VE572 Project Part 1

Yihao Liu 515370910207

August 3, 2018

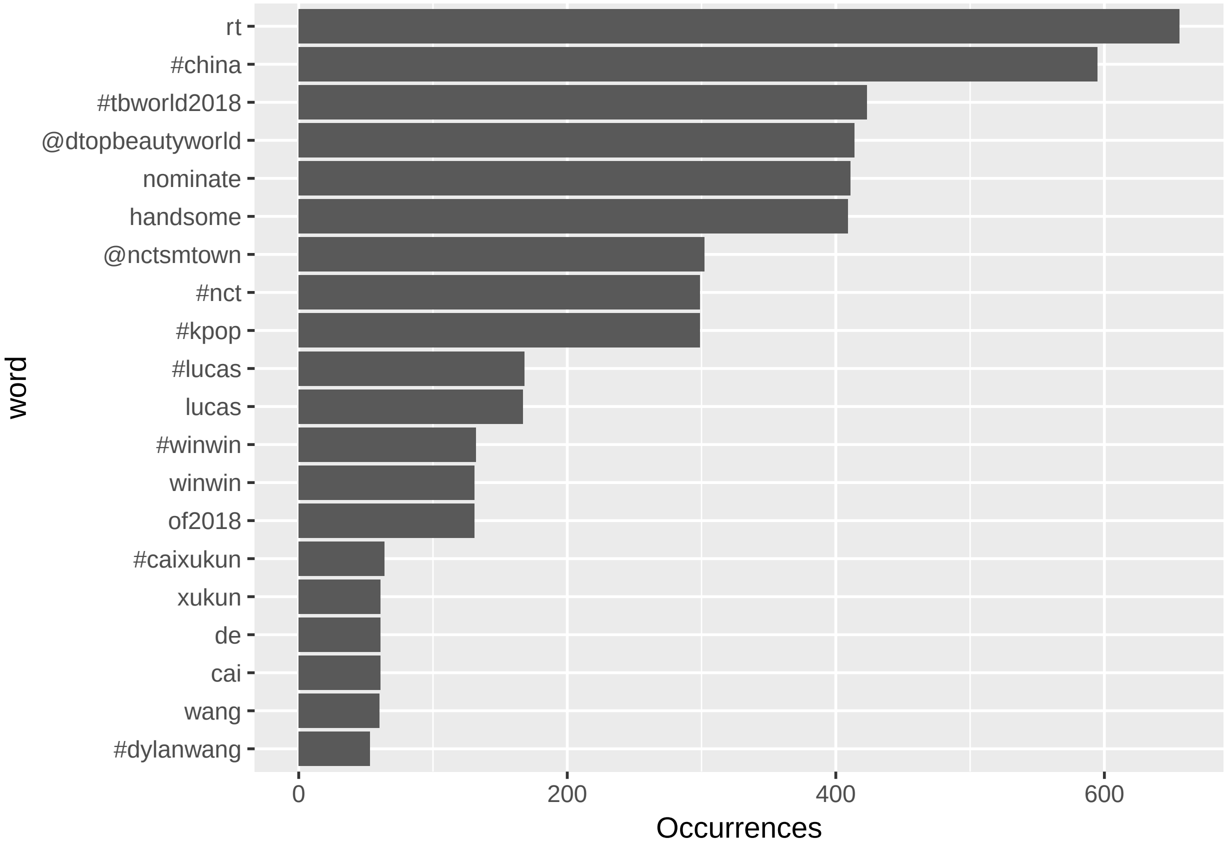
Task 1

(a)

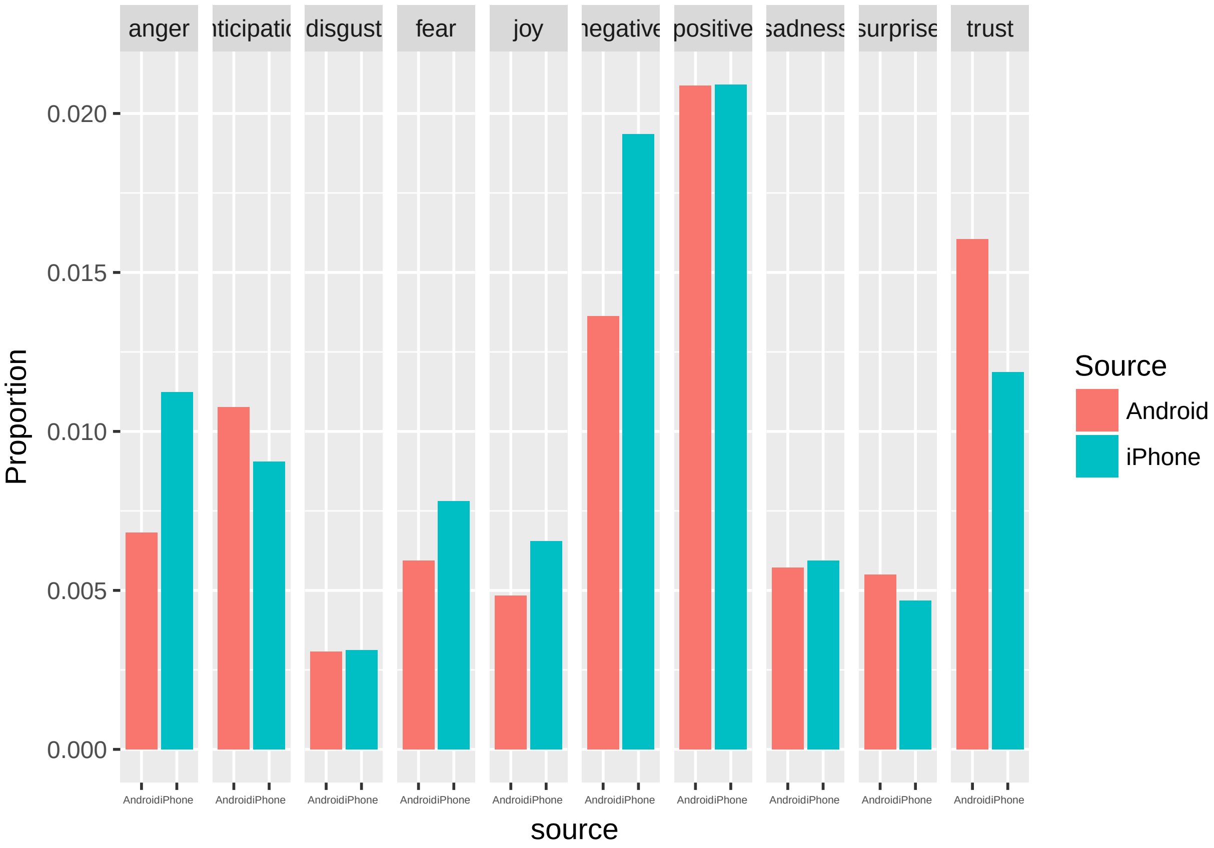
```
library(twitteR)
library(httr)
consumer_key = "bUMOqZlwm3JWqc2TJHutI7YQz"
consumer_secret = "6FORaUFkTSJP7dByIOgfVi8nLXndy9kkifN9G2DUC7loh5wnkR"
access_token = "1006882104150351872-LwKJICzZhxcEBUV4WzDTUnsKhcKABT"
