Strings: In-class Exercises (Part 2)

You can download this .qmd file from here. Just hit the Download Raw File button.

This uses parts of R4DS Ch 14: Strings and Ch 15: Regular Expressions (both the first and second editions).

Manipulating strings

str functions to know for manipulating strings:

- str_length()
- str_sub()
- str_c()
- str_to_lower()
- str_to_upper()
- str_to_title()
- str_replace() *not in video examples*

```r
library(tidyverse)

#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
  )
```

```r
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
```

Warm-up

0. Describe what EACH of the str_ functions below does. Then, create a new variable "month" which is the two digit month from album_release_date

```r
spot_new <- spot_smaller |>
  select(title, album_release_date) |>
  mutate(title_length = str_length(title),
         year = str_sub(album_release_date, 1, 4),
         month = str_sub(album_release_date, 6, 7),
         title_lower = str_to_lower(title),
         album_release_date2 = str_replace_all(album_release_date, "-", "/"))
spot_new
```

```
# A tibble: 10 x 7
   title                album_release_date title_length year  month title_lower
   <chr>                <chr>                     <int> <chr> <chr> <chr>
 1 Hear Me Now          2016-01-01                   11 2016  "01"  hear me now
 2 Run the World (Girls) 2011-06-24                  21 2011  "06"  run the wo~
 3 Formation            2016-04-23                    9 2016  "04"  formation
 4 7/11                 2014-11-24                    4 2014  "11"  7/11
 5 My Oh My (feat. DaBa~ 2019-12-06                  23 2019  "12"  my oh my (~
 6 It's Automatic       2013-11-28                   14 2013  "11"  it's autom~
 7 Poetic Justice        2012                            14 2012  ""    poetic jus~
 8 A.D.H.D               2011-07-02                       7 2011  "07"  a.d.h.d
 9 Ya Estuvo             1990-01-01                       9 1990  "01"  ya estuvo
10 Runnin (with A$AP Ro~ 2018-11-16                      49 2018  "11"  runnin (wi~
# i 1 more variable: album_release_date2 <chr>
```

```r
max_length <- max(spot_new$title_length)

str_c("The longest title is", max_length, "characters long.", sep = " ")
```

```
[1] "The longest title is 49 characters long."
```

- str_length returns a number, the number of characters in the title
- str_sub returns the just the characters that we tell it to
- str_to_lower makes all of the letters lowercase
- str_replace_all allows us to replace all of the instances of the second call by the third call.
- str_c strings together the elements that we tell it to include.

Important functions for identifying strings which match

str_view() : most useful for testing str_subset() : useful for printing matches to the console str_detect() : useful when working within a tibble

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

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

```
[1] │ indie <pop>timism
[2] │ post-teen <pop>
[3] │ hip <pop>
[4] │ hip <pop>
[5] │ latin <pop>
```

```r
typeof(str_view(spot_smaller$subgenre, "pop"))
```

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

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

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

```r
str_view(spot_smaller$subgenre, "pop", match = NA)
```

```
 [1] │ indie <pop>timism
 [2] │ post-teen <pop>
 [3] │ hip <pop>
 [4] │ hip <pop>
 [5] │ latin <pop>
 [6] │ latin hip hop
 [7] │ hip hop
 [8] │ southern hip hop
 [9] │ latin hip hop
[10] │ gangster rap
```

```r
str_view(spot_smaller$subgenre, "pop", html = TRUE)
```

```
indie poptimism
post-teen pop
hip pop
hip pop
latin pop
```

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

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

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

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

- Input is a character vector and the output is a vector of character strings showing which strings contain "pop" and where.
- Input is a character vector and the output is a vector of character strings showing which strings contain the word "pop" while also printing the strings that do not contain "pop".
- Input is a character vector and the output is an html window in r displaying a vector of strings showing which strings contain the word "pop" and where.
- Input is a character vector and the output is a vector of the character strings that contain the matches.
- Input is a character vector and the output is a 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)

```r
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 & R~
2 Run the World (Gi~ 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. 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.

