04 Homework

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(rvest)
```

```
Attaching package: 'rvest'

The following object is masked from 'package:readr':

    guess_encoding
```

```
library(httr)
```

```
library(tidyverse)

#spotify <- read_csv("Data/spotify.csv")
spotify <- read_csv("https://joeroith.github.io/264_spring_2025/Data/spotify.csv")

spot_smaller <- spotify |>
```

```
  select(
    title,
    artist,
    album_release_date,
    album_name,
    subgenre,
    playlist_name
  )

spot_smaller <- spot_smaller[c(5, 32, 49, 52, 83, 175, 219, 231, 246, 265), ]
spot_smaller
```

```
# A tibble: 10 x 6
   title              artist album_release_date album_name subgenre playlist_name
   <chr>              <chr>  <chr>              <chr>      <chr>    <chr>
 1 Hear Me Now        Alok   2016-01-01         Hear Me N~ indie p~ Chillout & R~
 2 Run the World (G~  Beyon~ 2011-06-24         4          post-te~ post-teen al~
 3 Formation          Beyon~ 2016-04-23         Lemonade   hip pop  Feeling Acco~
 4 7/11               Beyon~ 2014-11-24         BEYONCÉ [~ hip pop  Feeling Acco~
 5 My Oh My (feat. ~  Camil~ 2019-12-06         Romance    latin p~ 2020 Hits & ~
 6 It's Automatic     Frees~ 2013-11-28         It's Auto~ latin h~ 80's Freesty~
 7 Poetic Justice     Kendr~ 2012               good kid,~ hip hop  Hip Hop Cont~
 8 A.D.H.D            Kendr~ 2011-07-02         Section.80 souther~ Hip-Hop 'n R~
 9 Ya Estuvo          Kid F~ 1990-01-01         Hispanic ~ latin h~ HIP-HOP: Lat~
10 Runnin (with A$A~  Mike ~ 2018-11-16         Creed II:~ gangste~ RAP Gangsta
```

1. Identify the input type and output type for each of these examples:

str_view(spot_smaller$subgenre, "pop") $typeof(str_view(spot_smaller$subgenre, \text{“pop”}))$ class(str_view(spot_smaller$subgenre, “pop”)) input: character vector, output: stringr_view

str_view(spot_smaller$subgenre, “pop”, match = NA) input: character vector, output: stringr_view

str_view(spot_smaller$subgenre, “pop”, html = TRUE) input: list, output: str_view

str_subset(spot_smaller$subgenre, “pop”) input: character, output: character vector

str_detect(spot_smaller$subgenre, “pop”) input: character vector, output: logical vector

2. Use str_detect to print the rows of the spot_smaller tibble containing songs that have “pop” in the subgenre. (i.e. make a new tibble with fewer rows)

```
pop <- spot_smaller |>
  mutate(is_pop = str_detect(spot_smaller$subgenre, "pop")) |>
  filter(is_pop == "TRUE") |>
  select(-is_pop)
```

3. Find the mean song title length for songs with "pop" in the subgenre and songs without "pop" in the subgenre.

```
mean_pop <- pop |>
  mutate(title_length = str_length(title),
         mean_title_length = mean(title_length))

pop2 <- spot_smaller |>
  mutate(is_pop = str_detect(spot_smaller$subgenre, "pop")) |>
  mutate(title_length = str_length(title)) |>
  group_by(is_pop) |>
  summarize(mean_title_length = mean(title_length))
```

Producing a table like this would be great:

A tibble: 2 × 2

sub_pop mean_title_length 1 FALSE 18.6 2 TRUE 13.6

Producing a table like this would be SUPER great (hint: ifelse()):

A tibble: 2 × 2

sub_pop mean_title_length 1 Genre with pop 13.6 2 Genre without pop 18.6

4. In the bigspotify dataset, find the proportion of songs which contain "love" in the title (track_name) by playlist_genre.

```
bigspotify <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/
```

```
Rows: 32833 Columns: 23
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...

