String Manipulation: : CHEAT SHEET

Libraries

Library(tidyverse)
Library(stringr)

String Basics

1. Create your “Hello World” string:

```{r}
my_string <- "Hello World!"
writeLines(my_string)
```

```
Hello World!
```

2. There are some special symbols that can’t be directly coded in string, and therefore, we need to use ‘\’ to escape special symbols:

```{r}
keep_single_quote <- "keep single quote: \'\'"
writeLines(keep_single_quote)
keep_double_quote <- "keep double quote: \"\""
writeLines(keep_double_quote)
keep_back_slash <- "keep back slash: \\"
writeLines(keep_back_slash)
```

```
keep single quote: ''
keep double quote: ""
keep back quote: \
```

3. Some useful special characters: use ‘\t’ to insert tab and use ‘\n’ to start a new line

```{r}
tab <- "words\tare\tall\tseparated\tby\ttab"
writeLines(tab)
```

```
words	are	all	separated	by	tab
```

```{r}
new_line <- "start\na\nnew\nline"
writeLines(new_line)
```

```
start
a
new
line
```

Join, Collapse, and Slice Strings

1. Join multiple strings in a way similar to python string.join(iterable) by using str_c():

```{r}
str_c("str", "ing")
str_c("join", "with", "dash", "via", "sep", sep = "-")
str_c(c("join", "with", "dash", "via", "collapse"), collapse = "-")
```

```
[1] "string"
[1] "join-with-dash-via-sep"
[1] "join-with-dash-via-collapse"
```

? Difference between ‘sep = ’ and ‘collapse = ’:

‘sep = ’ creates new strings with the indicator, while ‘collapse = ’ collapse a vector of strings into a single string. The vector of strings is coded as c(“str1”, “str2”, “str3”)

```{r}
# sep creates new strings
str_c("x", c("y1", "y2"), "z", sep = "-")
# collapse gives you only one string
str_c("x", c("y1", "y2"), "z", collapse = "-")
```

```
[1] "x-y1-z" "x-y2-z"
[1] "xy1z-xy2z"
```

2. Slicing strings using str_sub() to get substrings:

```{r}
# keep the first 2 characters in string
str_sub("Hello", 1, 2)
str_sub(c("Hello", "World"), 1, 2)
# use str_sub to modify strings
s1 <- c("Apple", "Banana", "Pear")
str_sub(s1, 1, 1) <- str_to_lower(str_sub(s1, 1, 1))
s1

s2 <- "edav Community Contribution"
str_sub(s2, 1, 4) <- str_to_upper(str_sub(s2, 1, 4))
s2
```

```
[1] "He"
[1] "He" "Wo"
[1] "apple"  "banana" "pear"
[1] "EDAV Community Contribution"
```

Pattern Matching

1. Use str_view( ) to highlight matches, and use str_subset( ) to return strings in the collection that have the match:

```
x <- c("Apple", "Banana", "Pear")
# highlight the matches
(str_view(x, ".a."))
```

```
Apple
Banana
Pear
```

```
# return strings in your collection that have the match
(str_subset(x, ".a."))
```

```
## [1] "Banana" "Pear"
```

2. Pattern match:

- Use ‘.’ as place holder to match any character
- Use ‘^’ to match substrings start with the character following ‘^’
- Use ‘$’ to match substrings end with the character before ‘$’
- Use ‘[ ]’ to match characters inside the bracket
- Use ‘[^]’ to match anything except for characters in the bracket

```
x <- c("Apple", "Banana", "Pear", "Pineapple")
# match only "apple"
str_view(x, "^Apple$")
```

```
Apple
Banana
Pear
Pineapple
```

```
# match substrings of length 5 that start without vowels
# or substrings start with any four characters and end with a vowel
str_view(x, "^[^aeiou]....|....[aeiou]$")
```

```
Apple
Banana
Pear
Pineapple
```

3. Grouping and backreferences: Use ( ) to group expressions and use ‘.’ to match any single character; the expression ‘\ number’ refers to the ordinal position of the capturing group

```
# capture two groups, and each consists of only one character
# repeat the second group and then the first group
str_view(fruit, "(.)(.)\\2\\1",match = TRUE)
```

```
bell pepper
chili pepper
```

```
# the same setting as above, except that we allow one additional
# character or nothing follow the second captured group
str_view(fruit, "(.)(.).?\\2\\1",match = TRUE)
```

```
banana
bell pepper
chili pepper
```

```
# the same setting as above, except that we allow any number of
# characters follow the second captured group
str_view(fruit, "(.)(.).*\\2\\1", match = TRUE)
```

```
banana
bell pepper
chili pepper
clementine
```

```
# repeat the captured group 3 times and separate
# each repetition with a character
str_view(fruit, "(.).\\1.\\1",match = TRUE)
```

```
banana
papaya
```

Reference

https://r4ds.had.co.nz/strings.html