UCS654 Project: Sentiment Analysis and Customer Segmentation on Yelp Dataset

Name: Vikram Alagh

Roll number: 101803368

Group: 3COE17

Dataset name: Yelp Academic Dataset

Dataset link (the tips and users data-sets from the json download are being worked on): https://www.yelp.com/dataset/

Sentiment analysis:

Reading first 1000 entries of tip dataset:

```
review_file <- readLines("yelp_academic_dataset_tip.json", n = 1000)
```

This dataset contains "tips" from yelp users for businesses, these are short form reviews.

Loading the read entries into a json file for parsing:

```
library(jsonlite)
parsed_json <- jsonlite::stream_in(textConnection(review_file))
```

```
##  Found 500 records... Found 1000 records... Imported 1000 records. Simplifying...
```

Exporting parsed json to a tibble:

```
library(tibble)
parsed_json_tibble <- tibble::as_tibble(parsed_json)
```

Extracting all review rows from data frame:

```
parsed_json_tibble$unique_key <- paste(parsed_json_tibble$date,
    parsed_json_tibble$user_id)
tips <- dplyr::select(parsed_json_tibble, unique_key, text)

print(tips[0:5, ])
```

```
## # A tibble: 5 x 2
##   unique_key                    text
##   <chr>                         <chr>
## 1 2011-07-22 19:07:35 WCjg0jdH~ "Carne asada chips..."
## 2 2014-09-10 07:33:29 42-Z02y9~ "Best happy hour from 3pm to 6pm! $1 off martin~
## 3 2013-12-13 23:23:41 5u7E3LYp~ "Nice people, skilled staff, clean location - b~
## 4 2017-07-11 23:07:16 wDWoMG5N~ "1/2-price bowling & the \"Very\" Old Fashion a~
## 5 2016-11-30 08:46:36 JmuFlorj~ "Solid gold's. Great sauna. Great staff, too. E~
```

Splitting all tips into unigrams:

```
library(tidytext)
split_tips <- tidytext::unnest_tokens(tips, unigrams, text, token = "words",
    format = "text", to_lower = TRUE)
```

Getting sentiments from AFINN attaching values from it to the unigrams in tips:

```
afinn_sentiments <- tidytext::get_sentiments("afinn")
tips_with_unigram_scores = merge(split_tips, afinn_sentiments,
    by.x = "unigrams", by.y = "word")

tips_with_unigram_scores <- tips_with_unigram_scores[order(tips_with_unigram_scores$unique_key),
    ]

print(tips_with_unigram_scores[0:10, ])
```

```
##     unigrams                                unique_key value
## 133   better 2009-04-18 22:33:08 RquYSxdEHbg9Db_qPE5HGQ     2
## 891    sorry 2009-04-29 00:59:13 906lrlcCbiYI1SuvWFJS4g    -1
## 113     best 2009-05-02 18:22:49 V-2CHlifHvVC-3UXxipiNg     3
## 225    enjoy 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     2
## 530    great 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     3
## 594    happy 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     3
## 755     nice 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     3
## 950      top 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     2
```

```
## 215     drag 2009-07-31 01:39:00 fHS0bQ-l5rHME_xXKQSYXQ    -1
## 596     hard 2009-07-31 01:39:00 fHS0bQ-l5rHME_xXKQSYXQ    -1
```

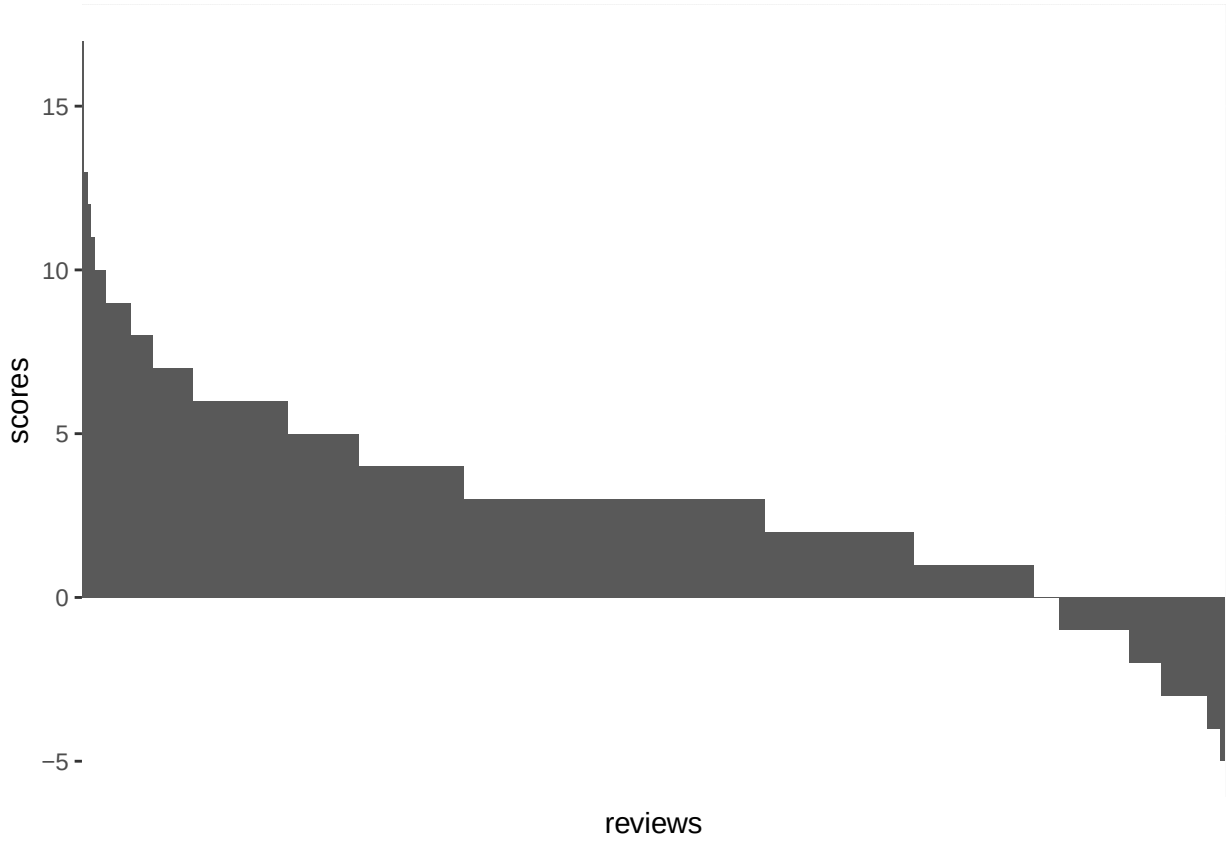
Scoring all reviews based on sentiment:

```
tips_with_score <- aggregate(tips_with_unigram_scores$value,
    by = list(unique_key = tips_with_unigram_scores$unique_key),
    FUN = sum)

print(tips_with_score[0:10, ])
```

```
##                                 unique_key  x
## 1  2009-04-18 22:33:08 RquYSxdEHbg9Db_qPE5HGQ  2
## 2  2009-04-29 00:59:13 906lrlcCbiYI1SuvWFJS4g -1
## 3  2009-05-02 18:22:49 V-2CHlifHvVC-3UXxipiNg  3
## 4  2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg 13
## 5  2009-07-31 01:39:00 fHS0bQ-l5rHME_xXKQSYXQ -2
## 6  2009-09-20 18:47:33 NENsz6vQJHTAO9RMF7765w  6
## 7  2010-01-18 16:35:54 -K29SbpviWPK9NR7xAedmg  4
## 8  2010-01-19 17:38:26 UPn--rhxC2fYe8VLa3jeHQ -3
## 9  2010-01-21 01:22:39 -K29SbpviWPK9NR7xAedmg  2
## 10 2010-01-22 04:06:37 -K29SbpviWPK9NR7xAedmg  2
```

