Practice5

Sebastian Montesinos

Due by midnight, Friday, March 25

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 - Justices of the Supreme Court of the United States

part a - Confirm that the following Wikipedia page allows automated scraping:

https://en.wikipedia.org/wiki/List_of_justices_of_the_Supreme_Court_of_the_United_States

Solution:

```
url <- "https://en.wikipedia.org/wiki/List_of_justices_of_the_Supreme_Court_of_the_United_States"

robotstxt::paths_allowed(url)
```

```
##  en.wikipedia.org
```

```
## [1] TRUE
```

part b - Scrape!

Go to the List of Justices of the Supreme Court of the United States. Create a new R script called "scrape-justices.R", and scrape the table of justices. Use `janitor::clean_names()` to tidy the names of the columns (do not do any additional wrangling beyond this for now), then write the final data frame to a csv file called "justices.csv" in the "data" folder using the `write_csv()` function. Commit and push both files in addition to this Rmd file when ready.

Solution:

```
url <- "https://en.wikipedia.org/wiki/List_of_justices_of_the_Supreme_Court_of_the_United_States"

justice_table <- url %>%
  read_html() %>%
  html_elements("#mw-content-text > div.mw-parser-output > table") %>%
  purrr::pluck(2) %>%
  html_table() %>%
  # Clean up variable names
  janitor::clean_names()
```

part c - Load "justices.csv" into this file using the `read_csv()` function. Then, modify the code below as needed and run the code to create the variable `tenure_years` (a numeric variable containing each justice's time spent on the bench). Create a visualization to show the distribution of tenure length of U.S. Supreme Court judges. Interpret the plot.

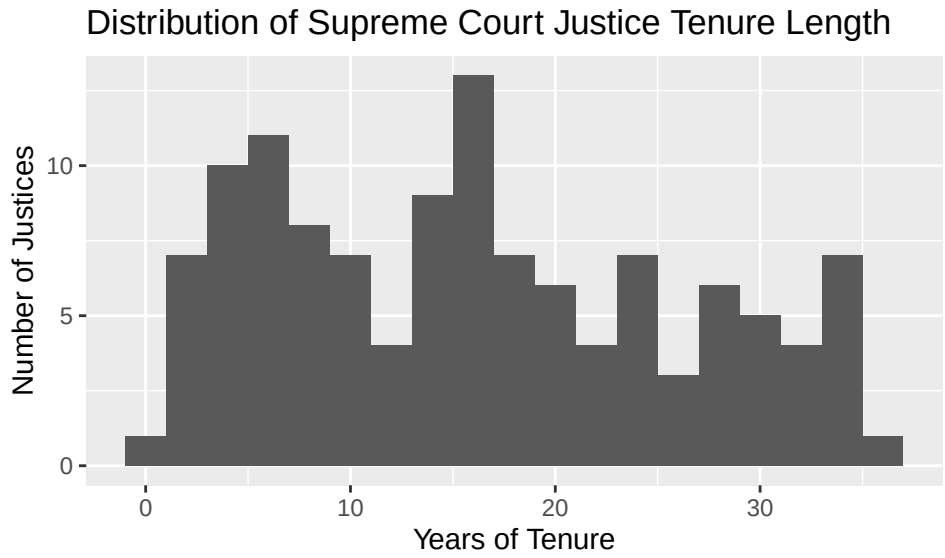
Solution: The histogram reveals that justices spend a wide variety of time on the supreme court bench. A large cluster of justices spend around 5-8 years on the bench, and another large cluster spent 14-15 years on the bench. 16-17 years appears to be the most common amount of time to spend on the bench. There are less justices who spend beyond 20 years on the bench but still quite a significant amount do.

```
# Remove this eval = FALSE before submitting!
justices <- read_csv("justices.csv")
```

```
## Rows: 121 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr (10): justice, justice_2, justice_3, state_c, position, succeeded, date_...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Run after loading your justices.csv file
justices <- justices %>%
  # Remove extra line that comes in at end of table
  filter(justice != "Justice") %>%
  # Some justices served <1 year; add "0 years," to make separating easier
  mutate(tenure_length = case_when(str_detect(tenure_length_d, "year") ~ tenure_length_d,
                                   TRUE ~ paste0("0 years, ", tenure_length_d))) %>%
  separate(tenure_length, into = c("years_char", "days_char"),
           sep = ", ") %>%
  mutate(tenure_years = parse_number(years_char) + (parse_number(days_char)/365)) %>%
  # Make date_confirmed_vote into date variable
  separate(date_confirmed_vote, into = c("date_confirmed", "vote"),
           sep = "\\(") %>%
  mutate(date_confirmed = lubridate::mdy(date_confirmed))

g <- ggplot(data = justices, aes(x = tenure_years)) +
  geom_histogram(binwidth = 2) +
  labs(title = "Distribution of Supreme Court Justice Tenure Length",
       x = "Years of Tenure", y = "Number of Justices")
g
```



