Week3_data_607

waheeb Algabri

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"

```
# load and read data
data <- read.csv(url("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/recer
```

```
# This code uses the grep function to search for the pattern "DATA" or "STATISTICS" in the Major column

selected_majors <- data[grep("DATA|STATISTICS", data$Major),]
print(selected_majors)
```

provide code that identifies the majors that contain either "DATA" or "STATISTICS"

```
##    Rank Major_code                                        Major Total   Men
## 25   25       6212 MANAGEMENT INFORMATION SYSTEMS AND STATISTICS 18713 13496
## 47   47       3702               STATISTICS AND DECISION SCIENCE  6251  2960
## 54   54       2101      COMPUTER PROGRAMMING AND DATA PROCESSING  4168  3046
##    Women          Major_category ShareWomen Sample_size Employed Full_time
## 25  5217                Business  0.2787901         278    16413     15141
## 47  3291 Computers & Mathematics  0.5264758          37     4247      3190
## 54  1122 Computers & Mathematics  0.2691939          43     3257      3204
##    Part_time Full_time_year_round Unemployed Unemployment_rate Median P25th
## 25      2420                13017       1015        0.05823961  51000 38000
## 47      1840                 2151        401        0.08627367  45000 26700
## 54       482                 2453        419        0.11398259  41300 20000
##    P75th College_jobs Non_college_jobs Low_wage_jobs
## 25 60000         6342             5741           708
## 47 60000         2298             1200           343
## 54 46000         2024             1033           263
```

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")

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

# Convert the vector to a data frame
fruits_df <- data.frame(fruits, row.names = 1:length(fruits))

# Rename the column
colnames(fruits_df) <- c("Fruit")

# Capitalize the first letter of each fruit name
fruits_df$Fruit <- sapply(fruits_df$Fruit, function(x) {
  paste0(toupper(substring(x, 1, 1)), substring(x, 2))
})

# Show the transformed data
print(fruits_df)
```

```
##           Fruit
## 1   Bell pepper
## 2      Bilberry
## 3    Blackberry
## 4  Blood orange
## 5     Blueberry
## 6    Cantaloupe
## 7  Chili pepper
## 8    Cloudberry
## 9    Elderberry
## 10         Lime
## 11       Lychee
## 12     Mulberry
## 13        Olive
## 14  Salal berry
```

```
(.)\1\1
```

Describe, in words, what these expressions will match: It will match any character (.) that appears three times in a row. For example, "aaa" would match this expression.

2-This expression will match any two characters (.) that appear in the order of the second character followed by the first character. For example, “abab” would match this expression

```
"(.)(.)\\2\\1"
```

for example

```
# Create a string to match
text <- "hello world"

# Use the regex expression to match
result <- regexpr("(.)(.)\\2\\1", text)

# Check if there is a match
if (result != -1) {
  # Use the regmatches function to extract the matching substring
  match <- regmatches(text, result)
  print(match)
} else {
  print("hello waheeb")
}
```

```
## [1] "hello waheeb"
```

3-This expression will match any two characters denoted by . that appear twice in a row. For example, “abab” would match this expression.

(..)\1

4-This expression will match any three characters (denoted by .) where the first and third characters are the same and the second character is different. For example, “abcabc” would match this expression.

```
"(.).\\1.\\1"
```

5-This expression will match any string of characters (.) where the first three characters appear in reverse order anywhere in the string. For example, “abcdcba” would match this expression.

```
"(.)(.)(.).*\\3\\2\\1"
```

Construct regular expressions to match words that: 1- Start and end with the same character.

This expression uses two capturing groups. The first (.) captures the first character of the word, and the second (.*\1) captures any number of characters in between, followed by the same character as the first group.

```
"(.)(.*\\1)"
```

As an example

```
text <- "hello world"
result <- regexpr("(.)(.*\\1)", text)

if (result != -1) {
  match <- regmatches(text, result)
  print(match)
} else {
  print("hello waheeb")
}
```

```
## [1] "hello waheeb"
```

2- Contain a repeated pair of letters (e.g. “church” contains “ch” repeated twice.)

This expression uses two capturing groups. The first (.) captures any single character, and the second .$\backslash 1 . \backslash 1$ captures any number of characters in between the repeated pair of letters.

```
"(.).*\\1.*\\1"
```

As an example

```
text <- "hello world"
result <- regexpr("(.).*\\1.*\\1", text)

if (result != -1) {
  match <- regmatches(text, result)
  print(match)
} else {
  print("hello waheeb")
}
```

```
## [1] "hello waheeb"
```

3- Contain one letter repeated in at least three places (e.g. “eleven” contains three “e”s.)

To match words that contain one letter repeated in at least three places, you can use the following regular expression:

```
"(.)(.*\\1){2,}"
```

As an example

```
text <- "hello world"
result <- regexpr("(.)(.*\\1){2,}", text)

if (result != -1) {
  match <- regmatches(text, result)
  print(match)
} else {
  print("hello waheeb")
}
```

```
## [1] "hello waheeb"
```