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.
```

```
bigspotify
```

```
# A tibble: 32,833 x 23
   track_id              track_name track_artist track_popularity track_album_id
   <chr>                 <chr>      <chr>                  <dbl> <chr>
 1 6f807x0ima9a1j3VPbc7~ I Don't C~ Ed Sheeran                66 2oCs0DGTsRO98~
 2 0r7CVbZTWZgbTCYdfa2P~ Memories ~ Maroon 5                  67 63rPSO264uRjW~
 3 1z1Hg7Vb0AhHDiEmnDE7~ All the T~ Zara Larsson              70 1HoSmj2eLcsrR~
 4 75FpbthrwQmzHlBJLuGd~ Call You ~ The Chainsm~             60 1nqYsOef1yKKu~
 5 1e8PAfcKUYoKkxPhrHqw~ Someone Y~ Lewis Capal~             69 7m7vv9wlQ4i0L~
 6 7fvUMiyapMsRRxr07cU8~ Beautiful~ Ed Sheeran               67 2yiy9cd2QktrN~
 7 2OAylPUDDfwRGfe0lYql~ Never Rea~ Katy Perry               62 7INHYSeusaFly~
 8 6b1RNvAcJjQH73eZO4BL~ Post Malo~ Sam Feldt                 69 6703SRPsLkS4b~
 9 7bF6tCO3gFb8INrEDcjN~ Tough Lov~ Avicii                   68 7CvAfGvq4RlIw~
10 1IXGILkPm0tOCNeq00kC~ If I Can'~ Shawn Mendes             67 4QxzbfSsVryEQ~
# i 32,823 more rows
# i 18 more variables: track_album_name <chr>, track_album_release_date <chr>,
#   playlist_name <chr>, playlist_id <chr>, playlist_genre <chr>,
#   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
#   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
#   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
#   duration_ms <dbl>
```

```
bigspotify |>
  mutate(track_name = str_to_lower(track_name)) |>
  group_by(playlist_genre) |>
  summarize(prop_love = mean(str_detect(track_name, "love"), na.rm = TRUE))
```

```
# A tibble: 6 x 2
  playlist_genre prop_love
  <chr>              <dbl>
1 edm               0.0399
```

```
2 latin            0.0258
3 pop              0.0481
4 r&b              0.0639
5 rap              0.0125
6 rock             0.0450
```

5. Given the corpus of common words in stringr::words, create regular expressions that find all words that:

- Start with "y".
- End with "x"
- Are exactly three letters long.
- Have seven letters or more.
- Start with a vowel.
- End with ed, but not with eed.
- Words where q is not followed by u. (are there any in `words`?)

Try using str_view() or str_subset()

str_view(words, "^y") str_view(words, "x")$str_view(words, "^.....")$ str_view(words, "[1]") str_view(words, "[^e]ed$") str_view(words, "q[^u]")

For example, to find words with "tion" at any point, I could use:

str_view(words, "tion") str_subset(words, "tion")

6. In bigspotify, how many track_names include a $? Be sure you print the track_names you find and make sure the dollar sign is not just in a featured artist!

```
bigspotify |>
  filter(str_detect(track_name, "\\$")) |>
  select(track_name, track_artist) |>
  filter(!str_detect(track_name, "(feat|with).*\\$")) # . indicates any character, * indicate
```

```
# A tibble: 25 x 2
   track_name                                          track_artist
   <chr>                                               <chr>
 1 Wing$                                               Macklemore & Ryan Lewis
```

[1] aeiou

```
 2 $Dreams                                                 Max Frost
 3 $ave Dat Money (feat. Fetty Wap & Rich Homie Quan) Lil Dicky
 4 NO TRU$T                                                NUGAT
 5 A$AP Forever                                            A$AP Rocky
 6 M'$ (feat. Lil Wayne)                                   A$AP Rocky
 7 Sie wollen meine Loui$ (Don Dollar)                     Kulturerbe Achim
 8 Foe Tha Love Of $                                       Bone Thugs-N-Harmony
 9 A$AP                                                    Dillom
10 $$$ - Remix                                             Saramalacara
# i 15 more rows
```

7. In bigspotify, how many track_names include a dollar amount (a $ followed by a number).

str_view(bigspotify$track_name, "
\d")

2

8. Modify the first regular expression above to also pick up "A.A" (in addition to "BEY-ONC" and "II"). That is, pick up strings where there might be a period between capital letters.

str_view(spot_smaller$album_name, "A-Z[A-Z]")

9. Create some strings that satisfy these regular expressions and explain.

- "^.*$"
- "\{.+\}"

str_view("example", "^.*$") #starts with any number of characters and ends with any number of characters

str_view("{example}", "\{.+\}") #starts with { and ends with } str_view("example", "\{.+\}")

10. Create regular expressions to find all `stringr::words` that:

- Start with three consonants.
- Have two or more vowel-consonant pairs in a row.

str_view(stringr::words, "[2]{3}.*([aeiou]{2,})")