2 - 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 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)
```

- 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.
- Remember that your solution must be reproducible. (Hint: this means you need to do something in your code.)

Solution: I used an elbow graph to look at how the amount of clusters affected the WGSS. Since 4 clusters is the point at which adding more clusters barely decreases the WGSS, I chose 4 as the number of clusters to use. I also chose to scale my variables for this analysis. The quantitative magnitude of career hits and runs batted is much greater than that of home runs and stolen bases, meaning that without scaling the analysis would weigh these two measures more strongly. The analysis revealed that one cluster (represented by the green in the graph) is marked by low career hits, home runs, and runs batted. A second cluster (the red) has very high career hits but low home runs, and high stolen bases. The blue cluster is high on every measure except stolen bases. Finally, the purple cluster is middling on all the measures and low on stolen bases.

```
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.1.3
```

```
## Package 'mclust' version 5.4.9
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

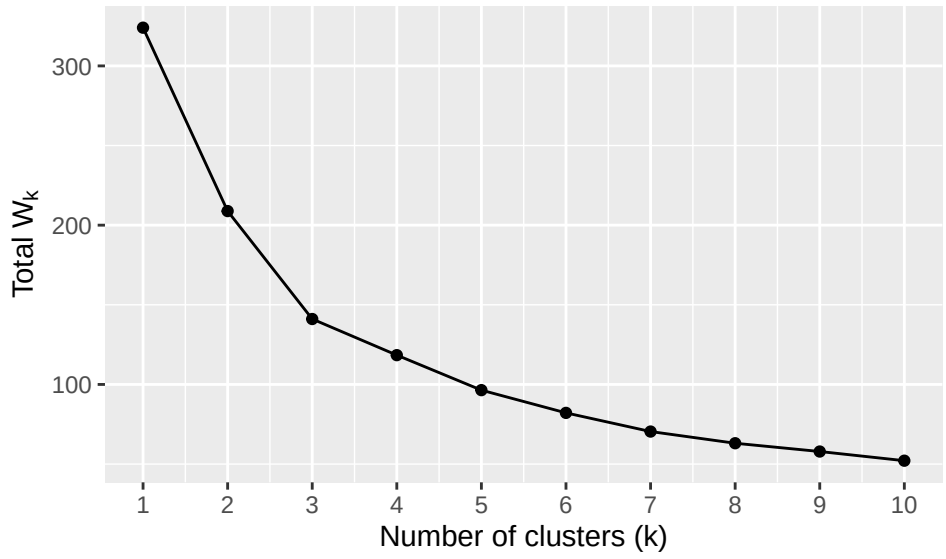
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
set.seed(200)
hof <- hof %>%
    mutate(across(tH:tSB, ~ as.numeric(.)),
```

```
        # Standardize or scale() numeric variables (subtract mean and divide by SD)
        across(where(is.numeric),  ~ scale(.)[,1], .names = "{.col}_scaled"))

