Applied Statistical Programming - Spring 2022

Kimberly Acquilano

Problem Set 4

Due Wednesday, March 23, 10:00 AM (Before Class)

Instructions

1. The following questions should each be answered within an Rmarkdown file. Be sure to provide many comments in your code blocks to facilitate grading. Undocumented code will not be graded.
2. Work on git. Continue to work in the repository you forked from https://github.com/johnsontr/AppliedStatisticalProgramming2022 and add your code for Problem Set 4. Commit and push frequently. Use meaningful commit messages because these will affect your grade.
3. You may work in teams, but each student should develop their own Rmarkdown file. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. For students new to programming, this may take a while. Get started.

`tidyverse`

Your task in this problem set is to combine two datasets in order to observe how many endorsements each candidate received using only `dplyr` functions. Use the same Presidential primary polls that were used for the in class worksheets on February 28 and March 2.

First, create two new objects `polls` and `Endorsements`. Then complete the following.

- Change the `Endorsements` variable name endorsee to `candidate_name`.
- Change the `Endorsements` dataframe into a `tibble` object.
- Filter the `poll` variable to only include the following 6 candidates: Amy Klobuchar, Bernard Sanders,Elizabeth Warren, Joseph R. Biden Jr., Michael Bloomberg, Pete Buttigieg **and** subset the dataset to the following five variables: `candidate_name, sample_size, start_date, party, pct`
- Compare the candidate names in the two datasets and find instances where the a candidates name is spelled differently i.e. Bernard vs. Bernie. Using only `dplyr` functions, make these the same across datasets.
- Now combine the two datasets by candidate name using `dplyr` (there will only be five candidates after joining).
- Create a variable which indicates the number of endorsements for each of the five candidates using `dplyr`.
- Plot the number of endorsement each of the 5 candidates have using `ggplot()`. Save your plot as an object `p`.
- Rerun the previous line as follows: `p + theme_dark()`. Notice how you can still customize your plot without rerunning the plot with new options.

- Now, using the knowledge from the last step change the label of the X and Y axes to be more informative, add a title. Save the plot in your forked repository.

```r
# Change eval=FALSE in the code block. Install packages as appropriate.
# install.packages('fivethirtyeight')
library(fivethirtyeight)
```

```
## Some larger datasets need to be installed separately, like senators and
## house_district_forecast. To install these, we recommend you install the
## fivethirtyeightdata package by running:
## install.packages('fivethirtyeightdata', repos =
## 'https://fivethirtyeightdata.github.io/drat/', type = 'source')
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
# URL to the data that you've used.
url <- "https://jmontgomery.github.io/PDS/Datasets/president_primary_polls_feb2020.csv"
polls <- read_csv(url)
```

```
## Rows: 16661 Columns: 33

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (21): state, pollster, sponsors, display_name, pollster_rating_name, fte...
## dbl  (8): question_id, poll_id, cycle, pollster_id, pollster_rating_id, samp...
## lgl  (3): internal, tracking, nationwide_batch
##
## 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
Endorsements <- endorsements_2020  # from the fiverthirtyeight package

# Alter the Endorsements file Change the column name from endorsee to
# candidate_name
Endorsements <- rename(Endorsements, candidate_name = endorsee)
# Turn it into a tibble to be able to manipulate data
Endorsements <- as_tibble(Endorsements)

# Alter the polls file, by paring it down filter the dataset to only be data
# for the following candidates select only the necessary columns:
# candidate_name, sample_size, start_date, party, and pct
polls <- polls %>%
    filter(candidate_name %in% c("Amy Klobuchar", "Bernard Sanders", "Elizabeth Warren",
        "Joseph R. Biden Jr.", "Michael Bloomberg", "Pete Buttigieg")) %>%
    select(candidate_name, sample_size, start_date, party, pct)
```

