Programación en R para el análisis de datos

Expresiones Regulares

Nicolás Schmidt

mail::nschmidt@cienciassociales.edu.uy
GitHub::@Nicolas-Schmidt

Departamento de Ciencia Política
Facultad de Ciencias Sociales

Ruta

1. Qué es una expresión regular?
2. Expresiones regulares
3. Carguemos datos
4. Funciones de base
5. Funciones de stringr

Qué es una expresión regular?

expresión regular?

Una expresión regular es un patrón que describe un conjunto de cadenas de caracteres. Es decir, es una secuencia de caracteres que conforma un patrón de bsqueda.

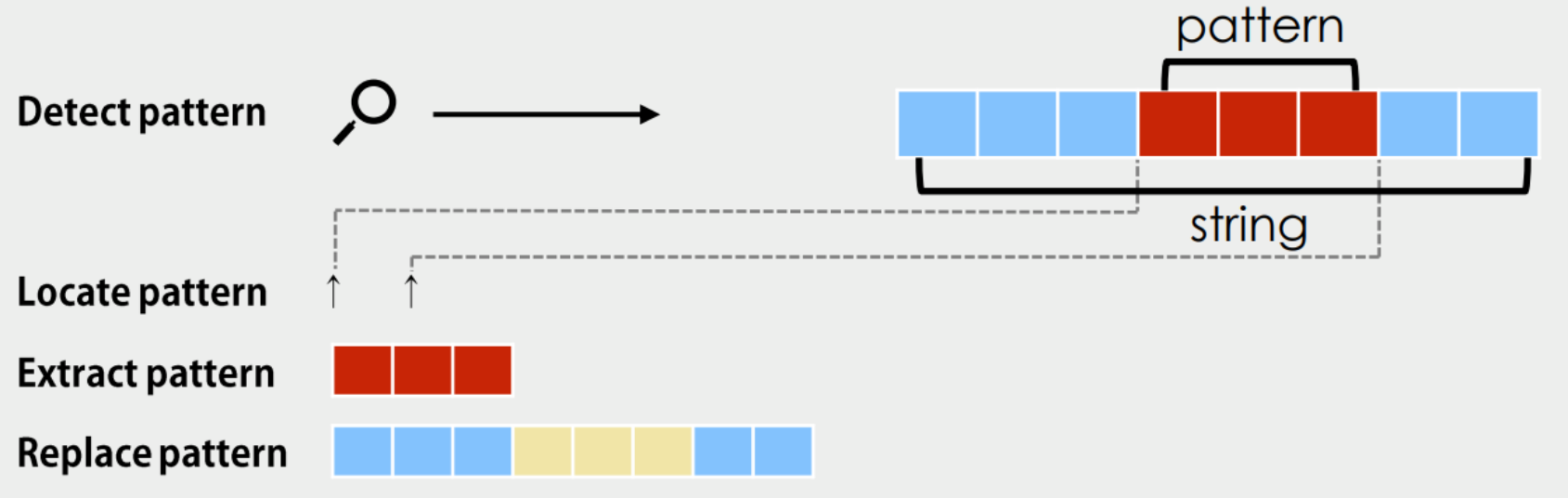
Este patrón de bsqueda a diferencia del reconocimiento de patrones, por ejemplo, la coincidencia buscada es comnmente exacta.

Cuándo usarla?

Obviamente, es necesario utilizar una *regular expression* cuando se quiere buscar dentro de datos de tipo `character`. De lo contrario se utilizaría lógica matemática. Esto no quiere decir que no se pueden incorporar

nmeros en la conformación de la expresión regular. Simplemente se deben tener caracteres (numéricos o alfabéticos) almacenado en modo texto.

regex


Detect pattern
pattern
string
Locate pattern
Extract pattern
Replace pattern

Expresiones regulares

Cuantificadores

`*`	Coincide al menos 0 veces.
`+`	Coincide al menos 1 vez.
`?`	Coincide a lo sumo 1 vez.
`{n}`	Coincide 'n' veces exactamente.
`{n,}`	Coincide al menos 'n' veces.
`{n,m}`	Coincide entre 'n' y 'm' veces.

Ubicacion dentro del patrón

^	Coicidencia con el inicio de la cadena.
$	Coicidencia con el fin de la cadena.
\b	Coincide con la cadena vacia en el bodere de una balabra.
\B	Coincide con la cadena vacia siempre que no esté en el borde de una palabra.height

Operadores

`.`	Coicidencia con cualquier caracter.
`[...]`	Lista de caracteres.
`[^ ...]`	Negación de lista de caracterets.
`|`	El operador 'o'.
`(...)`	Agrupacion de expresiones regulares.

