607 - Assignment 3

Chitrarth Kaushik

2/11/2020

#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"

```
col_majors<-read.csv("https://raw.githubusercontent.com/datasets/five-thirty-eight-datasets/master/datas

major<-col_majors$major


dat<-grepl("data", major)
major[dat]
```

```
## [1] computer programming and data processing
## 175 Levels: accounting actuarial science ... zoology
```

```
stat<-grepl("statistics",major)
major[stat]
```

```
## [1] management information systems and statistics
## [2] statistics and decision science
## 175 Levels: accounting actuarial science ... zoology
```

#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")

```
string1 = c("bell pepper", "bilberry", "blackberry","blood orange")
string2 = c("blueberry","cantalope","chili pepper","cloudberry")
string3 = c("elderberry","lime","lychee","mulberry")
string4 = c("olive","salal berry")
fin_string=c(string1,string2,string3,string4)
fin_string
```

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

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

(.)\1\1

The regex will not match anything as the backreferencing has not been done correctly. If the backreferencing is done correctly, ie., “(.)\1\1” - then it would match a character repeated 3 times consecutively in a string.

“(.)(.)\2\1”

The regex backrefences the 2nd group before the 1st group - so it will match a string of 4 characters wherein the 2nd character match is immediately repeated before the 1st character match. For example, “banana” will not throw any matches - however “hanna” will throw matches with “anna” highlighted - the 2nd character match “n” repeats before the 1st character match, ie, “a”.

(..)\1

The regex will not match anything as the backreferencing has not been done correctly. If the escape character would have been used twice, ie., backreferencing to the first group, then the regex would have matched a pattern of 4 characters wherein the 1st set of two characters would have been identical to the 2nd of set of characters. For example, “kaka” would have thrown a positive match but “kanka” would not have.

“(.).\1.\1”

The regex will match a pattern of lenght 5 in which the 1st character alternates after the subsequent character. For example, “banana” will throw a positive match with “anana” highlighted.

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

The regex will match a pattern of any length as long as the the 1st leading three characters repeat in reverse at the end of the pattern. For exmple, “abcehhthwriwriwcbafahfoashfo” will throw a positive matcg with “abcehhthwriwriwcba” highlighted

#4 Construct regular expressions to match words that:

Start and end with the same character. "^(.).*\1$"

Contain a repeated pair of letters (e.g. “church” contains “ch” repeated twice.) "([a-zA-Z])([a-zA-Z]).*\1\2"

Contain one letter repeated in at least three places (e.g. “eleven” contains three “e”s.) “(.). *1.*\1”