String Manipulation with stringr.

Evans_G02504972

2023-10-17

```
 library(dplyr)
library(tidyverse)
library(skimr)
library(stringr)
library(lubridate)
library(wordcloud2)
library(tidytext)
```

I import the tylor swift data into R

```
# Variables to keep
keeps <- c("track_name", "youtube_title", "youtube_duration", "full_lyrics")

# Importing CSV file
swiftSongs <- read_csv("https://raw.githubusercontent.com/dilernia/STA418-518/main/Data/swiftS
ongs.csv") %>% dplyr::select(keeps)
```

Explore high-level characteristics of the data using the glimpse() and skim() functions.

```
#Exploring swift data
skimr::skim(swiftSongs)
```

Data summary

Name	swiftSongs
Number of rows	151
Number of columns	4
_____	
Column type frequency:	
character	4
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
track_name	0	1	2	70	0	151	0
youtube_title	0	1	5	79	0	151	0
youtube_duration	0	1	4	7	0	92	0
full_lyrics	0	1	786	3505	0	151	0

First let’s print some songs lyrics from the Taylor

```
# Displaying lyrics
swiftSongs %>% filter(track_name == "It’s Nice To Have A Friend") %>%
  pull(full_lyrics)
```

```
## [1] "Ooh Ooh School bell rings, walk me home Sidewalk chalk covered in snow Lost my gloves, you give me one \"Wanna hang out?\" Yeah, sounds like fun Video games, you pass me a note Sleeping in tents It's nice to have a friend (Ooh) It's nice to have a friend (Ooh) Light pink sky, up on the roof Sun sinks down, no curfew 20 questions, we tell the truth You've been stressed out lately, yeah, me too Something gave you the nerve To touch my hand It's nice to have a friend (Ooh) It's nice to have a friend (Ooh) Church bells ring, carry me home Rice on the ground looks like snow Call my bluff, call you \"Babe\" Have my back, yeah, every day Feels like home, stay in bed The whole weekend It's nice to have a friend (Ooh) It's nice to have a friend (Ooh) It's nice to have a friend (Ooh) (Ooh)"
```

```
# Detecting if a string contains the substring 'Taylor'
str_detect(string = c("Taylor Swift", "Taylor Lautner", "Harry Styles"),
           pattern = "Taylor")
```

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

Using the str_detect() and mutate() functions, add a new boolean variable called contains_midnight to swiftSongs that indicates whether or not a song’s lyrics contain the word “midnight”. Using the str_detect() and mutate() functions, add a new boolean variable called contains_midnight to swiftSongs that indicates whether or not a song’s lyrics contain the word “midnight”.

```
#create anew varaible
swiftSongs <- swiftSongs %>%
  mutate(contains_midnight=str_detect(full_lyrics,
                                      pattern="midnight"))
swiftSongs %>%
  dplyr::count(contains_midnight)
```

contains_midnight <lgl>	n <int>
FALSE	145
TRUE	6

2 rows

```
swiftSongs <- swiftSongs %>%
  mutate(contains_midnight=str_detect(full_lyrics,
                                      pattern="midnight|Midnight"))
swiftSongs %>%
  dplyr::count(contains_midnight)
```

contains_midnight <lgl>	n <int>
FALSE	143
TRUE	8

2 rows

```
 # Counting instances of "man"
        str_count("I'm so sick of running as fast as I can Wondering if I'd get there quicker
if I was a man And I'm so sick of them coming at me again 'Cause if I was a man, then I'd be t
he man I'd be the man I'd be the man I'd be the man",
                  pattern = "man")
```

```
## [1] 5
```

Using the str_count() and mutate() functions, add a new variable called love_count to swiftSongs that indicates how many times each song mentions the word "love". ➡️ Which song mentions love the most times, and how many times is it mentioned?

```
#Tallying up love
         swiftSongs <- swiftSongs %>%
           mutate(love_count=str_count(str_to_lower(full_lyrics),pattern ="love"))
         swiftSongs %>% slice_max(love_count,n=1)
```

track_name <chr>	youtube_title <chr>	youtube_duration <chr>
This Love	This Love	PT4M11S