Escapes

Hay caracteres especiales que en R tienen un uso funcional. Por lo que se debe generer escape para poder incorporarlos como elementos de una expresión regular.

$ * + . ? [ ] ^ { } | ()

Para escapar hay que usar doble barra, por ejemplo: \\$

Clases

`[:digit:]`	Nmeros del 0 al 9: [0-9].
`\D`	No dígitos:[^ 0-9].
`[:lower:]`	letras minsculas:[a-z].
`[:upper:]`	letras maysculas:[A-Z]
`[:alnum:]`	caracteres alfabéticos:[[:lower:][:upper:]] o [A-z].
`\w`	caracteres de palabras:[[:alnum:]_]o [A-z0-9_].
`\W`	no palabra:[^ A-z0-9_].
`[:blank:]`	espacio y tabulación.
`[:space:]`	tabulación, nueva línea, tabulación vertical...
`\s`	espacio.
`\S`	no espacio.
`[:punct:]`	caracteres de puntuación

Clases

Importante:

- Las clases '[...]' deben usarse entre cortechetes: '[[...]]'
- El '\' es un caracter especial. Por lo de que se debe usar doble: \\D

Carguemos datos

Cargando datos

```
## Warning: package 'tidyverse' was built under R version 3.6.3
```

```
library(gapminder)
paises <- gapminder$country %>% as.character() %>% unique()
length(paises)
```

```
## [1] 142
```

```
paises[1:40]
```

```
##  [1] "Afghanistan"              "Albania"
##  [3] "Algeria"                  "Angola"
##  [5] "Argentina"                "Australia"
##  [7] "Austria"                  "Bahrain"
##  [9] "Bangladesh"               "Belgium"
## [11] "Benin"                    "Bolivia"
## [13] "Bosnia and Herzegovina"   "Botswana"
## [15] "Brazil"                   "Bulgaria"
## [17] "Burkina Faso"             "Burundi"
## [19] "Cambodia"                 "Cameroon"
## [21] "Canada"                   "Central African Republic"
## [23] "Chad"                     "Chile"
## [25] "China"                    "Colombia"
## [27] "Comoros"                  "Congo, Dem. Rep."
## [29] "Congo, Rep."              "Costa Rica"
## [31] "Cote d'Ivoire"            "Croatia"
## [33] "Cuba"                     "Czech Republic"
## [35] "Denmark"                  "Djibouti"
## [37] "Dominican Republic"       "Ecuador"
## [39] "Egypt"                    "El Salvador"
```

Funciones de base

Cargando datos

`grep()`	Devuelve los indices en los que se encuentra el patron
`grepl()`	Devuelve un vector logico con `TRUE` donde hay coincidencia
`regexpr()` `gregexpr()`	Devuelve los indices de la cadena en donde comineza la coincidencia y la duracion de la misma
`sub()` `gsub()`	Busca y reemplaza en donde hay coicidencia

Ejemplos

```
vec <- grep("ina", paises)
length(vec)
```

```
## [1] 5
```

```
vec
```

```
## [1]  5 13 17 25 56
```

Ejemplos

```r
a  <- grep("A", paises)
length(a)
```

```
## [1] 10
```

```r
aA <- grep("A", paises, ignore.case = TRUE, value = TRUE)
aA[1:10]
```

```
##  [1] "Afghanistan" "Albania"     "Algeria"     "Angola"      "Argentina"
##  [6] "Australia"   "Austria"     "Bahrain"     "Bangladesh"  "Bolivia"
```

```r
aA <- grep("[Aa]", paises)
aA[1:10]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 12
```

Ejemplos

```
grep(paises, pattern = "^U", value = TRUE)
```

```
## [1] "Uganda"         "United Kingdom" "United States"  "Uruguay"
```

```
grep(paises, pattern = "c$", value = TRUE)
```

```
## [1] "Central African Republic" "Czech Republic"
## [3] "Dominican Republic"       "Slovak Republic"
```

```
grep(paises, pattern = "\\s", value = TRUE)
```

```
##  [1] "Bosnia and Herzegovina"   "Burkina Faso"
##  [3] "Central African Republic" "Congo, Dem. Rep."
##  [5] "Congo, Rep."              "Costa Rica"
##  [7] "Cote d'Ivoire"            "Czech Republic"
##  [9] "Dominican Republic"       "El Salvador"
## [11] "Equatorial Guinea"        "Hong Kong, China"
## [13] "Korea, Dem. Rep."         "Korea, Rep."
## [15] "New Zealand"              "Puerto Rico"
## [17] "Sao Tome and Principe"    "Saudi Arabia"
## [19] "Sierra Leone"             "Slovak Republic"
## [21] "South Africa"             "Sri Lanka"
## [23] "Trinidad and Tobago"      "United Kingdom"
## [25] "United States"            "West Bank and Gaza"
## [27] "Yemen, Rep."
```

