



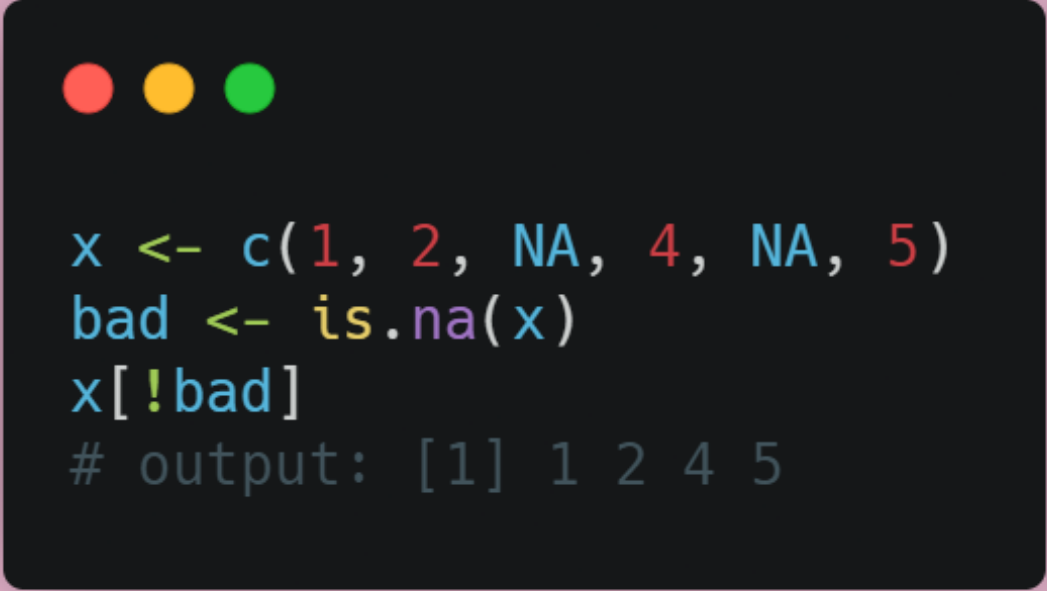
Valores NA, manejo de excepciones y leer Datos

Gutiérrez Cruz Abel Isaías

Remove values NA


Valores NA son valores faltantes en bases de datos y están denotados por NA or NaN para operaciones matemáticas sin definir.

Remove values NA de vectores



```
x <- c(1, 2, NA, 4, NA, 5)
bad <- is.na(x)
x[!bad]
# output: [1] 1 2 4 5
```

Remover valores teniendo en cuenta dos vectores



```
x <- c(1, 2, NA, 4, NA, 5)
y <- c("a", "b", NA, "d", NA, NA)
good <- complete.cases(x, y)

x[good]
# output: [1] 1 2 4
y[good]
# output: [1] "a" "b" "d"
```

Remover valores NA de un dataframe

```
> airquality[1:6, ]
#   Ozone Solar.R Wind Temp Month Day
#1   41    190   7.4  67    5    1
#2   36    118   8.0  72    5    2
#3   12    149  12.6  74    5    3
#4   18    313  11.5  62    5    4
#5   NA     NA  14.3  56    5    5
#6   28     NA  14.9  66    5    6
> good <- complete.cases(airquality)
> airquality[good, ][1:6, ]
#   Ozone Solar.R Wind Temp Month Day
#1   41    190   7.4  67    5    1
#2   36    118   8.0  72    5    2
#3   12    149  12.6  74    5    3
#4   18    313  11.5  62    5    4
#7   23    299   8.6  65    5    7
```

Descargar archivos

- Ayuda a la reproducibilidad de tu script
- Se utiliza la función "download.file()"
- Parámetros de la función "download.file()":
 - Url del sitio web en donde este alojado el archivo
 - Ubicación en la cual se quiera guardar el archivo (es necesario especificar la extensión del archivo)



```
url <- "direccion web"  
download.file(, destfile = "direccion/<nombre>.<extension del archivo>")
```