access_secret = "neAA3ShJDPMfFvRdvh4nrmWLkVGm9Hl7NPyomU691stuq"
options(httr_oauth_cache = TRUE)
Sys.setenv(http_proxy="http://127.0.0.1:8123")
setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)
tweets = searchTwitter('#China', resultType="popular", n=1000)
save(tweets, file = 'tweets.Rdata')
```

(b)

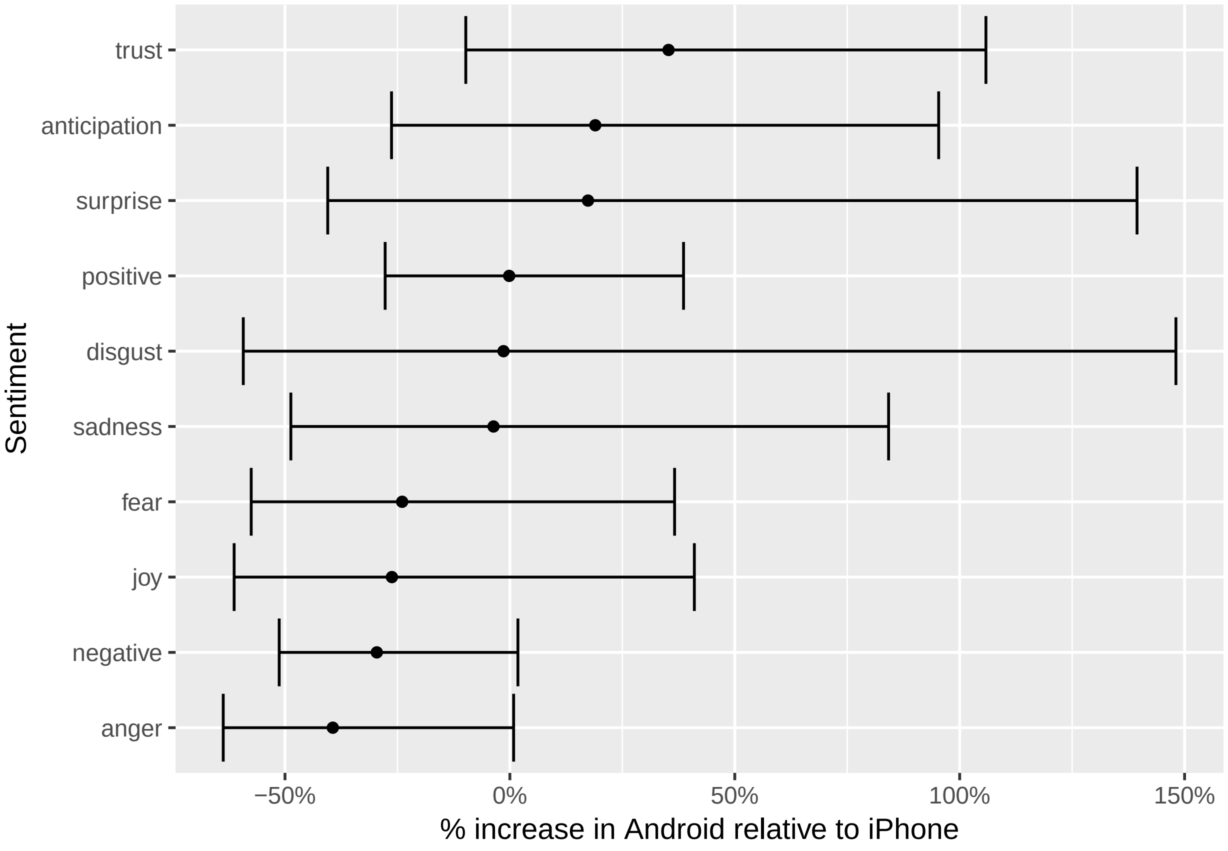
```
# i.
library(twitteR)
library(dplyr)
library(tidyr)
load(file = 'tweets.Rdata')
tweets_tb = as_tibble(purrr::map_dfr(tweets, as.data.frame)) %>%
select(id , statusSource , text, created) %>%
extract(statusSource, "source", "Twitter for (.*?)<") %>%
filter(source %in% c("iPhone", "Android"))

