

Introducción a R

Directorio de trabajo

Se recomienda usar proyectos en RStudio para que establecer el directorio de trabajo.



- code**
- data**
- output**
- figures**

Pedir Ayuda

Acceder a los archivos de ayuda

?mean

Pedir ayuda para una función en particular.

help.search('weighted mean')

Buscar archivos de ayuda.

help(package = 'dplyr')

Pedir ayuda para un paquete.

Información de un objeto

str(iris)

Resumen de la estructura de un objeto.

class(iris)

Obtener la clase de un objeto.

Objetos básicos

Existen cuatro objetos básicos en R, conocidos también como objetos de primer orden.

Estos son la base de todos los objetos en R.

vectores

Secuencia de elementos homogéneos.
Tienen una sola dimensión – longitud

```
> a  
[1] 'perro'
```

NULL

Objeto nulo.
Tienen longitud cero.

```
> a  
[1] NULL
```

listas

Conjunto de objetos heterogéneos.
Tienen una sola dimensión – longitud

```
> a  
[[1]]  
[1] 'perro'
```

funciones

Trozos de código que realizan ciertas operaciones. Se componen de tres cosas, todas opcionales:
argumentos, código, producto.

```
>mean(x,...)
```

Tipos de vectores

Los tres tipos comunes de vectores en R. Siempre se puede ir de un tipo superior a uno inferior en la lista.

No se pueden combinar tipos de elementos en un vector.

lógico

TRUE, FALSE, TRUE

Valores Booleanos.

numérico

1, 0, 1

Enteros o de punto flotante.

caracter

'1', '0', '1'

Cadena de caracteres.

factor

'1', '0', '1',
levels: '1', '0'

Cadena de caracteres con niveles.

Hoja de trucos

Usar Paquetería

install.packages('dplyr')

Bajar e instalar un paquete de CRAN.

library(dplyr)

Cargar un paquete en la sesión actual para que sus funciones (y objetos) estén disponibles.

dplyr::select(...)

Usar una función de un paquete particular.

data(package = 'palmerpenguins')

Cargar datos incorporados en un paquete.

Algunas funciones para vectores

sort(x)

Ordenar **x**.

rev(x)

Invertir **x**.

table(x)

Contar valores de **x**.

unique(x)

Valores únicos de **x**.

duplicated(x)

Conocer duplicados de **x**.

order(x)

Conocer el orden de **x**.

is.na(x)

Conocer valores de **x == NA**.

which(x)

Conocer valores de **x == TRUE**.

Vector **x** reordenado.

Vector **x** invertido.

Array nombrado con valores del conteo.

Nuevo vector con valores únicos de **x**.

Vector lógico indicando duplicados en **x**.

Nuevo vector numérico con el orden en **x**.

Vector lógico indicando NAs en **x**.

Vector numérico indicando donde **x** es igual a TRUE.

Vectores

Creando vectores

c(2, 4, 6)

2 4 6

Concatenar elementos

2:6

2 3 4 5 6

Secuencia de enteros

seq(2, 3, by=0.5)

2.0 2.5 3.0

Secuencia compleja de 'from' a 'to' cada 'by'

rep(1:2, times=3)

1 2 1 2 1 2

Repetición de un vector

rep(1:2, each=3)

1 1 1 2 2 2

Repetición de elementos de un vector

Seleccionar elementos de un vector

Por posición

x[4]

Elemento cuatro

x[-4]

Todos menos el cuatro

x[2:4]

Elementos dos al cuatro

x[c(1, 5)]

Elementos un y cinco

Por valor

Con vectores lógicos

x[x == 10]

Elementos que son iguales a 10.

x[x %in% c(1, 2, 5)]

Elementos en el conjunto 1, 2, 5.

Por valor

Con vectores numéricos

x[which(x == 10)]

x[which(x < 10)]

Listas

Creando listas

l <- list(x = 1:5, y = c('a', 'b'))

Una lista puede contener elementos de distinto tipo.

Seleccionar elementos de una lista

l[[2]]

Selecciona y extrae los elementos seleccionados.

l[1]

Genera una nueva lista con elementos seleccionados.

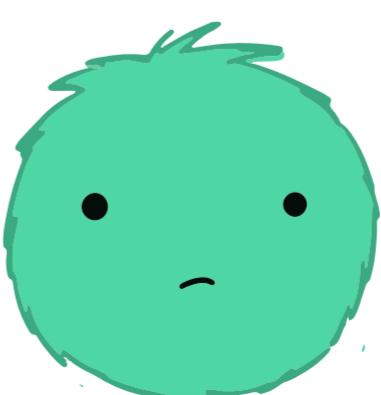
Este **l[[2]][1]** no es lo mismo que esto **l[2][1]**

Funciones sobre listas

lapply(x, FUN, ...)

Aplica sobre una lista **X** una función **FUN**.

lapply() es un caso especial de **apply()**.



Matrices

Creando matrices

```
m <- matrix(x, nrow = 3, ncol = 3)
```

Crear un matriz con vector `x`, que tiene que ser múltiplo de `nrow * ncol`.



