Wk3: R Character Manipulation and Date Processing

Dirk Hartog

2023-09-15

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## 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 errors
```

QUESTION 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"

The raw data can be read into a data frame

```
url <- "https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/majors-list.csv"
major_list <- read_csv(url, show_col_types = FALSE)
glimpse(major_list)
```

```
## Rows: 174
## Columns: 3
## $ FOD1P          <chr> "1100", "1101", "1102", "1103", "1104", "1105", "1106",~
## $ Major          <chr> "GENERAL AGRICULTURE", "AGRICULTURE PRODUCTION AND MANA~
## $ Major_Category <chr> "Agriculture & Natural Resources", "Agriculture & Natur~
```

Using the *stringr* library we can use regular expressions to identify majors that meet specific criteria. We can filter the original data frame to find the majors that include the word "DATA" or "STATISTICS"

```
major_list %>%
  filter(str_detect(Major, "DATA|STATISTICS"))
```

```
## # A tibble: 3 x 3
##   FOD1P Major                                          Major_Category
##   <chr> <chr>                                          <chr>
## 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
```

QUESTION 2: Write code that transforms data:

I first created a vector with a long string containing many fruits. My goal was to match each fruit, create a seperate string, and store it in a character vector.

```
fruits <- "bell pepper, bilberry, blackberry, blood orange, blueberry, cantaloupe, chili pepper, cloudb

# pattern to match each fruit
pattern <- "\\w+\\s?\\w+"

# find each fruit with str_match_all
fruit_list <- str_match_all(fruits, pattern)

#use unlist() to unpack the items into a vector
fruit_vector <- unlist(fruit_list)
fruit_vector
```

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

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

A. The expression **(.)\1\1** will match strings with the character captured by the (.) appearing three times in a row. See the example below.

```
x <- "aaaxyzccc"
# The regular expression needs to expressed in it's string form
str_view(x, "(.)\\1\\1")
```

```
## [1] | <aaa>xyz<ccc>
```

B. The expression **"(.)(.)\2\1"** will match any string with any two characters grouped by each (.) that appear consecutively in reverse order. See the example below.

```
str_view(words, "(.)(.)\\2\\1")
```

```
##  [19] | after<noon>
##  [43] | <appa>rent
##  [53] | <arra>nge
## [107] | b<otto>m
## [112] | br<illi>ant
## [174] | c<ommo>n
## [230] | d<iffi>cult
## [259] | <effe>ct
## [329] | f<ollo>w
## [422] | in<deed>
## [470] | l<ette>r
## [521] | m<illi>on
## [581] | <oppo>rtunity
## [582] | <oppo>se
## [877] | tom<orro>w
```

C. The expression **"(..)\1"** will return a string that repeats a group of any two characters "(..)". See example below.

```
z <- "gntntlw"
str_view(z, "(..)\\1")
```

```
## [1] | g<ntnt>lw
```

D. The expression **"(.).\1.\1"** will match any string that alternates the character grouped in (.) with any character. See the example below.

```
a <- "gafrabafa"
str_view(a,"(.).\\1.\\1" )
```

```
## [1] | gafr<abafa>
```

E. The expression **"(.)(.)(.).*\3\2\1"** will match any string with three groups of characters that are sperated by any character appearing zero or more times followed by the same cahracters in each group but in reverse order. See the example below.

```
b <- "habcxycbai"
str_view(b, "(.)(.)(.).*\\3\\2\\1")
```

```
## [1] | h<abcxycba>i
```

QUESTION 4: Construct regular expressions based on these criteria

A. Start and end with the same character.

```
# we will view the first five matches
head(str_view(words, "^(.).*\\1$"))
```

```
##  [36] | <america>
##  [49] | <area>
## [209] | <dad>
## [213] | <dead>
## [223] | <depend>
## [258] | <educate>
```

B. Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.).

```
# we will view the first 5 matches
head(str_view(words, "(..).*\\1"))
```

```
##  [48] | ap<propr>iate
## [152] | <church>
## [181] | c<ondition>
## [217] | <decide>
## [275] | <environmen>t
## [487] | l<ondon>
```

C. Contain one letter repeated in at least three places (e.g. "eleven" contains three "e"s.)

```r
# we will view the first five matches
head(str_view(words, "(.).*\\1.*\\1"))
```

```
##  [48] | a<pprop>riate
##  [62] | <availa>ble
##  [86] | b<elieve>
##  [90] | b<etwee>n
## [119] | bu<siness>
## [221] | d<egree>
```