Visualising all the review scores in descending order:

```
library(ggplot2)

ggplot(data = tips_with_score, aes(x = reorder(unique_key, -x),
    y = x)) + geom_bar(stat = "identity") + xlab("reviews") +
    ylab("scores") + theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
```



Finding top 10 most frequent positive and negative words; and top ten most positive and negative words (in terms of their score in AFINN); that appear in tips and plotting their frequencies:

```
unique_unigrams <- dplyr::distinct(tips_with_unigram_scores,
    unigrams, .keep_all = TRUE)

unique_unigrams <- merge(unique_unigrams, data.frame(table(tips_with_unigram_scores$unigrams)),
    by.x = "unigrams", by.y = "Var1")

unique_unigrams$sentiments <- ifelse(unique_unigrams$value >=
    0, "positive", "negative")

top_pos <- dplyr::filter(unique_unigrams, sentiments == "positive")
top_neg <- dplyr::filter(unique_unigrams, sentiments == "negative")
```

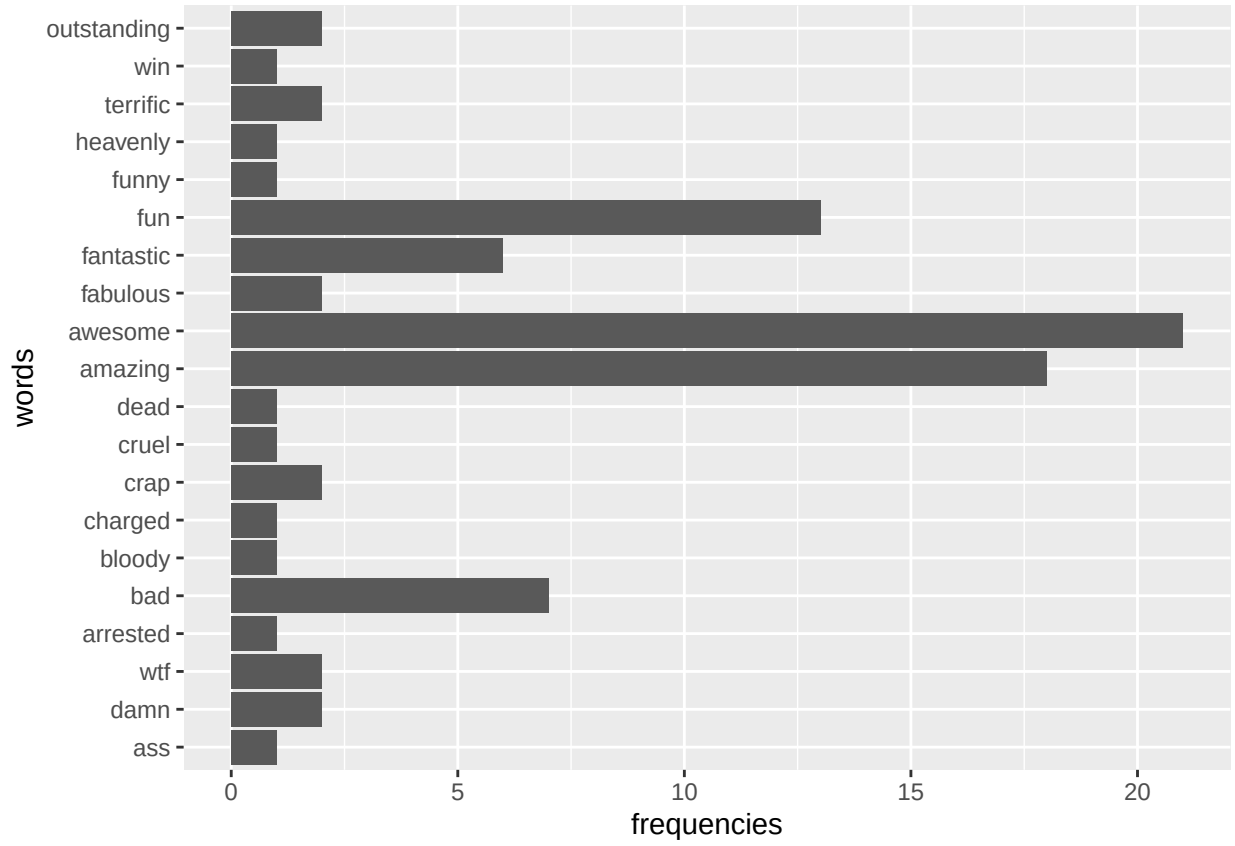
Plotting top 10 positive and negative words (according to scores) with their frequencies in decreasing order of sentiment value (5, -5):

```
top_score_pos <- top_pos[order(-top_pos$val), ]
top_score_neg <- top_neg[order(top_neg$val), ]

top10_score_posneg = rbind(top_score_neg[0:10, ], top_score_pos[0:10,
    ])
top10_score_posneg <- top10_score_posneg[order(top10_score_posneg$value),
    ]

print(top10_score_posneg)
```

```
##        unigrams                                unique_key value Freq
## 7           ass 2012-10-18 00:30:28 E4mINxIT0n2z0BhW6j81sA    -4    1
## 23         damn 2010-12-15 03:48:35 e0biEy1jBSHOvreBW8pATA    -4    2
## 99          wtf 2010-07-20 01:57:22 4NasTqV-SeT76ijXUt-nXQ    -4    2
## 5      arrested 2011-12-30 21:28:28 PI51J_fKhEfZKhi-MjWBpg    -3    1
## 10          bad 2010-03-04 18:33:01 1pMpk7naYYQJjHiM8fRJag    -3    7
## 11       bloody 2012-04-01 16:15:27 GnWwiVhE3RD__3qA9fxfSg    -3    1
## 15      charged 2013-08-18 03:20:30 vn9ixvK7Fv0BQP7uA0N9MQ    -3    1
## 18         crap 2010-02-06 17:34:16 5vh9CfVAG3tRgEreCHxX4w    -3    2
## 20        cruel 2015-01-12 04:24:30 tpi5TQyUbwGvvmgXJpCC_A    -3    1
## 26         dead 2011-12-25 19:41:16 TqPLT4SHNyDfjRVDIz2zeA    -3    1
## 3       amazing 2011-04-29 01:48:08 pE6P8uXIBtYyUHpPJ6fqGQ     4   18
## 51      awesome 2010-12-11 02:05:36 WfGBxC5EZzNQfIdsjJIPgQ     4   21
## 30     fabulous 2011-10-08 17:14:20 S5tOE7JAvaeXBEibnAV02g     4    2
## 33    fantastic 2010-01-31 15:04:27 uh0u3E7mbYz6dr2z9RdF5w     4    6
## 42          fun 2010-01-18 16:35:54 -K29SbpviWPK9NR7xAedmg     4   13
## 43        funny 2012-05-25 23:15:06 9YqYKkm42YvLkTaldBygPQ     4    1
## 58     heavenly 2013-02-08 23:39:10 mN6ii5R6_p7sP-buEnYAkg     4    1
## 115    terrific 2012-01-02 00:19:08 _PKXL0-QRclZQiOYxBbL6A     4    2
## 123         win 2011-04-30 01:17:19 nM8TyvrShtBtKwFK68eYhg     4    1
## 83  outstanding 2012-01-06 21:13:14 -h8OGC8dfT-llzlJDnOb9A     5    2
##     sentiments
## 7     negative
## 23    negative
## 99    negative
## 5     negative
## 10    negative
## 11    negative
## 15    negative
## 18    negative
## 20    negative
## 26    negative
## 3     positive
## 51    positive
## 30    positive
## 33    positive
## 42    positive
## 43    positive
## 58    positive
## 115   positive
## 123   positive
```

```
## 83    positive
```

```
library(ggplot2)

ggplot(data = top10_score_posneg, aes(x = Freq, y = reorder(unigrams,
    value))) + geom_bar(stat = "identity") + xlab("frequencies") +
    ylab("words")
```



