Assignment 3 - Data 607

Coco Donovan

2023-02-07

1. Identifying Majors with "DATA" or "STATISTICS"

```
library(readr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v dplyr   1.0.10
## v tibble  3.1.8      v stringr 1.5.0
## v tidyr   1.3.0      v forcats 0.5.2
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
majors_url = "https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/majors-list.
majors <- read_csv(majors_url)
```

```
## Rows: 174 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): FOD1P, Major, Major_Category
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
str_subset(majors$Major, pattern = "(DATA|STATISTICS)")
```

```
## [1] "MANAGEMENT INFORMATION SYSTEMS AND STATISTICS"
## [2] "COMPUTER PROGRAMMING AND DATA PROCESSING"
## [3] "STATISTICS AND DECISION SCIENCE"
```

2. Converting Data into a vector

```
library(stringr)

fruits <- '[1] "bell pepper"  "bilberry"     "blackberry"   "blood orange"
```

```
[5] "blueberry"    "cantaloupe"   "chili pepper" "cloudberry"
[9] "elderberry"   "lime"         "lychee"       "mulberry"
[13] "olive"        "salal berry"'

fruits_vector <- str_extract(fruits, "\".*\"")
```

3. Describe, in words, what these expressions will match:

1. Any character repeated three times (except a newline character).
2. Any two non-newline characters followed by the same two non-newline characters in reverse order.
3. Any two non-newline characters repeated in the same order (two characters followed by the same characters in the same order).
4. Any non-newline character followed by any non-newline character, followed by the original character, followed by any non-newline character and finally followed by the original character.
5. A pattern starting with any three non-newline characters, followed by any amount (including zero) of non-newline characters. This pattern is then ended by the three characters that started the pattern, but in reverse order.

4. Construct regular expressions to match words that:

Note: I am operating with the rule that words can only contain letters

```
words <-c('ono', 'jelly', 'pop', 'lol', 'theeth', 'jhoojh', 'qafqa', "eee",
          "elelel", "olpojo", "jokollo", 'chchlop')

# 4.1: Start and end with the same character
same_first_last <- str_subset(words,"^([a-zA-Z])[a-zA-Z]*\\1$")
same_first_last
```

```
## [1] "ono"    "pop"    "lol"    "eee"    "olpojo"
```

```
# 4.2: Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.)
repeated_pair <- str_subset(words, "([a-zA-Z][a-zA-Z])[a-zA-Z]*\\1")
repeated_pair
```

```
## [1] "theeth"  "jhoojh"  "qafqa"   "elelel"  "chchlop"
```

```
# 4.3: Contain one letter repeated in at least three places (e.g. "eleven" contains three "e"s.)
same_letter_3x <- str_subset(words, '([a-zA-Z])[a-zA-Z]*\\1[a-zA-Z]*\\1')
same_letter_3x
```

```
## [1] "eee"     "elelel"  "olpojo"  "jokollo"
```