Practice5S24

Sofia Guttmann

2024-03-08

Practice5 - Due Thursday, 3/7 by midnight to Gradescope

Reminder: Practice assignments may be completed working with other individuals.

Reading

The associated reading for the week is Chapter 19 and Section 12.1.

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (anything besides our textbook, course materials in the repo, labs, R help menu) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

-

I used the following sources to help complete this assignment:

Source(s) and corresponding problem(s)

-

1 - MDSR 12.6 (modified)

"Baseball players are voted into the Hall of Fame by the members of the Baseball Writers of America Association. Quantitative criteria are used by the voters, but they are also allowed wide discretion. The following code identifies the position players (not pitchers) who have been elected to the Hall of Fame and tabulates a few basic statistics, include their number of career hits (`tH`), home runs (`tHR`), runs batted in (`tRBI`), and stolen bases (`tSB`)." Only players with more than 1000 total hits are included as a way to obtain the position players only (not pitchers).

```
hof <- Batting %>%
  group_by(playerID) %>%
  inner_join(HallOfFame, by = "playerID") %>%
  filter(inducted == "Y" & votedBy == "BBWAA") %>%
  summarize(tH = sum(H), tHR = sum(HR), tRBI = sum(RBI), tSB = sum(SB)) %>%
  filter(tH > 1000)
```

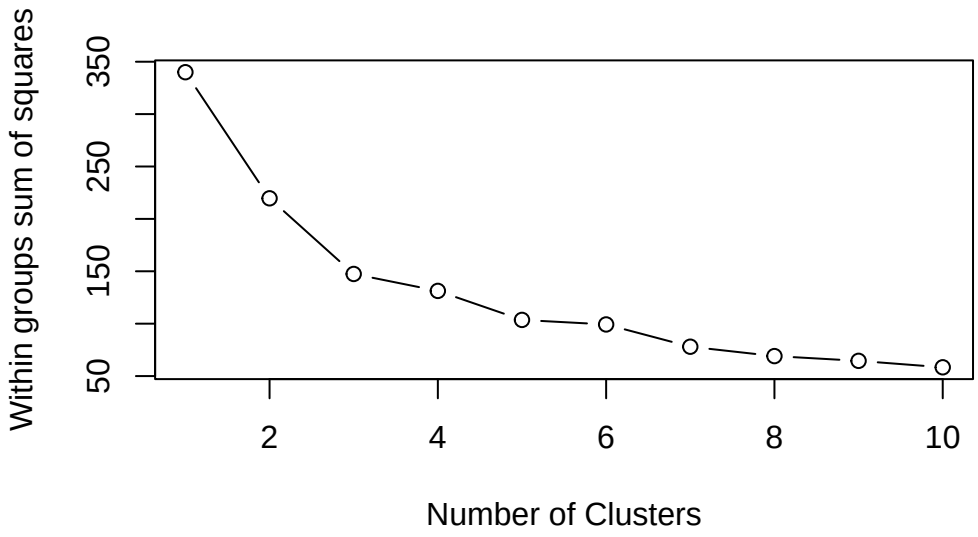
```
Warning in inner_join(., HallOfFame, by = "playerID"): Detected an unexpected many-to-many re
i Row 5 of `x` matches multiple rows in `y`.
i Row 72 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

- Use the `kmeans()` function to perform a cluster analysis on these players.
- Explain your choice of k, the number of clusters.
- Describe the properties that seem common to each cluster in your solution.
- Include at least one visual that helps explore the clusters found.
- Your solution should include some discussion of whether or not you chose to scale the variables and why. (You should determine whether or not you need to scale before clustering.)
- Remember that your solution must be reproducible. (Hint: this means you need to do something in your code.)

Solution:

```
library(dplyr)
library(ggplot2)

hof_data <- hof[, c("tH", "tHR", "tRBI", "tSB")]

scaled_data <- scale(hof_data)
```

```r
wss <- (nrow(scaled_data)-1)*sum(apply(scaled_data,2,var))
for (i in 2:10) wss[i] <- sum(kmeans(scaled_data, centers=i)$withinss)
plot(1:10, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```

