Four stringr functions to manipulate text

RStudio Certified Instructor Teaching Exam

Grace Lawley

2019-11-11

Learner Persona

Sam is Masters student studying chemistry and has been learning R in her free time. She is interested in computational linguistics and just finished taking a Natural Language Processing course online where she learned about regular expressions. She knows that most real world natural language data is very messy. She wants to learn how she can get started cleaning and manipulating text in R.

stringr

- A package for working with strings in R
- Contains functions for
    - Subsetting strings
    - Mutating strings
    - ....

We will only be looking at 4 of the stringr functions today (there are *many* more!)

Subsetting strings

`str_subset()`

`str_extract()`

Subsetting strings

`str_subset()`

`str_extract()`

Mutating strings

`str_sub()`

`str_replace()`

Subsetting strings

`str_subset()`

`str_extract()`

Mutating strings

`str_sub()`

`str_replace()`

Before we go further

```
# install.packages(“stringr”)
library(stringr)
```

Before we go further

```
# install.packages("stringr")
library(stringr)
```

```
fruit
```

```
##  [1] "apple"            "apricot"          "avocado"
##  [4] "banana"           "bell pepper"      "bilberry"
##  [7] "blackberry"       "blackcurrant"     "blood orange"
## [10] "blueberry"        "boysenberry"      "breadfruit"
## [13] "canary melon"     "cantaloupe"       "cherimoya"
## [16] "cherry"           "chili pepper"     "clementine"
## [19] "cloudberry"       "coconut"          "cranberry"
## [22] "cucumber"         "currant"          "damson"
## [25] "date"             "dragonfruit"      "durian"
## [28] "eggplant"         "elderberry"       "feijoa"
## [31] "fig"              "goji berry"       "gooseberry"
## [34] "grape"            "grapefruit"       "guava"
## [37] "honeydew"         "huckleberry"      "jackfruit"
## [40] "jambul"           "jujube"           "kiwi fruit"
## [43] "kumquat"          "lemon"            "lime"
## [46] "loquat"           "lychee"           "mandarine"
## [49] "mango"            "mulberry"         "nectarine"
## [52] "nut"              "olive"            "orange"
## [55] "pamelo"           "papaya"           "passionfruit"
## [58] "peach"            "pear"             "persimmon"
## [61] "physalis"         "pineapple"        "plum"
## [64] "pomegranate"      "pomelo"           "purple mangosteen"
## [67] "quince"           "raisin"           "rambutan"
## [70] "raspberry"        "redcurrant"       "rock melon"
## [73] "salal berry"      "satsuma"          "star fruit"
## [76] "strawberry"       "tamarillo"        "tangerine"
## [79] "ugli fruit"       "watermelon"
```

`str_subset(string, pattern)`

- `string` = a character vector
- `pattern` = pattern to look for (regular expression)

- Returns only the strings that contain a **`pattern`** match (a *subset* of the original set)

`str_subset()` in action

```
str_subset(fruit, “ine”)
```

```
## [1] "clementine" "mandarine"  "nectarine"  "pineapple"  "tangerine"
```

`str_subset()` in action

```
str_subset(fruit, “ine”)
```

```
## [1] "clementine" "mandarine"  "nectarine"  "pineapple"  "tangerine"
```

```
str_subset(fruit, “fruit”)
```

```
## [1] "breadfruit"   "dragonfruit"  "grapefruit"   "jackfruit"
## [5] "kiwi fruit"   "passionfruit" "star fruit"   "ugli fruit"
```

`str_subset()` in action

```
str_subset(fruit, “ine”)
```

```
## [1] "clementine" "mandarine"  "nectarine"  "pineapple"  "tangerine"
```

```
str_subset(fruit, “fruit”)
```

```
## [1] "breadfruit"   "dragonfruit"  "grapefruit"   "jackfruit"
## [5] "kiwi fruit"   "passionfruit" "star fruit"   "ugli fruit"
```

```
str_subset(fruit, “melon”)
```

```
## [1] "canary melon" "rock melon"   "watermelon"
```

`str_extract(string, pattern)`

- `string` = a character vector
- `pattern` = pattern to look for (regular expression)

- Extracts the first pattern match found in each string, returns as a vector
  - Can use `str_extract_all()` to extract *all* matches
- If a string doesn't contain a match it is changed to `NA`

str_extract() in action

```
str_extract(fruit, “melon”)
```

```
##  [1] NA      NA      NA      NA      NA      NA      NA      NA
##  [9] NA      NA      NA      NA      "melon" NA      NA      NA
## [17] NA      NA      NA      NA      NA      NA      NA      NA
## [25] NA      NA      NA      NA      NA      NA      NA      NA
## [33] NA      NA      NA      NA      NA      NA      NA      NA
## [41] NA      NA      NA      NA      NA      NA      NA      NA
## [49] NA      NA      NA      NA      NA      NA      NA      NA
## [57] NA      NA      NA      NA      NA      NA      NA      NA
## [65] NA      NA      NA      NA      NA      NA      NA      "melon"
## [73] NA      NA      NA      NA      NA      NA      NA      "melon"
```

