Class Activity 12

Your name here

March 19 2024

In this tutorial, we will learn about string manipulations using regular expressions and the `stringr` library in R. We will cover different examples and use cases to help you understand the concepts and functions related to string manipulation.

Group Activity 1

```
x <- "My SSN is 593-29-9502 and my age is 55"
y <- "My phone number is 612-643-1539"
z <- "My old SSN number is 39532 9423."
out <- str_flatten(c(x,y,z), collapse = ". ")
```

a. What characters in x will `str_view_all(x, "-..-")` find?

answer:

The pattern: "-" then "something" then "something" then "-"

```
str_view_all(x, "-..-")
[1] | My SSN is 593<-29->9502 and my age is 55
```

b. What pattern will `str_view_all(x, "-\\d{2}-")` find?

answer:

```
str_view_all(x, "-\\d{2}-")  # "-" then 2 digits then "-"
[1] | My SSN is 593<-29->9502 and my age is 55
```

c. What pattern will `str_view_all(out, "\\d{2}\\.*")` find?

answer:

```
str_view_all(out, "\\s\\d{2}\\.")  # 2 digits then "."
[1] | My SSN is 593-29-9502 and my age is< 55.> My phone number is 612-643-1539. My old SSN number is 3
```

d. Use `str_view_all` to determine the correct regex pattern to identify all SSN in `out`

answer:

We can get the SSN with the usual format (###-##-####) with a regex that has 3, 2, and 4 digits separated by a dash.

```
str_view_all(out,"([0-8]\\d{2})-(\\d{2})-(\\d{4})")
[1] | My SSN is <593-29-9502> and my age is 55. My phone number is 612-643-1539. My old SSN number is 3
```

This misses the oddly formatted SSN in the third entry. Rather than use a dash, we can specify the divider as `[-\\s]?` which allows either 0 or 1 occurrences of either a dash or space divider:

```
str_view_all(out,"([0-8]\\d{2})[-\\s]?(\\d{2})[-\\s]?(\\d{4})")
[1] | My SSN is <593-29-9502> and my age is 55. My phone number is 612-643-1539. My old SSN number is <
```

e. Write a regular expression to extract dates in the format YYYY-MM-DD from a given text.

```
date_pattern <- "\\d{4}-\\d{2}-\\d{2}"
text <- "The event will take place on 2023-07-20 and end on 2023-07-22."
str_extract_all(text, date_pattern)
[[1]]
[1] "2023-07-20" "2023-07-22"
```

answer:

f. Write a regular expression to extract all words that start with a capital letter in a given text.

answer:

```
capital_pattern <- "\\b[A-Z][a-zA-Z]*\\b"
text <- "Alice and Bob went to the Market to buy some Groceries."
str_extract_all(text, capital_pattern)
[[1]]
[1] "Alice"     "Bob"       "Market"    "Groceries"
```

Group Activity 2

a. Let's deal with a number string that is longer than 9 digits.

```
ssn <- "([0-8]\\d{2})[-\\s]?(\\d{2})[-\\s]?(\\d{4})"
test <- c("123-45-67890","1123 45 6789")
str_view_all(test, ssn)
[1] | <123-45-6789>0
[2] | 1<123 45 6789>
```

This example captures a 9-digit string as an SSN, but these strings are longer than 9 digits and may not represent an SSN. One way to deal with this is to use the negative lookbehind `?<!` and negative lookahead `?!` operators to ensure that the identified 9-digit string does not have a leading 0 or does not contain more digits.

If we "look behind" from the start of the SSN, we should not see another digit:

```
str_view_all(test, "(?<!\\d)([0-8]\\d{2})[-\\.\\s]?(\\d{2})[-\\.\\s]?(\\d{4})")
[1] | <123-45-6789>0
[2] | 1123 45 6789
```

And if we "look ahead" from the end of the SSN, we should not see another digit:

```
str_view_all(test, "(?<!\\d)([0-8]\\d{2})[-\\.\\s]?(\\d{2})[-\\.\\s]?(\\d{4})(?!\\d)")
[1] | 123-45-67890
[2] | 1123 45 6789
```

For parts b and c, consider the following string.

```
string1 <- "100 dollars 100 pesos"
```

b. Explain why the following matches the first 100 and not the second.

answer:

```
str_view(string1, "\\d+(?= dollars)")
[1] | <100> dollars 100 pesos
```

c. Explain why the following matches the second 100 and not the first.

answer:

```
str_view(string1, "\\d+(?!\\d| dollars)")
[1] | 100 dollars <100> pesos
```

For parts d and e, please take a look at `string2`.

```
string2 <- "USD100 PESO100"
```

d. Explain why the following matches the first 100 and not the second.

answer:

```
str_view(string2, "(?<=USD)\\d{3}")
[1] | USD<100> PESO100
```

e. Explain why the following matches the second 100 and not the first.

answer:

```
str_view(string2, "(?<!USD)\\d{3}")
[1] | USD100 PESO<100>
```

Group Activity 3

```
tweets<- read_csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/TrumpTweetData.csv")
```

a. What proportion of tweets (text) mention "America"?

```
tweets %>%
  summarize(prop = mean(str_detect(str_to_title(text), "America")))
# A tibble: 1 x 1
    prop
   <dbl>
1 0.0926
```

b. What proportion of these tweets include "great"?

```
tweets %>% filter(str_detect(str_to_title(text), "America")) %>%
  summarize(prop = mean(str_detect(str_to_lower(text), "great")))
# A tibble: 1 x 1
   prop
  <dbl>
1   0.4
```

c. What proportion of the tweets mention @?

```
tweets %>% mutate(ct = str_count(text, "@")) %>%
  select(text, ct) %>%
  summarize(prop = mean(ct>0))
# A tibble: 1 x 1
   prop
  <dbl>
1 0.317
```

d. Remove the tweets having mentions @.

```
Mentions <- c("@[^\\s]+")

tw_noMentions <- tweets %>% mutate(textNoMentions = str_replace_all(text, Mentions, ""))
tw_noMentions$text[38]
[1] "My daughter @IvankaTrump will be on @Greta tonight at 7pm. Enjoy! https://t.co/QySC5PLFMy"
```

e. What poportion of tweets originated from an iPhone?

```
tweets %>% group_by(source) %>% summarize(count = n()) %>%
  mutate(prop = count / sum(count)) %>%  filter(source == "iPhone")
# A tibble: 1 x 3
  source count  prop
  <chr>  <int> <dbl>
1 iPhone   628 0.415
```