HW4 Key

```
library(tidyverse)
library(rvest)
library(httr)

#spotify <- read_csv("Data/spotify.csv")
spotify <- read_csv("https://proback.github.io/264_fall_2024/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 & ~
 2 Run the World (G~  Beyon~ 2011-06-24         4          post-te~ "post-teen a~
 3 Formation          Beyon~ 2016-04-23         Lemonade   hip pop  "Feeling Acc~
 4 7/11               Beyon~ 2014-11-24         BEYONCÉ [~ hip pop  "Feeling Acc~
 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 Freest~
 7 Poetic Justice     Kendr~ 2012               good kid,~ hip hop  "Hip Hop Con~
 8 A.D.H.D            Kendr~ 2011-07-02         Section.80 souther~ "Hip-Hop 'n ~
```

```
 9 Ya Estuvo          Kid F~ 1990-01-01          Hispanic ~ latin h~ "HIP-HOP: La~
10 Runnin (with A$A~ Mike ~ 2018-11-16          Creed II:~ gangste~ "RAP Gangsta"
```

Exercises from 11_strings_part2.qmd

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, "pop"))
```

```
[1] "character"
```

```
class(str_view(spot_smaller$subgenre, "pop"))
```

```
[1] "stringr_view"
```

```
#str_view(spot_smaller$subgenre, "pop", match = NA)
#str_view(spot_smaller$subgenre, "pop", html = TRUE)

str_subset(spot_smaller$subgenre, "pop")
```

```
[1] "indie poptimism" "post-teen pop"   "hip pop"         "hip pop"
[5] "latin pop"
```

```
str_detect(spot_smaller$subgenre, "pop")
```

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

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)

```
spot_smaller |>
  filter(str_detect(subgenre, "pop"))
```

```
# A tibble: 5 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 & ~
2 Run the World (Gi~ Beyon~ 2011-06-24         4          post-te~ "post-teen a~
3 Formation          Beyon~ 2016-04-23         Lemonade   hip pop  "Feeling Acc~
4 7/11               Beyon~ 2014-11-24         BEYONCÉ [~ hip pop  "Feeling Acc~
5 My Oh My (feat. D~ Camil~ 2019-12-06         Romance    latin p~ "2020 Hits &~
```

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

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

```
# Solution 1: filter twice
spot_smaller |>
  mutate(title_length = str_length(title))  |>
  filter(str_detect(subgenre, "pop")) |>
  summarize(mean_title = mean(title_length))
```

```
# A tibble: 1 x 1
  mean_title
       <dbl>
1       13.6
```

```
spot_smaller |>
  mutate(title_length = str_length(title))  |>
  filter(!str_detect(subgenre, "pop")) |>
  summarize(mean_title = mean(title_length))
```

```
# A tibble: 1 x 1
  mean_title
       <dbl>
1       18.6
```

```
# Solution 2 (better!): mutate, group_by, summarize
spot_smaller |>
  mutate(title_length = str_length(title),
         sub_pop = str_detect(subgenre, "pop")) |>
  group_by(sub_pop) |>
  summarize(mean_title_length = mean(title_length))
```

```
# A tibble: 2 x 2
  sub_pop mean_title_length
  <lgl>               <dbl>
1 FALSE                18.6
2 TRUE                 13.6
```

```
# Produces a little nicer output
spot_smaller |>
  mutate(title_length = str_length(title),
         sub_pop = ifelse(str_detect(subgenre, "pop"), "Genre with pop",
                          "Genre without pop")) |>
  group_by(sub_pop) |>
  summarize(mean_title_length = mean(title_length))
```

```
# A tibble: 2 x 2
  sub_pop           mean_title_length
  <chr>                         <dbl>
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>
```

```
#str_view(str_to_lower(bigspotify$track_name), "love")


bigspotify |>
  filter(!is.na(track_name)) |>
  mutate(title_lower = str_to_lower(track_name),
         love = str_detect(title_lower, "love")) |>
  group_by(playlist_genre) |>
  summarize(prop_love = mean(love))
```