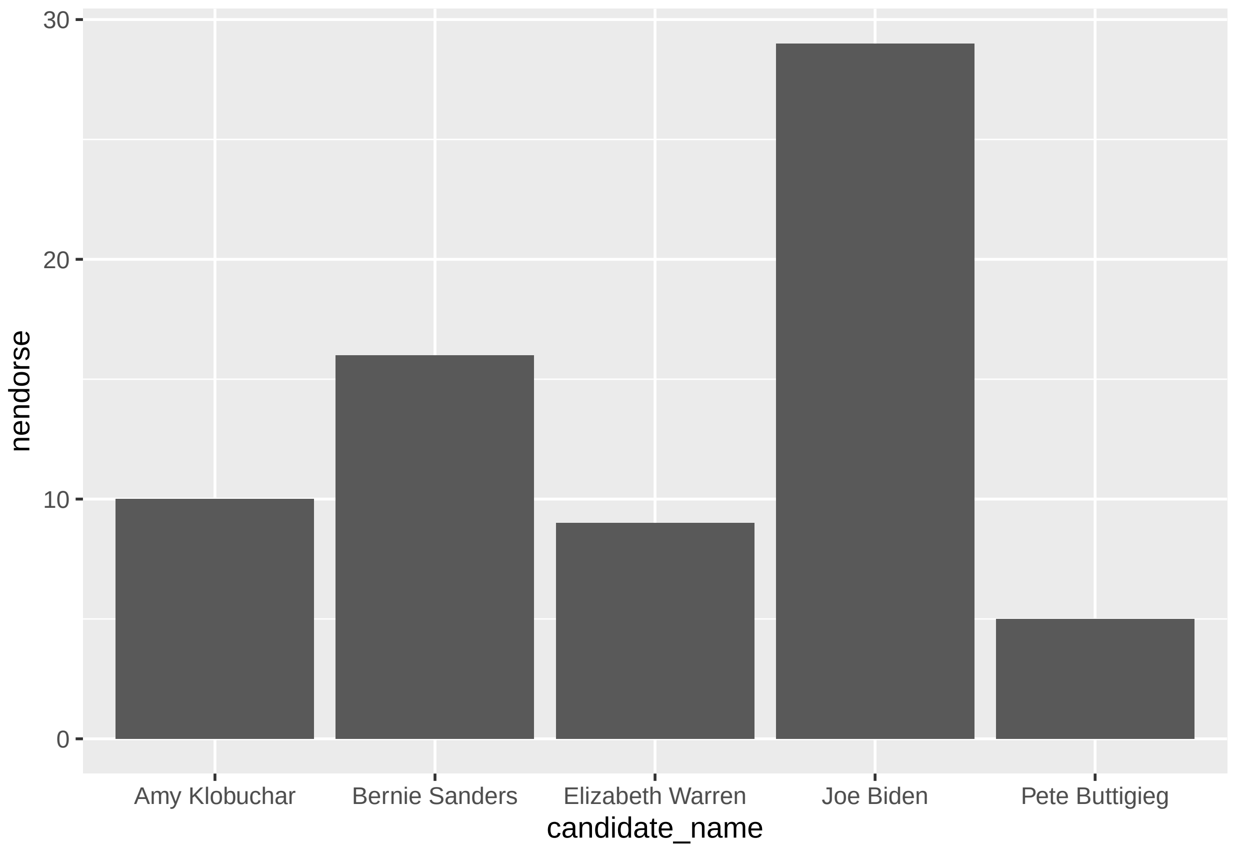
```r
# Check the Endorsements file to see what candidates are used in the dataset
# The purpose to match candidate information with those in the polls dataset
# Use distinct to only return unique values of candidate_names In other words,
# return only each candidate_name, I don't need every time each anme occurs Use
# summarise to get that list of names
Endorsements %>%
    distinct(candidate_name) %>%
    summarise(candidate_name)
```

```
## # A tibble: 16 x 1
##    candidate_name
##    <chr>
##  1 John Delaney
##  2 Joe Biden
##  3 Julian Castro
##  4 Kamala Harris
##  5 Bernie Sanders
##  6 Cory Booker
##  7 Amy Klobuchar
##  8 Elizabeth Warren
##  9 Jay Inslee
## 10 John Hickenlooper
## 11 Beto O'Rourke
## 12 Kirsten Gillibrand
## 13 Pete Buttigieg
## 14 Eric Swalwell
## 15 Steve Bullock
## 16 <NA>
```