Leer archivos

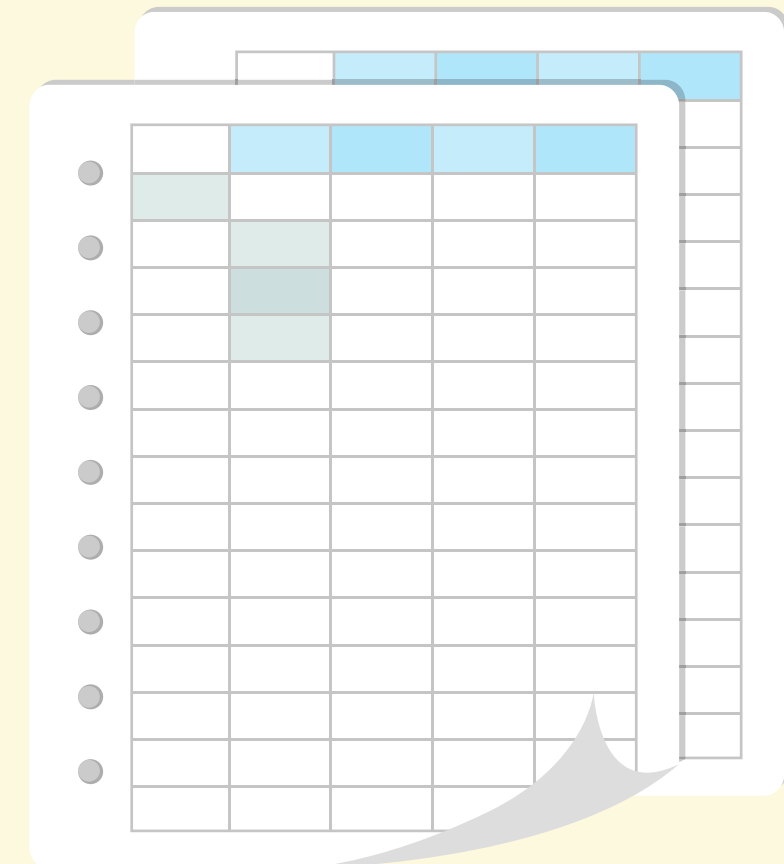
- Si tu archivo tiene la extensión .csv (datos separados por comas) es recomendable usar la función "read.csv()".
- Si tu archivo tiene otro formato, es recomendable usar la función "read.table()".

Parametros de la función read.table

- file: nombre del archivo o la conexión
- header: valor lógico indicando si el archivo tiene una línea de nombres de columnas o no
- sep: es un string indicando como están separadas las columnas
- colClasses: un vector de caracteres indicando que clase de valor guarda cada columna
- nrows: número de filas que guarda el dataframe
- comment.char: un string indicando el carácter de comentario
- skip: indicado por un número de líneas para saltar al inicio del documento
- na.strings: indica como están representados los valores faltantes

Leer archivos de excel

- Para leer archivos con extensión xlsx se usa la función "read.xlsx()". Antes de usarse esa función se tiene que cargar con la siguiente línea de código: "library(xlsx)".



Exploración de datos

Funciones

- head(): se obtienen las primeras cinco líneas de datos
- tail(): se obtienen las ultimas cinco líneas de datos
- summary(): resumen general del conjunto de datos a través de estadísticos como cuartiles, mínimo y máximo
- str(): se obtiene información de los datos de cada columna
- quantile(): se obtiene los cuartiles directamente. Se pueden especificar que probabilidad se desea. Parámetros de interés:
 - na.rm=TRUE: no toma en cuenta los datos de tipo NA
 - probs=c(<lista de probabilidades>)
- table(): determinar cuantas veces aparece cada elemento en el dataset

Funciones bucle

Se tratan de funciones repetitivas en estructuras de datos como vectores, matrices, o dataframes.