```
# 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()

# For example, to find words with "tion" at any point, I could use:
#str_view(words, "tion")
str_subset(words, "tion")
```

```
 [1] "condition" "function"  "mention"   "motion"    "nation"    "position"
 [7] "question"  "relation"  "section"   "station"
```

```
str_subset(words, "^y")
```

```
[1] "year"      "yes"       "yesterday" "yet"       "you"       "young"
```

```
str_subset(words, "x$")
```

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

```
str_subset(words, "^...$")
```

```
  [1] "act" "add" "age" "ago" "air" "all" "and" "any" "arm" "art" "ask" "bad"
 [13] "bag" "bar" "bed" "bet" "big" "bit" "box" "boy" "bus" "but" "buy" "can"
 [25] "car" "cat" "cup" "cut" "dad" "day" "die" "dog" "dry" "due" "eat" "egg"
 [37] "end" "eye" "far" "few" "fit" "fly" "for" "fun" "gas" "get" "god" "guy"
 [49] "hit" "hot" "how" "job" "key" "kid" "lad" "law" "lay" "leg" "let" "lie"
 [61] "lot" "low" "man" "may" "mrs" "new" "non" "not" "now" "odd" "off" "old"
 [73] "one" "out" "own" "pay" "per" "put" "red" "rid" "run" "say" "see" "set"
 [85] "sex" "she" "sir" "sit" "six" "son" "sun" "tax" "tea" "ten" "the" "tie"
 [97] "too" "top" "try" "two" "use" "war" "way" "wee" "who" "why" "win" "yes"
[109] "yet" "you"
```

```
str_subset(words, ".......")
```

```
  [1] "absolute"    "account"     "achieve"     "address"     "advertise"
  [6] "afternoon"   "against"     "already"     "alright"     "although"
 [11] "america"     "another"     "apparent"    "appoint"     "approach"
 [16] "appropriate" "arrange"     "associate"   "authority"   "available"
 [21] "balance"     "because"     "believe"     "benefit"     "between"
 [26] "brilliant"   "britain"     "brother"     "business"    "certain"
 [31] "chairman"    "character"   "Christmas"   "colleague"   "collect"
 [36] "college"     "comment"     "committee"   "community"   "company"
 [41] "compare"     "complete"    "compute"     "concern"     "condition"
 [46] "consider"    "consult"     "contact"     "continue"    "contract"
 [51] "control"     "converse"    "correct"     "council"     "country"
 [56] "current"     "decision"    "definite"    "department"  "describe"
 [61] "develop"     "difference"  "difficult"   "discuss"     "district"
 [66] "document"    "economy"     "educate"     "electric"    "encourage"
 [71] "english"     "environment" "especial"    "evening"     "evidence"
 [76] "example"     "exercise"    "expense"     "experience"  "explain"
 [81] "express"     "finance"     "fortune"     "forward"     "function"
 [86] "further"     "general"     "germany"     "goodbye"     "history"
 [91] "holiday"     "hospital"    "however"     "hundred"     "husband"
 [96] "identify"    "imagine"     "important"   "improve"     "include"
[101] "increase"    "individual"  "industry"    "instead"     "interest"
[106] "introduce"   "involve"     "kitchen"     "language"    "machine"
[111] "meaning"     "measure"     "mention"     "million"     "minister"
[116] "morning"     "necessary"   "obvious"     "occasion"    "operate"
[121] "opportunity" "organize"    "original"    "otherwise"   "paragraph"
[126] "particular"  "pension"     "percent"     "perfect"     "perhaps"
[131] "photograph"  "picture"     "politic"     "position"    "positive"
[136] "possible"    "practise"    "prepare"     "present"     "pressure"
[141] "presume"     "previous"    "private"     "probable"    "problem"
```