clustering_vars <- c("tH_scaled", "tHR_scaled", "tRBI_scaled", "tSB_scaled")
hof_clust <- hof %>%
  select(clustering_vars)
```

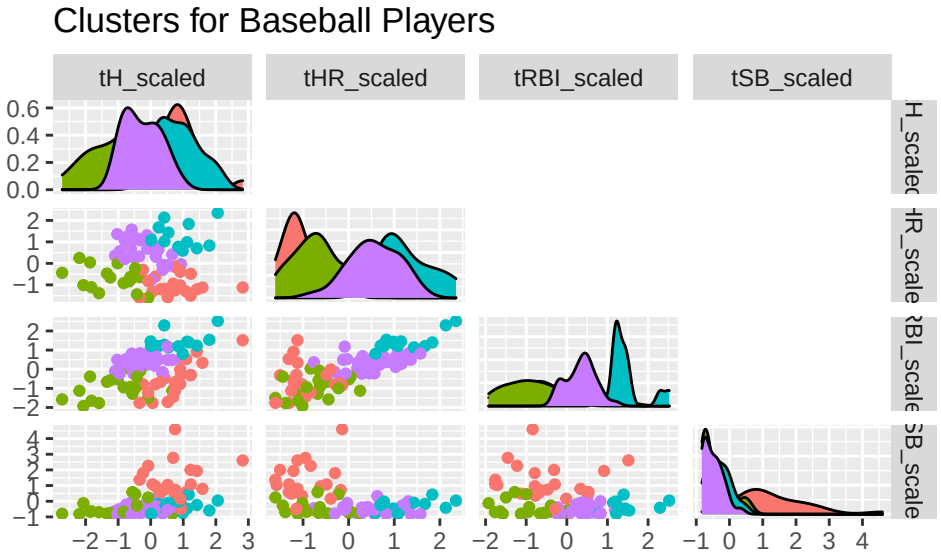
```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(clustering_vars)‘ instead of ‘clustering_vars‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
#Set up elbow plot
elbow_plot <- data.frame(clusters = 1:10,
                         within_ss = rep(NA, 10))
set.seed(75)
for (i in 1:10){
  hof_kms_out <- hof %>%
    select(clustering_vars) %>%
    kmeans(centers = i, nstart = 20)

  elbow_plot$within_ss[i] <- hof_kms_out$tot.withinss
}

# Construct elbow plot
ggplot(elbow_plot, aes(x = clusters, y = within_ss)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Number of clusters (k)", y = expression("Total W"[k]))
```




```
hof_clust <- hof_clust %>%
  kmeans(centers = 4, nstart = 20)
```

```r
hof <- hof %>%
  add_column(cluster = factor(hof_clust$cluster))


#Construct comparison cluster plot
GGally::ggpairs(data = hof,
                aes(color = cluster),
                columns = c("tH_scaled",
                            "tHR_scaled",
                            "tRBI_scaled",
                            "tSB_scaled"),
                upper = list(continuous = "blank")) +
  labs(title = "Clusters for Baseball Players")
```



3 - 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.

```
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:

```
#Confirming that all tweets are from Trump
tweets_check <- trump_tweets_df %>%
  group_by(screenName) %>%
  summarise(n())

#Selecting Relevant Variables
tweets <- trump_tweets_df %>%
  select(text, created, statusSource)
```

> 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: Answer in table.

```
#Looking at the different sources for the tweets
tweets_count <- tweets %>%
group_by(statusSource) %>%
  rename(source = statusSource) %>%
  summarise("count" = n())

knitr::kable(tweets_count,
             caption = "Source of Trump Tweets")
```

Table 1: Source of Trump Tweets

source	count
Instagram	1
Twitter Web Client	120
Twitter for iPad	1
Twitter for Android	762

source	count
Twitter for iPhone	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.

```
# remove eval = FALSE when working on this!

tweets <- tweets %>%
  extract(col = statusSource, into = "source",
#Defines the string from which to extract the values
          regex = "Twitter for (.*)<",
#Keeping original statustSource column in the data frame
          remove = FALSE) %>%
#Keeps only rows where the source is Android or iPhone
  filter(source %in% c("Android", "iPhone"))
```

Solution: extract() pulls out something from a previously existing column and turns it into groups in a new column. In this case, we are pulling out the source as either 'iphone' or 'android' and turning that into a new column.

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".