```
str_extract(spot_smaller$album_release_date, "\\d{4}-\\d{2}")
```

[2] âeiou

```
 [1] "2016-01" "2011-06" "2016-04" "2014-11" "2019-12" "2013-11" NA
 [8] "2011-07" "1990-01" "2018-11"
```

```
spot_smaller |>
  select(album_release_date) |>
  mutate(year_month = str_extract(album_release_date, "\\d{4}-\\d{2}"))
```

```
# A tibble: 10 x 2
   album_release_date year_month
   <chr>              <chr>
 1 2016-01-01         2016-01
 2 2011-06-24         2011-06
 3 2016-04-23         2016-04
 4 2014-11-24         2014-11
 5 2019-12-06         2019-12
 6 2013-11-28         2013-11
 7 2012               <NA>
 8 2011-07-02         2011-07
 9 1990-01-01         1990-01
10 2018-11-16         2018-11
```

```
spot_smaller |>
  select(artist) |>
  mutate(n_vowels = str_count(artist, "[aeiou]"))
```

```
# A tibble: 10 x 2
   artist            n_vowels
   <chr>                <int>
 1 Alok                     1
 2 Beyoncé                  2
 3 Beyoncé                  2
 4 Beyoncé                  2
 5 Camila Cabello           6
 6 Freestyle                3
 7 Kendrick Lamar           4
 8 Kendrick Lamar           4
 9 Kid Frost                2
10 Mike WiLL Made-It        5
```

11. In the spot_smaller dataset, how many words are in each title? (hint \b)

```
str_count(spot_smaller$title, "\\b[^ ]+\\b")
```

```
 [1] 3 4 1 1 5 2 2 1 2 8
```

12. In the spot_smaller dataset, extract the first word from every title. Show how you would print out these words as a vector and how you would create a new column on the spot_smaller tibble. That is, produce this:

```
str_extract(spot_smaller$title, "[^ ]*")
```

```
 [1] "Hear"      "Run"       "Formation" "7/11"      "My"        "It's"
 [7] "Poetic"    "A.D.H.D"   "Ya"        "Runnin"
```

```
# [1] "Hear"      "Run"       "Formation" "7/11"      "My"        "It's"
# [7] "Poetic"    "A.D.H.D"   "Ya"        "Runnin"
```

Then this:

```
spot_smaller |>
  select(title) |>
  mutate(first_word = str_extract(title, "[^ ]*"))
```

```
# A tibble: 10 x 2
   title                                              first_word
   <chr>                                              <chr>
 1 Hear Me Now                                        Hear
 2 Run the World (Girls)                              Run
 3 Formation                                          Formation
 4 7/11                                               7/11
 5 My Oh My (feat. DaBaby)                            My
 6 It's Automatic                                     It's
 7 Poetic Justice                                     Poetic
 8 A.D.H.D                                            A.D.H.D
 9 Ya Estuvo                                          Ya
10 Runnin (with A$AP Rocky, A$AP Ferg & Nicki Minaj) Runnin
```

```
# A tibble: 10 × 2
#   title                                                first_word
#   <chr>                                                <chr>
# 1 Hear Me Now                                          Hear
# 2 Run the World (Girls)                                Run
# 3 Formation                                            Formation
# 4 7/11                                                 7/11
# 5 My Oh My (feat. DaBaby)                              My
# 6 It's Automatic                                       It's
# 7 Poetic Justice                                       Poetic
# 8 A.D.H.D                                              A.D.H.D
# 9 Ya Estuvo                                            Ya
#10 Runnin (with A$AP Rocky, A$AP Ferg & Nicki Minaj) Runnin
```

13. Which decades are popular for playlist_names? Using the bigspotify dataset, try doing each of these steps one at a time!

- filter the bigspotify dataset to only include playlists that include something like "80's" or "00's" in their title.
- create a new column that extracts the decade
- use count to find how many playlists include each decade
- what if you include both "80's" and "80s"?
- how can you count "80's" and "80s" together in your final tibble?