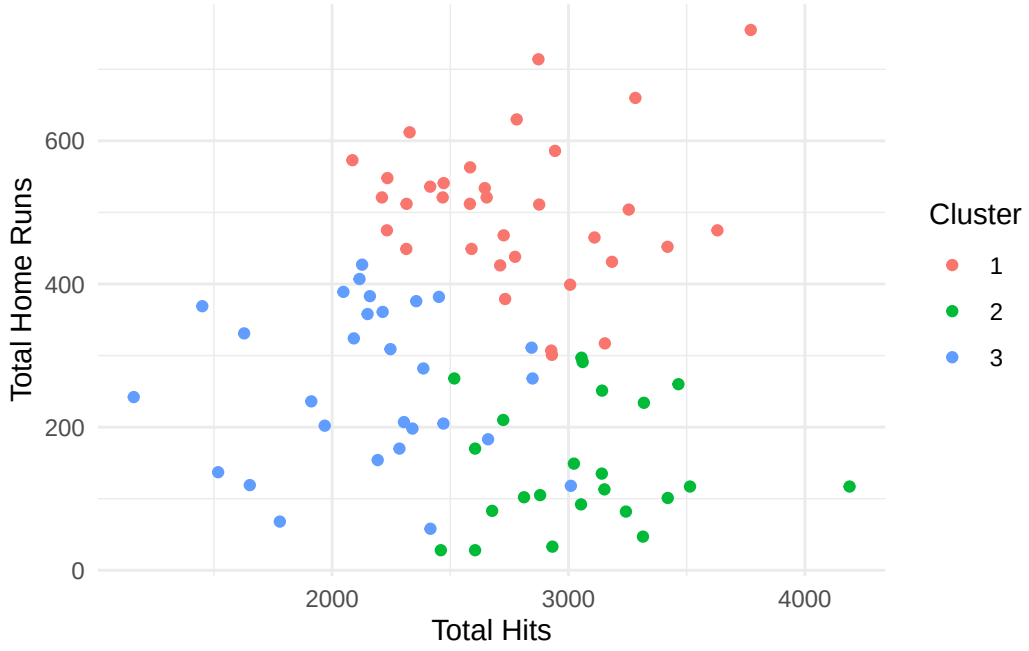



```r
k <- 3
set.seed(123) # For reproducibility
kmeans_result <- kmeans(scaled_data, centers = k)

hof$cluster <- kmeans_result$cluster

cluster_summary <- hof %>%
  group_by(cluster) %>%
  summarize(
    avg_hits = mean(tH),
    avg_home_runs = mean(tHR),
    avg_runs_batted_in = mean(tRBI),
    avg_stolen_bases = mean(tSB),
    players_count = n()
  )

print(cluster_summary)
```

```
# A tibble: 3 x 6
  cluster avg_hits avg_home_runs avg_runs_batted_in avg_stolen_bases
```

```
    <int>    <dbl>         <dbl>              <dbl>            <dbl>
1       1    2771.          502.               1724.            125.
2       2    3057.          144.               1205.            542.
3       3    2165.          261.               1206.             93.9
# i 1 more variable: players_count <int>
```

```r
ggplot(hof, aes(x = tH, y = tHR, color = as.factor(cluster))) +
  geom_point() +
  labs(x = "Total Hits", y = "Total Home Runs", color = "Cluster") +
  theme_minimal()
```



2 - Trump Tweets

David Robinson, Chief Data Scientist at DataCamp, wrote a blog post "Text analysis of Trump's tweets confirms he writes only the (angrier) Android half". He provides a dataset with over 1,500 tweets from the account realDonaldTrump between 12/14/2015 and 8/8/2016. We'll use this dataset to explore the tweeting behavior of @realDonaldTrump during this time period.

First, read in the file. Note that there is a `TwitteR` package which provides an interface to the Twitter web API. We'll use this R dataset David Robinson created using that package so that you don't have to set up Twitter authentication.

```
# the .rda file is also provided if this website ever breaks
load(url("http://varianceexplained.org/files/trump_tweets_df.rda"))
```

> part a - Wrangling! There are a number of variables in the dataset we won't need. First, confirm that all the observations in the dataset are from the screen-name *realDonaldTrump*. Then, create a new dataset called `tweets` that only includes the variables `text`, `created` and `statusSource`.

