R Character Manipulation and Date Processing

Jawaid Hakim

2022-09-13

Contents


```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

1 Assignment

1.1 Search for majors containing “DATA” or “STATISTICS” - 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”

Load file from GitHub.

```
ds <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/2d2ff3e9457549d51f8e571c52099bfe
```

Let’s take a look at the unique values of *Major* and *Major_Category* columns. There are 174 unique majors and 16 major categories.

```
glimpse(as.factor(ds$Major))
```

```
##  Factor w/ 174 levels "ACCOUNTING","ACTUARIAL SCIENCE",..: 70 6 5 7 67 151 164 117 68 132 ...
```

```
glimpse(as.factor(ds$Major_Category))
```

```
##  Factor w/ 16 levels "Agriculture & Natural Resources",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Look for all majors that contain either "DATA" or "STATISTICS". Let's assume the search terms may appear in either Major or Major Category columns and we want to look consider both columns.

The search terms are found in 3 rows:

1. MANAGEMENT INFORMATION SYSTEMS AND STATISTICS
2. COMPUTER PROGRAMMING AND DATA PROCESSING
3. STATISTICS AND DECISION SCIENCE

```
ds %>%
    filter(
            grepl("DATA|STATISTICS", ignore.case = TRUE, Major) |
            grepl("DATA|STATISTICS", ignore.case = TRUE, Major_Category)) %>%
    arrange(Major)
```

```
##   FOD1P                                        Major          Major_Category
## 1  2101      COMPUTER PROGRAMMING AND DATA PROCESSING Computers & Mathematics
## 2  6212 MANAGEMENT INFORMATION SYSTEMS AND STATISTICS                Business
## 3  3702               STATISTICS AND DECISION SCIENCE Computers & Mathematics
```

1.2 Transform data - 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")

Let's generate the input data as a string.

```
input_data <- '[1] "bell pepper"  "bilberry"     "blackberry"   "blood orange"
[5] "blueberry"    "cantaloupe"   "chili pepper" "cloudberry"
[9] "elderberry"   "lime"         "lychee"       "mulberry"
[13] "olive"        "salal berry"'
input_data
```

```
## [1] "[1] \"bell pepper\"  \"bilberry\"     \"blackberry\"   \"blood orange\"\n[5] \"blueberry\"    \
```

Let's extract fruit names from the input, *unlist* to flatten the list into a vector.

```
input_split <- unlist(str_extract_all(input_data, '"[a-zA-Z ]+"'))
input_split
```

```
##  [1] "\"bell pepper\""  "\"bilberry\""     "\"blackberry\""   "\"blood orange\""
##  [5] "\"blueberry\""    "\"cantaloupe\""   "\"chili pepper\"" "\"cloudberry\""
##  [9] "\"elderberry\""   "\"lime\""         "\"lychee\""       "\"mulberry\""
## [13] "\"olive\""        "\"salal berry\""
```

Almost there. All that is needed is remove the pesky double quotes. And we are done!

```
input_clean <- str_remove_all(input_split, '"')
input_clean
```

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

1.3 Describe Data - 3

1. (.)\1\1 : matches any character followed by 2 repeats of matching character. Example, aaa*, BBB, 111, etc.
2. (.)(.)\2\1 : matches any two characters followed by 2ˆnd matching character followed by the 1ˆst matching character. For example, abba, 1221, etc. In other words, *palindromes* of length 4.
3. (..)\1 : two character repeats. For example, abab, 1212.
4. (.).\1.\1 : matches 5 character strings, where 1ˆst, 3ˆrd, and 5ˆth characters are the same. Example, abaka, *2141, etc.
5. (.)(.)(.).*\3\2\1 : matches strings of length > 5, where the first 3 characters are repeated at the end in reverse order. For example, abc12345cba.

1.4 Construct regular expressions to match words - 4

1. Start and end with same character: (.).*\1
2. Contain a repeated pair of letters (e.g. “church” contains “ch” repeated twice.): .*([a-zA-Z])([a-zA-Z]).*\1\2.*
3. Contain one letter repeated in at least three places (e.g. “eleven” contains three “e”s.): .*([a-zA-Z]).*\1.*\1.*