Guide 4 strings in R

September 8, 2019

1 Guía 3: Strings en R

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

1.1 Instalar!

Usaremos los siguientes paquetes para manipular cadenas de caracteres:

```
In [6]: library(tidyverse)
        library(stringr)
```

```
 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 Lo básico

Puedes crear cadenas de caracteres usando comillas simples o dobles. A diferencia de otros lenguages de programación, esto no hace ninguna diferencia.

```
In [1]: string1 <- "Esto es un string"
        string2 <- 'Si quieres incluir "comillas" dentro
                    de un string, usa comillas simples'
```

Si olvidas cerrar las comillas en una línea, en la siguiente línea verás el caracter + en la consola de R. Sucede lo mismo cuando dejas una llave, paréntesis o corchete sin cerrar.

Para incluir comillas en el texto (en tu cadena de caracteres), puedes usar \ para evitar que la comilla se interprete como el cierre de tu string:

```
In [2]: comillas_dobles  <- "\"" # ó '"'
        comillas_simples <- '\'' # ó "'"
```

Esto quiere decir que si quieres incluir un backslash de manera literal, tendrás que escribirlo dos veces: `"\\"`.

Considera que la versión impresa de un arreglo no es el arreglo por sí mismo, ya que la versión que se imprime muestra los "escapes". Para ver el contenido crudo de un string, usa la función `writeLines()`:

```
In [3]: x <- c("\"", "\\")
        x
        writeLines(x)
```

1. '"' 2. '\\'

```
"
\
```

mira lo que sucede usando print:

```
In [8]: print(x)
```

```
[1] "\"" "\\"
```

Hay un abanico enorme de caracteres especiales. Lo más comunes son `"\n"`, que es salto de línea, y `"\t"`, que es el espacio de tabulación, pero puedes ver una lista completa usando `?'"'`, ó `?"'"`. También habrán veces que verás cosas como `"\u00b5"`, esto es una manera de escribir caracteres que no necesariamente están en tu distribución de teclado, de tal forma que funcione en todas las plataformas:

```
In [10]: x <- "\u00b5"
         x
```

'μ'

Caracteres múltiples son almacenados comunmente en vectores de caracteres, que puedes crear con `c()`

```
In [11]: c("uno", "dos", "tres")
```

1. 'uno' 2. 'dos' 3. 'tres'

3 El largo de una cadena de caracteres

Dentro de la base de R están contenidas muchas funciones para trabajar con cadenas de caracteres; pero como hemos visto en guías anteriores, las funciones básicas de R pueden sufrir de inconsistencia de sintaxis, por lo que mejor evitaremos su uso. En su lugar, usaremos funciones del paquete `stringr`. Las funciones del paquete son más intuitivas y empiezan con `str_`. Por ejemplo, `str_length()` te dice el número de caracteres de un string:

```
In [12]: str_length(c("a", "Computación II", "\u00b5", NA))
```

1. 1 2. 14 3. 1 4. <NA>

4 Combinando strings

Para combinar dos o más strings, usa `str_c()`:

```
In [13]: str_c("x", "y")
         str_c("x", "y", "z")
```

'xy'
'xyz'
puedes controlar la manera en que se separa usando el argumento `sep`:

```
In [15]: str_c("x", "y", sep = ", ")
```

'x, y'
Como muchas funciones en R, los valores perdidos son un problema bastante común (más de lo que te imaginas). Si los quieres mostrar como `"NA"`, usa `str_replace_na()`:

```
In [17]: x <- c("abc", NA)
         str_c("|-", x, "-|")

         str_c("|-", str_replace_na(x), "-|")
```

1. '|-abc-|' 2. NA
1. '|-abc-|' 2. '|-NA-|'
Como se mostró anteriormente, `str_c()` es vectorizado, y automáticamente reciclará vectores más cortos al mismo largo que el de mayor largo:

```
In [21]: str_c("prefijo-", c("a", "b", "c"), "-subfijo")
```

1. 'prefijo-a-subfijo' 2. 'prefijo-b-subfijo' 3. 'prefijo-c-subfijo'
Los objetos de largo 0, son silenciosamente borrados. Esto es particularmente útil en conjunción con `if`:

```
In [26]: nombre <- "Roberto"
         momento_del_dia <- "morning"
         cumple <- FALSE

         str_c(
           "good ", momento_del_dia, " ", nombre,
           if (cumple) " and HAPPY BIRTHDAY",
           "."
         )
```

'good morning Roberto.'

5 Actividad

Cree una función en R donde se retorne una vaca enviando un mensaje, donde el cuadro de diálogo se ajuste al largo del texto. Si el largo del texto es demasiado grande, haga que el resultado se imprima en más de una línea.

Lo más básico de una función (puedes usar o no el `return()` a conveniencia:

```
In [33]: f <- function(a,b,c)
             return(str_c("el resultado es", a+b+c,sep=": "))
         f(1,2,3)
```

'el resultado es: 6'

```
In [ ]:  _____
         < moo >
          -----
                 \   ^__^
                  \  (oo)\_______
                     (__)\       )\/\
                         ||----w |
                         ||     ||
```