Guide 5.2 strings in R

September 25, 2019

1 Guía 5.2: Strings en R

Computación 2, IES. Profesor: Eduardo Jorquera, eduardo.jorquera@postgrado.uv.cl

1.1 Librerías!

Usaremos los siguientes paquetes para manipular cadenas de caracteres:

```
In [2]: library(tidyverse)
        library(stringr)
        options(jupyter.rich_display=T)
```

```
 Attaching packages  tidyverse 1.2.1
 ggplot2 3.2.1      purrr   0.3.2
 tibble  2.1.3      dplyr   0.8.3
 tidyr   0.8.3      stringr 1.4.0
 readr   1.3.1      forcats 0.4.0
 Conflicts  tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
```

2 Repetición

El siguiente paso para subir de nivel, es controlar la cantidad de veces que un patrón se empareja: * ?: 0 ó 1 * +: 1 o más * *: 0 o más

```
In [7]: x <- "1888 es el año más largo en números romanos: MDCCCLXXXVIII"
        str_view(x, "CC?")
```

```
HTML widgets cannot be represented in plain text (need html)
```

```
In [8]: str_view(x, "CC+")
```

```
HTML widgets cannot be represented in plain text (need html)
```

```
In [9]: str_view(x, 'C[LX]+')
```

```
HTML widgets cannot be represented in plain text (need html)
```

Nota la cómo es que los operadores se introducen para reconocer patrones en las expresiones regulares. Puedes escribir `colo?r` para emparejar color o colour. La mayoría de las veces se usa el paréntesis, como `plata(no)+`.

También puedes especificar el número de veces que se empareja: * `{n}`: exactamente n * `{n,}`: n o más * `{,m}`: a lo más m * `{n,m}`: entre n y m

```
In [12]: str_view(x, "C{2}")
         str_view(x, "C{2,}")
         str_view(x, "C{2,3}")
```

```
HTML widgets cannot be represented in plain text (need html)
```

```
HTML widgets cannot be represented in plain text (need html)
```

```
HTML widgets cannot be represented in plain text (need html)
```

Por defecto estos emparejamientos son codiciosos. Puedes reducir el cóigo usando `?` después de éstos. Esta es una característica avanzada de las expresioner regulares

```
In [14]: str_view(x, 'C{2,3}?')
         str_view(x, 'C[LX]+?')
```

```
HTML widgets cannot be represented in plain text (need html)
```

```
HTML widgets cannot be represented in plain text (need html)
```

3 Ejercicios

1. Describa lo equivalente de `?`, `+`, `*` en forma de `{n,m}`.
2. Describa a qué es que éstas expresiones regulares (o strings) se emparejan. Lee atentamente para ver si se trata de una expresión regular, o un string que define una expresión regular.
   - `^.*$`
   - `"\\{.+\\}"`
   - `\d{4}-\d{2}-\d{2}`
   - `"\\\\{4}"`
3. Crea una expresión regular que encuentre todas las palabras que:
   - Empiezan con consonantes.
   - Tienen tres o más vocales seguidas en la palabra.
   - Tienen dos o más pares de vocales de forma consecutiva.

```
In [ ]:
```

4 Agrupando referencias previas (backreferences)

Antes, aprendiste a usar los parentesis para emparejar expresiones complejas. Parentesis también crean un grupo enumerado de captura. Un grupo de captura almacena *la parte de un string emparejado* según la parte de la expresión regular dentro del paréntesis. Te puedes referir al mismo texto que previamente emparejaste por un grupo de captura con backreferences, como `\1`, `\2`, etc. Por ejemplo, la siguiente expresión regular encuentra todas las frutas que tengan un par de letras repetidas:

```
In [21]: str_view(fruit, "(..)\\1", match = TRUE)
```

```
HTML widgets cannot be represented in plain text (need html)
```

5 Ejercicios

1. Describa, en palabras, a qué se emparejarán estas expresiones:

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

2. Construya expresiones regulares que se emparejen con palabras que:

   - Empiecen y terminen con el mismo caracter.
   - Contengan un par repetido de letras (por ejemplo, "rara" contiene "ra" dos veces).
   - Contengan una letra repetida al menos tres veces (por ejemplo, "hipopotamo" contiene tres "o"es).

```
In [ ]:
```