# jjimenez_data607_hw3

Jean Jimenez

2023-09-11

## Data 607 Homework #3

### #1

#### Question

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".

#### Solution

First, I imported the csv from the raw github url and placed the data into a data frame named majors. Then I loaded dplyr and tidyverse. Afterwards, I piped my majors data frame through a filter. The filter uses regex to filter out majors that contain either 'DATA' or 'STATISTICS' and spits it out into a new data frame called infosci_maj.

```
#Importing the majors data

library(readr)

majors=read.csv(url("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/majors

names(majors)
```

```
## [1] "FOD1P"          "Major"          "Major_Category"
```

```
#loading, tidyverse, dplyr
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
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
#filtering out only DATA or STATISTICS using regex

infosci_maj = majors %>%
  filter(grepl("(?=.*DATA)|(?=.*STATISTICS)", Major, perl = TRUE))

head(infosci_maj)
```

```
##   FOD1P                                        Major          Major_Category
## 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
```

## #2

### Question

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")

### Solution

First, I imported each of the fruits above into a list. Then, I pasted the characters "c(" in front of that list. This gives me a character version of the format asked in the questions.

```r
#input data

shopping_list = c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe", "
# Create the final output
shopping_vector = paste("c(", paste( shopping_list,  sep = "", collapse = ", "), ")", sep = "")


print(shopping_vector)
```

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

## #3

### Question

Describe, in words, what these expressions will match:

1. (.)\1\1
2. "(.)(.)\\2\\1"
3. (..)\1
4. "(.).\\1.\\1"
5. "(.)(.)(.).*\\3\\2\\1"

### Solution

1. This regex will match any character that is repeated three times in a row, like 'zzz', '555', or 'AAAinsurance'.
2. This regex will match any two characters followed by the same two characters in reverse order. For example, 'toot' would be matched but not 'ttoo'.
3. This one will match any two characters that are immediately repeated like '2121' or 'momo'.
4. This one is a bit more complicated to explain in words. Lets say you have three characters, 'e' ,'m' and 'w'. This one will match 'emewe' but e can be any other letter. It can also match 'tmtwt' or '42484'.
5. This one will match any three characters that are followed by either non or any character [.*] and the same first three but in reverse. Think '123…anything…321' or 'xyz…zyx'. Anagrams would be interesting to use for example 'racecar…racecar'.

## #4

### Question

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.)

**Solution**

To do this question, first I created a sample text with a bunch of U.S. City names to use in this question. I made sure to have examples that can represent each condition and added some extra values.

Afterwards, I processed the data in the cities text into the city_vector. I used str_split and it split the cities text every space. Afterwards, I removed empty strings because there were some.

To filter out cities that start and end with the same character, I first converted each value in city_vector to lower case. Afterwards, I ran str_detect with the regex filter "^(.).*\\1$" which matches characters that repeat at the beginning and end of a string. Then, I printed it out.

Similar to the previous example, in aa, I used str_detect to filter out cities containing repeating pair of characters by using the regex code "(.).*\\1" and printed.

For the final example, I used the regex code "(.).*\\1.*\\1" which matched the cities that had a character that repeated 3 times.

```r
#sample text containing city names to match
cities = "NewYork Chicago Alameda LosAngeles Mississippi Houston Miami Philadelphia Elma Atlanta Dallas

#split the cities string into a vector of city names

city_vector = str_split(cities, " ")[[1]]

#remove empty strings
city_vector = city_vector[city_vector != ""]

# filter cities that start and end with the same character

abba = city_vector[str_detect(tolower(city_vector), "^(.).*\\1$")]
print(paste("Cities that start and end with the same character:", paste(abba, collapse = ", ")))
```

```
## [1] "Cities that start and end with the same character: Alameda, Atlanta, Aurora"
```

```r
# filter cities that contain one repeting pair
aa = city_vector[str_detect(city_vector, "(.).*\\1")]
print(paste("Cities that contain a repeated pair of letters:", paste(aa, collapse = ", ")))
```

```
## [1] "Cities that contain a repeated pair of letters: Alameda, LosAngeles, Mississippi, Houston, Miam
```

```r
# Filter cities that contain one letter repeated 3 times
aaa = city_vector[str_detect(city_vector, "(.).*\\1.*\\1")]
print(paste("Cities that contain one letter repeated in at least three places:", paste(aaa, collapse =
```

```
## [1] "Cities that contain one letter repeated in at least three places: Mississippi, Tallahassee, Cin
```