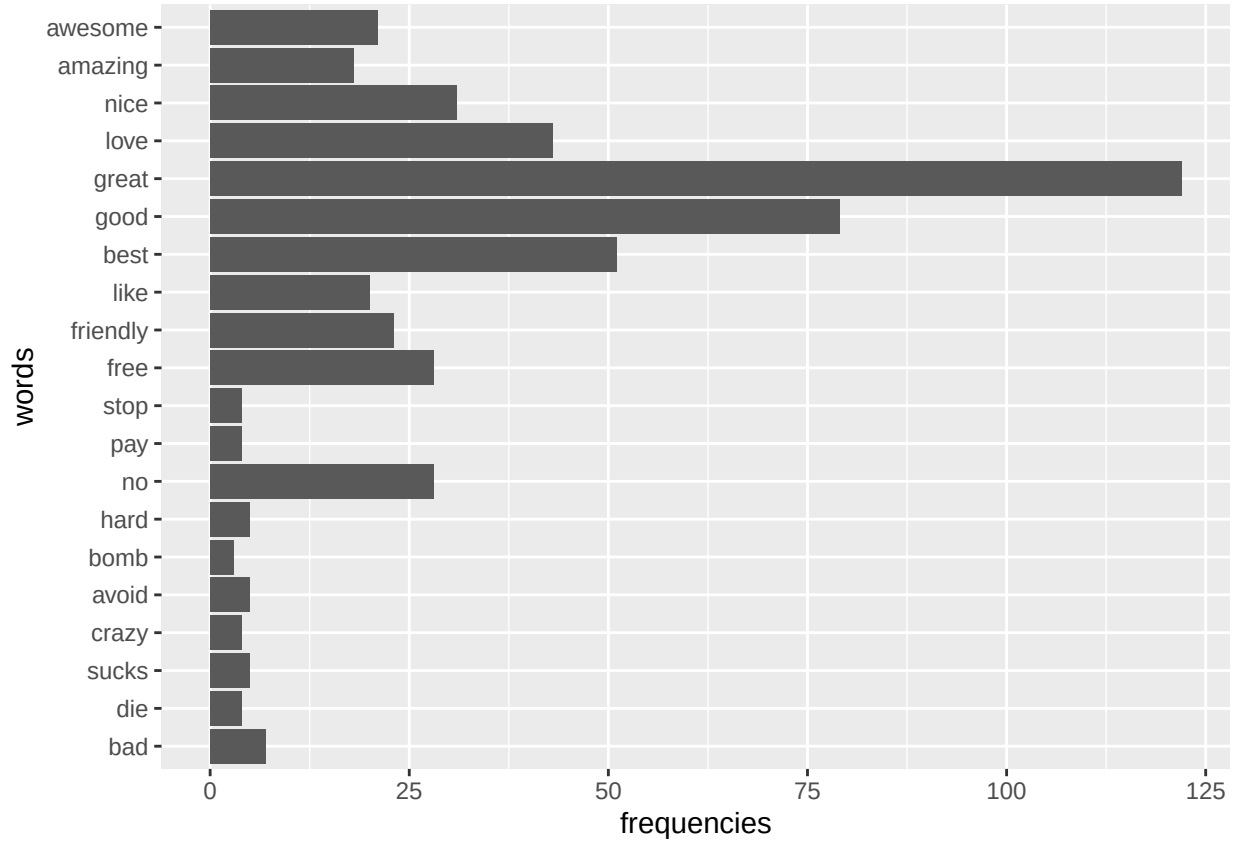
Plotting top 10 most frequent positive and negative words with their frequencies in decreasing order of sentiment value (5, -5):

```
top_freq_pos <- top_pos[order(-top_pos$Freq), ]
top_freq_neg <- top_neg[order(-top_neg$Freq), ]

top10_freq_posneg = rbind(top_freq_neg[0:10, ], top_freq_pos[0:10,
    ])
top10_freq_posneg <- top10_freq_posneg[order(top10_freq_posneg$value),
    ]

print(top10_freq_posneg)
```

```
##     unigrams                                 unique_key value Freq sentiments
## 10       bad 2010-03-04 18:33:01 1pMpk7naYYQJjHiM8fRJag    -3    7   negative
## 89     sucks 2011-04-15 03:11:38 ho_3BORkrN4hNwld2R73eg    -3    5   negative
## 27       die 2012-03-19 11:02:15 MtHD0Wdh4FtbyFJCb4yfUQ    -3    4   negative
```

```
## 19     crazy 2010-11-23 16:43:11 1jWX85cb5FAN78WbLJTUZA    -2    4   negative
## 68        no 2010-02-26 07:13:46 cLmZqkWBI4NxIAx5kYi5lg    -1   28   negative
## 8      avoid 2010-12-08 04:39:12 GAEgSERjNA9lN3EG7Ren8A    -1    5   negative
## 49      hard 2009-07-31 01:39:00 fHS0bQ-l5rHME_xXKQSYXQ    -1    5   negative
## 71       pay 2010-05-28 03:19:46 v2aaBL6CxCZ7uiWGDnr21g    -1    4   negative
## 84      stop 2010-03-17 02:29:17 uq4IinFuA8FfdgP3JaPIhg    -1    4   negative
## 12      bomb 2010-11-23 07:30:16 ayF5zFFro_QWrus2dST5_A    -1    3   negative
## 38      free 2010-05-16 01:19:47 t903_es-gp3abvdrIQutQA     1   28   positive
## 41  friendly 2010-02-19 03:29:49 IuzdwKo2Aqn0_2bRXyAcAw     2   23   positive
## 75      like 2011-07-16 21:20:25 Vc_8ovbnoLxrCUy5zt6ilQ     2   20   positive
## 491    great 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     3  122   positive
## 46      good 2009-09-20 18:47:33 NENsz6vQJHTAO9RMF7765w     3   79   positive
## 7       best 2009-05-02 18:22:49 V-2CHlifHvVC-3UXxipiNg     3   51   positive
## 78      love 2009-09-20 18:47:33 NENsz6vQJHTAO9RMF7765w     3   43   positive
## 82      nice 2009-06-23 20:48:30 or7SmjPAL6mq0w0sJC6Kwg     3   31   positive
## 5    awesome 2010-12-11 02:05:36 WfGBxC5EZzNQfIdsjJIPgQ     4   21   positive
## 3    amazing 2011-04-29 01:48:08 pE6P8uXIBtYyUHpPJ6fqGQ     4   18   positive
```

```
library(ggplot2)

ggplot(data = top10_freq_posneg, aes(x = Freq, y = reorder(unigrams,
    value))) + geom_bar(stat = "identity") + xlab("frequencies") +
    ylab("words")
```