```
bigspotify |>
  filter(str_detect(playlist_name, "\\d\\d('?)s")) |>
  mutate(playlist_decade = str_extract(playlist_name, "\\d\\d('?)s"),
         playlist_decade = str_replace(playlist_decade, "'", "")) |>
  count(playlist_decade) |>
  filter(playlist_decade != "08s")
```

```
# A tibble: 6 x 2
  playlist_decade     n
  <chr>           <int>
1 00s                45
2 10s               281
3 50s               100
4 70s               442
5 80s               682
6 90s              1013
```

14. Describe to your groupmates what these expressions will match, and provide a word or expression as an example:

- (.)\1\1 Any character repeated 3 times

str_view("aaa", "(.)\1\1")

- "(.)(.)(.).*\3\2\1" Any word that has any three characters, followed by any number of characters, and then the three characters are repeated backwards.

str_view("abcdecba", "(.)(.)(.).*\3\2\1")

Which words in `stringr::words` match each expression?

str_view(stringr::words, "(.)\1\1")

none

str_view(stringr::words, "(.)(.)(.).*\3\2\1")

paragraph

15. Construct a regular expression to match words in `stringr::words` that contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice) but *not* match repeated pairs of numbers (e.g. 507-786-3861).

str_view(stringr::words, "([a-z][a-z]).*\1")

16. Reformat the album_release_date variable in spot_smaller so that it is MM-DD-YYYY instead of YYYY-MM-DD. (Hint: str_replace().)

```
spot_smaller |>
  mutate(album_release_date = str_replace(album_release_date, "(\\d{4})-(\\d{2})-(\\d{2})", '
```

```
# A tibble: 10 x 6
   title             artist album_release_date album_name subgenre playlist_name
   <chr>             <chr>  <chr>              <chr>      <chr>    <chr>
 1 Hear Me Now       Alok   01-01-2016         Hear Me N~ indie p~ Chillout & R~
 2 Run the World (G~ Beyon~ 06-24-2011         4          post-te~ post-teen al~
 3 Formation         Beyon~ 04-23-2016         Lemonade   hip pop  Feeling Acco~
 4 7/11              Beyon~ 11-24-2014         BEYONCÉ [~ hip pop  Feeling Acco~
 5 My Oh My (feat. ~ Camil~ 12-06-2019         Romance    latin p~ 2020 Hits & ~
 6 It's Automatic    Frees~ 11-28-2013         It's Auto~ latin h~ 80's Freesty~
 7 Poetic Justice    Kendr~ 2012               good kid,~ hip hop  Hip Hop Cont~
 8 A.D.H.D           Kendr~ 07-02-2011         Section.80 souther~ Hip-Hop 'n R~
 9 Ya Estuvo         Kid F~ 01-01-1990         Hispanic ~ latin h~ HIP-HOP: Lat~
10 Runnin (with A$A~ Mike ~ 11-16-2018         Creed II:~ gangste~ RAP Gangsta
```

17. BEFORE RUNNING IT, explain to your partner(s) what the following R chunk will do:

It will switch the 2nd and 3rd words.

```
sentences |>
  str_replace("([^ ]+) ([^ ]+) ([^ ]+)", "\\1 \\3 \\2") |>
  head(5)
```

```
[1] "The canoe birch slid on the smooth planks."
[2] "Glue sheet the to the dark blue background."
[3] "It's to easy tell the depth of a well."
[4] "These a days chicken leg is a rare dish."
[5] "Rice often is served in round bowls."
```

1. Describe the equivalents of ?, +, * in {m,n} form.

{0,1}, {1, >1}, {0, > 0}r

2. Describe, in words, what the expression "(.)(.)\2\1" will match, and provide a word or expression as an example.

Words where some two letter sequence is repeated right after in reverse.

```
word <- c("abba", "abbaca", "abcabc") #the first two are matches
str_detect(word, "(.)(.)\\2\\1")
```

```
[1]  TRUE  TRUE FALSE
```

3. Produce an R string which the regular expression represented by "\..\..\.." matches. In other words, find a string `y` below that produces a TRUE in `str_detect`.

```
expression <- "Y.O.L.O."
str_detect(expression, "\\..\\..\\..")
```

```
[1] TRUE
```

4. Solve with `str_subset()`, using the words from `stringr::words`:

- Find all words that start or end with x.
- Find all words that start with a vowel and end with a consonant.
- Find all words that start and end with the same letter