`m[2,]` Selecciona renglón dos.



`m[, 1]` Selecciona columna uno.



`m[2, 3]` Selecciona elemento en renglón 2, columna 3.

Las matrices son vectores y por lo tanto sólo pueden ser de un solo tipo. Si se combinan tipos, se aplican las mismas reglas que para vectores:

lógico → numérico → carácter

Seleccionar elementos de una matriz

Por posición

`x[4,]` Renglón cuatro

`x[, -4]` Todos las columnas, menos la cuatro.

`x[2:4]` Elementos dos al cuatro

`x[c(1, 3)]` Elementos uno y tres

Cuando el resultado incluye un sólo elemento, una sola columna o un único renglón, el resultado es siempre un vector.

`x[1:4,] x[, 2:4]` Resulta un matriz.

Funciones sobre matrices

<code>t(x)</code>	<code>upper.tri(x)</code>
<code>diag(x)</code>	<code>lower.tri(x)</code>
<code>nrow(x)</code>	<code>dim(x)</code>

Data Frames

Creando data frames

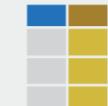
```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
```

Un data frame es un caso especial de un objeto tipo lista, donde todos sus elementos tienen la misma longitud.

Seleccionar elementos

Como matriz

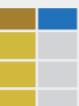
`df[, 2]`



Como lista

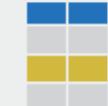
Como lista

`df[2,]`



Resultado es siempre un vector.

`df[2,]`

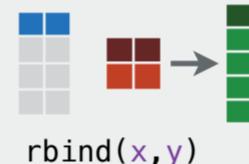


Resultado es siempre un data frame.

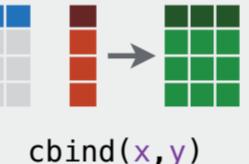
Funciones sobre data frames

`nrow(x)` `dim(x)` `ncol(x)`

Juntando data frames



`rbind(x,y)`



`cbind(x,y)`

Creamos un nuevo elemento del data frame mediante la asignación a un objeto NULL.

`df$x <- c(1:4)`

Ler y escribir archivos de datos

```
df <- read.table('file.txt')
```

```
write.table(df, 'file.txt')
```

Funciones genéricas para leer y escribir archivos de texto delimitados por caracteres particulares (delimitadores entre columnas).

```
df <- read.csv('file.csv')
```

```
write.csv(df, 'file.csv')
```

Caso especial de la función anterior. Para leer y escribir archivos de texto delimitados por comas.

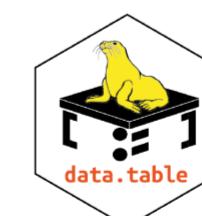
```
load('file.RData')
```

```
save(df, file = 'file.Rdata')
```

Cargar y guardar archivos binarios de R. Se usan para guardar objetos con estructuras más complejas que una tabla.



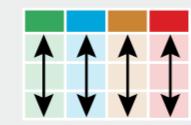
```
install.packages('readxl')
library(readxl)
read_xlsx("Archivo.xls")
```



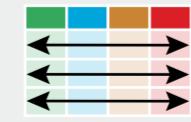
```
install.packages('data.table')
data.table::fread("Archivo.txt")
```



Datos prolíjos (tidy)



Cada variable se encuentra en su propia columna.



Cada observación se encuentra en su propio renglón.

`tibble(...)`
`as_tibble(...)`

Los datos 'tidy' se complementan con las operaciones vectorizadas de R. Automáticamente se preservan las observaciones cuando se manipulan variables y viceversa.

Se trabaja con una grámatica para manipulación de datos, con un conjunto de verbos para realizar operaciones.

```
x[which(x$esto == 10),]
x[, 'esto']
x$aquello <- c(TRUE)
table(x$esto)
```

```
filter(x, a == 10)
select(x, esto)
mutate(x, aquello = TRUE)
count(x, esto)
```

Programación

Ejecutar código de manera programática

For loops

```
for (vector in vector){
  x <- vector + 2
  cat(vector)
}
```

Asigna a un `vector` los elementos de un `vector` de manera iterada y ejecuta un `trozo` de código.

Ejemplo

```
for (i in 1:4){
  x <- i + 2
  print(x)
}
```

Asigna a un `i` los elementos de `1:4` de manera iterada y ejecuta el `trozo` de código.

Primera iteración:

```
i <- 1
x <- 1 + 2
print(x)
```

Segunda iteración:

```
i <- 2
x <- 2 + 2
print(x)
```

Tercera iteración:

```
i <- 3
x <- 3 + 2
print(x)
```

Cuarta iteración:

```
i <- 4
x <- 4 + 2
print(x)
```

While loops

```
while(condicion){
  x <- vector + 2
  cat(vector)
}
```

Ejecuta un `trozo` de código, mientras se cumpla una `condición`.

Declaraciones condicionales

```
if (condicion){
  trozo
} else {
  trozo
}
```

Si una `condición` se cumple, ejecuta un `trozo` de código, si no se cumple ejecuta otro `trozo` de código.