```
[146] "proceed"     "process"     "produce"     "product"     "programme"
[151] "project"     "propose"     "protect"     "provide"     "purpose"
[156] "quality"     "quarter"     "question"    "realise"     "receive"
[161] "recognize"   "recommend"   "relation"    "remember"    "represent"
[166] "require"     "research"    "resource"    "respect"     "responsible"
[171] "saturday"    "science"     "scotland"    "secretary"   "section"
[176] "separate"    "serious"     "service"     "similar"     "situate"
[181] "society"     "special"     "specific"    "standard"    "station"
[186] "straight"    "strategy"    "structure"   "student"     "subject"
[191] "succeed"     "suggest"     "support"     "suppose"     "surprise"
[196] "telephone"   "television"  "terrible"    "therefore"   "thirteen"
[201] "thousand"    "through"     "thursday"    "together"    "tomorrow"
[206] "tonight"     "traffic"     "transport"   "trouble"     "tuesday"
[211] "understand"  "university"  "various"     "village"     "wednesday"
[216] "welcome"     "whether"     "without"     "yesterday"
```

```
str_subset(words, "^[aeiou]")
```

```
  [1] "a"           "able"        "about"       "absolute"    "accept"
  [6] "account"     "achieve"     "across"      "act"         "active"
 [11] "actual"      "add"         "address"     "admit"       "advertise"
 [16] "affect"      "afford"      "after"       "afternoon"   "again"
 [21] "against"     "age"         "agent"       "ago"         "agree"
 [26] "air"         "all"         "allow"       "almost"      "along"
 [31] "already"     "alright"     "also"        "although"    "always"
 [36] "america"     "amount"      "and"         "another"     "answer"
 [41] "any"         "apart"       "apparent"    "appear"      "apply"
 [46] "appoint"     "approach"    "appropriate" "area"        "argue"
 [51] "arm"         "around"      "arrange"     "art"         "as"
 [56] "ask"         "associate"   "assume"      "at"          "attend"
 [61] "authority"   "available"   "aware"       "away"        "awful"
 [66] "each"        "early"       "east"        "easy"        "eat"
 [71] "economy"     "educate"     "effect"      "egg"         "eight"
 [76] "either"      "elect"       "electric"    "eleven"      "else"
 [81] "employ"      "encourage"   "end"         "engine"      "english"
 [86] "enjoy"       "enough"      "enter"       "environment" "equal"
 [91] "especial"    "europe"      "even"        "evening"     "ever"
 [96] "every"       "evidence"    "exact"       "example"     "except"
[101] "excuse"      "exercise"    "exist"       "expect"      "expense"
[106] "experience"  "explain"     "express"     "extra"       "eye"
[111] "idea"        "identify"    "if"          "imagine"     "important"
[116] "improve"     "in"          "include"     "income"      "increase"
```

```
[121] "indeed"      "individual"  "industry"    "inform"      "inside"
[126] "instead"     "insure"      "interest"    "into"        "introduce"
[131] "invest"      "involve"     "issue"       "it"          "item"
[136] "obvious"     "occasion"    "odd"         "of"          "off"
[141] "offer"       "office"      "often"       "okay"        "old"
[146] "on"          "once"        "one"         "only"        "open"
[151] "operate"     "opportunity" "oppose"      "or"          "order"
[156] "organize"    "original"    "other"       "otherwise"   "ought"
[161] "out"         "over"        "own"         "under"       "understand"
[166] "union"       "unit"        "unite"       "university"  "unless"
[171] "until"       "up"          "upon"        "use"         "usual"
```