Lapply

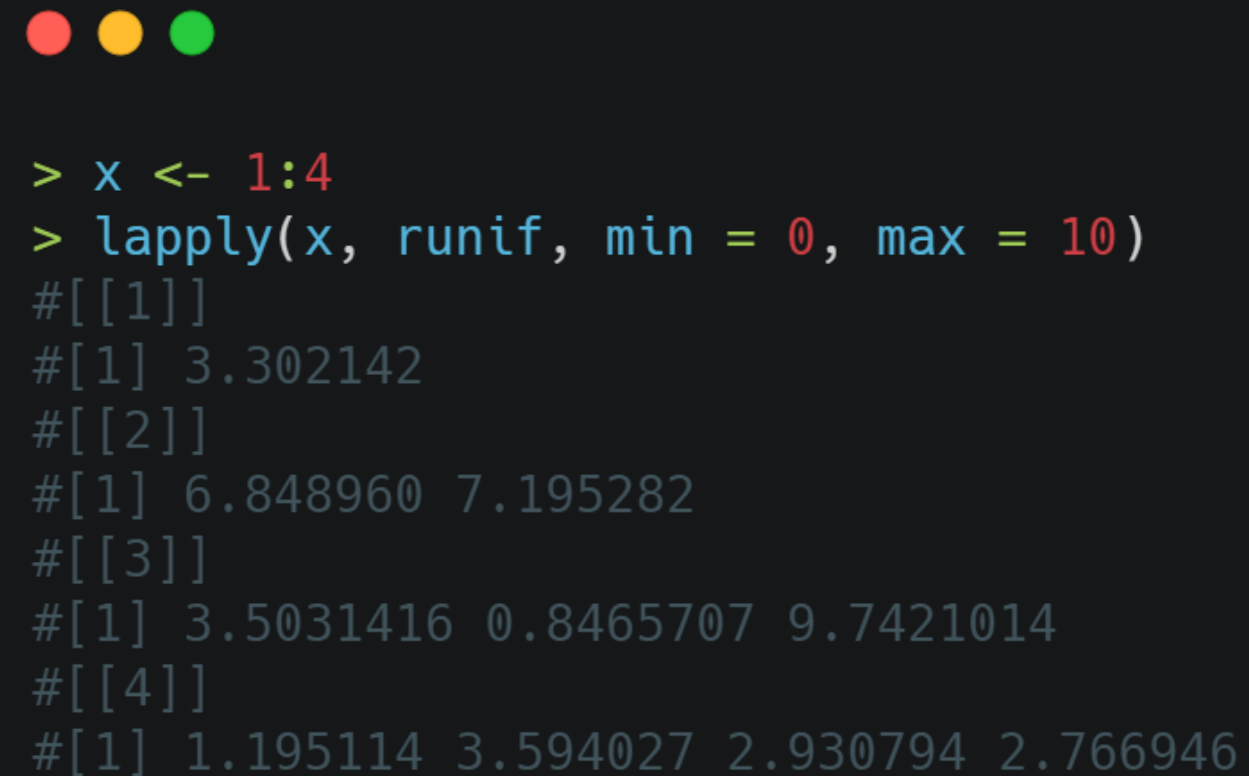
Necesita los siguientes parámetros:

- Una lista x
- Una función (o el nombre de la función)
- Otros argumentos para la función ingresada

Nota: Siempre regresa una lista

```
x <- list(a = 1:5, b = rnorm(10))
lapply(x, mean)
#$a
#[1] 3
#$b
#[1] 0.4671
```

Se agregan los parámetros de la función de la siguiente manera:



```
> x <- 1:4
> lapply(x, runif, min = 0, max = 10)
#[[1]]
#[1] 3.302142
#[[2]]
#[1] 6.848960 7.195282
#[[3]]
#[1] 3.5031416 0.8465707 9.7421014
#[[4]]
#[1] 1.195114 3.594027 2.930794 2.766946
```

La función `runif` genera cierto número de dígitos aleatorios según un máximo y un mínimo

Sapply

sapply tratará de simplificar el resultados de lapply si es posible.

