Strings

MGinorio

Character Manipulation & Data Processing

STRINGS - You can create strings with either single quotes or double quotes. Unlike other languages, there is no difference in behaviour. I recommend always using “, unless you want to create a string that contains multiple”.

DATES -Always use the simplest possible data type that works for your needs. That means if you can use a date instead of a date-time, you should. Date-times are substantially more complicated because of the need to handle time zones.

Implementation

String Manipulation with string:: The string package provides a set of internally consistent tools for working with character strings, i.e sequences of characters surrounded by quotation marks

Results

Overview

Figure 1: String Manipulation

```
url.show("https://fivethirtyeight.com/features/the-economic-guide-to-picking-a-college-major/")
```

Packages

```
#Packages used

library(tidytext)
library(DT)
library(tm)
```

```
## Loading required package: NLP
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::annotate() masks NLP::annotate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
```

```
library(stringr)
```

1.- Str_detect

1.1 String_Detect Using the 173 majors listed in fivethirtyeight.com's College Majors dataset, provide code that identifies the majors that contain either "DATA" or "STATISTICS"

```
college_majors <-read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/

glimpse(college_majors)
```

```
## Rows: 174
## Columns: 3
## $ FOD1P          <chr> "1100", "1101", "1102", "1103", "1104", "1105", "110...
## $ Major          <chr> "GENERAL AGRICULTURE", "AGRICULTURE PRODUCTION AND M...
## $ Major_Category <chr> "Agriculture & Natural Resources", "Agriculture & Na...
```

```
data_majors <- str_detect(college_majors$Major, fixed("DATA"))
college_majors[data_majors, ]
```

```
##    FOD1P                                        Major          Major_Category
## 52  2101 COMPUTER PROGRAMMING AND DATA PROCESSING Computers & Mathematics
```

There is only one result that contains "DATA" = 2101

```
stats_majors <- str_detect(college_majors$Major, fixed("STATISTICS"))
college_majors[stats_majors, ]
```

```
##    FOD1P                                          Major          Major_Category
## 44  6212 MANAGEMENT INFORMATION SYSTEMS AND STATISTICS                 Business
## 59  3702               STATISTICS AND DECISION SCIENCE Computers & Mathematics
```

There is only one result that contains "STATISTICS" = 6212 + 3702

2.- String_extract

2.1 Fruits

Create Produce List for Next Week

Figure 2: String Manipulation

[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”)

```
str_1 = c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry")
str_2 = c("blueberry", "cantaloupe", "chili pepper", "cloudberry")
str_3 = c("elderberry","lime","lychee","mulberry")
str_4 = c("olive", "salad berry")

fruit_list = c(str_1, str_2, str_3, str_4)
fruit_list
```

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

3.- String Expressions

Expressions

(.)\1\1

This expression refers to groups - () - we use parenthesis to set precedent (order of evaluation) in this case.

we need to type “(.)\1\1” in string to mean this in regexp (.)\1\1. it means the first capturing group in the matched expression. however, the expression is not complete. regexp will match (. . .)

“(.)(.)\2\1”

The regexp back-references - that means it picks the second group before the fist. ex. "

```
string_1 <- c("abba", "aaabbb","aaabbb", "abba", "bbaabbaa")
string_1 %>%
  str_detect(pattern = "(a)(b)\\2\\1")
```

```
## [1]  TRUE FALSE FALSE  TRUE  TRUE
```

(..)\1

This will search for two characters, repeated once, like “mama” or “5656”. The correct expression would be “(..)\1”.

(.).\1.\1"

This will search for a five character term, three of which are the same, like “momma” or “75717”.

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

This will construct a set of characters that begin and end with the same three characters, except the second instance is reversed, like “racecar” or “12378724321”.

4.- Construct

Strings

Construct regular expressions to match words that:

```
str_bring <- function(string, pattern) {
  string[str_detect(string, pattern)]
}
```

```
str_bring(fruit_list, "(.).*\\1$")
```

Start and end with the same character.

```
## [1] "lychee"
```

```
str_subset(fruit_list, "([A-Za-z][A-Za-z]).*\\1")
```

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

```
## [1] "bell pepper"  "chili pepper" "elderberry"
```

```
str_subset(fruit_list, "([a-z]).*\\1.*\\1")
```

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

```
## [1] "bell pepper"  "blood orange" "chili pepper" "elderberry"
```