```
str_subset(words, "[^e]ed$")
```

```
[1] "bed"     "hundred" "red"
```

```
str_subset(words, "q[^u]")
```

```
character(0)
```

```
#str_view("wqwrw", "q[^u]")
```

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)
```

```
# A tibble: 83 x 2
   track_name                                                 track_artist
   <chr>                                                      <chr>
 1 Do You Mean (feat. Ty Dolla $ign & bülow) - Myon Remix     The Chainsmokers
 2 Midnight Hour with Boys Noize & Ty Dolla $ign              Skrillex
 3 Do You Mean (feat. Ty Dolla $ign & bülow)                  The Chainsmokers
 4 Work from Home (feat. Ty Dolla $ign)                       Fifth Harmony
 5 Timber (feat. Ke$ha)                                       Pitbull
 6 Work from Home (feat. Ty Dolla $ign)                       Fifth Harmony
 7 Swalla (feat. Nicki Minaj & Ty Dolla $ign)                 Jason Derulo
 8 Timber (feat. Ke$ha)                                       Pitbull
```

```
 9 My First Kiss - feat. Ke$ha                                  3OH!3
10 Work from Home (feat. Ty Dolla $ign)                         Fifth Harmony
# i 73 more rows
```

```
bigspotify |>
  filter(str_detect(track_name, "\\$")) |>
#  filter(!str_detect(track_name, "\\(.*\\$.*\\)")) |>
  filter(!str_detect(track_name, "(feat|with).*\\$")) |>
  mutate(only_name = str_remove(track_name, "\\(?(feat|with).*")) |>
  select(track_name, only_name, track_artist) |>
  print(n = Inf)
```

```
# A tibble: 25 x 3
   track_name                                          only_name     track_artist
   <chr>                                               <chr>         <chr>
 1 Wing$                                               "Wing$"       Macklemore ~
 2 $Dreams                                             "$Dreams"     Max Frost
 3 $ave Dat Money (feat. Fetty Wap & Rich Homie Quan) "$ave Dat Mo~ Lil Dicky
 4 NO TRU$T                                            "NO TRU$T"    NUGAT
 5 A$AP Forever                                        "A$AP Foreve~ A$AP Rocky
 6 M'$ (feat. Lil Wayne)                               "M'$ "        A$AP Rocky
 7 Sie wollen meine Loui$ (Don Dollar)                 "Sie wollen ~ Kulturerbe ~
 8 Foe Tha Love Of $                                   "Foe Tha Lov~ Bone Thugs-~
 9 A$AP                                                "A$AP"        Dillom
10 $$$ - Remix                                         "$$$ - Remix" Saramalacara
11 Fre$h                                               "Fre$h"       Lil Whigga
12 $ENHOR                                              "$ENHOR"      FBC
13 $20 Fine                                            "$20 Fine"    Jimi Hendrix
14 A$IAN BOY                                           "A$IAN BOY"   Chriilz
15 ¿Cuánto E$?                                         "¿Cuánto E$?" Jhay Cortez
16 $. A. N. T. E. R. Í. A.                             "$. A. N. T.~ Doble Porci~
17 A$IAN BOY                                           "A$IAN BOY"   Chriilz
18 Bernice Burgo$                                      "Bernice Bur~ Nino Khayyam
19 $100 (feat. Polo Donatello)                         "$100 "       Mibbs
20 M'$                                                 "M'$"         A$AP Rocky
21 Dat $tick                                           "Dat $tick"   Rich Brian
22 $ave Dat Money (feat. Fetty Wap & Rich Homie Quan) "$ave Dat Mo~ Lil Dicky
23 Love$ick                                            "Love$ick"    Mura Masa
24 A$IAN BOY                                           "A$IAN BOY"   Chriilz
25 CA$H                                                "CA$H"        Olly James
```

Note: the commented filter misses a few since not all featured artists are in parentheses. Also,

the current solution has issues if "with" is part of the song title, or if both the title and featured artist contain $ (could solve by first deleting rows with feat|with, then picking off names with $).

Also the mutate line is not necessary, but it picks off only the track name.

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

```
bigspotify |>
  filter(str_detect(track_name, "\\$\\d")) |>
  select(track_name, track_artist)
```

```
# A tibble: 2 x 2
  track_name                  track_artist
  <chr>                       <chr>
1 $20 Fine                    Jimi Hendrix
2 $100 (feat. Polo Donatello) Mibbs
```

```
bigspotify |>
  filter(str_detect(track_name, "\\$[0-9]")) |>
  select(track_name, track_artist)
```

```
# A tibble: 2 x 2
  track_name                  track_artist
  <chr>                       <chr>
1 $20 Fine                    Jimi Hendrix
2 $100 (feat. Polo Donatello) Mibbs
```