1 row | 1-3 of 6 columns

Use the str_replace_all() function to replace certain patterns with a replacement of our choice.

```
       str_replace_all("I'm so sick of running as fast as I can Wondering if I'd get there qu
icker if I was a man And I'm so sick of them coming at me again 'Cause if I was a man, then
I'd be the man I'd be the man I'd be the man",
                        pattern = "man", replacement = "!!!")
```

```
## [1] "I'm so sick of running as fast as I can Wondering if I'd get there quicker if I was a
!!! And I'm so sick of them coming at me again 'Cause if I was a !!!, then I'd be the !!! I'd
be the !!! I'd be the !!!"
```

To explore an instance where the str_subset() function is useful, let me view the youtube_duration variable, which gives the duration of Taylor's YouTube videos in a format that is not the easiest to work with.

```
    Create a new variable called youtube_time that is the same as youtube_duration, but with
a : symbol replacing the M.
```

```
swiftSongs <- swiftSongs %>%
         mutate(youtube_time = str_replace_all(youtube_duration,
                                               pattern ="M",
                                               replacement = ":"))
```


```
swiftSongs <- swiftSongs %>%
         mutate(youtube_time = str_replace_all(youtube_duration,
                                               pattern ="M",
                                               replacement = ":"))
```

```
 #cleaning up the youtube duration
        swiftSongs <- swiftSongs %>%
          mutate(youtube_time = str_replace_all(youtube_time,
                                                pattern ="PT|S",
                                                replacement = ""))
```

```
 swiftSongs <- swiftSongs %>%
  mutate(youtube_time = case_when(
    str_length(youtube_time) == 2 ~ str_c(youtube_time, "00"),
    str_length(youtube_time) == 3 ~ str_replace_all(youtube_time, pattern = ":", replacement =
":0"),
    TRUE ~ youtube_time
  ))
```

#Capitalization and spacing

```
str_to_lower("Its nice to have a friend")
```

```
## [1] "its nice to have a friend"
```

```
str_to_upper("Its nice to have a friend")
```

```
## [1] "ITS NICE TO HAVE A FRIEND"
```

```
str_to_title("Its nice to have a friend")
```

```
## [1] "Its Nice To Have A Friend"
```

```
# Removing spaces at start and end of string
str_trim(" Best believe I'm still bejeweled     When I walk in the room     I can still make t
he whole place shimmer ")
```

```
## [1] "Best believe I'm still bejeweled     When I walk in the room     I can still make the
whole place shimmer"
```

```
## [1] "Best believe I'm still bejeweled     When I walk in the room     I can still make the
whole place shimmer"
# Removing spaces at start and end of string and repetitive spaces
str_squish(" Best believe I'm still bejeweled     When I walk in the room     I can still make
the whole place shimmer ")
```

```
## [1] "Best believe I'm still bejeweled When I walk in the room I can still make the whole pl
ace shimmer"
```

```
swiftSongs <- swiftSongs %>%
  dplyr::mutate(youtube_time=lubridate::parse_date_time(youtube_time, orders="%M:%s"))
```