Exercise 1

What would the following code return?

```
veggies ← c(“asparagus”, “cabbage”, “onion”, “kale”
```

1.
```
str_subset(veggies, “a”)
```

2.
```
str_extract(veggies, “a”)
```

Exercise 1

What would the following code return?

```
veggies ← c(“asparagus”, “cabbage”, “onion”, “kale”
```

1.
```
str_subset(veggies, “a”)
```

```
## [1] "asparagus" "cabbage"   "kale"
```

2.
```
str_extract(veggies, “a”)
```

```
## [1] "a" "a" NA  "a"
```

Subsetting strings

`str_subset()`

`str_extract()`

Mutating strings

`str_sub()`

`str_replace()`

`str_sub(string, start, end)`

- `string` = a character vector
- `start` = start index
- `end` = end index

- Returns the substring between the **`start`** and **`end`** indices

str_sub() in action

```
str_sub(“watermelon”, start=1, end=5)
```

```
## [1] "water"
```

`str_sub()` in action

```
str_sub(“watermelon”, start=1, end=5)
```

```
## [1] "water"
```

```
str_sub(fruit, start=1, end=3)
```

```
##  [1] "app" "apr" "avo" "ban" "bel" "bil" "bla" "bla" "blo" "blu" "boy"
## [12] "bre" "can" "can" "che" "che" "chi" "cle" "clo" "coc" "cra" "cuc"
## [23] "cur" "dam" "dat" "dra" "dur" "egg" "eld" "fei" "fig" "goj" "goo"
## [34] "gra" "gra" "gua" "hon" "huc" "jac" "jam" "juj" "kiw" "kum" "lem"
## [45] "lim" "loq" "lyc" "man" "man" "mul" "nec" "nut" "oli" "ora" "pam"
## [56] "pap" "pas" "pea" "pea" "per" "phy" "pin" "plu" "pom" "pom" "pur"
## [67] "qui" "rai" "ram" "ras" "red" "roc" "sal" "sat" "sta" "str" "tam"
## [78] "tan" "ugl" "wat"
```

```
str_replace(string, pattern,
            replacement)
```

- `string` = a character vector
- `pattern` = pattern to look for (regular expression)
- `replacement` = string to replace with

- Replaces the first matched pattern in each string
  - Can use `str_replace_all()` to replace *all* matches found

str_replace() in action

```
str_replace(fruit, pattern = “[aeiou]”, replacement = “_”)
```

```
##  [1] "_pple"        "_pricot"       "_vocado"
##  [4] "b_nana"       "b_ll pepper"   "b_lberry"
##  [7] "bl_ckberry"   "bl_ckcurrant"  "bl_od orange"
## [10] "bl_eberry"    "b_ysenberry"   "br_adfruit"
## [13] "c_nary melon" "c_ntaloupe"    "ch_rimoya"
##  ...
```

str_replace() in action

```
str_replace(fruit, pattern = “[aeiou]”, replacement = “_”)
```

```
##  [1] "_pple"        "_pricot"       "_vocado"
##  [4] "b_nana"       "b_ll pepper"   "b_lberry"
##  [7] "bl_ckberry"   "bl_ckcurrant"  "bl_od orange"
## [10] "bl_eberry"    "b_ysenberry"   "br_adfruit"
## [13] "c_nary melon" "c_ntaloupe"    "ch_rimoya"
##  ...
```

```
str_replace_all(fruit, pattern = “[aeiou]”, replacement = “_”)
```

```
##  [1] "_ppl_"        "_pr_c_t"       "_v_c_d_"
##  [4] "b_n_n_"       "b_ll p_pp_r"   "b_lb_rry"
##  [7] "bl_ckb_rry"   "bl_ckc_rr_nt"  "bl__d _r_ng_"
## [10] "bl__b_rry"    "b_ys_nb_rry"   "br__dfr__t"
## [13] "c_n_ry m_l_n" "c_nt_l__p_"    "ch_r_m_y_"
##  ...
```

Exercise 2

What would the following code return?

```
veggies ← c(“asparagus”, “cabbage”, “onion”, “kale”
```

1.
```
str_sub(veggies, 1, 1)
```

2.
```
str_replace(veggies, “[aeiou]”, “_”)
```

Exercise 2

What would the following code return?

```
veggies ← c(“asparagus”, “cabbage”, “onion”, “kale”
```