Use at least 1 repetition symbol when solving 8-10 below

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

```
# preferred solution
#str_view(spot_smaller$album_name, "[A-Z](\\.?)[A-Z]")

# not bad but picks up .'s on either side of A.A
#str_view(spot_smaller$album_name, "[A-Z.]{2,}")
```

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

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

```
#str_view(c("dog", "$1.23", "weird stuff"), "^.*$")   # anything
#str_view(c("{a}", "{abc}", "{}"), "\\{.+\\}")

test <- c("dog", "$1.23", "weird stuff")
str_subset(test, "^.*$")   # anything
```

```
[1] "dog"         "$1.23"       "weird stuff"
```

```
test <- c("{a}", "{abc}", "{}")
str_subset(test, "\\{.+\\}")
```

```
[1] "{a}"   "{abc}"
```

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(words, "^[^aeiou]{3}")
# produces a long list, but includes words like "dry"
#str_view(words, "^[^aeiou][^aeiouy]{2}")

#str_view(words, "([aeiou][^aeiou]){2,}")
# another long list - prints first 20

str_subset(words, "^[^aeiou][^aeiouy]{2}")
```

```
 [1] "Christ"    "Christmas" "mrs"       "scheme"    "school"    "straight"
 [7] "strategy"  "street"    "strike"    "strong"    "structure" "three"
[13] "through"   "throw"
```

```
str_subset(words, "([aeiou][^aeiou]){2,}")
```

```
  [1] "absolute"    "agent"       "along"       "america"     "another"
  [6] "apart"       "apparent"    "authority"   "available"   "aware"
 [11] "away"        "balance"     "basis"       "become"      "before"
 [16] "begin"       "behind"      "benefit"     "business"    "character"
```

```
 [21] "closes"      "community"   "consider"    "cover"       "debate"
 [26] "decide"      "decision"    "definite"    "department"  "depend"
 [31] "design"      "develop"     "difference"  "difficult"   "direct"
 [36] "divide"      "document"    "during"      "economy"     "educate"
 [41] "elect"       "electric"    "eleven"      "encourage"   "environment"
 [46] "europe"      "even"        "evening"     "ever"        "every"
 [51] "evidence"    "exact"       "example"     "exercise"    "exist"
 [56] "family"      "figure"      "final"       "finance"     "finish"
 [61] "friday"      "future"      "general"     "govern"      "holiday"
 [66] "honest"      "hospital"    "however"     "identify"    "imagine"
 [71] "individual"  "interest"    "introduce"   "item"        "jesus"
 [76] "level"       "likely"      "limit"       "local"       "major"
 [81] "manage"      "meaning"     "measure"     "minister"    "minus"
 [86] "minute"      "moment"      "money"       "music"       "nature"
 [91] "necessary"   "never"       "notice"      "okay"        "open"
 [96] "operate"     "opportunity" "organize"    "original"    "over"
[101] "paper"       "paragraph"   "parent"      "particular"  "photograph"
[106] "police"      "policy"      "politic"     "position"    "positive"
[111] "power"       "prepare"     "present"     "presume"     "private"
[116] "probable"    "process"     "produce"     "product"     "project"
[121] "proper"      "propose"     "protect"     "provide"     "quality"
[126] "realise"     "reason"      "recent"      "recognize"   "recommend"
[131] "record"      "reduce"      "refer"       "regard"      "relation"
[136] "remember"    "report"      "represent"   "result"      "return"
[141] "saturday"    "second"      "secretary"   "secure"      "separate"
[146] "seven"       "similar"     "specific"    "strategy"    "student"
[151] "stupid"      "telephone"   "television"  "therefore"   "thousand"
[156] "today"       "together"    "tomorrow"    "tonight"     "total"
[161] "toward"      "travel"      "unit"        "unite"       "university"
[166] "upon"        "visit"       "water"       "woman"
```

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

```
#str_view(spot_smaller$title, "\\b[^ ]+\\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:

[1] “Hear” “Run” “Formation” “7/11” “My” “It’s”
[7] “Poetic” “A.D.H.D” “Ya” “Runnin”