# ii.
library(stringr)
library(tidytext)
library(ggplot2)
reg = "([^A-Za-z\\d#@']|'(?![A-Za-z\\d#@]))"
tweets_tb = tweets_tb %>%
filter(!str_detect(text, '^"')) %>%
mutate(text = str_replace_all(text, "https://t.co/[A-Za-z\\d]+|&amp;", ""))

words = tweets_tb %>%
unnest_tokens(word, text, token = "regex", pattern = reg) %>%
filter (!word %in% stop_words$word, str_detect(word, "[a-z]"))

words %>%
count (word, sort = TRUE) %>%
head(20) %>%
mutate(word = reorder(word, n)) %>%
ggplot(aes(word, n)) +
geom_bar(stat = "identity") +
ylab("Occurrences") +
coord_flip()
```



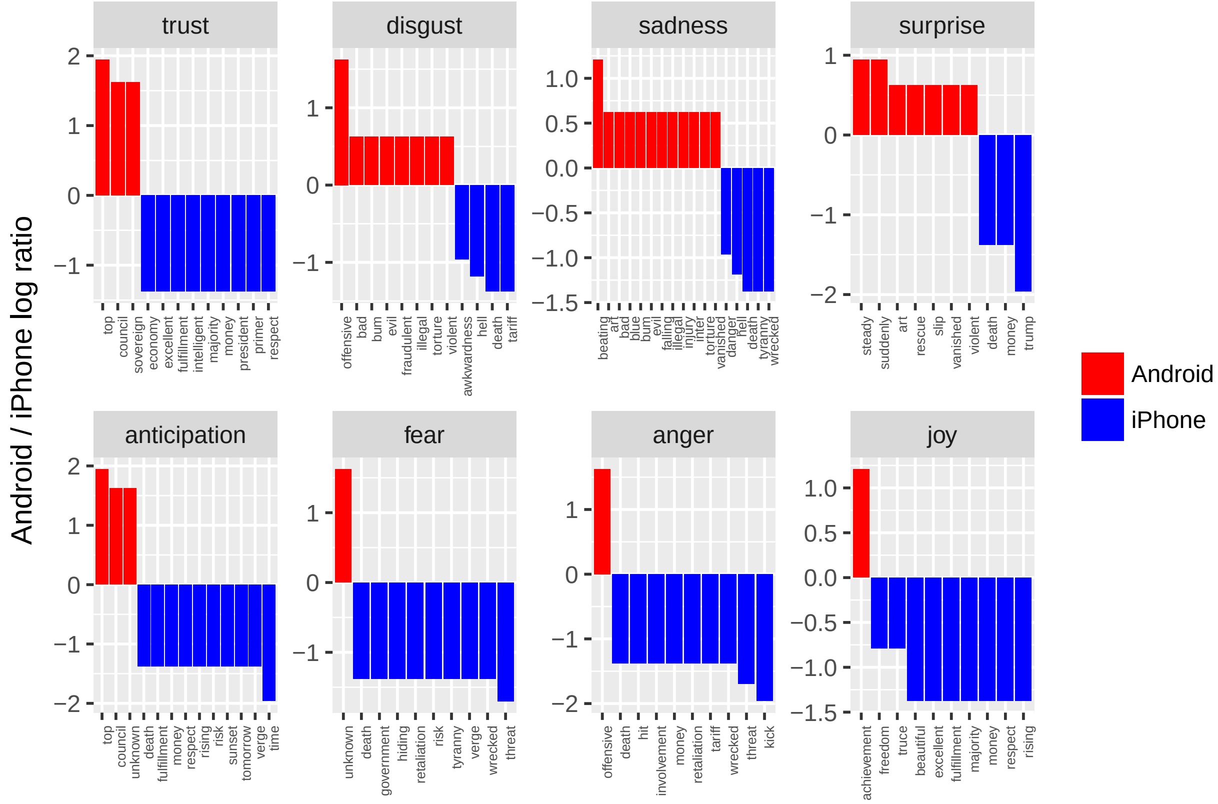
```
# iii.
nrc = sentiments %>%
filter (lexicon == "nrc") %>%
select (word , sentiment)

sources = words %>%
group_by (source) %>%
mutate (total = n ()) %>%
ungroup () %>%
distinct (id , source , total)

words_by_source_sentiment = words %>%
inner_join(nrc , by = "word") %>%
count(sentiment , id) %>%
ungroup() %>%
complete(sentiment , id , fill = list (n = 0)) %>%
inner_join(sources) %>%
group_by(source , sentiment , total) %>%
summarize(counts = sum (n)) %>%
ungroup()

words_by_source_sentiment %>%
ggplot (aes (source , counts / total , fill = source)) +
geom_bar (stat = "identity" , position = "dodge") +
labs (y = "Proportion" , fill = "Source") +
facet_grid ( ~ sentiment) +
theme(axis.text.x = element_text(size = 4))
```



```
sentiment_differences =
words_by_source_sentiment %>%
group_by (sentiment) %>%
do (broom::tidy (poisson.test (.$counts , .$total)))

sentiment_differences %>%
ungroup () %>%
mutate (sentiment = reorder (sentiment , estimate)) %>%
mutate_at (c ("estimate" , "conf.low" , "conf.high") , funs (. - 1)) %>%
ggplot (aes (estimate , sentiment)) +
geom_point () +
geom_errorbarh (aes (xmin = conf.low , xmax = conf.high)) +
scale_x_continuous (labels = scales::percent_format ()) +
labs(x = "% increase in Android relative to iPhone", y = "Sentiment")
```



```
android_iphone_ratios = words %>%
count (word , source) %>%
spread (source, n , fill = 0) %>%
mutate_at (c ("Android" , "iPhone") , funs ((. + 1) / sum (. + 1))) %>%
mutate (logratio = log2 (Android / iPhone)) %>%
arrange (desc (logratio))

android_iphone_ratios %>%
inner_join (nrc , by = "word") %>%
filter (!sentiment %in% c ("positive" , "negative")) %>%
mutate (sentiment = reorder (sentiment ,-logratio),
word = reorder (word ,-logratio)) %>%
group_by (sentiment) %>%
top_n (10 , abs (logratio)) %>%
ungroup () %>%
ggplot (aes (word , logratio , fill = logratio < 0)) +
facet_wrap ( ~ sentiment , scales = "free" , nrow = 2) +
geom_bar (stat = "identity") +
theme (axis.text.x = element_text (
size = 5,
angle = 90 ,
hjust = 1
)) +
labs (x = "" , y = "Android / iPhone log ratio") +
scale_fill_manual (
name = " " ,
values = c ("red" , "blue"),
labels = c ("Android" , "iPhone")
)
```



```
# iv. (visulization has been implemented in the above parts)
```

Task 3

```
# (a)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
```

```
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, month, week, year
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```
song_tbl = fread("msd_onevalue.csv", header = TRUE)