```
spot_smaller |>
  mutate(sub_pop = ifelse(str_detect(subgenre, "pop"), "Genre with pop", "Genre without po
  group_by(sub_pop) |>
  summarize(mean_title_length = mean(str_length(title)))
```

```
# A tibble: 2 x 2
  sub_pop           mean_title_length
  <chr>                         <dbl>
1 Genre with pop                 13.6
2 Genre without pop              18.6
```

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/tidytuesd
```

```
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.
```

```r
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>
```

```r
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
```

Matching patterns with regular expressions

^abc string starts with abc abc$ string ends with abc . any character [abc] a or b or c [^abc] anything EXCEPT a or b or c

```r
# Guess the output!

str_view(spot_smaller$artist, "^K")
```

```
[7] | <K>endrick Lamar
[8] | <K>endrick Lamar
[9] | <K>id Frost
```

```r
str_view(spot_smaller$album_release_date, "01$")
```

```
[1] | 2016-01-<01>
[9] | 1990-01-<01>
```

```r
str_view(spot_smaller$title, "^.. ")
```

```
[5] | <My >Oh My (feat. DaBaby)
[9] | <Ya >Estuvo
```

```r
str_view(spot_smaller$artist, "[^A-Za-z ]")
```

```
 [2] | Beyonc<é>
 [3] | Beyonc<é>
 [4] | Beyonc<é>
[10] | Mike WiLL Made<->It
```

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`?)

```r
# Try using str_view() or str_subset()

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

```
[181] │ condi<tion>
[347] │ func<tion>
[516] │ men<tion>
[536] │ mo<tion>
[543] │ na<tion>
[631] │ posi<tion>
[667] │ ques<tion>
[695] │ rela<tion>
[732] │ sec<tion>
[804] │ sta<tion>
```

```r
str_subset(words, "tion")
```

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

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

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

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

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

```r
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)
```

More useful regular expressions:

\d - any number \s - any space, tab, etc \b - any boundary: space, ., etc.

```
str_view(spot_smaller$album_name, "\\d")
```

```
[2] | <4>
[8] | Section.<8><0>
```

```
str_view(spot_smaller$album_name, "\\s")
```

```
 [1] | Hear< >Me< >Now
 [4] | BEYONCÉ< >[Platinum< >Edition]
 [6] | It's< >Automatic
 [7] | good< >kid,< >m.A.A.d< >city< >(Deluxe)
 [9] | Hispanic< >Causing< >Panic
[10] | Creed< >II:< >The< >Album
```

```
str_view_all(spot_smaller$album_name, "\\b")
```

```
Warning: `str_view_all()` was deprecated in stringr 1.5.0.
i Please use `str_view()` instead.
```

```
 [1] | <>Hear<> <>Me<> <>Now<>
 [2] | <>4<>
 [3] | <>Lemonade<>
 [4] | <>BEYONCÉ<> [<>Platinum<> <>Edition<>]
 [5] | <>Romance<>
 [6] | <>It<>'<>s<> <>Automatic<>
 [7] | <>good<> <>kid<>, <>m<>.<>A<>.<>A<>.<>d<> <>city<> (<>Deluxe<>)
 [8] | <>Section<>.<>80<>
 [9] | <>Hispanic<> <>Causing<> <>Panic<>
[10] | <>Creed<> <>II<>: <>The<> <>Album<>
```

Here are the regular expression special characters that require an escape character (a preceding): ^ $. ? * | + ( ) [{

For any characters with special properties, use to "escape" its special meaning ... but is itself a special character ... so we need two \! (e.g. \$, \., etc.)

```
str_view(spot_smaller$title, "$")
```

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

```
str_view(spot_smaller$title, "\\$")
```

```
[10] | Runnin (with A<$>AP Rocky, A<$>AP Ferg & Nicki Minaj)
```

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!

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

```
# A tibble: 25 x 2
   track_name                                            track_artist
   <chr>                                                 <chr>
 1 Wing$                                                 Macklemore & Ryan Lewis
 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
11 Fre$h                                                 Lil Whigga
12 $ENHOR                                                FBC
13 $20 Fine                                              Jimi Hendrix
14 A$IAN BOY                                             Chriilz
15 ¿Cuánto E$?                                           Jhay Cortez
16 $. A. N. T. E. R. Í. A.                               Doble Porcion
17 A$IAN BOY                                             Chriilz
18 Bernice Burgo$                                        Nino Khayyam
19 $100 (feat. Polo Donatello)                           Mibbs
20 M'$                                                   A$AP Rocky
21 Dat $tick                                             Rich Brian
22 $ave Dat Money (feat. Fetty Wap & Rich Homie Quan) Lil Dicky
23 Love$ick                                              Mura Masa
24 A$IAN BOY                                             Chriilz
25 CA$H                                                  Olly James
```

- 25 track_names

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

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