Then this:

A tibble: 10 × 2

title first_word
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

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

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

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

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

```
# 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
```

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")) |>
  count(playlist_decade)
```

```
# A tibble: 5 x 2
  playlist_decade     n
  <chr>           <int>
1 00's               45
2 08's               99
3 70's               67
4 80's              348
5 90's              405
```

```
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)
```

```
# A tibble: 7 x 2
  playlist_decade     n
  <chr>           <int>
1 00s                45
2 08s                99
3 10s               281
4 50s               100
5 70s               442
6 80s               682
7 90s              1013
```

```r
# What's up with the 99 "08s"?
bigspotify |>
  filter(str_detect(playlist_name, "08('?)s")) |>
  select(playlist_name) |>
  print(n = Inf)
```

```
# A tibble: 99 x 1
   playlist_name
   <chr>
 1 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 2 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 3 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 4 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 5 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 6 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 7 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 8 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
 9 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
10 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
11 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
12 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
13 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
14 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
15 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
16 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
17 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
18 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
19 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
20 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
21 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
22 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
23 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
24 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
25 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
26 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
27 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
28 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
29 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
30 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
31 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
32 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
33 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
```

```
34 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
35 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
36 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
37 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
38 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
39 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
40 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
41 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
42 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
43 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
44 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
45 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
46 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
47 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
48 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
49 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
50 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
51 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
52 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
53 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
54 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
55 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
56 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
57 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
58 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
59 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
60 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
61 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
62 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
63 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
64 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
65 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
66 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
67 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
68 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
69 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
70 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
71 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
72 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
73 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
74 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
75 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
76 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
```

```
77 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
78 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
79 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
80 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
81 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
82 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
83 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
84 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
85 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
86 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
87 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
88 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
89 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
90 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
91 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
92 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
93 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
94 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
95 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
96 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
97 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
98 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
99 "\U0001f525\U0001f4b5 Hip Hop, Rap, Heavy 808's - New School"
```

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

- (.)\1\1
- "(.)(.)(.).*\3\2\1"

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

```
# (.)\1\1 : The same character appearing three times in a row. E.g. "aaa"
#str_view("abcccde", "(.)\\1\\1")
# "(.)(.)(.).*\\3\\2\\1" Three characters followed by zero or more characters of any kind fo
#str_view("We built strong forts.", "(.)(.)(.).*\\3\\2\\1")

str_subset(words, "(.)\\1\\1")
```

```
character(0)
```

```
str_subset(words, "(.)(.)(.).*\\3\\2\\1")
```

```
[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(words, "([A-Za-z][A-Za-z]).*\\1")
str_subset(words, "([A-Za-z][A-Za-z]).*\\1")
```

```
 [1] "appropriate" "church"      "condition"   "decide"      "environment"
 [6] "london"      "paragraph"   "particular"  "photograph"  "prepare"
[11] "pressure"    "remember"    "represent"   "require"     "sense"
[16] "therefore"   "understand"  "whether"
```

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 |>
  select(album_release_date) |>
  mutate(new_date = str_replace(album_release_date,
           "(\\d{4})-(\\d{2})-(\\d{2})", "\\2-\\3-\\1"))
```

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

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

```
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."
```

The second and third words will be flipped.

Exercises from 12_strings_part3.qmd

On Your Own - Extra practice with strings and regular expressions

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

```
# ? = {0,1}
# + = {1,}
# * = {0,}
```

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

```
# A pair of characters followed by the same pair in reversed order.
str_detect("ABBA", "(.)(.)\\2\\1")
```

```
[1] TRUE
```

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`.