```
str_subset(stringr::words, "(^x)|(x$)")
```

```
[1] "box" "sex" "six" "tax"
```

```
str_subset(stringr::words, "^[auiou].*[^aeiou]$")
```

```
 [1] "about"       "accept"      "account"     "across"      "act"
 [6] "actual"      "add"         "address"     "admit"       "affect"
[11] "afford"      "after"       "afternoon"   "again"       "against"
[16] "agent"       "air"         "all"         "allow"       "almost"
[21] "along"       "already"     "alright"     "although"    "always"
[26] "amount"      "and"         "another"     "answer"      "any"
[31] "apart"       "apparent"    "appear"      "apply"       "appoint"
[36] "approach"    "arm"         "around"      "art"         "as"
[41] "ask"         "at"          "attend"      "authority"   "away"
[46] "awful"       "identify"    "if"          "important"   "in"
[51] "indeed"      "individual"  "industry"    "inform"      "instead"
[56] "interest"    "invest"      "it"          "item"        "obvious"
[61] "occasion"    "odd"         "of"          "off"         "offer"
[66] "often"       "okay"        "old"         "on"          "only"
[71] "open"        "opportunity" "or"          "order"       "original"
[76] "other"       "ought"       "out"         "over"        "own"
[81] "under"       "understand"  "union"       "unit"        "university"
[86] "unless"      "until"       "up"          "upon"        "usual"
```

```
str_subset(stringr::words, "^(.)(.*)\\1$")
```

```
 [1] "america"    "area"       "dad"        "dead"       "depend"
 [6] "educate"    "else"       "encourage"  "engine"     "europe"
[11] "evidence"   "example"    "excuse"     "exercise"   "expense"
[16] "experience" "eye"        "health"     "high"       "knock"
[21] "level"      "local"      "nation"     "non"        "rather"
[26] "refer"      "remember"   "serious"    "stairs"     "test"
[31] "tonight"    "transport"  "treat"      "trust"      "window"
[36] "yesterday"
```

5. What words in `stringr::words` have the highest number of vowels? What words have the highest proportion of vowels? (Hint: what is the denominator?) Figure this out using the tidyverse and piping, starting with `as_tibble(words) |>`.

```
as_tibble(stringr::words) |>
  mutate(num_lett = str_count(value),
         num_vowel = str_count(value, "[aeiou]")) |>
  slice_max(num_vowel)
```

```
# A tibble: 8 x 3
  value       num_lett num_vowel
  <chr>          <int>     <int>
1 appropriate       11         5
2 associate          9         5
3 available          9         5
4 colleague          9         5
5 encourage          9         5
6 experience        10         5
7 individual        10         5
8 television        10         5
```

```
as_tibble(stringr::words) |>
  mutate(num_lett = str_count(value),
         num_vowel = str_count(value, "[aeiou]"),
         vowel_prop = num_vowel / num_lett) |>
  slice_max(vowel_prop, n = 5)
```

```
# A tibble: 21 x 4
   value  num_lett num_vowel vowel_prop
   <chr>     <int>     <int>      <dbl>
 1 a             1         1      1
 2 area          4         3      0.75
 3 idea          4         3      0.75
 4 age           3         2      0.667
 5 ago           3         2      0.667
 6 air           3         2      0.667
 7 die           3         2      0.667
 8 due           3         2      0.667
 9 eat           3         2      0.667
10 europe        6         4      0.667
# i 11 more rows
```

6. From the Harvard sentences data, use `str_extract` to produce a tibble with 3 columns: the sentence, the first word in the sentence, and the first word ending in "ed" (NA if there isn't one).

```
as_tibble(sentences) |>
  mutate(first_word = str_extract(value, "[^ ]*"),
         first_word_ed = str_extract(value, "[^ ]*ed\\b"))
```

```
# A tibble: 720 x 3
   value                                          first_word first_word_ed
   <chr>                                          <chr>      <chr>
 1 The birch canoe slid on the smooth planks.     The        <NA>
 2 Glue the sheet to the dark blue background.    Glue       <NA>
 3 It's easy to tell the depth of a well.         It's       <NA>
 4 These days a chicken leg is a rare dish.       These      <NA>
 5 Rice is often served in round bowls.           Rice       served
 6 The juice of lemons makes fine punch.          The        <NA>
 7 The box was thrown beside the parked truck.    The        parked
 8 The hogs were fed chopped corn and garbage.    The        fed
 9 Four hours of steady work faced us.            Four       faced
10 A large size in stockings is hard to sell.     A          <NA>
# i 710 more rows
```

7. Find and output all contractions (words with apostrophes) in the Harvard sentences, assuming no sentence has multiple contractions.