```
# A tibble: 2 x 23
  track_id               track_name track_artist track_popularity track_album_id
  <chr>                  <chr>      <chr>                   <dbl> <chr>
1 7pse475uICmWRY5hEkvPvI $20 Fine   Jimi Hendrix               44 0EfHWQeb3T1UJ~
2 1ivrsBwgQe5KwjMHu8xje4 $100 (fea~ Mibbs                      20 6DjgLGR3L3LjG~
# 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>
```

- 2 songs

Repetition

? 0 or 1 times + 1 or more * 0 or more {n} exactly n times {n,} n or more times {,m} at most m times {n,m} between n and m times

```
str_view(spot_smaller$album_name, "[A-Z]{2,}")
```

```
 [4] │ <BEYONC>É [Platinum Edition]
[10] │ Creed <II>: The Album
```

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

```
 [1] │ <2016-01>-01
 [2] │ <2011-06>-24
 [3] │ <2016-04>-23
 [4] │ <2014-11>-24
 [5] │ <2019-12>-06
 [6] │ <2013-11>-28
 [8] │ <2011-07>-02
 [9] │ <1990-01>-01
[10] │ <2018-11>-16
```

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

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

```
str_view(spot_smaller$album_name, "([A-Z]\\.?){2,}")
```

```
 [4] | <BEYONC>É [Platinum Edition]
 [7] | good kid, m.<A.A.>d city (Deluxe)
[10] | Creed <II>: The Album
```

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

```
test <- c("88", "", " $ ", "")
str_view(test, "^.*$")
```

```
[1] | <88>
[2] | <>
[3] | < $ >
[4] | <>
```

```
test1 <- c("{a}", "{ab}")
str_view(test1, "\\{.+\\}")
```

```
[1] | <{a}>
[2] | <{ab}>
```

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

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, "^[^AEIOUIaeiou]{3}")
```

```
[150] | <Chr>ist
[151] | <Chr>istmas
[249] | <dry>
[328] | <fly>
[538] | <mrs>
[724] | <sch>eme
[725] | <sch>ool
```

```
[811] │ <str>aight
[812] │ <str>ategy
[813] │ <str>eet
[814] │ <str>ike
[815] │ <str>ong
[816] │ <str>ucture
[836] │ <sys>tem
[868] │ <thr>ee
[869] │ <thr>ough
[870] │ <thr>ow
[895] │ <try>
[901] │ <typ>e
[952] │ <why>
```

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

```
  [4] │ abs<olut>e
 [23] │ <agen>t
 [30] │ <alon>g
 [36] │ <americ>a
 [39] │ <anot>her
 [42] │ <apar>t
 [43] │ app<aren>t
 [61] │ auth<orit>y
 [62] │ ava<ilab>le
 [63] │ <awar>e
 [64] │ <away>
 [70] │ b<alan>ce
 [75] │ b<asis>
 [81] │ b<ecom>e
 [83] │ b<efor>e
 [84] │ b<egin>
 [85] │ b<ehin>d
 [87] │ b<enefit>
[119] │ b<usines>s
[143] │ ch<arac>ter
... and 149 more
```

Useful functions for handling patterns

str_extract() : extract a string that matches a pattern str_count() : count how many times a pattern occurs within a string

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

```
 [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, "[AEIOUaeiou]"))
```

```
# A tibble: 10 x 2
   artist            n_vowels
   <chr>                <int>
 1 Alok                     2
 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       6
```

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

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

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

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

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

```
# [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
#   <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, "\\b(\\d){2}\\'?s"))|>
  mutate(decade = str_extract(playlist_name, "\\b(\\d){2}\\'?s"),
         decade = str_replace_all(decade, "0s", "0's")) |>
  count(decade)
```

```
# A tibble: 3 x 2
  decade     n
  <chr>  <int>
1 70's     442
2 80's     682
3 90's    1013
```