Many possibilities.... here are two examples:

```
#y <- "???"
#str_detect(y, "\\..\\..\\..")

y <- "stuff.h.i.stuff"
str_detect(y, "\\..\\..\\..")
```

```
[1] TRUE
```

```
y <- ".h.i.i"
str_detect(y, "\\..\\..\\..")
```

```
[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(words, "^x|x$")
```

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

```
str_subset(words, "^[aeiou].*[^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"       "each"        "early"       "east"        "easy"
 [51] "eat"         "economy"     "effect"      "egg"         "eight"
 [56] "either"      "elect"       "electric"    "eleven"      "employ"
 [61] "end"         "english"     "enjoy"       "enough"      "enter"
 [66] "environment" "equal"       "especial"    "even"        "evening"
 [71] "ever"        "every"       "exact"       "except"      "exist"
 [76] "expect"      "explain"     "express"     "identify"    "if"
 [81] "important"   "in"          "indeed"      "individual"  "industry"
 [86] "inform"      "instead"     "interest"    "invest"      "it"
 [91] "item"        "obvious"     "occasion"    "odd"         "of"
 [96] "off"         "offer"       "often"       "okay"        "old"
[101] "on"          "only"        "open"        "opportunity" "or"
[106] "order"       "original"    "other"       "ought"       "out"
```

```
[111] "over"        "own"         "under"       "understand"  "union"
[116] "unit"        "university"  "unless"      "until"       "up"
[121] "upon"        "usual"
```

```
str_subset(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(words) |>
  mutate(num_vowels = str_count(value, "[aeiou]"),
         prop_vowels = num_vowels / str_length(value)) |>
  arrange(-num_vowels)
```

```
# A tibble: 980 x 3
   value       num_vowels prop_vowels
   <chr>            <int>       <dbl>
 1 appropriate          5       0.455
 2 associate            5       0.556
 3 available            5       0.556
 4 colleague            5       0.556
 5 encourage            5       0.556
 6 experience           5       0.5
 7 individual           5       0.5
 8 television           5       0.5
 9 absolute             4       0.5
10 achieve              4       0.571
# i 970 more rows
```

```r
as_tibble(words) |>
  mutate(num_vowels = str_count(value, "[aeiou]"),
         prop_vowels = num_vowels / str_length(value)) |>
  arrange(-prop_vowels)
```

```
# A tibble: 980 x 3
   value  num_vowels prop_vowels
   <chr>       <int>       <dbl>
 1 a               1       1
 2 area            3       0.75
 3 idea            3       0.75
 4 age             2       0.667
 5 ago             2       0.667
 6 air             2       0.667
 7 die             2       0.667
 8 due             2       0.667
 9 eat             2       0.667
10 europe          4       0.667
# i 970 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).

```r
as_tibble(sentences) |>
  rename("sentence" = "value") |>
  mutate(first_word = str_extract(sentences, "[a-zA-Z']+"),
         first_ed_word = str_extract(sentence, "\\b[A-Za-z']+ed\\b"))
```

```
# A tibble: 720 x 3
   sentence                                    first_word first_ed_word
   <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.

```
contraction <- "([A-Za-z]+)'([A-Za-z]+)"
sentences |>
  str_subset(contraction) |>
  str_extract(contraction)
```

```
 [1] "It's"       "man's"      "don't"      "store's"    "workman's"
 [6] "Let's"      "sun's"      "child's"    "king's"     "It's"
[11] "don't"      "queen's"    "don't"      "don't"      "don't"
[16] "don't"      "pirate's"   "neighbor's"
```

```
as_tibble(sentences) |>
  filter(str_detect(value, contraction)) |>
  mutate(contraction = str_extract(value, contraction))
```

```
# A tibble: 18 x 2
   value                                          contraction
   <chr>                                          <chr>
 1 It's easy to tell the depth of a well.         It's
 2 The soft cushion broke the man's fall.         man's
 3 Open the crate but don't break the glass.      don't
 4 Add the store's account to the last cent.      store's
 5 The beam dropped down on the workman's head.   workman's
 6 Let's all join as we sing the last chorus.     Let's
 7 The copper bowl shone in the sun's rays.       sun's
 8 A child's wit saved the day for us.            child's
 9 A ripe plum is fit for a king's palate.        king's
10 It's a dense crowd in two distinct ways.       It's
11 We don't get much money but we have fun.       don't
12 Ripe pears are fit for a queen's table.        queen's
13 Cheap clothes are flashy but don't last.       don't
14 The facts don't always show who is right.      don't
15 Pack the kits and don't forget the salt.       don't
16 We don't like to admit our small faults.       don't
17 Dig deep in the earth for pirate's gold.       pirate's
18 She saw a cat in the neighbor's house.         neighbor's
```