Making a word cloud of top 10 positive and negative words (according to AFINN score):

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     tips
```

```
library(wordcloud)
```

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

```
acast(top10_score_posneg, unigrams ~ sentiments, value.var = "Freq",
    fill = 0) %>%
    comparison.cloud(colors = c("red", "dark green"), max.words = 100)
```

Making a word cloud of top 10 most frequent positive and negative words:

```
acast(top10_freq_posneg, unigrams ~ sentiments, value.var = "Freq",
    fill = 0) %>%
    comparison.cloud(colors = c("red", "dark green"), max.words = 100)
```

Customer Segmentation:

Reading first 100 entries of users dataset:

```
user_file <- readLines("yelp_academic_dataset_user.json", n = 100)
```

This dataset contains user information related to their reviews including the kinds of reactions their reviews generated in other users.

Loading the read entries into a json file for parsing:

```
library(jsonlite)
parsed_json <- jsonlite::stream_in(textConnection(user_file))
```

```
##  Found 100 records... Imported 100 records. Simplifying...
```

Exporting parsed json to a tibble:

```
library(tibble)
parsed_json_tibble <- tibble::as_tibble(parsed_json)
```

Removing unused columns from parsed tibble:

```
user_data <- dplyr::select(parsed_json_tibble, -friends, -elite,
    -yelping_since)
print(user_data[0:10, ])
```

```
## # A tibble: 10 x 19
##    user_id          name     review_count useful funny  cool  fans average_stars
##    <chr>            <chr>           <int>  <int> <int> <int> <int>         <dbl>
##  1 q_QQ5kBBwlCcbL1~ Jane             1220  15038 10030 11291  1357          3.85
##  2 dIIKEfOgo0KqUfG~ Gabi             2136  21272 10289 18046  1025          4.09
##  3 D6ErcUnFALnCQN4~ Jason             119    188   128   130    16          3.76
##  4 JnPIjvC0cmooNDf~ Kat               987   7234  4722  4035   420          3.77
##  5 37Hc8hr3cw0iHLo~ Christi~          495   1577   727  1124    47          3.72
##  6 n-QwITZYrXlKQRi~ Natasha           229    476   101   140    17          3.59
##  7 eCJoZqpV1fDKJGA~ Bridget            51     53    14    16     1          3.86
##  8 cojecOwQJpsYDxn~ Steven             51    136    47    44     4          3.79
##  9 1jXmzuIFKxTnEnR~ Clara             299    381   106   121    23          3.43
## 10 -8QoOIfvwwxJ4sY~ Antoine~          288    752   220   306    25          3.88
## # ... with 11 more variables: compliment_hot <int>, compliment_more <int>,
## #   compliment_profile <int>, compliment_cute <int>, compliment_list <int>,
## #   compliment_note <int>, compliment_plain <int>, compliment_cool <int>,
## #   compliment_funny <int>, compliment_writer <int>, compliment_photos <int>
```

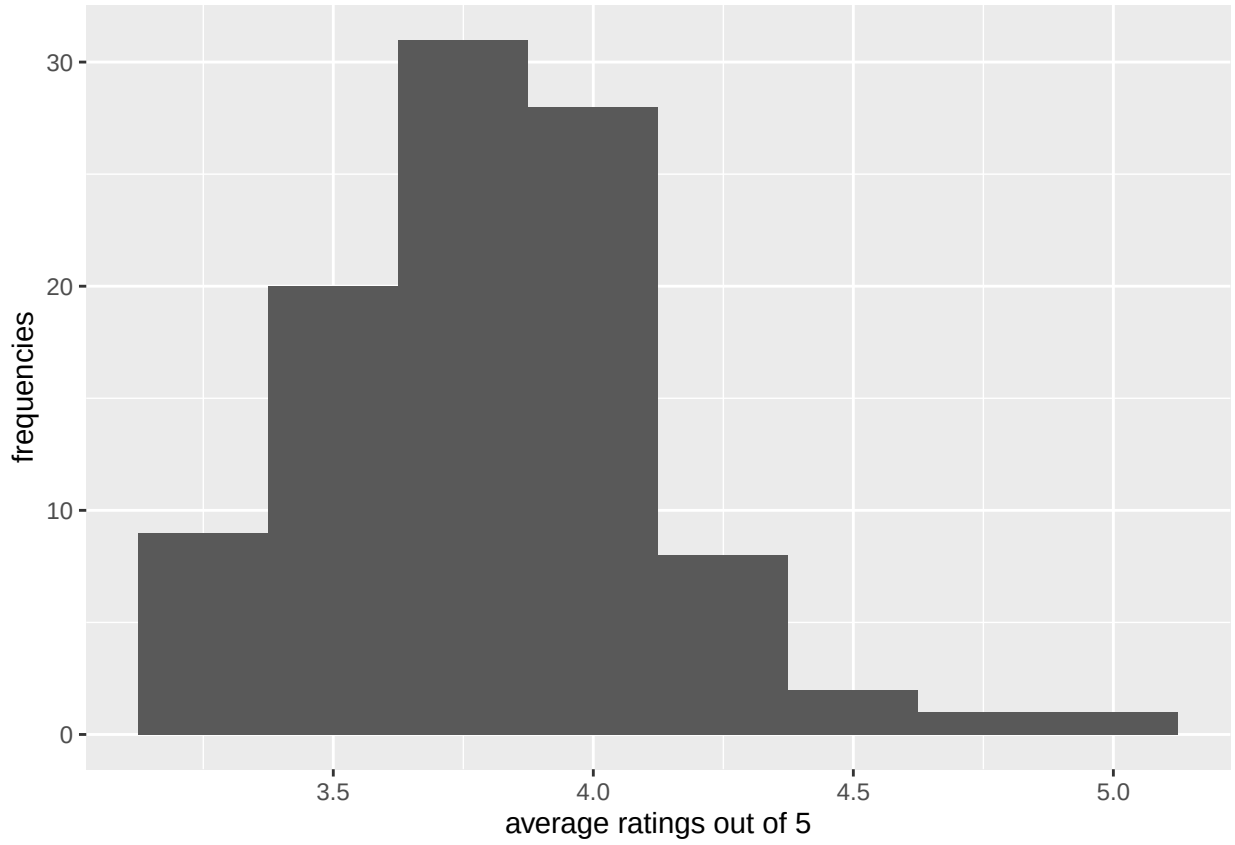
Summary of average user ratings

