <- matrix(0, nrow(yelp), length(new_words))
colnames(new_X) <- new_words
for (i in 1:length(new_words)){
  new_X[,i] <- str_count(yelp$text, regex(new_words[i], ignore_case=T)) # ignore the upp
er/lower case in the text
}

# plotting the word count against star rating
plotWordStar <- function(stars, wordcount, wordname){
  meancount <- rep(0,5)
  names(meancount) <- 1:5
  for (i in 1:5)    meancount[i] <- mean(wordcount[stars==i])
  barplot(meancount, main=wordname, xlab="Stars", ylab="Average word count")
}

par(mfrow=c(2,4))
for (i in 1:length(new_words)){
  plotWordStar(yelp$stars, new_X[,i], colnames(new_X)[i])
}
for (i in c(15,16,34,35)){
  plotWordStar(yelp$stars, yelp[,i], colnames(yelp)[i])
}
```



```
# testing if a specific word count is associated with star rating
new_pvals <- rep(0,4)
names(new_pvals) <- new_words
for (i in 1:4){
  ctable <- table(yelp$stars, new_X[,i])
  new_pvals[i] <- fisher.test(ctable, simulate.p.value = T)$p.value
}
new_pvals
```

```
##     not good          tip          her         veal
## 0.0004997501 0.0004997501 0.0004997501 0.5462268866
```

```
# Generate word count for the entire dictionary of words
library(dplyr)
library(tidytext)
library(tm)
# Here I am only building the dictionary using the first 5 reviews. If you really want t
o do this for all the reviews, be prepared for the run time and the size of the dictiona
ry
yelp_text_tbl <- tbl_df(data.frame(uniqueID = 1:5,yelp[1:5,]))
yelp_text_tbl_words <- yelp_text_tbl %>% select(uniqueID,text) %>%
                       unnest_tokens(word, text) %>% filter(str_detect(word,"^[a-z']+$"))
%>%
                       group_by(uniqueID) %>% count(word)
ReviewWordMatrix <- yelp_text_tbl_words %>% cast_dtm(uniqueID, word, n)
dim(ReviewWordMatrix)
```

```
## [1]   5 289
```

```
as.matrix(ReviewWordMatrix)[,1:10]
```

```
##     Terms
## Docs a accompanied again all ambience and as at bacon bar
##    1 4           1     1   1        1   4  1  1     1   1
##    2 0           0     0   0        0   1  0  0     0   0
##    3 9           0     0   0        0   7  0  4     0   0
##    4 6           0     0   1        0   9  0  0     0   0
##    5 1           0     0   0        0   3  0  0     0   0
```

The benchmark model

```
dat <- yelp[,-c(1,3:5,9,13)]
colnames(dat)
```

```
##   [1] "stars"         "useful"        "funny"         "cool"
##   [5] "nchar"         "nword"         "sentiment"     "gem"
##   [9] "incredible"    "perfection"    "heaven"        "phenomenal"
##  [13] "divine"        "die"           "deliciously"   "highly"
##  [17] "heavenly"      "superb"        "amazing"       "favorites"
##  [21] "delectable"    "perfect"       "sourced"       "knowledgeable"
##  [25] "wonderfully"   "deliciousness" "knowledgable"  "fantastic"
##  [29] "adorable"      "wonderful"     "fabulous"      "scrumptious"
##  [33] "hidden"        "notch"         "favorite"      "disappoint"
##  [37] "watering"      "delightful"    "yum"           "outstanding"
##  [41] "awesome"       "everyday"      "relaxed"       "dream"
##  [45] "amazingly"     "delicious"     "affordable"    "scones"
##  [49] "excellent"     "rotating"      "paired"        "secret"
##  [53] "beautifully"   "terrace"       "skeptical"     "perfectly"
##  [57] "delish"        "supposed"      "mcdonald.s"    "undercooked"
##  [61] "management"    "sucks"         "flag"          "messed"
##  [65] "unpleasant"    "dirty"         "mediocre"      "manager"
##  [69] "burned"        "nope"          "upset"         "zero"
##  [73] "stale"         "screw"         "subpar"        "inattentive"
##  [77] "edible"        "complained"    "rubbery"       "lukewarm"
##  [81] "garbage"       "contact"       "flavorless"    "poor"
##  [85] "apologize"     "gross"         "charged"       "receipt"
##  [89] "worse"         "ignored"       "response"      "poorly"
##  [93] "nasty"         "proceeded"     "terrible"      "waste"
##  [97] "tasteless"     "inedible"      "downhill"      "awful"
## [101] "rude"          "horrible"      "refused"       "apology"
## [105] "disgusting"    "worst"         "refund"
```