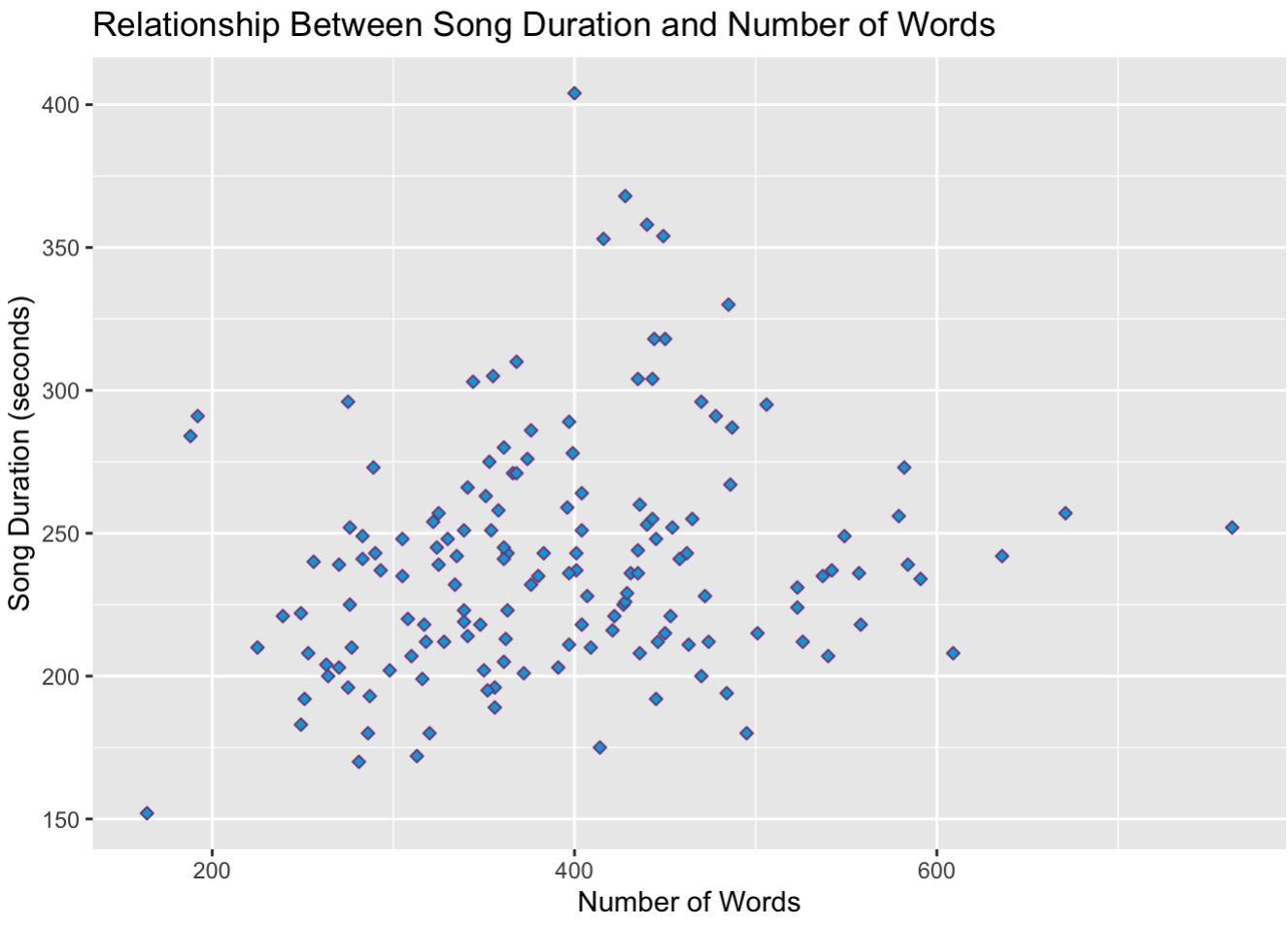
```
#Creating song_duration_s variable
swiftSongs <- swiftSongs %>%
  dplyr::mutate(song_duration_s = lubridate::second(youtube_time) +
                                  60*lubridate::minute(youtube_time))
```

```
# Creating song_words variable
swiftSongs <- swiftSongs %>%
  dplyr::mutate(song_words = str_count(full_lyrics, pattern = "\\w+"))
```

Reproduce the plot below showing the relationship between the duration of each song in seconds and its number of words. Hint: to match the style of the points, use fill = '#01a7d9', pch = 23, color = '#7d488e' inside of the geom_point() layer.

```
# Assuming you have a 'swiftSongs' data frame with 'song_duration' and 'song_words' columns

ggplot(swiftSongs, aes(x = song_words, y = song_duration_s)) +
  geom_point(
    shape = 23,
    fill = '#01a7d9',
    color = '#7d488e'
  ) +
  labs(
    title = "Relationship Between Song Duration and Number of Words",
    x = "Number of Words",
    y = "Song Duration (seconds)"
  )+
theme_update(text = element_text(face = "bold"))
```



#creating a word cloud

```r
# Tokenize the text into individual words and count their frequencies
wordFreqs <- swiftSongs %>%
  unnest_tokens(word, full_lyrics) %>%
  count(word, sort = TRUE)

# Removing 'stop words' (common but not very meaningful words)
wordFreqs <- wordFreqs %>%
  anti_join(stop_words)

# Creating the word cloud
 wordcloud2(wordFreqs)
```