```
summary(user_data$average_stars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.200   3.605   3.805   3.804   3.980   4.910
```

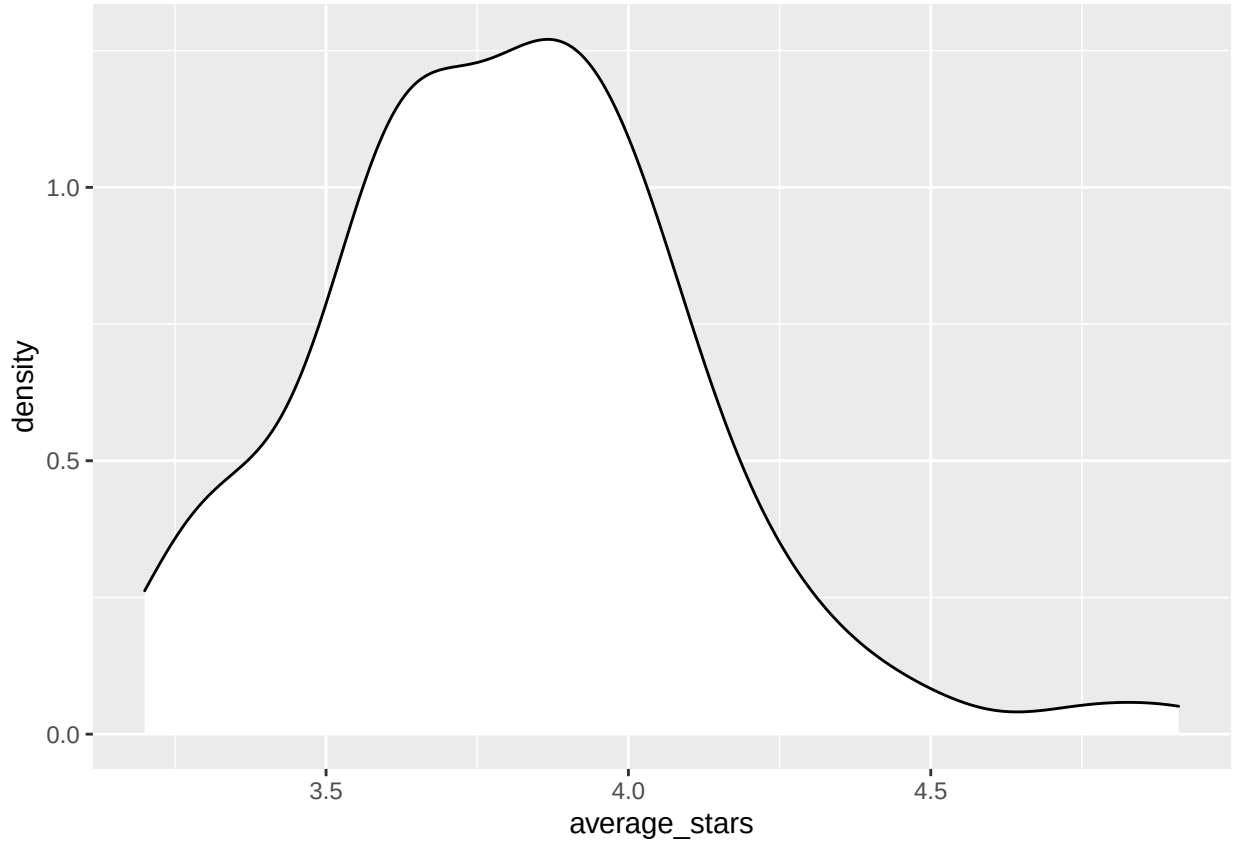
Generating histogram for average user ratings