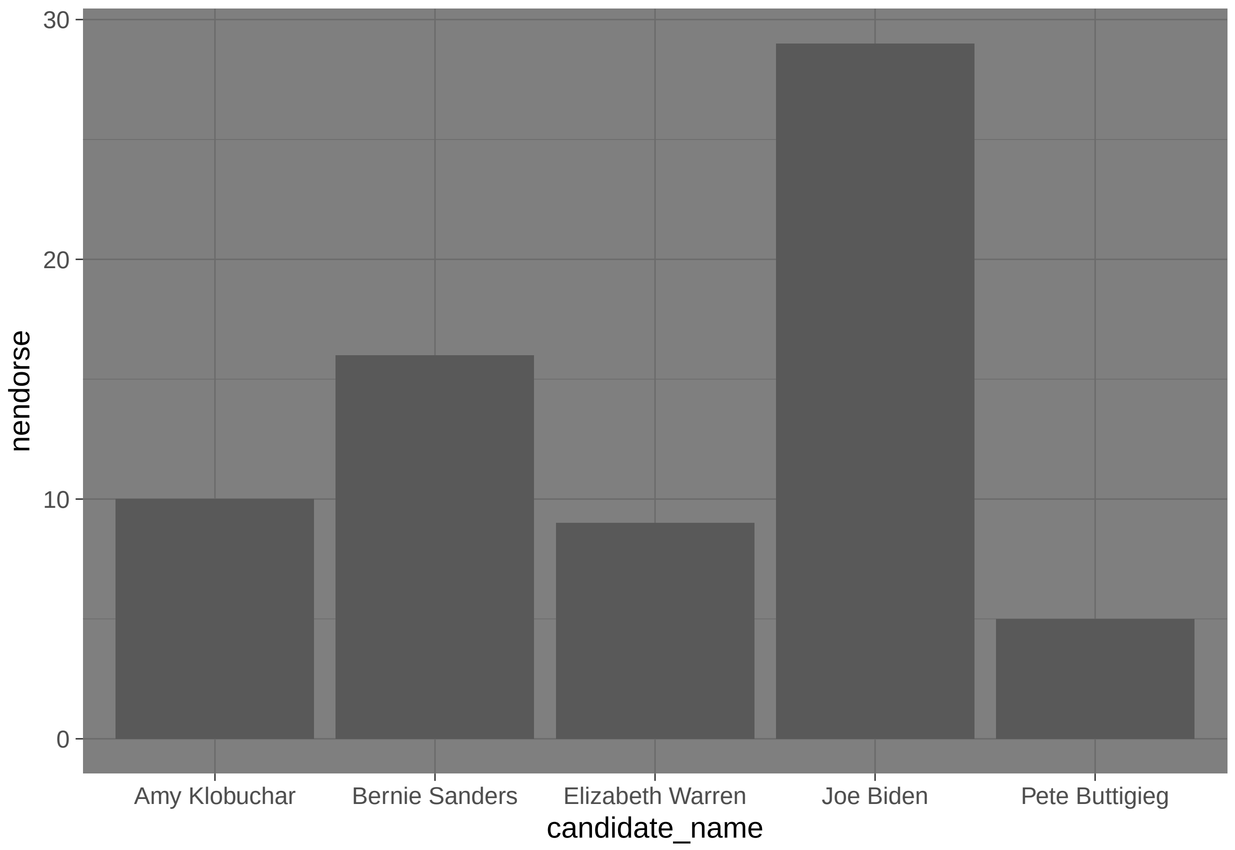
```r
# polls uses: Amy Klobuchar Bernard Sanders Elizabeth Warren Joseph R. Biden
# Jr. Michael Bloomberg Pete Buttigieg

# Endorsements uses: Joe Biden Bernie Sanders Amy Klobuchar Elizabeth Warren
# Pete Buttigieg Michael Bloomberg is missing

# Match names, as they are written, in polls to be those in Endorsements Amy
# Klobuchar, Elizabeth Warrent, and Pete Buttigieg already match Alter the
# polls dataset so that the Bernie Sanders name matches that in Endorsements
# Use mutate function. Replace Bernard Sanders in polls to be Bernie Sanders
polls <- polls %>%
    mutate(candidate_name = replace(candidate_name, candidate_name == "Bernard Sanders",
        "Bernie Sanders"))

# Alter the polls dataset again, so that Joe Biden's name matches what is in
# Endorsements Use mutate function. Replace Joseph R. Biden Jr. in polls to be
# Joe Biden
polls <- polls %>%
    mutate(candidate_name = replace(candidate_name, candidate_name == "Joseph R. Biden Jr.",
        "Joe Biden"))

# Combine the two datasets Create a new tibble for this combo set Use
# inner_join to combine all matching rows, according to candidate name, and
# drop rows that do not match
newPoll <- polls %>%
    inner_join(Endorsements, by = "candidate_name")

# Verify there are only 5 candidate names, and that they are the correct names
newPoll %>%
    distinct(candidate_name) %>%
    summarise(candidate_name)
```

```
## # A tibble: 5 x 1
##   candidate_name
##   <chr>
## 1 Bernie Sanders
## 2 Pete Buttigieg
## 3 Joe Biden
## 4 Amy Klobuchar
## 5 Elizabeth Warren
```

```r
# Create an new tibble Thhis new tibble will only have each candidate's name
# and their total number of endorsements Group the data by candidate's name Use
# summarise to create the new column nendorse, which is the number of
# endorsements (count of number of distinct endorers)
groupednewPoll <- newPoll %>%
    group_by(candidate_name) %>%
    summarise(nendorse = n_distinct(endorser))

# Create a plot called c
library(ggplot2)
# Plot groupednewPoll data Candidate name on x-axis Number of endorsements on
# y-axis Use geom shape columns
p <- ggplot(groupednewPoll, aes(x = candidate_name, y = nendorse)) + geom_col()

# Print plot p
p
```



```r
# Add the dark theme to p
p + theme_dark()
```



