assignment_3

Matthew Tillmawitz

2024-09-13

Instructions and the relevant questions can be found in the Instructions.txt file in the parent folder of this project.

Question 1

There are only three majors in the 538 dataset of majors found in the majors-list.csv file that contain "DATA" or "STATISTICS".

```
college_majors <- read_csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors
    filter(grepl("DATA|STATISTICS", Major))
college_majors
```

```
## # A tibble: 3 x 3
##   FOD1P Major                                        Major_Category
##   <chr> <chr>                                        <chr>
## 1 6212  MANAGEMENT INFORMATION SYSTEMS AND STATISTICS Business
## 2 2101  COMPUTER PROGRAMMING AND DATA PROCESSING      Computers & Mathematics
## 3 3702  STATISTICS AND DECISION SCIENCE              Computers & Mathematics
```

Question 2

I found this question a little unclear, as the solution appears to be part of the question. Regardless, the code below creates a vector of the requested strings in the order requested.

```
fruits <- c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry",
    "cantaloupe", "chili pepper", "cloudberry", "elderberry", "lime", "lychee",
    "mulberry", "olive", "salal berry")
print(fruits)
```

```
##  [1] "bell pepper"  "bilberry"     "blackberry"   "blood orange" "blueberry"
##  [6] "cantaloupe"   "chili pepper" "cloudberry"   "elderberry"   "lime"
## [11] "lychee"       "mulberry"     "olive"        "salal berry"
```

Question 3

3.1

The first regex `(.)\1\1` will match any sequence of three matching characters since `.` matches any character and `\1` matches the same text as was most recently matched by the first capture group, in this case `(.)`. An

example is provided for clarity, we can see “aaa” matches, the substrings “bbb” and “ccc” in “bbbccc” match, non-alphanumeric characters “!!!” match and the repeating pattern of non-matching characters “abcabc” generates no matches.

```
strings <- c("aaa", "bbbccc", "!!!", "abcabc")
str_view(strings, "(.)\\1\\1")
```

```
## [1] | <aaa>
## [2] | <bbb><ccc>
## [3] | <!!!>
```

3.2

The second regex `"(.)(.)\\2\\1"` matches any four character palindrome. The extra slash in `\\2` escapes the required slash since the regex is presented as a plain string. `\2` matches the most recent match in the second capture group while `\1` matches the most recent match in the first capture group, and since both the first and second capture groups match any character we end up with four character palindromes.

```
strings <- c("aaa", "bbbccc", "!!!", "xxxx", "xyyx", "xyxy", "1!!1")
str_view(strings, "(.)(.)\\2\\1")
```

```
## [4] | <xxxx>
## [5] | <xyyx>
## [7] | <1!!1>
```

3.3

The third regex `(..)\1` matches any four character string where the first and second characters match the third and fourth characters respectively. The capture group `(..)` matches any two characters and `\1` matches any match of the first capture group.

```
strings <- c("aaa", "bbbccc", "!!!", "xxxx", "xyyx", "xyxy", "1!!1")
str_view(strings, "(..)\\1")
```

```
## [4] | <xxxx>
## [6] | <xyxy>
```

3.4

The fourth regex `"(.).\\1.\\1"` matches any five character string where the first, third, and fifth characters match. The second and third characters can be any non-newline character.

```
strings <- c("aaa", "bbbccc", "!!!", "xxxx", "xyyx", "xyxy", "1!!1", "abaca",
    "!2!r!")
str_view(strings, "(.).\\1.\\1")
```

```
## [8] | <abaca>
## [9] | <!2!r!>
```

3.5

The fifth regex "(.)(.)(.).*\3\2\1" matches any string were the last three characters mirror the first three characters. The string can be any length six characters or greater that matches this pattern.

```
strings <- c("aaa", "bbbccc", "!!!", "xxxx", "xyyx", "xyxy", "1!!1", "abaca",
    "!2!r!", "abccba", "xyz!231abzyx")
str_view(strings, "(.)(.)(.).*\\3\\2\\1")
```

```
## [10] | <abccba>
## [11] | <xyz!231abzyx>
```

Question 4

Due to the ambiguity of the term "words" in the instructions, two solutions are provided for each question. The first assumes "word" to mean "any substring of the input string". The second solution assumes a "word" is "a substring of the input string containing only consecutive english letters". Hopefully both sets of solutions will sufficiently demonstrate a grasp of the topic.

4.1

The regex `(.).*\1` will match any string that starts and ends with the same character. This can include whitespace and non-alphabet characters. If we follow the stricter definition of words we can use `([a-zA-Z])\w*\1` instead.

```
strings <- c("aaa", "bbbccc", "!!!", "xxxx", "xyyx", "xyxy", "1!!1", "abaca",
    "!2!r!", "abccba", "xyz!231abzyx", "my name is tom", "mom says hi")
str_view(strings, "(.).*\\1")
```

```
##  [1] | <aaa>
##  [2] | <bbb><ccc>
##  [3] | <!!!>
##  [4] | <xxxx>
##  [5] | <xyyx>
##  [6] | <xyx>y
##  [7] | <1!!1>
##  [8] | <abaca>
##  [9] | <!2!r!>
## [10] | <abccba>
## [11] | <xyz!231abzyx>
## [12] | <my name is tom>
## [13] | <mom>< says >hi
```

```
str_view(strings, "([a-zA-Z])\\w*\\1")
```

```
##  [1] | <aaa>
##  [2] | <bbb><ccc>
##  [4] | <xxxx>
##  [5] | <xyyx>
##  [6] | <xyx>y
##  [8] | <abaca>
## [10] | <abccba>
## [13] | <mom> <says> hi
```

4.2

The regex `([a-zA-Z][a-zA-Z]).*\1` will match any string that contains a repeated pair of letters Alternatively, for the more strict definition we can use `([a-zA-Z][a-zA-Z])\w*\1`.

```
strings <- c("aaa", "bbbccc", "!!!!", "xxxx", "xyyx", "xyxy", "1!!1", "abaca",
    "!2!r!", "abccba", "xyz!231abzyx", "my name is tom", "church", "he went to church",
    "cheese chess")
str_view(strings, "([a-zA-Z][a-zA-Z]).*\\1")
```

```
##  [4] | <xxxx>
##  [6] | <xyxy>
## [13] | <church>
## [14] | he went to <church>
## [15] | <cheese ch>ess
```

```
str_view(strings, "([a-zA-Z][a-zA-Z])\\w*\\1")
```

```
##  [4] | <xxxx>
##  [6] | <xyxy>
## [13] | <church>
## [14] | he went to <church>
```

4.3

The regex `([a-zA-Z]).*\1.*\1` will match any string with an english letter that occurs at least three times. Once again, if we want to restrict it to the stricter word definition we can instead use `([a-zA-Z])\w*\1\w*\1`.

```
strings <- c("aaa", "bbbccc", "!!!", "xxxx", "xyyx", "xyxy", "1!!1", "abaca",
    "!2!r!", "abccba", "xyz!231abzyx", "my name is tom", "church", "she shears sheep")
str_view(strings, "([a-zA-Z]).*\\1.*\\1")
```

```
##  [1] | <aaa>
##  [2] | <bbb><ccc>
##  [4] | <xxxx>
##  [8] | <abaca>
## [12] | <my name is tom>
## [14] | <she shears s>heep
```

```
str_view(strings, "([a-zA-Z])\\w*\\1\\w*\\1")
```

```
## [1] | <aaa>
## [2] | <bbb><ccc>
## [4] | <xxxx>
## [8] | <abaca>
```