Ejemplos

```
grep(paises, pattern = "[[:punct:]]", value = TRUE)
```

```
## [1] "Congo, Dem. Rep." "Congo, Rep."      "Cote d'Ivoire"    "Guinea-Bissau"
## [5] "Hong Kong, China" "Korea, Dem. Rep." "Korea, Rep."      "Yemen, Rep."
```

```
grep(paises, pattern = "[[:punct:]]", value = TRUE)
```

```
## [1] "Congo, Dem. Rep." "Congo, Rep."      "Cote d'Ivoire"    "Guinea-Bissau"
## [5] "Hong Kong, China" "Korea, Dem. Rep." "Korea, Rep."      "Yemen, Rep."
```

Ejemplos

```r
strings <- c("a", "ab", "acb", "accb", "acccb", "accccb")
strings
```

```
## [1] "a"      "ab"     "acb"    "accb"   "acccb"  "accccb"
```

```r
grep("ac*b", strings, value = TRUE)
```

```
## [1] "ab"     "acb"    "accb"   "acccb"  "accccb"
```

```r
grep("ac+b", strings, value = TRUE)
```

```
## [1] "acb"    "accb"   "acccb"  "accccb"
```

```r
grep("ac?b", strings, value = TRUE)
```

```
## [1] "ab"  "acb"
```

Ejemplos

```r
strings <- c("a", "ab", "acb", "accb", "acccb", "accccb")
strings
```

```
## [1] "a"      "ab"     "acb"    "accb"   "acccb"  "accccb"
```

```r
grep("ac{2}b", strings, value = TRUE)
```

```
## [1] "accb"
```

```r
grep("ac{2,}b", strings, value = TRUE)
```

```
## [1] "accb"   "acccb"  "accccb"
```

```r
grep("ac{2,3}b", strings, value = TRUE)
```

```
## [1] "accb"  "acccb"
```

Ejemplos

```r
strings <- c("abcd", "cdab", "cabd", "c abd")
strings
```

```
## [1] "abcd"  "cdab"  "cabd"  "c abd"
```

```r
grep("ab", strings, value = TRUE)
```

```
## [1] "abcd"  "cdab"  "cabd"  "c abd"
```

```r
grep("^ab", strings, value = TRUE)
```

```
## [1] "abcd"
```

Ejemplos

```
strings
```

```
## [1] "abcd"  "cdab"  "cabd"  "c abd"
```

```
grep("ab$", strings, value = TRUE)
```

```
## [1] "cdab"
```

```
grep("\\bab", strings, value = TRUE)
```

```
## [1] "abcd"  "c abd"
```

Ejemplos

```r
strings <- c("^ab", "ab", "abc", "abd", "abe", "ab 12")
strings
```

```
## [1] "^ab"   "ab"    "abc"   "abd"   "abe"   "ab 12"
```

```r
grep("ab.", strings, value = TRUE)
```

```
## [1] "abc"   "abd"   "abe"   "ab 12"
```

```r
grep("ab[c-e]", strings, value = TRUE)
```

```
## [1] "abc" "abd" "abe"
```

```r
grep("ab[^c]", strings, value = TRUE)
```

```
## [1] "abd"   "abe"   "ab 12"
```

Ejemplos

```r
strings <- c("^ab", "ab", "abc", "abd", "abe", "ab 12")
strings
```

```
## [1] "^ab"   "ab"    "abc"   "abd"   "abe"   "ab 12"
```

```r
grep("^ab", strings, value = TRUE)
```

```
## [1] "ab"    "abc"   "abd"   "abe"   "ab 12"
```

```r
grep("\\^ab", strings, value = TRUE)
```

```
## [1] "^ab"
```

```r
grep("abc|abd", strings, value = TRUE)
```

```
## [1] "abc" "abd"
```

```r
gsub("(ab) 12", "\\1 34", strings)
```

```
## [1] "^ab"   "ab"    "abc"   "abd"   "abe"   "ab 34"
```

Funciones de stringr

Stringr

El paquete `stringr` ofrece un conjunto de fucniones para trabajar con expresione sregulares que agregan y mejoran en lagunos casos funciones existentes en el paquete base. Asimismo, en muchas circunstancias el rendimiento es mejor.

De todas maneras la gran virtud refiere a que las funciones tienen un nombre intuitivo. Todas las funciones comienzan con el prefijo `str_*`