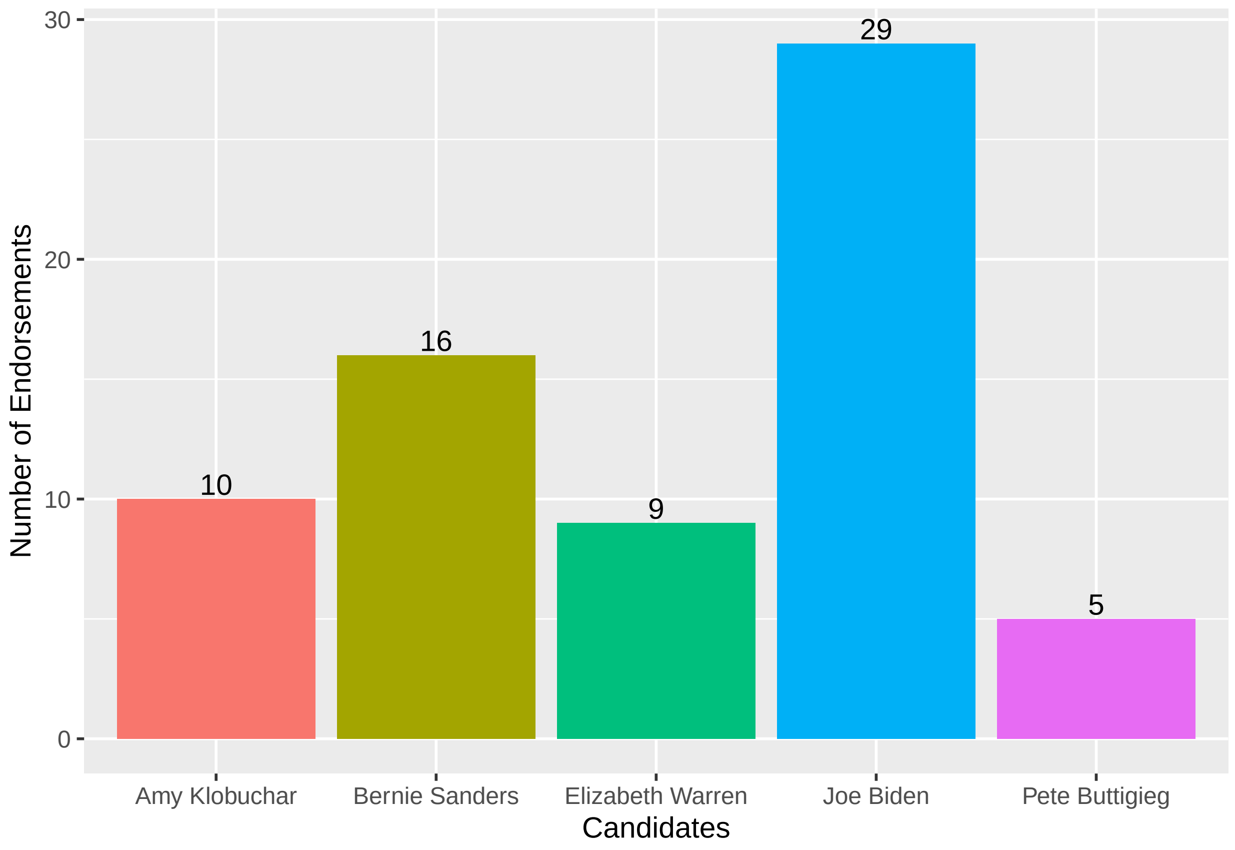
```
# Customize plot p load a new color palatte install.packages('viridis')
library(viridis)
```

```
## Loading required package: viridisLite
```

```
# Use plot p Add in column labels, to show the number of endorsements on top of
# each column Color each column, using the viridis color palette (5 colors from
# the inferno option) Change x-axis label to 'Candidates' Change y-axis label
# to 'Number of Endorsements' Remove color legend
p <- p + geom_text(aes(label = nendorse, vjust = -0.2)) + geom_col(aes(fill = inferno(5))) +
    xlab("Candidates") + ylab("Number of Endorsements") + guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
# Print new plot p
p
```