```
str_extract(sentences, "[^ ]*'[^ ]*")
```

```
  [1] NA           NA           "It's"       NA           NA
  [6] NA           NA           NA           NA           NA
 [11] NA           NA           NA           NA           NA
 [16] NA           NA           "man's"      NA           NA
 [21] NA           NA           NA           NA           NA
 [26] NA           NA           NA           NA           NA
 [31] NA           NA           NA           NA           NA
 [36] NA           NA           NA           NA           NA
 [41] NA           NA           NA           NA           NA
 [46] NA           NA           NA           NA           NA
 [51] NA           NA           NA           NA           NA
 [56] NA           NA           NA           NA           NA
 [61] NA           NA           NA           NA           NA
 [66] NA           NA           NA           NA           NA
 [71] NA           NA           NA           NA           NA
 [76] NA           NA           NA           NA           NA
 [81] NA           NA           NA           NA           NA
 [86] NA           NA           NA           NA           NA
 [91] NA           NA           NA           NA           NA
 [96] NA           NA           NA           NA           NA
[101] NA           NA           NA           "don't"      NA
[106] NA           NA           NA           NA           NA
```

```
[111] NA          NA          NA          NA          NA
[116] NA          NA          NA          NA          NA
[121] NA          NA          NA          NA          NA
[126] NA          NA          NA          NA          NA
[131] NA          NA          NA          NA          NA
[136] NA          NA          NA          NA          NA
[141] NA          NA          NA          NA          NA
[146] NA          NA          NA          NA          NA
[151] NA          NA          NA          NA          NA
[156] NA          NA          NA          NA          NA
[161] NA          NA          NA          NA          NA
[166] NA          NA          NA          NA          NA
[171] NA          NA          NA          NA          NA
[176] NA          NA          NA          NA          "store's"
[181] NA          NA          NA          NA          NA
[186] NA          NA          NA          NA          NA
[191] NA          NA          NA          NA          NA
[196] NA          NA          NA          NA          NA
[201] NA          NA          NA          NA          NA
[206] NA          NA          NA          NA          NA
[211] NA          NA          NA          NA          NA
[216] NA          NA          NA          NA          NA
[221] NA          NA          NA          NA          NA
[226] NA          NA          NA          NA          NA
[231] NA          NA          NA          NA          NA
[236] NA          NA          NA          NA          NA
[241] NA          NA          NA          NA          NA
[246] NA          NA          NA          NA          NA
[251] NA          NA          NA          NA          NA
[256] NA          NA          NA          NA          NA
[261] NA          NA          NA          NA          NA
[266] NA          NA          NA          NA          NA
[271] NA          NA          NA          NA          NA
[276] NA          NA          NA          NA          NA
[281] NA          NA          NA          NA          NA
[286] NA          NA          NA          NA          NA
[291] NA          NA          NA          NA          NA
[296] NA          NA          NA          NA          NA
[301] NA          NA          "workman's" NA          NA
[306] NA          NA          "Let's"     NA          NA
[311] NA          NA          NA          NA          NA
[316] NA          NA          NA          NA          NA
[321] NA          NA          NA          NA          NA
```

```
[326] "sun's"      NA           NA           NA           NA
[331] NA           NA           NA           NA           NA
[336] NA           NA           NA           NA           NA
[341] NA           NA           NA           NA           NA
[346] NA           NA           "child's"    NA           NA
[351] NA           NA           NA           NA           NA
[356] NA           "king's"     NA           NA           NA
[361] NA           NA           NA           NA           NA
[366] NA           NA           NA           NA           NA
[371] NA           NA           NA           NA           NA
[376] NA           NA           NA           NA           NA
[381] NA           NA           NA           NA           NA
[386] NA           NA           NA           NA           NA
[391] NA           NA           "It's"       NA           NA
[396] NA           NA           NA           NA           NA
[401] NA           NA           NA           NA           NA
[406] NA           NA           NA           NA           NA
[411] NA           NA           NA           NA           NA
[416] NA           NA           NA           NA           NA
[421] NA           NA           NA           NA           NA
[426] NA           NA           NA           NA           NA
[431] NA           NA           NA           NA           NA
[436] NA           NA           NA           NA           NA
[441] NA           NA           NA           NA           NA
[446] NA           NA           NA           NA           NA
[451] NA           NA           NA           NA           NA
[456] NA           NA           NA           NA           NA
[461] NA           NA           NA           NA           NA
[466] NA           NA           NA           NA           NA
[471] NA           NA           NA           NA           NA
[476] NA           "don't"      NA           NA           NA
[481] NA           NA           NA           NA           NA
[486] NA           NA           NA           NA           NA
[491] NA           NA           NA           NA           NA
[496] NA           NA           NA           NA           NA
[501] NA           NA           NA           NA           NA
[506] NA           NA           NA           NA           NA
[511] NA           NA           "queen's"    NA           NA
[516] NA           NA           NA           NA           NA
[521] NA           NA           NA           NA           NA
[526] NA           NA           NA           NA           NA
[531] NA           NA           NA           NA           NA
[536] "don't"      NA           NA           NA           NA
```

