Lenguaje R

Conceptos Básicos

Lenguaje de programación, no necesita compilar

X = 3

Values	
x	3

Es capaz de inferir tipos
No se exigen declarar tipos —> es más seguro

```
> x = 3
> y = 4
> log (x + y)
[1] 1.94591
```

Values	
x	3
y	4

>?log —> se nos abre una ayuda

R: Logarithms and Exponentials Find in Topic

log {base} R Documentation

Logarithms and Exponentials

Description

`log` computes logarithms, by default natural logarithms, `log10` computes common (i.e., base 10) logarithms, and `log2` computes binary (i.e., base 2) logarithms. The general form `log(x, base)` computes logarithms with base `base`.

`log1p(x)` computes $\log(1+x)$ accurately also for $|x| \ll 1$.

`exp` computes the exponential function.

`expm1(x)` computes $\exp(x) - 1$ accurately also for $|x| \ll 1$.

Usage

```
log(x, base = exp(1))
```

Usage

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)

log1p(x)

exp(x)
expm1(x)
```

```
> log (10, base = 10)
[1] 1
> #sobreescribo
```

log (10,19) = log (10, base = 19)

Crear Hola Mundo

1. Creamos proyecto
2. Creamos un R Script —> botón verde +

hola_mundo.R Source on Save Run Source

```
nombre = "Paloma " #nombre <- "Paloma"
print(nombre)
```

```
> source("~/Desktop/2ndo_carrera/estadistica_I/archivos_estadistica/hola_mundo.R")
[1] "Paloma "
```

Ejecutar programa linea a linea: cmd + Enter

Vectorización

```
#calcular logaritmos de 1, 2, 3
log(3)
sum(my_vector)
log(3)

#las funciones están vectorizadas
my_vector= c(1,2,3) #c es de concatenar
log(my_vector)
```

```
> source("~/Desktop/2ndo_carrera/es
> log(my_vector)
[1] 0.0000000 0.6931472 1.0986123
```

```
> sum(my_vector)
[1] 6
```

R representa todo en vectores
[1] 1º posición <—> índice 1

```
#crear un vector de 1 a 100
my_long_vector = 1 : 1000
my_long_vector

#Crear solo numeros pares
seq(2, 10, by = 2)

#repetir 10 veces
rep ("Gauss" , 5)
```

EJERCICIO ESTADÍSTICA COVID

#' documentar el código

covid19.csv —> import datase

```
library(readr)
covid19 <- read_csv("covid19.csv")
View(covid19)
```

—> To Source (mayúscula + abajo)

(Librerías suelen ir arriba)

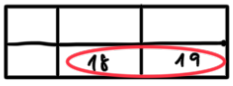
data.frame

data.frame → matriz con ≠ tipos de datos
int, char, string, ...

ACCEDER A UN DATO

→ covid19[2,2] ---> 18
fila columna

en R empezamos a contar en 1 (java → 0)



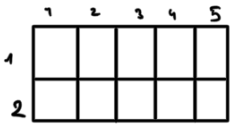
→ covid19[2, c(2,3)]

ACCEDER A UNA COLUMNA

covid19 [1 : nrow (covid19) , 2]

1 hasta el índice final — columna

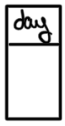
Sintactic Sugar : Forma Concisa de Hacer algo → covid19 [, 2]



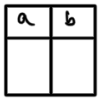
covid19 [2 , seq (1, 5 , by = 2)]

coger primeras 5 de dos en dos

Acceder por nombre



covid19 [, "day"]



covid19 [, c ("a", "b")]

covid19 $ day → R nos ayuda

INDEXAR UN VECTOR

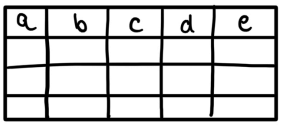
day = covid19 $ day

Es un vector

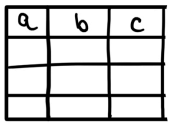
day [3 : 5] ---> 3 4 5

Indexado

covid 19



new covid



new_covid = covid19 [, c ("a", "b", "c")]

cambiar nombre columnas

actual_name = colnames (new_covid) — vector con nombre columnas

col_names (new_covid) = c ("A", "B", "C")

Cambiar solo la columna 6 y 7 —> `colnames(new_covid)[6:7]=c("countries", "cum_rate")`

NUEVA COLUMNA A PARTIR DE OTRAS

Ejemplo Tiempo

tiempo = day + (month - 1) * 30 + (year - 2019) * 365

objetivo :

day	month	year	Time

→ añado nueva columna : new_covid $ Time = new_covid $ day + (new - covid $ month...

DAY

→ Con Método with with (new-covid , day + (month - 1) * 30 + (year - 2019) * 365)

Permite especificar un data.frame

new-covid $ day

A veces es confuso

Borrar columna

- new_covid $ day = NULL
- new-covid [, c("time" , "month",...

no incluyendo columna day

- new_covid [, c(-1 , -2)]

↳ Borrar columnas 1 y 2

↳ R utiliza índices negativos para borrar

DIBUJAR GRÁFICA

notes = c(10, 9, 8, 4, 0)

↳ notes > 5 --> TRUE TRUE TRUE FALSE FALSE

↳ sum(notes > 5) --> 1 + 1 + 1 + 0 + 0 = 3 aprobados

↳ notes [c(TRUE, FALSE, TRUE, FALSE, TRUE)] --> 10 8 0

| OR
& AND
|| ó && no se utilizan en vectores

notas excepcionalmente altas y bajas: notes [(notes > 9) | (notes < 1)] → 10 0

Selecciona solo los datos q sean de España:

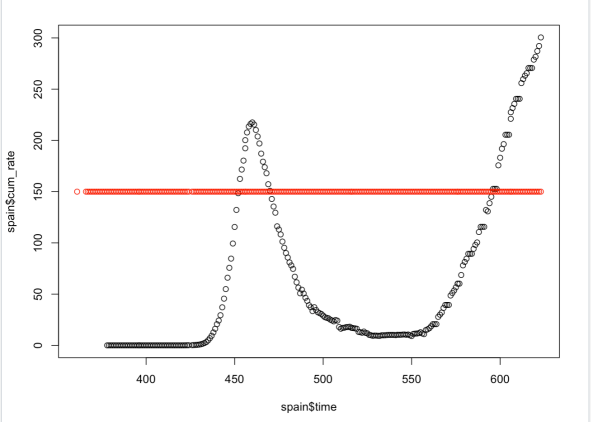
countries → Spain

new_covid $ countries == "Spain" devuelve lista TRUE y FALSE

Spain = new_covid [new_covid $ countries == "Spain" ,]

filas: solo TRUE

Columnas Todas



Función Plot

Dibuja que columna va en "x" y cual en "y"

plot (spain $ time , spain $ cum_rate)

x y

Función Points

Añadir otra curva al gráfico

constante = rep (150 , length (spain $ time)) Repite el número 150 hasta que termine spain $ time

points (spain $ time , constante , col = "red")

color

Más Países en la gráfica :

Spain = new_covid [new_covid $ countries == "Spain"]

plot (Spain $ time , Spain $ cum_rate , col = "red)

Portugal = new_covid [new_covid $ countries == "Portugal"]

points (Portugal $ time , Portugal $ cum_rate , col = "black ")

Cambiar a línea suave → plot (, , type = "l")

Título Gráfica → main = "Título"

Título eje x ó eje y → xlab = "eje x" ylab = "eje y"

Añadir límites → ylim = c(0, 500)

→ Leyenda:

Spain
Portugal

Spain
Portugal

Legend ("topleft", c("Spain", "Portugal"), col = c("black", "red"), lty = 1

Posición

Con muchos países:

countries = c ("Spain", "Portugal", "Venezuela", ...)

plot — points ...

1. Crear Variable ⇒ "Spain" = [Spain | | |]
   "Portugal" = [Portugal | | |]
   ...

2. Para el Primer country → Plot

3. Resto Countries → Points