Solution:

```
load(url("http://varianceexplained.org/files/trump_tweets_df.rda"))

str(trump_tweets_df)
```

```
tibble [1,512 x 16] (S3: tbl_df/tbl/data.frame)
 $ text         : chr [1:1512] "My economic policy speech will be carried live at 12:15 P.M.
 $ favorited    : logi [1:1512] FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ favoriteCount: num [1:1512] 9214 6981 15724 19837 34051 ...
 $ replyToSN    : chr [1:1512] NA NA NA NA ...
 $ created      : POSIXct[1:1512], format: "2016-08-08 15:20:44" "2016-08-08 13:28:20" ...
 $ truncated    : logi [1:1512] FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ replyToSID   : logi [1:1512] NA NA NA NA NA NA ...
 $ id           : chr [1:1512] "762669882571980801" "762641595439190016" "762439658911338496
 $ replyToUID   : chr [1:1512] NA NA NA NA ...
 $ statusSource : chr [1:1512] "<a href=\"http://twitter.com/download/android\" rel=\"nofollo
 $ screenName   : chr [1:1512] "realDonaldTrump" "realDonaldTrump" "realDonaldTrump" "realDon
 $ retweetCount : num [1:1512] 3107 2390 6691 6402 11717 ...
 $ isRetweet    : logi [1:1512] FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ retweeted    : logi [1:1512] FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ longitude    : chr [1:1512] NA NA NA NA ...
```

```
 $ latitude     : chr [1:1512] NA NA NA NA ...
```

```
unique(trump_tweets_df$screen_name)  # Check unique screen names
```

```
Warning: Unknown or uninitialised column: `screen_name`.
```

```
NULL
```

```
tweets <- trump_tweets_df[, c("text", "created", "statusSource")]

head(tweets)
```

```
# A tibble: 6 x 3
  text                                          created             statusSource
  <chr>                                         <dttm>              <chr>
1 "My economic policy speech will be carried l~ 2016-08-08 15:20:44 "<a href=\"~
2 "Join me in Fayetteville, North Carolina tom~ 2016-08-08 13:28:20 "<a href=\"~
3 "#ICYMI: \"Will Media Apologize to Trump?\" ~ 2016-08-08 00:05:54 "<a href=\"~
4 "Michael Morell, the lightweight former Acti~ 2016-08-07 23:09:08 "<a href=\"~
5 "The media is going crazy. They totally dist~ 2016-08-07 21:31:46 "<a href=\"~
6 "I see where Mayor Stephanie Rawlings-Blake ~ 2016-08-07 13:49:29 "<a href=\"~
```

part b - Using the `statusSource` variable, compute the number of tweets from each source. How many different sources are there? How often are each used?

Hint: You could answer the questions with a nice table printed to the screen.

Solution:

```
tweets_by_source <- table(tweets$statusSource)

num_sources <- length(tweets_by_source)

cat("Number of different sources:", num_sources, "\n")
```

```
Number of different sources: 5
```

```
print(tweets_by_source)
```

```
                        <a href="http://instagram.com" rel="nofollow">Instagram</a>
                                                                                    1
                  <a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
                                                                                  120
   <a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for iPad</a>
                                                                                    1
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>
                                                                                  762
  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
                                                                                  628
```

part c - We're going to compare the language used between the Android and iPhone sources, so we only want to keep tweets coming from those sources. Explain what the `extract()` function (from the **tidyverse** package) is doing below. Include in your own words what each argument is doing.

```
tweets <- tweets %>%
  extract(col = statusSource, into = "source",
          regex = "Twitter for (.*)<",
          remove = FALSE) %>%
  filter(source %in% c("Android", "iPhone"))
```

Solution:

The 'extract()' function from the 'tidyverse' package is utilized to isolate a specific pattern from a column within a dataframe and create a new column to store this pattern. Here's a breakdown of each argument:

'col = statusSource': This indicates the column from which the extraction will be carried out. In this context, it refers to the 'statusSource' column that contains the source information for each tweet.

'into = "source"': This specifies the name of the newly created column where the isolated pattern will be stored. Here, it establishes a new column labeled 'source'.

'regex = "Twitter for (.*)<"': This defines the regular expression pattern used to extract information from the 'statusSource' column. In this pattern:

'Twitter for ': This segment matches the literal phrase "Twitter for".