Text-as-Data with `tidyverse`

For this question you will be analyzing Tweets from President Trump for various characteristics. Load in the following packages and data:

```r
# Change eval=FALSE in the code block. Install packages as appropriate.
library(tidyverse)
# install.packages('tm')
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
# install.packages('lubridate')
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# install.packages('wordcloud')
library(wordcloud)
```

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

```
trump_tweets_url <- "https://politicaldatascience.com/PDS/Datasets/trump_tweets.csv"
tweets <- read_csv(trump_tweets_url)
```

```
## Rows: 32974 Columns: 6

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): source, text, created_at
## dbl (2): retweet_count, favorite_count
## lgl (1): is_retweet
##
## 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.
```

First separate the `created_at` variable into two new variables where the date and the time are in separate columns. After you do that, then report the range of dates that is in this dataset.

```
# Extract date and create new variable, Use created_at, which is originally
# written as month/date/year
tweets$date <- as.Date(tweets$created_at, "%m/%d/%y")

# Extract time and create new variable, starting at fifth to last character and
# ending at last character of created_at
tweets$times <- str_sub(tweets$created_at, start = -5, end = -1)
# Turn times into time of day, with hours and minutes
tweets$times <- hm(tweets$times)

# get range of dates
range(tweets$date)
```

```
## [1] "2020-01-01" "2020-12-31"
```

Range of dates are: 01-01-2020 to 12-31-2020

Using `dplyr` subset the data to only include original tweets (remove retweents) and show the text of the President's **top 5** most popular and most retweeted tweets. (Hint: The `match` function can help you find the index once you identify the largest values.)

```
# create a new dataframe, a subset of tweets include only tweets, no retweets
tweets_only <- tweets[tweets$is_retweet == FALSE, ]

# reorder the dataframe according to retweet_count, with highest at the top,
```

```r
# put NAs at the end
tweets_only$retweet_count <- sort(tweets_only$retweet_count, decreasing = TRUE, na.last = TRUE)
# save top 5 tweets, the 5 tweets with highest retweet count
most_retweets <- head(tweets_only$text, 5)

# reorder the dataframe according to favorite_count, with the highest at the
# top, put NAs at the end
tweets_only$favorite_count <- sort(tweets_only$favorite_count, decreasing = TRUE,
    na.last = TRUE)
# save top 5 tweets, the 5 tweets with the most favorites
mostpop <- head(tweets_only$text, 5)

# check the two objects, most_retweets and mostpop they are the same
most_retweets
```

```
## [1] "I'm seeing Governor Cuomo today at The White House. He must understand that National Security fa
## [2] "....which he actually has a military and legal obligation to do. His incredible wife Karen who I
## [3] "When I terminated John Kelly which I couldn't do fast enough he knew full well that he was way o
## [4] "DRAIN THE SWAMP! We want bad people out of our government!"
## [5] "Mini Mike is a 5'4" mass of dead energy who does not want to be on the debate stage with these p
```

```r
mostpop
```

```
## [1] "I'm seeing Governor Cuomo today at The White House. He must understand that National Security fa
## [2] "....which he actually has a military and legal obligation to do. His incredible wife Karen who I
## [3] "When I terminated John Kelly which I couldn't do fast enough he knew full well that he was way o
## [4] "DRAIN THE SWAMP! We want bad people out of our government!"
## [5] "Mini Mike is a 5'4" mass of dead energy who does not want to be on the debate stage with these p
```

