Class Activity 11

Your name here

March 19 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 = "")
my_sentence
[1] "Central Park is great for jogging."
```

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

Answer: When an entry in `str_c` is a vector, it will combine the strings with each element of the vector, creating multiple combined strings.

```
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: You can combine strings with `str_glue` using curly braces `{}` to insert variables directly into the string.

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

my_sentences1 <- str_glue("{place} is great for {activities}.")
my_sentences1
Central Park is great for jogging.
Central Park is great for picnicking.
Central Park is great for boating.
```

c. What does `str_flatten` do?

Answer: `str_flatten` collapses a character vector into a single string by concatenating the elements with a specified separator.

```
p <- str_flatten(my_sentences, collapse = " and ")
writeLines(p)
Central Park is great for jogging. and Central Park is great for picnicking. and Central Park is great
```

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

Answer: Using a `\n` separator in the command will insert a newline character between the strings being combined, making them display on separate lines when printed.

```
p <- str_c(place, " is great for ", activity, sep = "\n")
writeLines(p, sep = "\n")
Central Park
 is great for
jogging
```

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

Answer: Yes, `str_length` counts spaces and special characters as part of the string's length.

```
p
[1] "Central Park\n is great for \njogging"
str_length(p)
[1] 35
```

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

Answer: You can count the number of `e`'s in a string using `str_count` with a pattern that matches the character 'e'.

```
text <- "The quick brown fox jumps over the lazy dog."
pattern <- "e"
vowel_count <- str_count(text, pattern)
vowel_count
[1] 3
```

g. What happens with negative positions?

Answer: Negative positions in `str_sub` count the positions from the end of the string rather than from the beginning.

```
str_sub(my_sentence, start = -3, end = -1)
[1] "ng."
```

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

Answer: You can extract a `substring` with positive and negative positions using `str_sub` and specifying the start and end positions with either positive or negative numbers.

```
my_sentence <- "Central Park is great for jogging."
positive_substr <- str_sub(my_sentence, start = 1, end = 12)
negative_substr <- str_sub(my_sentence, start = -8, end = -1)
positive_substr
[1] "Central Park"
negative_substr
[1] "jogging."
```

i. With a vector of positions?

Answer: Using a vector of positions with `str_sub` will extract `substrings` starting and ending at the specified positions in the vector.

```
str_sub(my_sentence, start = c(1, 9), end = c(4, 15))
[1] "Cent"    "Park is"
```

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

Answer: You can extract multiple `substrings` using a vector of positions with `str_sub` by specifying the start and end positions in separate vectors.

```
my_sentence <- "Central Park is great for jogging."
substrs <- str_sub(my_sentence, start = c(1, 14, 24), end = c(12, 19, 30))
substrs
[1] "Central Park" "is gre"       "or jogg"
```

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_replace(s1, "%", "")
s2_clean <- str_replace(s2, "_\\*", "")
s3_clean <- str_replace(s3, ",", "")
s4_clean <- str_extract(s4, "#\\w+")
s5_clean <- str_split(s5, ",\\s?")

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

```
s1 <- "25%"  # remove %
s2 <- "Los Angeles_#"  # remove _#
s3 <- "1,250"    # remove comma(,)
s4 <- "Discover #machinelearning today!"   # extract #machinelearning
s5 <- "456 Main St, San Francisco, CA, 94107"   # separate info
```

```
# Cleaning steps
s1_clean <- str_replace(s1, "%", "")
s2_clean <- str_replace(s2, "_#", "")
s3_clean <- str_replace(s3, ",", "")
s4_clean <- str_extract(s4, "#\\w+")
s5_clean <- str_split(s5, ",\\s?")

# Print cleaned strings
s1_clean
[1] "25"
s2_clean
[1] "Los Angeles"
s3_clean
[1] "1250"
s4_clean
[1] "#machinelearning"
s5_clean
[[1]]
[1] "456 Main St"   "San Francisco" "CA"            "94107"
```

Problem 4

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.integer(str_extract(title, "\\d{4}")),
    decade = floor(release_year / 10) * 10
  ) %>%
  count(decade) %>%
  rename(num_movies = n)

# Print the processed dataset
movies_processed
# A tibble: 4 x 2
  decade num_movies
   <dbl>      <int>
1   1970          1
2   1990          5
3   2000          1
4   2010          3
```