- Si el resultado es una lista donde cada elemento tiene un longitud de 1, el vector es regresado
- Si el resultado es una lista donde cada elemento es un vector de la misma longitud (mayor a 1), una matriz es devuelta
- Si no se puede simplificar los objetos, una lista es devuelta

Apply

Es usado para evaluar una función (a menudo una función anónima) sobre las márgenes de un array

- Es comúnmente usado para aplicar una función a las filas o columnas de una matriz
- Puede ser usado en arrays, por ejemplo, tomar el promedio de un array de matrices
- No es más rápido que escribir un bucle, pero funciona en una sola línea.

Necesita los siguientes parámetros:

- Un array
- Margin, el cual es un número que indica que margen debe de ser usado (número 1 indica las filas y el número 2 indica las columnas)
- Una función para ser aplicada

Ejemplo

Cálculo de el promedio de los datos incluidos en cada una de las columnas:



```
x <- matrix(rnorm(200), 20, 10)
apply(x, 2, mean)
#[1] 0.04868268 0.35743615 -0.09104379
#[4] -0.05381370 -0.16552070 -0.18192493
#[7] 0.10285727 0.36519270 0.14898850
#[10] 0.26767260
```

Ejemplo

Cálculo de los cuartiles de los datos incluidos en cada una de las filas:



```
x <- matrix(rnorm(200), 20, 10)
apply(x, 1, quantile, probs = c(0.25, 0.75))
```

Mapply

Es una especie de aplicación multivariante que aplica una función en paralelo sobre un set de argumentos

Necesita los siguientes parámetros:

- Una función para aplicar
- Argumentos necesarios para la función anterior
- Otros argumentos para la función

Ejemplo

Forma tediosa de elaborar cuatro series de números



```
list(rep(1, 4), rep(2, 3), rep(3, 2), rep(4, 1))
```

Forma eficiente de elaborar cuatro series de números



```
mapply(rep, 1:4, 4:1)
#[[1]]
#[1] 1 1 1 1
#[[2]]
#[1] 2 2 2
#[[3]]
#[1] 3 3
#[[4]]
#[1] 4
```


Ejemplo vectorizar una función

Forma original de implementar una función:



```
rnorm(n = 1:5, mean = 1:5, sd = 2)
```

Vectorización de la función:



```
mapply(rnorm, 1:5, 1:5, 2)
```

Tapply

Es usada para aplicar una función sobre subsets de un vector

Necesita los siguientes parámetros:

- Un vector
- Un factor o una lista de factores
- Una función para ser aplicada
- Argumentos de la función que se quiere aplicar

Ejemplo

Calculo del promedio en tres categorías distintas dentro de un vector



```
# Creación de 30 elementos numericos
x <- c(rnorm(10), runif(10), rnorm(10, 1))
# Creación de 30 factores. 1, 2 y 3
f <- gl(3, 10)
# Calculo del promedio para cada factor
tapply(x, f, mean)
# 1 2 3
# 0.1144464 0.5163468 1.2463678
```

Split

Toma un vector u otros objetos y los separa en grupos de acuerdo con un factor o una lista de factores


Necesita los siguientes parámetros:

- Un vector, lista o dataframe
- Un factor o una lista de factores



```
# Creación de 30 elementos numericos
x <- c(rnorm(10), runif(10), rnorm(10, 1))
# Creación de 30 factores. 1, 2 y 3
f <- gl(3, 10)
# Separación del vector en grupos
split(x, f)
```

Calculo del promedio en cada uno de los elementos de la lista generada con ayuda de la función `lapply`



```
> lapply(split(x, f), mean)
#$ '1'
#[1] 0.1144464
#$ '2'
#[1] 0.5163468
#$ '3'
#[1] 1.246368
```

Separar en grupos un dataframe

Calculo del promedio en tres categorías distintas dentro de un vector



```
# separar dataframe conforme los valores de "Month"  
s <- split(airquality, airquality$Month)  
# obtener el promedio de una lista de columnas segun el mes  
lapply(s, function(x) colMeans(x[, c("Ozone", "Solar.R", "Wind")]))
```