1.
```
str_sub(veggies, 1, 1)
```

```
## [1] "a" "c" "o" "k"
```

2.
```
str_replace(veggies, “[aeiou]”, “_”)
```

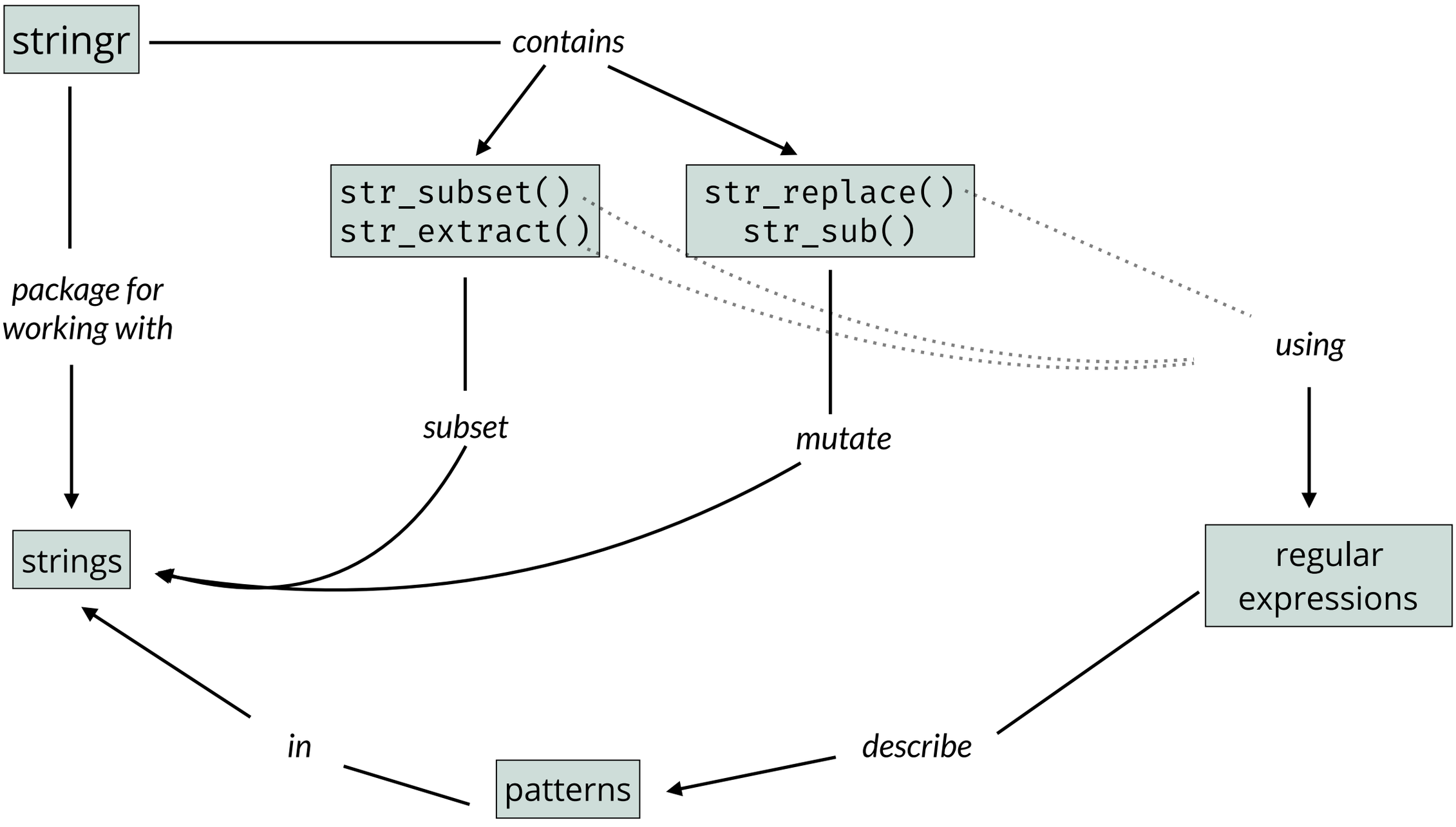
```
## [1] "_sparagus" "c_bbage"   "_nion"     "k_le"
```

Thanks!

Resources

- The Strings Chapter in R for Data Science
- “Introduction to stringr” vignette
- The stringr documentation
- Official stringr cheatsheet

Concept Map


stringr
contains
str_subset()
str_extract()
str_replace()
str_sub()
package for working with
subset
mutate
using
strings
regular expressions
in
describe
patterns