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

```
temp <- str_replace_all(words, "^([A-Za-z])(.*)([a-z])$", "\\3\\2\\1")
as_tibble(words) |>
  semi_join(as_tibble(temp)) |>
  print(n = Inf)
```

```
Joining with `by = join_by(value)`

# A tibble: 45 x 1
   value
   <chr>
 1 a
 2 america
 3 area
 4 dad
 5 dead
 6 deal
 7 dear
 8 depend
 9 dog
10 educate
11 else
12 encourage
13 engine
14 europe
15 evidence
16 example
17 excuse
18 exercise
19 expense
20 experience
21 eye
22 god
23 health
24 high
25 knock
26 lead
27 level
28 local
29 nation
30 no
```

```
31 non
32 on
33 rather
34 read
35 refer
36 remember
37 serious
38 stairs
39 test
40 tonight
41 transport
42 treat
43 trust
44 window
45 yesterday
```

The first line switches the first and last letters in all words. Then we start with the words as a tibble and perform a filtering join, where only words that also appear in the list of words with first and last letters exchanged are kept and printed. The overall effect is a tibble containing 45 words that also appear in words after the first and last letters are exchanged.

Coco and Rotten Tomatoes

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")
```

```
 www.rottentomatoes.com
```

```
[1] TRUE
```

```
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.

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

The user reviews have extra characters and spaces at the beginning and end that we need to trim off.

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.

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

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.

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
str_subset(str_to_lower(top_reviews), "cried|cry|emotion")
```

```
[1] "a wonderful return to form for pixar, who again deliver the emotional and creative goods
[2] "at worst it suggests that the brains trust at pixar, after 22 years of peerless output
[3] "funny, irreverent and eye-popping. it will also make you want to cry at least once but
```

```r
str_replace_all(top_reviews, "Pixar", "Pixar Animation Studios")
```

```
 [1] "A fine addition to the Pixar Animation Studios legacy… a very sweet film about family,
 [2] "An unexpectedly brilliant and dynamic story about lineage, connection, and self-discov
 [3] "In a country with an ever increasing Hispanic and Mexican population, a film like Coco
 [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 realize why t
 [6] "A wonderful return to form for Pixar Animation Studios, who again deliver the emotiona
 [7] "The film has a galloping rhythm, and the animation is scrupulous and ravishing, from i
 [8] "On paper, the mythology scans as complicated and dark, but in the capable hands of Aca
 [9] "Pixar Animation Studios's latest project is a glittering return to non-franchise form
[10] "Its victorious denouement offers everyone a different way to think about what it means
[11] "At worst it suggests that the brains trust at Pixar Animation Studios, after 22 years
[12] "Funny, irreverent and eye-popping. It will also make you want to cry at least once but
[13] "This is a charming and very memorable film."
[14] "Despite the fact that it's so well told and really beautifully directed, it doesn't ha
[15] "... Coco is another triumph for Pixar Animation Studios..."
[16] "Funny and heart-tugging with some knockout tunes, the movie glows with warmth. And how
[17] "Not top-tier Pixar Animation Studios. But decent enough."
[18] "Pixar Animation Studios has raised the animation bar again, with its most musical - an
[19] "While the animation is Pixar Animation Studios perfect, I don't think the story grips
[20] "Every plot point and thematic implication slots into place, but the pleasures of Coco
```

```r
str_count(str_to_lower(user_reviews), "\\b[i]\\b")
```

```
 [1] 0 0 0 3 0 0 3 1 4 2 0 0 0 1 0 0 1 2 0 0
```

```r
mean(str_count(top_reviews, "[^\\d],[^\\d]"))
```

```
[1] 1.35
```

```r
mean(str_count(user_reviews, "[^\\d],[^\\d]"))
```

```
[1] 2.2
```