```
benchmark <- lm(stars~., data=dat)
summary(benchmark)
```

```
##
## Call:
## lm(formula = stars ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8846 -0.5763  0.0710  0.6563  3.8330
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.0891864  0.0096830 319.033  < 2e-16 ***
## useful        -0.0577548  0.0028668 -20.146  < 2e-16 ***
## funny         -0.0829936  0.0042098 -19.714  < 2e-16 ***
## cool           0.1852480  0.0045599  40.625  < 2e-16 ***
## nchar          0.0010083  0.0001165   8.658  < 2e-16 ***
## nword         -0.0063289  0.0006130 -10.324  < 2e-16 ***
## sentiment      0.3998109  0.0041778  95.699  < 2e-16 ***
## gem            0.2799987  0.0334504   8.371  < 2e-16 ***
## incredible     0.8036893  0.0709230  11.332  < 2e-16 ***
## perfection     0.0775190  0.0409644   1.892 0.058449 .
## heaven         0.3746516  0.0474424   7.897 2.91e-15 ***
## phenomenal     0.3243697  0.0408534   7.940 2.06e-15 ***
## divine         0.1855574  0.0656607   2.826 0.004715 **
## die            0.1665433  0.0127144  13.099  < 2e-16 ***
## deliciously    0.0173388  0.0801952   0.216 0.828827
## highly         0.3329703  0.0202531  16.440  < 2e-16 ***
## heavenly      -0.0836687  0.0867021  -0.965 0.334542
## superb        -0.0003135  0.0455265  -0.007 0.994506
## amazing        0.2136531  0.0108947  19.611  < 2e-16 ***
## favorites      0.0306767  0.0358525   0.856 0.392203
## delectable     0.2381609  0.0803750   2.963 0.003047 **
## perfect        0.2199324  0.0142467  15.437  < 2e-16 ***
## sourced        0.1594288  0.0616291   2.587 0.009687 **
## knowledgeable  0.2195770  0.0398752   5.507 3.67e-08 ***
## wonderfully    0.1546324  0.0710143   2.177 0.029449 *
## deliciousness -0.0243057  0.0761661  -0.319 0.749641
## knowledgable   0.3196812  0.0850073   3.761 0.000170 ***
## fantastic      0.1866539  0.0163602  11.409  < 2e-16 ***
## adorable       0.2012420  0.0711092   2.830 0.004656 **
## wonderful      0.1487266  0.0177296   8.389  < 2e-16 ***
## fabulous       0.1179691  0.0359097   3.285 0.001020 **
## scrumptious    0.2657841  0.0804797   3.302 0.000959 ***
## hidden         0.1921794  0.0508022   3.783 0.000155 ***
## notch          0.3632581  0.0372226   9.759  < 2e-16 ***
## favorite       0.3175478  0.0121941  26.041  < 2e-16 ***
## disappoint    -0.1088343  0.0125365  -8.681  < 2e-16 ***
## watering       0.2469797  0.0677308   3.646 0.000266 ***
## delightful     0.2408598  0.0440981   5.462 4.73e-08 ***
## yum            0.1591653  0.0155897  10.210  < 2e-16 ***
## outstanding    0.1031219  0.0267578   3.854 0.000116 ***
## awesome        0.1777371  0.0142045  12.513  < 2e-16 ***
## everyday       0.2923938  0.0736289   3.971 7.16e-05 ***
## relaxed        0.2940164  0.0537608   5.469 4.55e-08 ***
```