```
[541] NA          NA          NA          NA          NA
[546] NA          NA          NA          NA          NA
[551] NA          NA          NA          NA          NA
[556] NA          NA          NA          NA          NA
[561] NA          NA          NA          NA          NA
[566] NA          NA          NA          NA          NA
[571] NA          NA          NA          NA          NA
[576] NA          NA          NA          NA          NA
[581] NA          NA          NA          NA          NA
[586] NA          NA          NA          NA          NA
[591] NA          NA          NA          NA          NA
[596] NA          NA          NA          NA          NA
[601] NA          NA          NA          NA          NA
[606] NA          NA          NA          NA          NA
[611] NA          NA          NA          NA          NA
[616] NA          NA          NA          NA          NA
[621] NA          NA          NA          "don't"     NA
[626] NA          NA          NA          NA          NA
[631] NA          NA          NA          NA          NA
[636] "don't"     NA          NA          NA          NA
[641] NA          NA          NA          NA          NA
[646] NA          NA          NA          NA          NA
[651] NA          NA          NA          NA          NA
[656] NA          NA          NA          NA          NA
[661] NA          NA          NA          NA          NA
[666] NA          NA          NA          NA          NA
[671] NA          NA          NA          NA          NA
[676] NA          NA          "don't"     NA          "pirate's"
[681] NA          NA          NA          NA          NA
[686] NA          NA          NA          NA          NA
[691] NA          NA          NA          NA          NA
[696] NA          NA          NA          NA          NA
[701] NA          NA          NA          NA          NA
[706] NA          NA          NA          NA          NA
[711] NA          NA          "neighbor's" NA         NA
[716] NA          NA          NA          NA          NA
```

8. *Carefully* explain what the code below does, both line by line and in general terms.

It takes the word and reorders the characters so the last becomes the first and the first becomes the last. Then it joins creates a tibble of words that are the same after being rearranged.

Code: (I had an issue with rendering even though the code was successful)

temp <- str_replace_all(stringr::words, “^([A-Za-z])(.*)([a-z])$“,”\3\2\1”) #replaces the first character with the last and the last with the first as_tibble(words) |> #turns the list of words into a tibble semi_join(as_tibble(temp), by = c(“word” = “value”)) |> #semi_joins the adjusted words with the whole list of words so only words that are the same or match a different word after being rearranged are shown. print(n = Inf) #prints the list of matched words

We will check out the Rotten Tomatoes page for the 2017 movie Coco, scrape information from that page (we'll get into web scraping in a few weeks!), clean it up into a usable format, and answer some questions using strings and regular expressions.

```
# used to work
# coco <- read_html("https://www.rottentomatoes.com/m/coco_2017")

# robotstxt::paths_allowed("https://www.rottentomatoes.com/m/coco_2017")

library(polite)
coco <- "https://www.rottentomatoes.com/m/coco_2017" |>
  bow() |>
  scrape()

top_reviews <-
  "https://www.rottentomatoes.com/m/coco_2017/reviews?type=top_critics" |>
  bow() |>
  scrape()
top_reviews <- html_nodes(top_reviews, ".review-text")
top_reviews <- html_text(top_reviews)

user_reviews <-
  "https://www.rottentomatoes.com/m/coco_2017/reviews?type=user" |>
  bow() |>
  scrape()
user_reviews <- html_nodes(user_reviews, ".js-review-text")
user_reviews <- html_text(user_reviews)
```

9. `top_reviews` is a character vector containing the 20 most recent critic reviews (along with some other junk) for Coco, while `user_reviews` is a character vector with the 10 most recent user reviews.

a) Explain how the code below helps clean up both `user_reviews` and `top_reviews` before we start using them.

It removes the empty space before and after a string.

```r
user_reviews <- str_trim(user_reviews)
top_reviews <- str_trim(top_reviews)
```

b) Print out the critic reviews where the reviewer mentions "emotion" or "cry". Think about various forms ("cried", "emotional", etc.) You may want to turn reviews to all lower case before searching for matches.