Grouping and backreferences

```
# find all fruits with repeated pair of letters.
fruit = stringr::fruit
fruit
```

```
 [1] "apple"             "apricot"           "avocado"
 [4] "banana"            "bell pepper"       "bilberry"
 [7] "blackberry"        "blackcurrant"      "blood orange"
[10] "blueberry"         "boysenberry"       "breadfruit"
[13] "canary melon"      "cantaloupe"        "cherimoya"
[16] "cherry"            "chili pepper"      "clementine"
[19] "cloudberry"        "coconut"           "cranberry"
[22] "cucumber"          "currant"           "damson"
[25] "date"              "dragonfruit"       "durian"
[28] "eggplant"          "elderberry"        "feijoa"
[31] "fig"               "goji berry"        "gooseberry"
[34] "grape"             "grapefruit"        "guava"
[37] "honeydew"          "huckleberry"       "jackfruit"
[40] "jambul"            "jujube"            "kiwi fruit"
[43] "kumquat"           "lemon"             "lime"
[46] "loquat"            "lychee"            "mandarine"
[49] "mango"             "mulberry"          "nectarine"
[52] "nut"               "olive"             "orange"
[55] "pamelo"            "papaya"            "passionfruit"
[58] "peach"             "pear"              "persimmon"
[61] "physalis"          "pineapple"         "plum"
```

```
[64] "pomegranate"       "pomelo"            "purple mangosteen"
[67] "quince"            "raisin"            "rambutan"
[70] "raspberry"         "redcurrant"        "rock melon"
[73] "salal berry"       "satsuma"           "star fruit"
[76] "strawberry"        "tamarillo"         "tangerine"
[79] "ugli fruit"        "watermelon"
```

```
str_view(fruit, "(..)\\1", match = TRUE)
```

```
 [4] | b<anan>a
[20] | <coco>nut
[22] | <cucu>mber
[41] | <juju>be
[56] | <papa>ya
[73] | s<alal> berry
```

```
# why does the code below add "pepper" and even "nectarine"?
str_view(fruit, "(..)(.*)\\1", match = TRUE)
```

```
 [4] | b<anan>a
 [5] | bell <peppe>r
[17] | chili <peppe>r
[20] | <coco>nut
[22] | <cucu>mber
[29] | eld<erber>ry
[41] | <juju>be
[51] | <nectarine>
[56] | <papa>ya
[73] | s<alal> berry
```

Tips with backreference: - You must use () around the the thing you want to reference. - To backreference multiple times, use \1 again. - The number refers to which spot you are referencing… e.g. \2 references the second set of ()

```
x1 <- c("abxyba", "abccba", "xyaayx", "abxyab", "abcabc")
str_view(x1, "(.)(.)(..)\\2\\1")
```

```
[1] | <abxyba>
[2] | <abccba>
[3] | <xyaayx>
```

```r
str_view(x1, "(.)(.)(..)\\1\\2")
```

```
[4] | <abxyab>
```

```r
str_view(x1, "(.)(.)(.)\\1\\2\\3")
```

```
[5] | <abcabc>
```

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

- (.)\1\1
  - This will return words or expressions that have the same letter three times in a row.
  - Example expression: abcccb
- "(.)(.)(.).*\3\2\1"
  - This will return words of expressions that have three different letters then any number of letters in between and then the 3rd letter will be repeated once followed by the second followed by the first.
  - Example expression: abcghjgldcba

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

```r
str_view(words, "(.)\1\1")
```

- no words match the first expression

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

```
[598] | <paragrap>h
```

- "paragraph" matches.

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).

```r
str_view(words, "([a-z])([a-z]).*\\1\\2")
```

```
 [48] │ ap<propr>iate
[152] │ <church>
[181] │ c<ondition>
[217] │ <decide>
[275] │ <environmen>t
[487] │ l<ondon>
[598] │ pa<ragra>ph
[603] │ p<articular>
[617] │ <photograph>
[638] │ p<repare>
[641] │ p<ressure>
[696] │ r<emem>ber
[698] │ <repre>sent
[699] │ <require>
[739] │ <sense>
[858] │ the<refore>
[903] │ u<nderstand>
[946] │ w<hethe>r
```

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().)

```r
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:

```r
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."
```

This chunk will take sentences and rearrange them so that what was originally the second word in the sentence will now be the third and what was originally the third word will now be the second.