```
## dream          0.4193816  0.0493936   8.491  < 2e-16 ***
## amazingly      0.0119979  0.0668639   0.179 0.857595
## delicious      0.3025384  0.0087442  34.599  < 2e-16 ***
## affordable     0.2206039  0.0407279   5.417 6.10e-08 ***
## scones         0.1493058  0.0433957   3.441 0.000581 ***
## excellent      0.2093327  0.0119949  17.452  < 2e-16 ***
## rotating       0.1736878  0.0737394   2.355 0.018505 *
## paired         0.0129436  0.0539736   0.240 0.810476
## secret         0.2705495  0.0576278   4.695 2.68e-06 ***
## beautifully   -0.0192309  0.0580524  -0.331 0.740443
## terrace        0.2789547  0.0539091   5.175 2.29e-07 ***
## skeptical      0.4071186  0.0765952   5.315 1.07e-07 ***
## perfectly     -0.0978024  0.0245952  -3.976 7.00e-05 ***
## delish         0.2499442  0.0467518   5.346 9.02e-08 ***
## supposed      -0.2958733  0.0333197  -8.880  < 2e-16 ***
## mcdonald.s    -0.2505454  0.0543776  -4.608 4.08e-06 ***
## undercooked   -0.4705585  0.0545825  -8.621  < 2e-16 ***
## management    -0.4977050  0.0489716 -10.163  < 2e-16 ***
## sucks         -0.3381900  0.0639893  -5.285 1.26e-07 ***
## flag          -0.2420702  0.0546143  -4.432 9.34e-06 ***
## messed        -0.1268703  0.0731721  -1.734 0.082948 .
## unpleasant    -0.4572210  0.0740836  -6.172 6.80e-10 ***
## dirty         -0.2087756  0.0281203  -7.424 1.15e-13 ***
## mediocre      -0.5955654  0.0286939 -20.756  < 2e-16 ***
## manager       -0.1785734  0.0174063 -10.259  < 2e-16 ***
## burned        -0.4695676  0.0684943  -6.856 7.18e-12 ***
## nope          -0.2933806  0.0623935  -4.702 2.58e-06 ***
## upset         -0.1358210  0.0608534  -2.232 0.025623 *
## zero          -0.2718498  0.0496903  -5.471 4.50e-08 ***
## stale         -0.4502562  0.0503892  -8.936  < 2e-16 ***
## screw         -0.3502364  0.0514256  -6.811 9.82e-12 ***
## subpar        -0.8263314  0.0826360 -10.000  < 2e-16 ***
## inattentive   -0.4397579  0.0837088  -5.253 1.50e-07 ***
## edible        -0.4493075  0.0645477  -6.961 3.42e-12 ***
## complained    -0.0973106  0.0669722  -1.453 0.146230
## rubbery       -0.5104571  0.0722095  -7.069 1.58e-12 ***
## lukewarm      -0.6072710  0.0600648 -10.110  < 2e-16 ***
## garbage       -0.2395133  0.0837916  -2.858 0.004259 **
## contact       -0.2901692  0.0538260  -5.391 7.04e-08 ***
## flavorless    -0.5430040  0.0423707 -12.816  < 2e-16 ***
## poor          -0.3622912  0.0275626 -13.144  < 2e-16 ***
## apologize     -0.0026555  0.0391204  -0.068 0.945881
## gross         -0.4270127  0.0394664 -10.820  < 2e-16 ***
## charged       -0.2787452  0.0364785  -7.641 2.18e-14 ***
## receipt       -0.2412892  0.0569810  -4.235 2.29e-05 ***
## worse         -0.2173959  0.0402611  -5.400 6.70e-08 ***
## ignored       -0.2947661  0.0602022  -4.896 9.79e-07 ***
## response      -0.4095200  0.0603077  -6.791 1.13e-11 ***
## poorly        -0.1095021  0.0688542  -1.590 0.111763
## nasty         -0.2653322  0.0613285  -4.326 1.52e-05 ***
## proceeded     -0.3021905  0.0706331  -4.278 1.89e-05 ***
## terrible      -0.3734419  0.0242651 -15.390  < 2e-16 ***
## waste         -0.4374465  0.0406594 -10.759  < 2e-16 ***
## tasteless     -0.5760351  0.0452082 -12.742  < 2e-16 ***
```

```
## inedible      -0.0694323  0.0876383  -0.792 0.428213
## downhill      -0.7373187  0.0831679  -8.865  < 2e-16 ***
## awful         -0.3307123  0.0316455 -10.451  < 2e-16 ***
## rude          -0.4684396  0.0230964 -20.282  < 2e-16 ***
## horrible      -0.4368221  0.0303162 -14.409  < 2e-16 ***
## refused       -0.2368852  0.0756638  -3.131 0.001744 **
## apology       -0.3306422  0.0594842  -5.558 2.73e-08 ***
## disgusting    -0.3345741  0.0459934  -7.274 3.53e-13 ***
## worst         -0.5122428  0.0259276 -19.757  < 2e-16 ***
## refund        -0.3205240  0.0530401  -6.043 1.52e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9313 on 55235 degrees of freedom
## Multiple R-squared:  0.4913, Adjusted R-squared:  0.4903
## F-statistic: 503.3 on 106 and 55235 DF,  p-value: < 2.2e-16
```

```
star_out <- data.frame(Id=yelp_out$Id, Expected=predict(benchmark, newdata = yelp_out))
write.csv(star_out, file='Pixu_submission.csv', row.names=FALSE)
```

Things you can consider to improve your model

- Adding more predictors: you are not confined to the provided predictors.
    - You can extract more words or phrases from the review text as new predictors.
    - You can combine similar words such as okay and ok.
    - You can create new predictors based on the available predictors.
- Transformation of variables: log, exponential, square, square root, inverse, ratio of variables, etc.
- Models other than multiple linear regression: for example but not limited to ridge regression, lasso, logistic regression, multinomial logistic regression, principle component regression, etc.
- BE CAREFUL not to overfit your model!! You can manipulate your model such that it predicts the best on your training data, or even on the public leader board. But it doesn't necessarily mean that your model does well on the testing data.
    - My suggestion: Randomly choose a small proportion of your data as your own testing data when looking for the best method to build the model. When doing the final prediction, use all the available data for model estimation.