Create a *corpus* of the tweet content and put this into the object `Corpus` using the `tm` (text mining) package. (Hint: Do the assigned readings.)

```r
# install.packages('tm')
library(tm)

# Create a volitile corpus of all of the tweets in the tweets_only datatframe
Corpus <- VCorpus(VectorSource((tweets_only$text)))
```

Remove extraneous whitespace, remove numbers and punctuation, convert everything to lower case and remove 'stop words' that have little substantive meaning (the, a, it).

```
# remove all the whitespace from the tweets
TweetCorpus <- tm_map(Corpus, stripWhitespace)

# remove numbers
TweetCorpus <- tm_map(TweetCorpus, removeNumbers)

# remove punctuation
TweetCorpus <- tm_map(TweetCorpus, removePunctuation)

# Convert everything to lower case
TweetCorpus <- tm_map(TweetCorpus, content_transformer(tolower))

# remove stopwords, such as the, a, it.
TweetCorpus <- tm_map(TweetCorpus, removeWords, stopwords("english"))
```

Now create a `wordcloud` to visualize the top 50 words the President uses in his tweets. Use only words that occur at least three times. Display the plot with words in random order and use 50 random colors. Save the plot into your forked repository.

```
# To create a word-cloud: Note: followed steps laid out by
# https://towardsdatascience.com/create-a-word-cloud-with-r-bde3e7422e8a for
# manipulating TDM and creating word-cloud. Followed their code for this task.
# 1. create a term document matrix
TDMTweets <- TermDocumentMatrix(TweetCorpus)
# 2. remove sparse terms, because the object is too big otherwise the closer to
# 1, the more sparse the words are that are being removed
TDMTweets <- removeSparseTerms(TDMTweets, 0.99)
# 3. turn it into a matrix
TDMTweets <- as.matrix((TDMTweets))
# 4. sort the rows to be decreasing
TDMTweets <- sort(rowSums(TDMTweets), decreasing = TRUE)
# 5. create a dataframe with the frequency count of each word
TDMTweets <- data.frame(word = names(TDMTweets), freq = TDMTweets)
# 6. Create word-cloud a. create pdf for word-cloud, saved to the repository.
pdf(file = "./wordcloud.pdf")
# b. set seed for replicability
set.seed(1234)
# c. install.packages('pals') for color palette
library(pals)
```

```
##
## Attaching package: 'pals'

## The following objects are masked from 'package:viridis':
##
##     cividis, inferno, magma, plasma, turbo, viridis

## The following objects are masked from 'package:viridisLite':
##
```

```
##     cividis, inferno, magma, plasma, turbo, viridis
```

```
# d. wordcloud function use words that occur at least 3 times word-cloud will
# have 50 most frequent words display in a random order set random.color to
# FALSE, so that colors will not be assigned by frequency Have 25% of the words
# rotated 90 degrees use the colors created above
wordcloud(words = TDMTweets$word, freq = TDMTweets$freq, min.freq = 3, max.words = 50,
    random.order = TRUE, random.color = FALSE, ordered.colors = FALSE, rot.per = 0.25,
    colors = kovesi.cyclic_mygbm_30_95_c78(length(TDMTweets$word)))
# e. add a title, using mtext. Options side=3 is to place the text at the top
# of the page, cex=2 is the font size
mtext("Top 50 words tweeted by Trump in 2020", side = 3, cex = 2)
# f. add a note to describe what the colors and words sizes mean. Use '\n' in
# text to move to a new line. Option side=1 places the text at the bottom on
# the page, cex=.5 is the font size.
mtext("Note: The colors do not add any information; \n they are only for aesthetic purposes.\n Text siz
    side = 1, cex = 0.5)
# e. end of pdf
dev.off()
```

```
## pdf
##   2
```

