Assignment 3: Strings and Regexes

By Tony Fraser, CUNY MSDS Student

2023-09-17

Assignment:

Use Quarto and markdown to asnwer the following questions.

Question 1:

Using the 173 majors listed in fivethirtyeight.com's College Majors dataset used for this study, and provide code that identifies the majors that contain either "DATA" or "STATISTICS"

```
library(tidyverse)
library(gt)
fivethirtyeight <- "https://raw.githubusercontent.com/fivethirtyeight/data/master/"
m_list <- "college-majors/majors-list.csv"
filter_for <- ".*data.*|*.statistics*."
majors <- read_csv(paste(fivethirtyeight, m_list, sep="")) |>
    filter(grepl(filter_for, Major, ignore.case = TRUE) |
           grepl(filter_for, Major_Category, ignore.case = TRUE)) |>
   rename(ID = FOD1P)
majors |>
    gt() %>%
    tab_options(table.font.size = px(10)) %>%
    tab_style(style = cell_text(size = 'smaller'), locations = cells_body())
```

ID	Major	Major_Category
6212	MANAGEMENT INFORMATION SYSTEMS AND STATISTICS	Business
2101	COMPUTER PROGRAMMING AND DATA PROCESSING	Computers & Mathematics

Question 2

Write code that transforms input data like *[1] "bell pepper" "bilberry" "blackberry" "blood orange" [5] "blueberry" "cantaloupe" ...* and writes output data like *c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe", "chili ...* Or basically, read a string input, puts it into a vector, then prints it out so it looks like a vector.

```
library(magrittr)
library(stringr)

to_vector <- function(input_string) {
  cleaned_vector <- input_string %>%
    gsub("\\[\\d+\\]|\\\"", "", .) %>% # remove []"
    gsub(" {2,}|\n", ",", .) %>%       # replace 2+ \s with ,
    str_split(",") %>%                 # split str to list w/ vetor
    `[[`(1) %>%                        # same as .[[1]]%>%, extract first element
    str_trim()                         # trim whitespace off elements
  return(cleaned_vector)
}

to_string <- function(v) {
  var_name <- deparse(substitute(v))
  v %>%
   lapply(function(x) paste0("\"", x, "\"")) %>%   # surround elements with quotes
   unlist() %>%                                    # convert list to a character vector
   paste(collapse = ", ") %>%                      # collapse to string
   paste(var_name, " <- c(", ., ")", sep = "") %>% # finish return string
  return(v)
}

input_string <- paste(
    '[1] "bell pepper"  "bilberry"     "blackberry"   "blood orange"',
    '[5] "blueberry"    "cantaloupe"   "chili pepper" "cloudberry"',
    '[9] "elderberry"   "lime"         "lychee"       "mulberry"',
    '[13] "olive"        "salal berry"',
    sep = "\n"
)

jack <- to_vector(input_string)
cat(to_string(jack)) # use cat instead of print, doesn't print out "'s.\newpage
```

```
jack <- c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe"
```

Question 3

Describe what these regexes match

1. (.)\1\1

This matches one character three times in a row, like ppQQ:

```
> regex <- "(.)\\1\\1"
> matches <- str_match_all("tony aaabb", regex)
> matches[[1]][ ,1]
[1] "aaa"
```

2. (.)(.)\2\1

Matches two characters and their reverse, like abba!

```
regex <-"(.)(.)\\2\\1"
matches <- str_match_all("dancing queen abba")
> matches[[1]][ ,1]
[1] "abba"
```

3. (..)\1

Matches two characters right after each other, like lulu.

```
regex <-"(..)\\1"
matches <- str_match_all("eat me some hohos", regex)
> matches[[1]][ ,1]
[1] "hoho"
```

4. (.).\1.\1

Think R?R?R, or a triple decker sandwich of characters with anything in between. The first (.) picks what those 3 matching characters will be, and the periods in there mean “anything.”

```
> matches <- str_match_all("rxrxrx rarbrc r?r*r", regex)
> matches
[[1]]
     [,1]    [,2]
```

```
[1,] "rxrxr" "r"
[2,] "rarbr" "r"
[3,] "r?r*r" "r"
```

5. (.)(.)(.).*\3\2\1

Think of this one like a sandwich also, three characters as the bread, and whatever you want in between. Except the ending bread slice needs to be backwards.

```
> regex<- "(.)(.)(.).*\\3\\2\\1"
> matches <- str_match_all("abc what's this cba  def tony fed", regex)
> matches
[[1]]
     [,1]                 [,2] [,3] [,4]
[1,] "abc what's this cba" "a"  "b"  "c"
[2,] "def tony fed"        "d"  "e"  "f"
```

Question 4, build regexes that match:

4.1 Words that start and end with the same character

\b means "word boundary." So in the word "word", the boundaries are w and b. "./*" means anything, and \1\b means it looks at the other word boundary and sees if it matches the first group.

```
> regex <- "(\\b.).*\\1\\b"
> matches <- str_extract_all("bob tot tonyfraserwashere", regex)
> matches
[[1]]
[1] "bob"   " tot "
```

Note, snippit 4.1 searches for all words within a string, not just a string exact match like `"^(.).*\1$"`

4.2 Words contain a repeated pairs of letters

- like church contains ch repeated twice
- `"\\b(\\w*(\\w{2,})\\w*\\2\\w*)\\b"` is the regex we are going to use.
- \w means "word", which is shorthand for this. [a-zA-Z0-9_].

- The outer _'s are for word boundaries, meaning we are looking for groups of characters surrounded by spaces, characters, newlines, etc.

- The first capture group is here, `(\\w*(\\w{2,})\\w*\\2\\w*)` it's the entire word.
- The second capture group is `\\w{2,})`, and this is what \2 looks to see if there is a repeat of. If \2 exists, the first capture group matches.

```
> regex <- "\\b(\\w*(\\w{2,})\\w*\\2\\w*)\\b"
matches <- str_extract_all("church goer and chacha dancers", regex)
> matches
[[1]]
[1] "church" "chacha"
```

4.3 Words that contain one letter repeated three times

- like eleven with 3 e's.
- `"\\b\\w*(\\w)\\w*\\1\\w*\\1\\w*\\b"` is the regex we are going to use, and is very similar to the last example.
- \b matches words
- the first * mean pretty much any character
- the (\w) is the important capture group, and the next two \1's are repeats of that capture
- the \w*'s between the \1's mean any word character (or no character)

```
> regex <- "\\b\\w*(\\w)\\w*\\1\\w*\\1\\w*\\b"
> matches <- str_extract_all("eleven twelve altavista speaker", regex)
> matches
[[1]]
[1] "eleven"    "altavista"
```