Class Activity 11

Arthur Viegas Eguia

April 17 2024

Problem 1

Let's learn about combining strings with different separators first.

```
place <- "Central Park"
activity <- "jogging"
activities <- c("jogging", "picnicking", "boating")
my_sentence <- str_c(place, " is great for ", activity, ".", sep = "")
```

a. What happens when a `str_c` entry is a vector?

Answer: In this case, without the vector, it prints "Central Park is great for jogging." With the vector, it will print an array, with one item for each item in the array. So, like "Central Part is great for jogging.", "Central Part is great for picknicking", "Central Part is great for boating." THis happens because it will print all, separated items in the list concatenated with the rest of the sentence.

```
my_sentences <- str_c(place, " is great for ", activities, ".", sep = "")
my_sentences
[1] "Central Park is great for jogging."
[2] "Central Park is great for picnicking."
[3] "Central Park is great for boating."
```

b. How do you combine strings with `str_glue`?

Answer:

```
str_glue("{place} is great for {activity}.")
Central Park is great for jogging.
```

c. What does `str_flatten` do?

Answer: This will concatenate all of the sentences into a single string, they will be divided by a whitespace as it is the defined separator.

```
str_flatten(my_sentences, collapse = " ")
[1] "Central Park is great for jogging. Central Park is great for picnicking. Central Park is great for
```

d. What will using a `\n` separator do in the command below?

Answer: Will also print the `\n` in the string.

```
str_c(place, "is great for", activity, sep = "\n")
[1] "Central Park\nis great for\njogging"
```

e. Does `str_length` count spaces and special characters??

Answer: It does count `\n`, but it is counted as a single character.

```
str_length(str_c(place, "is great for", activity))
[1] 31
str_length(str_c(place, "is great for", activity, sep = "\n"))
[1] 33
```

f. How do you count the number of `e`'s in a string?

Answer: use the pattern.

```
text <- "The quick brown fox jumps over the lazy dog."
pattern <- "e"
vowel_count <- str_count(text, pattern = pattern)
```

g. What happens with negative positions?

Answer: This returns the last elements of the array. All the characters from the 7th to the first part of the string.

```
str_sub(text, start = 1, end = 7)
[1] "The qui"
str_sub(text, start = -7, end = -1)
[1] "zy dog."
```

h. How do you extract a `substring` with positive and negative positions?

Answer: Goes from the positive index, until the other, counted from the end of the string.

```
my_sentence <- "Central Park is great for jogging."
str_sub(sentences[3], start = 1, end = 7)
[1] "It's ea"
str_sub(sentences[3], start = -7, end = -1)
[1] "a well."
str_sub(sentences[3], start = 5, end = -3)
[1] " easy to tell the depth of a wel"
```

i. With a vector of positions?

Answer:

```
str_sub(my_sentence, start = c(1, 9), end = c(7, 12))
[1] "Central" "Park"
str_sub(my_sentence, start = c(1, 9, 27), end = c(7, 12, -2))
[1] "Central" "Park"    "jogging"
```

j. How do you extract multiple `substrings` using a vector of positions?

Answer:

```
str_sub(my_sentence, start = c(1, 9, 27), end = c(7, 12, -2))
[1] "Central" "Park"    "jogging"
```

Problem 2

a. Use the string parsing functions that you learned today to do tasks described in the comments below:

```
s1 <- "12%"  # remove %
s2 <- "New Jersey_*"  # remove _*
s3 <- "2,150"    # remove comma(,)
s4 <- "Learning #datascience is fun!"   # extract #datascience
s5 <- "123 Main St, Springfield, MA, 01101"   # separate info

# Cleaning steps
s1_clean <- str_remove(s1, "%"); s1_clean
[1] "12"
s2_clean <- str_remove(s2, "_\\*"); s2_clean
[1] "New Jersey"
s3_clean <- parse_number(s3); s3_clean
[1] 2150
s3_clean <- str_replace(s3, ",", ""); s3_clean
[1] "2150"
s4_clean <- str_extract(s4, "#\\w+");s4_clean
[1] "#datascience"
s5_clean <- str_split(s5, ",");s5_clean
[[1]]
[1] "123 Main St"  " Springfield" " MA"          " 01101"

# Print cleaned strings
s1_clean
[1] "12"
s2_clean
[1] "New Jersey"
s3_clean
[1] "2150"
s4_clean
[1] "#datascience"
s5_clean
[[1]]
[1] "123 Main St"  " Springfield" " MA"          " 01101"
```

Problem 3

a. Let's look at the following dataset containing information about movies and their release years. We'll extract the release year from the movie title, create a new column with decades, and count the number of movies in each decade.

```
# Sample dataset
movies <- tibble(
  title = c(
    "The Godfather (1972)", "Pulp Fiction (1994)", "The Dark Knight (2008)",
    "Forrest Gump (1994)", "The Shawshank Redemption (1994)", "The Matrix (1999)",
    "Inception (2010)", "Interstellar (2014)", "Parasite (2019)", "Fight Club (1999)"
  )
)
movies
# A tibble: 10 x 1
   title
```

```
   <chr>
 1 The Godfather (1972)
 2 Pulp Fiction (1994)
 3 The Dark Knight (2008)
 4 Forrest Gump (1994)
 5 The Shawshank Redemption (1994)
 6 The Matrix (1999)
 7 Inception (2010)
 8 Interstellar (2014)
 9 Parasite (2019)
10 Fight Club (1999)
```

```
# Processing the dataset
movies_processed <- movies %>%
  mutate(
    release_year = as.numeric(str_extract(title, pattern = "\\d{4}")) ,
    decade = floor(release_year/10) *10
  ) %>%
  count()

# Print the processed dataset
movies_processed
# A tibble: 1 x 1
      n
  <int>
1    10
```