```r
tibble(review = top_reviews) |>
  mutate(review = str_to_lower(review),
         review = str_extract(review, ".*cry.*\\.|.*cri.*\\.|.*emotion.*\\.")) |>
  filter(!is.na(review)) |>
  print()
```

```
# A tibble: 3 x 1
  review
  <chr>
1 a wonderful return to form for pixar, who again deliver the emotional and cre~
2 at worst it suggests that the brains trust at pixar, after 22 years of peerle~
3 funny, irreverent and eye-popping. it will also make you want to cry at least~
```

c) In critic reviews, replace all instances where "Pixar" is used with its full name: "Pixar Animation Studios".

```r
tibble(review = top_reviews) |>
  mutate(review = str_replace(review, "pixar", "Pixar Animation Studios"))
```

```
# A tibble: 20 x 1
   review
   <chr>
 1 "A fine addition to the Pixar legacy… a very sweet film about family, very t~
 2 "An unexpectedly brilliant and dynamic story about lineage, connection, and ~
 3 "In a country with an ever increasing Hispanic and Mexican population, a fil~
 4 "I don't think there's any question that Coco is really great."
 5 "Several times I found myself sobbing without knowing exactly why only to re~
 6 "A wonderful return to form for Pixar, who again deliver the emotional and c~
 7 "The film has a galloping rhythm, and the animation is scrupulous and ravish~
 8 "On paper, the mythology scans as complicated and dark, but in the capable h~
 9 "Pixar's latest project is a glittering return to non-franchise form after 2~
10 "Its victorious denouement offers everyone a different way to think about wh~
11 "At worst it suggests that the brains trust at Pixar, after 22 years of peer~
12 "Funny, irreverent and eye-popping. It will also make you want to cry at lea~
```

```
13 "This is a charming and very memorable film."
14 "Despite the fact that it's so well told and really beautifully directed, it~
15 "... Coco is another triumph for Pixar..."
16 "Funny and heart-tugging with some knockout tunes, the movie glows with warm~
17 "Not top-tier Pixar. But decent enough."
18 "Pixar has raised the animation bar again, with its most musical - and argua~
19 "While the animation is Pixar perfect, I don't think the story grips the vie~
20 "Every plot point and thematic implication slots into place, but the pleasur~
```

d) Find out how many times each user uses "I" in their review. Remember that it could be used as upper or lower case, at the beginning, middle, or end of a sentence, etc.

```
tibble(review = user_reviews) |>
  mutate(review = str_to_lower(review),
         i_count = str_count(review, "i "))
```

```
# A tibble: 20 x 2
   review                                                              i_count
   <chr>                                                                 <int>
 1 "\n                    i loved the plot twist at the end, and the ~       3
 2 "\n                    captivatingly beautiful. will tug at your h~      0
 3 "\n                    such a beautiful film, this story is disney~      0
 4 "\n                    i find myself as a grown man watching this ~      2
 5 "\n                    this was a good and funny musical that ever~      0
 6 "\n                    definitely the best movie of 2017 and what ~      2
 7 "\n                    great movie excited about the second movie\~      0
 8 "\n                    flawless film. just stunning. \n\n        ~       0
 9 "\n                    all time classic. just masterfully done acr~      2
10 "\n                    i've seen this movie a hundred times, and e~      0
11 "\n                    the film's visuals and narrative take you o~      0
12 "\n                    how can i not love this movie? coco is my f~      4
13 "\n                    what's not to love?  this tender story of t~      0
14 "\n                    another enjoyable movie from pixar.  heartw~      0
15 "\n                    it's one of the best movies pixar has made ~      1
16 "\n                    very colorful much like encanto and had a f~      0
17 "\n                    mild spoilers: \n\ni've heard people make f~      2
18 "\n                    so good, visually stunning.\n             ~       0
19 "\n                    excellent coulorful movie explaining the me~      0
20 "\n                    a very good and well made pixar film. the m~      0
```

e) Do critics or users have more complex reviews, as measured by average number of commas used? Be sure your code weeds out commas used in numbers, such as "12,345".

```r
tibble(review = top_reviews) |>
  mutate(comma_count = str_count(review, "[^\\n], ")) |>
  summarize(mean = mean(comma_count))
```

```
# A tibble: 1 x 1
   mean
  <dbl>
1  1.35
```

```r
tibble(review = user_reviews) |>
  mutate(comma_count = str_count(review, "[^\\n], ")) |>
  summarize(mean = mean(comma_count))
```

```
# A tibble: 1 x 1
   mean
  <dbl>
1     2
```

Users have more complex reviews.