Create a *document term matrix* called DTM that includes the argument `control = list(weighting = weightTfIdf)`

```
# subset tweets by only those that have been retweeted at least 1,000 times
retweeted <- tweets_only %>%
    filter(retweet_count >= 1000)

# Turn this document into a corpus
RetweetCorpus <- VCorpus(VectorSource((retweeted$text)))
# Clean this corpus just like the one before remove all the whitespace from the
# tweets
ReweetCorpus <- tm_map(RetweetCorpus, stripWhitespace)

# remove numbers
RetweetCorpus <- tm_map(RetweetCorpus, removeNumbers)

# remove punctuation
RetweetCorpus <- tm_map(RetweetCorpus, removePunctuation)

# Convert everything to lower case
RetweetCorpus <- tm_map(RetweetCorpus, content_transformer(tolower))

# remove stopwords, such as the, a, it.
RetweetCorpus <- tm_map(RetweetCorpus, removeWords, stopwords("english"))

# Turn RetweetCorpus into Document-Term_Matrix
```

```
DTM_tfidf <- DocumentTermMatrix(RetweetCorpus, control = list(weighting = weightTfIdf,
    lowfreq = 0.8))
```

```
## Warning in weighting(x): empty document(s): 480 482 1824 8946 10551 10552 10553
## 10554 10555 10556 10557 10558 10559 10560 10561 10562 10563 10564 10565 10566
## 10567 10568 10569 10570 10571 10572 10573 10574 10575 10576 10577 10578 10579
## 10580 10581 10582 10583 10584 10585 10586 10587 10588 10589 10590 10591 10592
## 10593 10594 10595 10596 10597 10598 10599 10600 10601 10602 10603 10604 10605
## 10606 10607 10608 12200
```

Finally, report the 50 words with the the highest tf.idf scores using a lower frequency bound of .8.

```
# turn the DTM into a dataframe using tidy() install.packages('tidytext')
library(tidytext)
DTM_tfidf_df <- tidy(DTM_tfidf)
# The tidy function turned my DTM into a data frame. Now the columns are: 1.
# the document the observation came from; 2. the term, which is word used in
# the original tweet (this is really the observation for practical purposes);
# and 3. the observation's count, which is the td.idf score.

# Get the 50 terms with the highest td.idf scores 1. start with renaming the
# column name from count to tf_idf
DTM_tfidf_df2 <- DTM_tfidf_df %>%
    rename(tf_idf = "count") %>%
    # 2. then reorder the tf_idf column, so that its value is decreasing
arrange(desc(tf_idf)) %>%
    # 3 now remove all words that have http in them
filter(!(str_detect(term, ".[(http)]."))) %>%
    # 4. keept only the distinct terms, in other words remove duplicate words
    # Be sure to set .keep_all to TRUE to be able to keep all variables
distinct(term, .keep_all = TRUE) %>%
    # 4. select only the top 50 words
slice_head(n = 50)

# print the top 50 terms with the highest td.idf scores
print(DTM_tfidf_df2$term)
```

```
##  [1] "pressure""                  "gma"
##  [3] "treason"                    "boring"
##  [5] "name"                       "winred"
##  [7] "usembassyjerusalem"         "congressionalbaseballgame"
##  [9] "americasmerkel"             "nfib"
## [11] "enjoy"                      "remembering"
## [13] "nike"                       "peaceofficersmemorialday"
## [15] "jfkfiles"                   "memorialday"
## [17] "weeklyaddress"              "foxandfriends"
## [19] "armynavygame"               "makeamericasafeagain"
## [21] "usa"                        "revealing"
## [23] "neverforget"                "obamacarefail"
## [25] "wave"                       "donlemon"
## [27] "piersmorgan"                "unga"
## [29] "merry"                      "scavino"
## [31] "confirmed"                  "drain"
## [33] "july"                       "americas"
## [35] "nadler"                     "juliet"
## [37] "adriana"                    "rena"
## [39] "anne"                       "hugh"
## [41] "convenient"                 "genius""
## [43] "israelusaforevr"            "realwayneroot"
## [45] "summervilleguy"             "negop"
## [47] "gosakasummit"               "mcguire"
## [49] "boneszoom"                  "wargs"
```