```
library(ggplot2)
ggplot(user_data, aes(x = average_stars)) + geom_histogram(binwidth = 0.25) +
    xlab("average ratings out of 5") + ylab("frequencies")
```



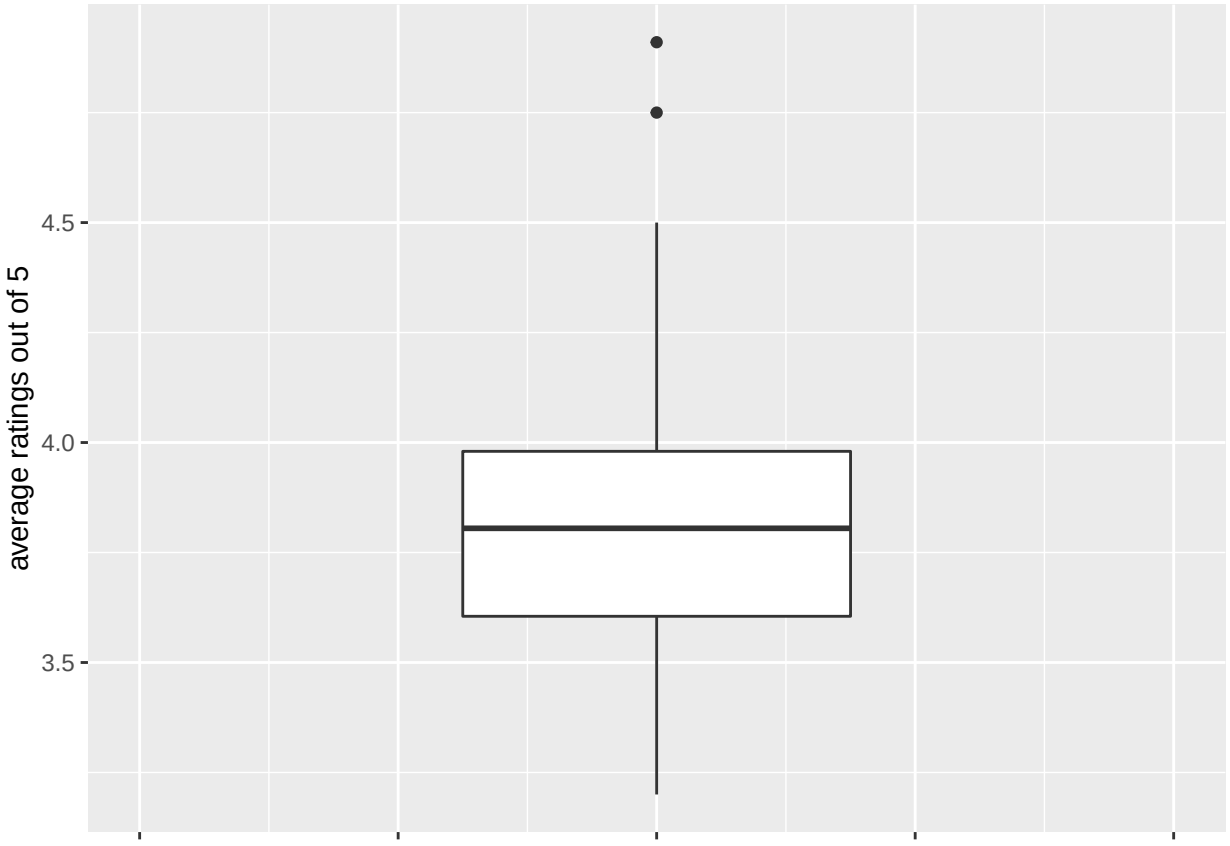
Density plot for average ratings

```
ggplot(user_data, aes(x = average_stars)) + geom_density(fill = "white")
```



Box plot for average ratings

```
ggplot(user_data, aes(y = average_stars)) + geom_boxplot() +
    xlim(-1, 1) + ylab("average ratings out of 5") + theme(axis.text.x = element_blank())
```



Finding averages of compliment recieved columns in user_data:

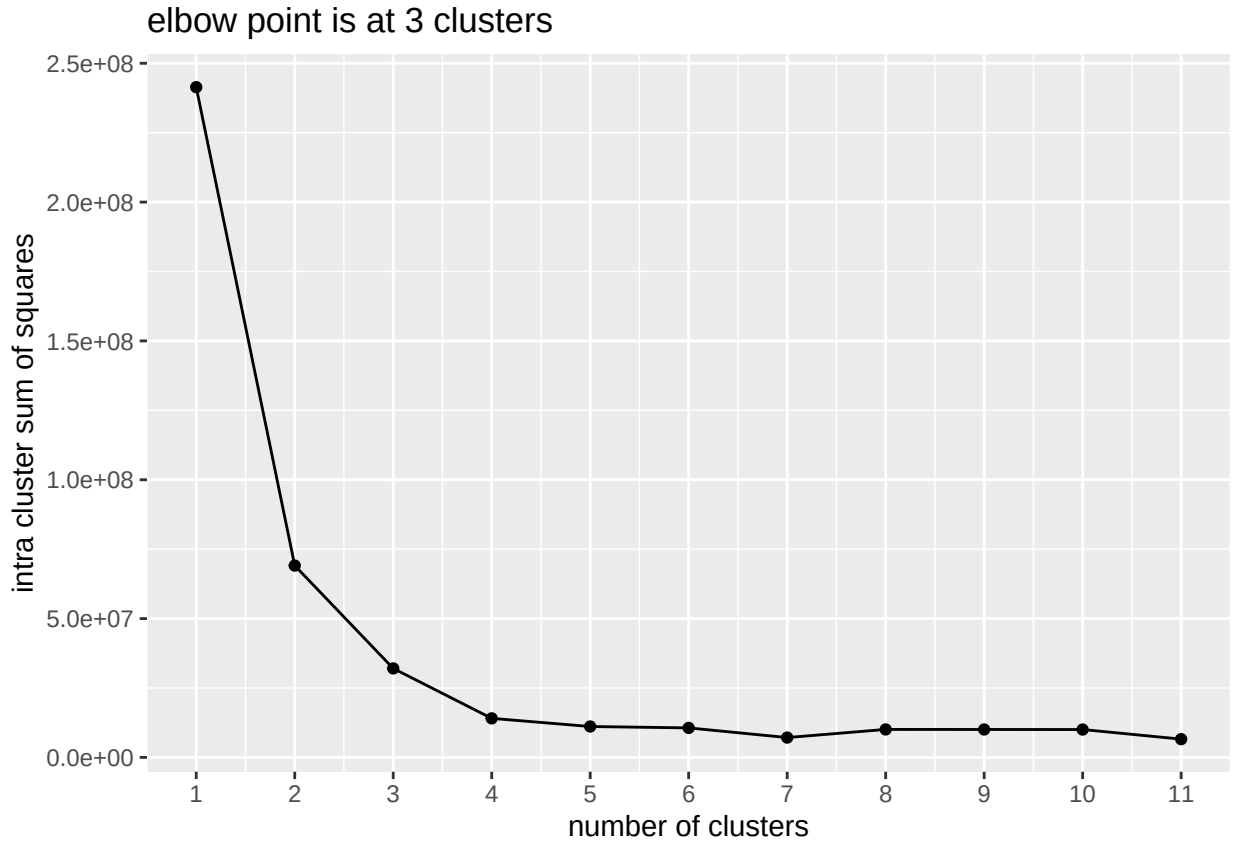
```
user_data_compliments <- dplyr::select(user_data, contains("compliment"))
print(sort(colMeans(user_data_compliments)))
```