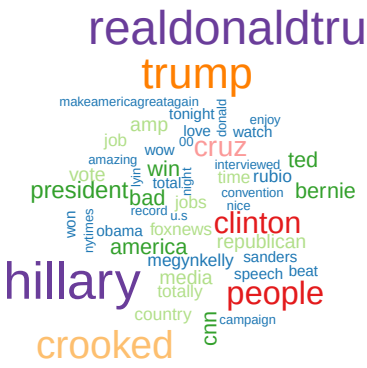
Solution:

```
tweet_words <- tweets %>%
  unnest_tokens(output = word, input = text)

data(stop_words)
tweet_cloud_iphone <- tweet_words %>%
  filter(source == "iPhone") %>%
  anti_join(stop_words,
            by = "word") %>%
  count(word,
        sort = TRUE) %>%
  filter(word != "https" & word != "t.co")

tweet_cloud_android <- tweet_words %>%
  filter(source == "Android") %>%
  anti_join(stop_words,
            by = "word") %>%
  count(word,
        sort = TRUE) %>%
```

```r
  filter(word != "https" & word != "t.co")


mypal <- brewer.pal(10, "Paired")

set.seed(231)
tweet_cloud_iphone %>%
  with(wordcloud(words = word,
                 freq = n,
                 min.freq = 5,
                 max.words = 50,
                 # plot the words in a random order
                 random.order = TRUE,
                 # specify the range of the size of the words
                 scale = c(2, 0.3),
                 # specify proportion of words with 90 degree rotation
                 rot.per = 0.15,
                 # colors words from least to most frequent
                 colors = mypal,
                 # font family
                 family = "sans"))
```


```
tweet_cloud_android %>%
  with(wordcloud(words = word,
                 freq = n,
                 min.freq = 5,
                 max.words = 50,
                 # plot the words in a random order
                 random.order = TRUE,
                 # specify the range of the size of the words
                 scale = c(2, 0.3),
                 # specify proportion of words with 90 degree rotation
                 rot.per = 0.15,
                 # colors words from least to most frequent
                 colors = mypal,
                 # font family
```

```
                family = "sans"))
```



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: The proportion of words classed as angry and joy on the android were approximately 9% and 6% respectively, versus approximately 8% each on iphone. Approximately 16% of tweets were positive on the android and 18% were negative, compared to approximately 13% negative and 21.5% positive on the iphone.

```r
nrc_lexicon <- get_sentiments("nrc")


#Group by number of tweets in each sentiment category
nrc_tweets_android <- tweet_words %>%
  filter(source == "Android") %>%
  anti_join(stop_words,
            by = "word") %>%
  inner_join(nrc_lexicon, by = "word") %>%
  group_by(sentiment) %>%
  summarise(n = n())

#Compute the total amount of tweets on iphone (3975)
total_words_android <- nrc_tweets_android %>%
  summarise(total_words = sum(n))

#Calculate proportion of tweets in each valence category
nrc_tweets_android <- nrc_tweets_android %>%
  mutate(proportion = n/3975)

#Group by number of tweets in each sentiment category
nrc_tweets_iphone <- tweet_words %>%
  filter(source == "iPhone") %>%
  anti_join(stop_words,
            by = "word") %>%
  inner_join(nrc_lexicon, by = "word") %>%
  group_by(sentiment) %>%
  summarise(n = n())

#Compute the total amount of tweets on iphone (1939)
total_words_iphone <- nrc_tweets_iphone %>%
  summarise(total_words = sum(n))

#Calculate proportion of tweets in each valence category
nrc_tweets_iphone <- nrc_tweets_iphone %>%
  mutate(proportion = n/1939)

knitr::kable(nrc_tweets_iphone,
             caption = "Proportion of Trump Tweet sentiments on iphone")
```

Table 2: Proportion of Trump Tweet sentiments on iphone

sentiment	n	proportion
anger	170	0.0876741
anticipation	175	0.0902527
disgust	96	0.0495101
fear	138	0.0711707
joy	161	0.0830325
negative	260	0.1340897
positive	417	0.2150593
sadness	148	0.0763280
surprise	113	0.0582775

sentiment	n	proportion
trust	261	0.1346055

```
knitr::kable(nrc_tweets_android,
             caption = "Proportion of Trump Tweet sentiments on android")
```

Table 3: Proportion of Trump Tweet sentiments on android

sentiment	n	proportion
anger	363	0.0913208
anticipation	335	0.0842767
disgust	227	0.0571069
fear	314	0.0789937
joy	265	0.0666667
negative	647	0.1627673
positive	732	0.1841509
sadness	348	0.0875472
surprise	270	0.0679245
trust	474	0.1192453

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 some evidence for Robinson's claim, given that the overall valence of tweets from the android was more negative than the overall valence of tweets from the iphone. Additionally, the tweet cloud reveals that the most common words tweeted from the iphone were slogans like 'make america great again' and 'trump2016' while tweets from the iphone often contained the terms 'crooked' and 'hillary'. Thus, it appears that tweets from the android were on the whole more negative and combative, suggesting that it was Trump tweteing from the android and his team tweeting from an iphone.