Assignment 3

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

#1. Using the 173 majors listed in fivethirtyeight.com's College Majors dataset [https://fivethirtyeight.com/features/the-economic-guide-to-picking-a-college-major/], provide code that identifies the majors that contain either "DATA" or "STATISTICS"

```
library(readr)
library(RCurl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
majorsList <- read.csv(url('https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors
str(majorsList)
```

```
## 'data.frame':    174 obs. of  3 variables:
##  $ FOD1P         : chr  "1100" "1101" "1102" "1103" ...
##  $ Major         : chr  "GENERAL AGRICULTURE" "AGRICULTURE PRODUCTION AND MANAGEMENT" "AGRICULTURAL E
##  $ Major_Category: chr  "Agriculture & Natural Resources" "Agriculture & Natural Resources" "Agricult
```

```
datamajor <- majorsList$Major[grepl("DATA", majorsList$Major)]; datamajor
```

```
## [1] "COMPUTER PROGRAMMING AND DATA PROCESSING"
```

```
statsmajor <- majorsList$Major[grepl("STATISTICS", majorsList$Major)]; statsmajor
```

```
## [1] "MANAGEMENT INFORMATION SYSTEMS AND STATISTICS"
## [2] "STATISTICS AND DECISION SCIENCE"
```

#2 Write code that transforms the data below:

[1] “bell pepper” “bilberry” “blackberry” “blood orange”

[5] “blueberry” “cantaloupe” “chili pepper” “cloudberry”

[9] “elderberry” “lime” “lychee” “mulberry”

[13] “olive” “salal berry”

Into a format like this:

c(“bell pepper”, “bilberry”, “blackberry”, “blood orange”, “blueberry”, “cantaloupe”, “chili pepper”, “cloudberry”, “elderberry”, “lime”, “lychee”, “mulberry”, “olive”, “salal berry”)

```
## -- Attaching packages -------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v forcats 0.5.0
## v tidyr   1.1.1

## -- Conflicts ------------------------------------------------------------------------------
## x tidyr::complete() masks RCurl::complete()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()

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

## [1] "c(bell pepper, bilberry, blackberry, blood orange, blueberry, cantaloupe, chili pepper, cloudbe
```

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

(.)\1\1 - The dot matches a single character, except newline. The backreference \1 references the first capturing group and matches the exact same text that was matched by the first capturing group. Here the backreference is repeated twice, in essence the single character is repeated three times.

```r
library(tidyverse)
library(stringr)

a <- c("aaa", "bbbb", "adadad", "ghfdaert")
part31 <- str_extract(a, "(.)\\1\\1"); part31
```

```
## [1] "aaa" "bbb" NA    NA
```

“(.)(.)\2\1” - This signified a pair of characters followed by the same pair in reverse order.

```r
library(tidyverse)
library(stringr)

b <- c("abba", "baooab", "garrison", "aabbcdeffgg")
part32 <- str_extract(b, "(.)(.)\\2\\1"); part32
```

```
## [1] "abba" "aooa" NA     NA
```

(..)\1 - This finds a repeated pair of characters.

```r
library(tidyverse)
library(stringr)

c <- c("kitchen", "banana", "ccfcfcff", "aabbcdeffgg")
part33 <- str_extract(c, "(..)\\1"); part33
```

```
## [1] NA     "anan" "cfcf" NA
```

"(.).\1.\1" - A character followed by another character (non-matching), followed by the original character (\1) and any other character as signified by the period and finally ending with the original character.

```r
library(tidyverse)
library(stringr)

d <- c("kitchen", "banana", "malayalam", "aabbcdeffgg")
part34 <- str_extract(d, "(.).\\1.\\1"); part34
```

```
## [1] NA      "anana" "alaya" NA
```

"(.)(.)(.).*\3\2\1" - The* signifies 0 or more times, so this expression signifies any 3 characters followed by none or more characters of any kind with the original three characters repeated in the reverse order.

```r
library(tidyverse)
library(stringr)

e <- c("kitchen", "abcdcba", "malayalam", "bcddcb")
part35 <- str_extract(e, "(.)(.)(.).*\\3\\2\\1"); part35
```

```
## [1] NA          "abcdcba"   "malayalam" "bcddcb"
```

#4 Construct regular expressions to match words that:

Start and end with the same character. Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.) Contain one letter repeated in at least three places (e.g. "eleven" contains three "e"s.)

```r
#Part I: Start and end with the same character.

library(tidyverse)
library(stringr)

x <- c("america", "russia", "armenia", "canada")
part1 <- str_extract(x, '^(.).*\\1$'); part1
```

```
## [1] "america" NA        "armenia" NA
```

```r
#Part II: Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.)

y <- c("aabbccaa", "abcdefg", "aaa", "abac", "bbbbbb")
part2 <- str_extract(y, '(..).*\\1'); part2
```

```
## [1] "aabbccaa" NA         NA         NA         "bbbbbb"
```

```r
#Part III: Contain one letter repeated in at least three places (e.g. "eleven" contains three "e"s.)

z <- c("bookkeeper", "successful", "tattoo", "dog", "cat")
part3 <- str_extract(z, '([a-z]).*\\1.*\\1'); part3
```

```
## [1] "eepe"    "success" "tatt"    NA        NA
```