```
##    compliment_list compliment_profile    compliment_cute    compliment_more
##              11.91              15.42              18.55              21.19
##  compliment_photos  compliment_writer    compliment_note     compliment_hot
##              58.66              82.45             115.23             185.29
##    compliment_cool   compliment_funny   compliment_plain
##             241.38             241.38             305.71
```

Using K-means to group the user data using all compliments recieved by user; finding optimal number of clusters first using elbow method then silhouette method:

```
library(purrr)
set.seed(243)
```

```
intra_cluster_square_sums <- function(k) {
    invisible(kmeans(user_data_compliments, k, iter.max = 200,
        nstart = 256, algorithm = "Lloyd"))$tot.withinss
}

k_values <- 1:11

intra_cluster_square_sums_values <- map_dbl(k_values, intra_cluster_square_sums)

plot_points = data.frame(x = k_values, y = intra_cluster_square_sums_values)

ggplot(plot_points, aes(x = x, y = y)) + geom_line() + geom_point() +
    xlab("number of clusters") + ylab("intra cluster sum of squares") +
    ggtitle("elbow point is at 3 clusters") + scale_x_continuous(labels = min(plot_points["x"]):max(plo
    breaks = min(plot_points["x"]):max(plot_points["x"]))
```



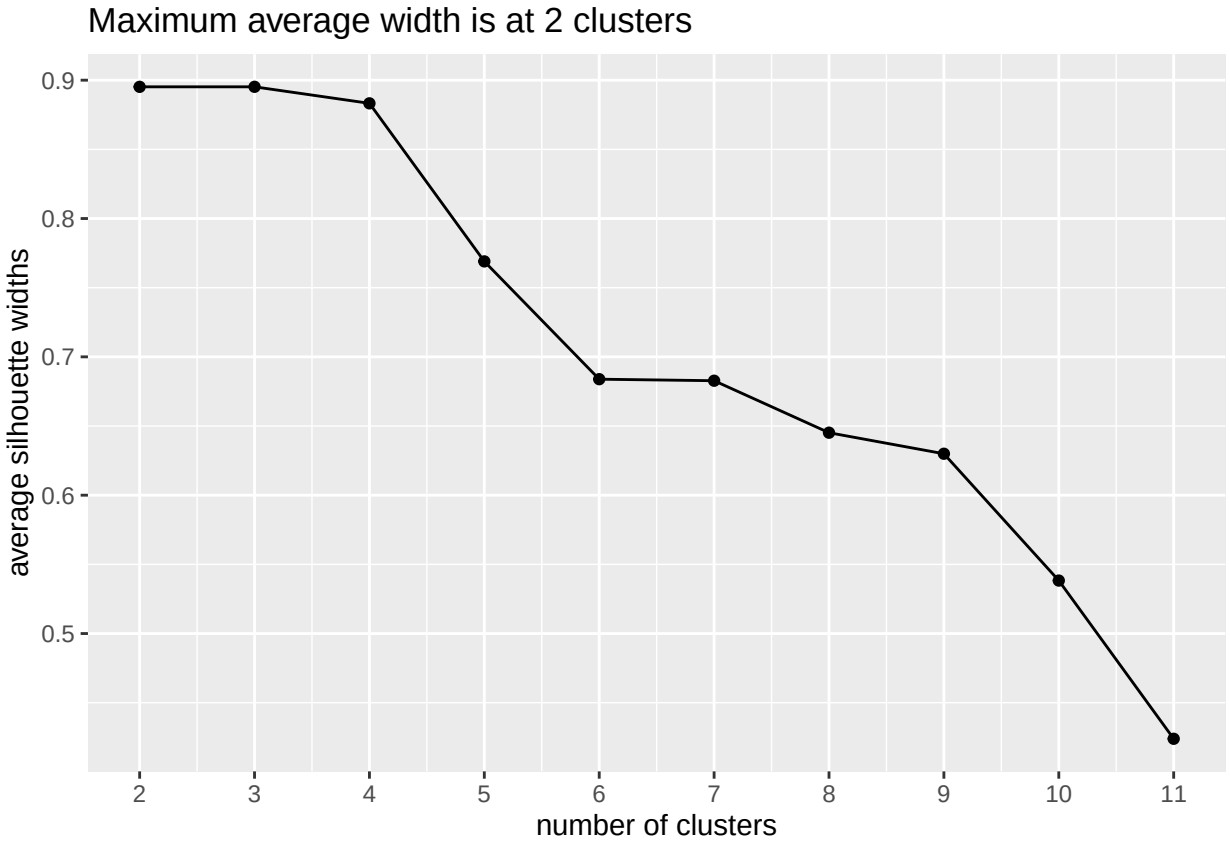

```
library(cluster)
library(gridExtra)
library(grid)

silhouette_width <- function(k) {
```

```
    k_ <- kmeans(user_data_compliments, k, iter.max = 200, nstart = 256,
        algorithm = "Lloyd")
    s_ <- silhouette(k_$cluster, dist(user_data_compliments,
        "euclidean"))
    mean(s_[, 3])
}


k_values = 2:11
avg_widths <- map_dbl(k_values, silhouette_width)
plot_points = data.frame(x = k_values, y = avg_widths)