# (b)
library(dplyr)
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 minutes 25 seconds
##     H2O cluster timezone:       Asia/Shanghai
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.20.0.4
##     H2O cluster version age:    5 days
##     H2O cluster name:           H2O_started_from_R_liu_aoo481
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.33 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.4.4 (2018-03-15)
```

```
selection = c("artist_familiarity","artist_hotttnesss","duration","loudness","tempo","song_id")
feature = c("artist_familiarity", "artist_hotttnesss", "duration", "loudness", "tempo")
song.h2o = as.h2o(select(song_tbl, selection))
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
(song.kmeans1 = h2o.kmeans(training_frame = song.h2o, k = 1, x = feature))
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
## Model Details:
## ==============
##
## H2OClusteringModel: kmeans
## Model ID:  KMeans_model_R_1533545724074_3
## Model Summary:
##   number_of_rows number_of_clusters number_of_categorical_columns
## 1        1000000                  1                             0
##   number_of_iterations within_cluster_sum_of_squares total_sum_of_squares
## 1                    2                 4999798.00000        4999798.00000
##   between_cluster_sum_of_squares
## 1                        0.00000
##
##
## H2OClusteringMetrics: kmeans
## ** Reported on training data. **
##
##
## Total Within SS:  4999798
## Between SS:  0
## Total SS:  4999798
## Centroid Statistics:
##   centroid          size within_cluster_sum_of_squares
## 1        1 1000000.00000                 4999797.97948
```

```r
(song.kmeans2 = h2o.kmeans(training_frame = song.h2o, k = 2, x = feature))
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
## Model Details:
## ==============
##
## H2OClusteringModel: kmeans
## Model ID:  KMeans_model_R_1533545724074_4
## Model Summary:
##   number_of_rows number_of_clusters number_of_categorical_columns
## 1        1000000                  2                             0
##   number_of_iterations within_cluster_sum_of_squares total_sum_of_squares
## 1                   10                 4521126.51187        4999798.00000
##   between_cluster_sum_of_squares
## 1                   478671.48813
##
##
## H2OClusteringMetrics: kmeans
## ** Reported on training data. **
##
```

```
##
## Total Within SS:  4489866
## Between SS:  509932.4
## Total SS:  4999798
## Centroid Statistics:
##   centroid         size within_cluster_sum_of_squares
## 1        1 872495.00000                 3561156.21964
## 2        2 127505.00000                  928709.34725
```

```
(song.kmeans3 = h2o.kmeans(training_frame = song.h2o, k = 3, x = feature))
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