'(.*)': This segment captures any characters ('.') occurring zero or more times ('*') within parentheses '()'. These captured characters constitute the portion of the pattern to be isolated and saved in the new column.

‘<‘: This matches the literal character “<” that appears at the end of the source string, signaling the endpoint for the extraction.

‘remove = FALSE‘: This determines whether to retain the original column (‘statusSource‘) following the extraction. By setting it to ‘FALSE‘, the original column remains intact within the dataframe.

Subsequent to extracting the source information into the new ‘source‘ column, the ‘filter()‘ function is employed to retain only the tweets originating from “Android” or “iPhone” sources. This is accomplished by filtering the dataframe based on whether the ‘source‘ column contains either “Android” or “iPhone”.

> part d - How does the language of the tweets differ by source? Create a word cloud for the top 50 words used in tweets sent from the Android. Create a second word cloud for the top 50 words used in tweets sent from the iPhone. How do these word clouds compare? (Are there some common words frequently used from both sources? Are the most common words different between the sources?)

Note: Don’t forget to remove stop words before creating the word cloud. Also remove the terms “https” and “t.co”.

Solution:

```
library(tm)
library(wordcloud)
library(dplyr)
library(tidytext)


load(url("http://varianceexplained.org/files/trump_tweets_df.rda"))


data(stop_words)


custom_stopwords <- c(stop_words$word, "https", "t.co")


android_tweets <- tweets %>%
  filter(statusSource == "Twitter for Android") %>%
  mutate(text = tolower(text)) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  filter(!word %in% custom_stopwords)
```

```
iphone_tweets <- tweets %>%
  filter(statusSource == "Twitter for iPhone") %>%
  mutate(text = tolower(text)) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  filter(!word %in% custom_stopwords)


set.seed(123) # for reproducibility

android_wordcloud_data <- android_tweets %>%
  count(word) %>%
  arrange(desc(n)) %>%
  head(50)

android_wordcloud <- with(android_wordcloud_data,
                          wordcloud(words = word, freq = n, max.words = 50, colors = brewer.

iphone_wordcloud_data <- iphone_tweets %>%
  count(word) %>%
  arrange(desc(n)) %>%
  head(50)

iphone_wordcloud <- with(iphone_wordcloud_data,
                         wordcloud(words = word, freq = n, max.words = 50, colors = brewer.pa


par(mfrow=c(1,2))
android_wordcloud
title(main = "Top 50 Words in Tweets from Android")

iphone_wordcloud
title(main = "Top 50 Words in Tweets from iPhone")
```

part e - Consider the sentiment. Compute the proportion of words among the tweets within each source classified as "angry" and the proportion of words classified as "joy" based on the NRC lexicon. How does the proportion of "angry" and "joy" words compare between the two sources? What about "positive" and "negative" words?

Solution:

```
library(tidyr)


nrc_lexicon <- get_sentiments("nrc")

compute_proportion <- function(tweets_data) {
  tweets_sentiments <- tweets_data %>%
    inner_join(nrc_lexicon, by = "word") %>%
    select(-word) %>%
    group_by(source, sentiment) %>%
    summarize(count = n()) %>%
    pivot_wider(names_from = sentiment, values_from = count, values_fill = 0) %>%
    mutate(total_words = rowSums(across(-source))) %>%
    mutate(across(angry:negative, ~ . / total_words))

  return(tweets_sentiments)
}

android_sentiments <- compute_proportion(android_tweets)

iphone_sentiments <- compute_proportion(iphone_tweets)

print("Proportions of words by sentiment in Android tweets:")
print(android_sentiments)

print("Proportions of words by sentiment in iPhone tweets:")
print(iphone_sentiments)
```

> part f - Lastly, based on your responses above, do you think there is evidence to support Robinson's claim that Trump only writes the Android half of the tweets from realDonaldTrump? In 2-4 sentences, please explain.

Solution: There is evidence to suggest that there are differences between tweets posted from Android and iPhone sources. The proportions of certain feelings, such as "angry" and "positive," differ between the two sources, indicating potential variations in the tone and content of tweets. However, further analysis would be necessary to definitively support Robinson's claim that Trump exclusively writes the "angrier" tweets from the Android half.