print(avg_widths)
```

Using silhouette method to find optimal number of clusters:

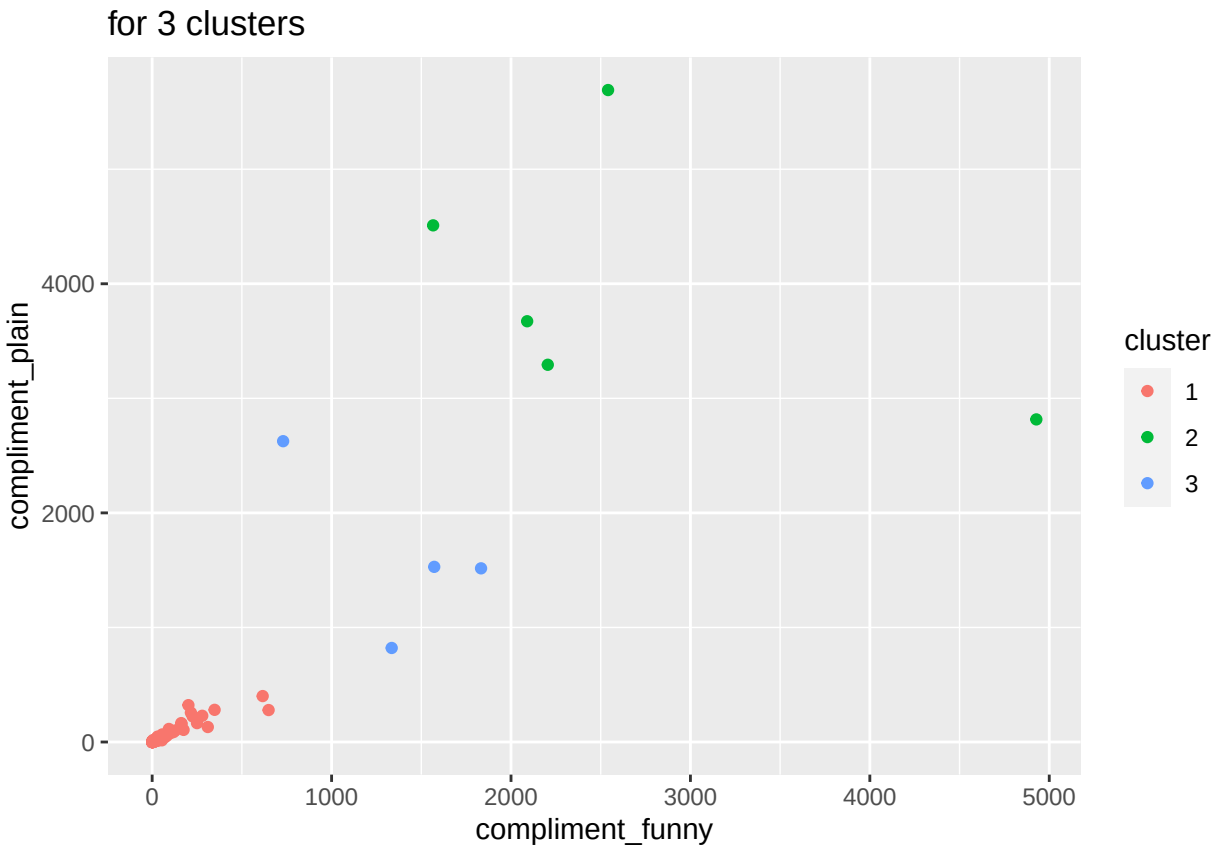
```
##  [1] 0.8951983 0.8952105 0.8832658 0.7690026 0.6838575 0.6827461 0.6451682
##  [8] 0.6299576 0.5382912 0.4239144
```

```
ggplot(plot_points, aes(x = x, y = y)) + geom_line() + geom_point() +
    xlab("number of clusters") + ylab("average silhouette widths") +
    ggtitle("Maximum average width is at 2 clusters") + scale_x_continuous(labels = min(plot_points["x"]
    breaks = min(plot_points["x"]):max(plot_points["x"]))
```

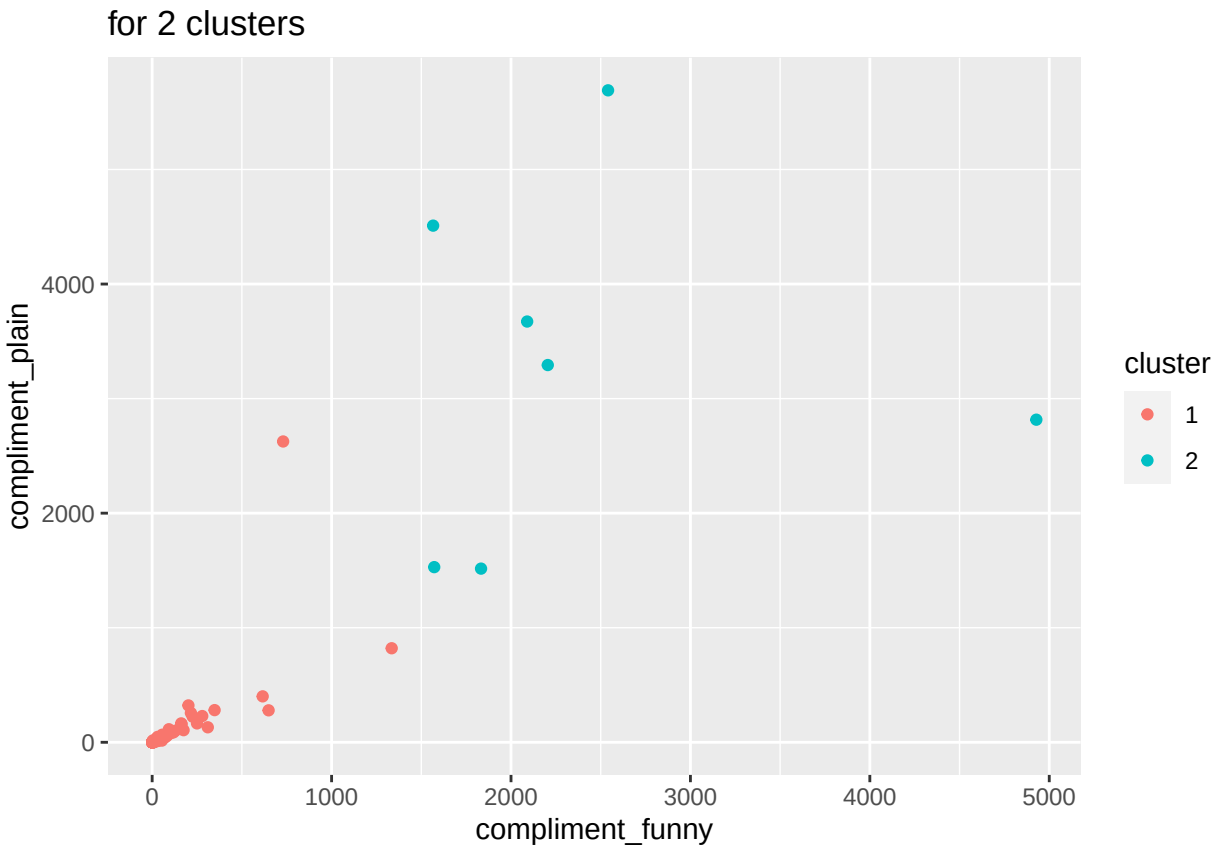


Plotting K-Means clusters in 2d using values for number of clusters computed from above, the compliments with the most mean are used, namely: funny and plain

```
library(plotly)

user_data_compliments$cluster = factor(kmeans(user_data_compliments,
    3)$cluster)

ggplot() + geom_point(data = user_data_compliments, mapping = aes(x = compliment_funny,
    y = compliment_plain, colour = cluster)) + ggtitle("for 3 clusters")
```




```
user_data_compliments$cluster = factor(kmeans(user_data_compliments,
    2)$cluster)

ggplot() + geom_point(data = user_data_compliments, mapping = aes(x = compliment_funny,
    y = compliment_plain, colour = cluster)) + ggtitle("for 2 clusters")
```


for 2 clusters
compliment_plain
4000
2000
0
0
1000
2000
3000
4000
5000
compliment_funny
cluster
1
2