## Model Details:
## ==============
##
## H2OClusteringModel: kmeans
## Model ID:  KMeans_model_R_1533545724074_5
## Model Summary:
##   number_of_rows number_of_clusters number_of_categorical_columns
## 1        1000000                  3                             0
##   number_of_iterations within_cluster_sum_of_squares total_sum_of_squares
## 1                   10               3540214.32960        4999798.00000
##   between_cluster_sum_of_squares
## 1                  1459583.67040
##
##
## H2OClusteringMetrics: kmeans
## ** Reported on training data. **
##
##
## Total Within SS:  3535451
## Between SS:  1464347
## Total SS:  4999798
## Centroid Statistics:
##   centroid         size within_cluster_sum_of_squares
## 1        1 229992.00000                 1213042.77047
## 2        2 345425.00000                 1088355.17011
## 3        3 424583.00000                 1234052.68797
```

```
(song.kmeans4 = h2o.kmeans(training_frame = song.h2o, k = 4, x = feature))
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
## Model Details:
## ==============
##
## H2OClusteringModel: kmeans
## Model ID:  KMeans_model_R_1533545724074_6
## Model Summary:
##   number_of_rows number_of_clusters number_of_categorical_columns
## 1        1000000                  4                             0
##   number_of_iterations within_cluster_sum_of_squares total_sum_of_squares
## 1                   10                 3136239.85568        4999798.00000
##   between_cluster_sum_of_squares
## 1                  1863558.14432
##
##
## H2OClusteringMetrics: kmeans
## ** Reported on training data. **
##
##
## Total Within SS:  3127324
## Between SS:  1872474
## Total SS:  4999798
## Centroid Statistics:
##   centroid         size within_cluster_sum_of_squares
## 1        1 494228.00000                 1232746.86077
## 2        2  78459.00000                  444598.46045
## 3        3 211867.00000                  794494.73742
## 4        4 215446.00000                  655484.21249
```

```
(song.kmeans5 = h2o.kmeans(training_frame = song.h2o, k = 5, x = feature))
```

```
##
  |
  |                                                                 |   0%
  |
  |====================================================             |  80%
  |
  |=================================================================| 100%
```

```
## Model Details:
## ==============
##
## H2OClusteringModel: kmeans
## Model ID:  KMeans_model_R_1533545724074_7
## Model Summary:
##   number_of_rows number_of_clusters number_of_categorical_columns
## 1        1000000                  5                             0
##   number_of_iterations within_cluster_sum_of_squares total_sum_of_squares
## 1                   10                 2950706.39640        4999798.00000
##   between_cluster_sum_of_squares
## 1                  2049091.60360
##
##
## H2OClusteringMetrics: kmeans
## ** Reported on training data. **
##
##
## Total Within SS:  2948977
## Between SS:  2050821
## Total SS:  4999798
## Centroid Statistics:
```

```
##   centroid         size within_cluster_sum_of_squares
## 1        1 294066.00000                  999289.42928
## 2        2   5524.00000                   94049.36238
## 3        3 172644.00000                  568132.44150
## 4        4 381617.00000                  913127.56615
## 5        5 146149.00000                  374378.31744
```

```
# (c)
# I found that the k-means of different runnings are different, so the initial state is random.
```