```
i = 1
for (country in countries) {
  current_df = new_covid[new_covid$countries == country, ]
  if (country == countries[1]) { #españa --> plot
    plot(current_df$time, current_df$cumul_rate, col = i, type = "l")
  } else {
    points(current_df$time, current_df$cumul_rate, col = i, type = "l")
  }
  i = i + 1
```

countries = c ("Spain", "Portugal", "Venezuela", ...)

plot — points ...

1. Hacer función Plot → Spain

2. Filtrar por país → hacer Points

```
spain = new_covid[new_covid$countries == "Spain", ]
plot(spain$time, spain$cumul_rate, type = "l",
     xlab = "Time (days since 1/1/2019)",
     ylab = "Cumulative inf. rate",
     main = "Covid 19 pandemic",
     #ylim = c(0, 500)
)
i = 2
for (country in countries[-1]) {
  # filtrar por pais
  current_df = new_covid[new_covid$countries == country, ]
  points(current_df$time, current_df$cumul_rate, col = i, type = "l")
  i = i + 1
}
```

FUNCIONES

my_sum = function (x, y) {
  z = x+y
  z
}

my_sum (1, 2) → 1+2 = 3

funciones con parámetros predet.

my_sum = function (x, y = 10) {
  z = x+y
  z
}

my_sum (9) = 9+10
my_sum (9, 9) = 9+9

data.frame

```
count_by_country = function(df, country) {
  current_country = df[df$countries == country, ]
  sum(current_country$cases)
}
```

FUNCIÓN

Ejemplo: contar casos de cada pais

```
countries = c("Venezuela", "Spain", "Portugal", "Italy", "France", "Germany")
cases_by_country = c() #variable declarada para usarla en el bucle
  for (country in countries){
    current_cases = count_by_country(new_covid, country)
    cases_by_country = c(cases_by_country , current_cases)
  }
```

FOR ≃ FOR EACH!!
FOR ≃ FOR EACH!!
FOR ≃ FOR EACH!!
FOR ≃ FOR EACH!!

```
print(cases_by_country)
[1]  65174 640040  67176 294932 428696 270070
```

asigna a cada nº un nombre →

```
names(cases_by_country) = countries
cases_by_country
Venezuela     Spain  Portugal     Italy    France   Germany
    65174    640040     67176    294932    428696    270070
```

PROGRAMACIÓN FUNCIONAL

Programación Funcional → todo se hace con f(x)s bucle for → f(x)

sapply (countries, count_by_country)

→ Convierte en vector: un vector con una f(x) [a | b | c] —sapply→ [f(a) | f(b) | f(c)]

```
sapply(countries, count_by_country, df = new_covid)
 sapply(
  countries,
  function(country) {
    current_country = new_covid[new_covid$countries == country, ]
    sum(current_country$cases)},
    )
```

Variables GLOBALES ⇒ solo en f(x) LAMBDA
(ante la duda declaramos todos los elementos)

Como modificar variables

```
demo = function(x){
  x = 1
  x
}

x = 5
demo (x) #sale 1
print(x) #sale 5
```

Variable x=1 → solo dentro del bucle

java (PASO POR REFERENCIA): lo cambia dentro, lo cambia fuera

R (PASO POR COPIA): lo cabia dentro, NO cambia fuera