**Sesión 9 *Loop function***

Contenido: Lappy, Sapply, Tapply, Split, Apply, Mapply

R. **Actividad 9**

Aplicaciones en R utilizando data frame de esta sesión

**EJEMPLO 1: apply**

la función apply aplica una función a todos los elementos de una matriz. apply tiene tres argumentos:

* X: Una matriz o un objeto que pueda coercionarse a una matriz, generalmente, un data frame.
* MARGIN: La dimensión (margen) que agrupará los elementos de la matriz X, para aplicarles una función. Son identificadas con números, 1 son renglones y 2 son colummnas.
* FUN: La función que aplicaremos a la matriz X en su dimención MARGIN. • DATA FRAME

df <- data.frame(x = 1:4, y = 5:8, z = 10:13)

df

## x y z

## 1 1 5 10

## 2 2 6 11

## 3 3 7 12

## 4 4 8 13

apply(X = df, MARGIN = 1, FUN = sum) *#Suma por filas*

## [1] 16 19 22 25

apply(df, 1, sum) *#Otra forma de sumar por filas*

## [1] 16 19 22 25

apply(df[c(1, 2), ], 1, sum) *#Sumar por filas de la fila 1 y la fila 2*

## 1 2

## 16 19

apply(df, 2, sum) *# Sumar por columnas*

## x y z

## 10 26 46

apply(df[, c(1, 3)], 2, sum) *#Sumar por columnas de la columna 1 y 3*

## x z

## 10 46

apply(df, c(1, 2), sum)

## x y z

## [1,] 1 5 10

## [2,] 2 6 11

## [3,] 3 7 12

## [4,] 4 8 13

apply(df, c(2, 1), sum)

## [,1] [,2] [,3] [,4]

## x 1 2 3 4

## y 5 6 7 8

## z 10 11 12 13

apply(df, 1, mean, na.rm = TRUE)

## [1] 5.333333 6.333333 7.333333 8.333333

• FUNCION

fun <- **function**(x, character = FALSE) {

**if** (character == FALSE) {

x ^ 2

} **else** {

as.character(x ^2)

}

}

apply(df, 1, fun)

## [,1] [,2] [,3] [,4]

## x 1 4 9 16

## y 25 36 49 64

## z 100 121 144 169

apply(df, 1, fun, character = TRUE)

## [,1] [,2] [,3] [,4]

## [1,] "1" "4" "9" "16"

## [2,] "25" "36" "49" "64"

## [3,] "100" "121" "144" "169"

apply(df, 2, fun)

## x y z

## [1,] 1 25 100

## [2,] 4 36 121

## [3,] 9 49 144

## [4,] 16 64 169

apply(df, c(1, 2), fun)

## x y z

## [1,] 1 25 100

## [2,] 4 36 121

## [3,] 9 49 144

## [4,] 16 64 169

f <- **function**(x) sum(exp(x))

apply(df, 1, f)

## [1] 22177.60 60284.96 163871.51 445448.95

apply(df, 2, f)

## x y z

## 84.79102 4629.43310 687068.79094

apply(df, 1:2, f)

## x y z

## [1,] 2.718282 148.4132 22026.47

## [2,] 7.389056 403.4288 59874.14

## [3,] 20.085537 1096.6332 162754.79

## [4,] 54.598150 2980.9580 442413.39

apply(df, 2, min)

## x y z

## 1 5 10

apply(df, 2, range)

## x y z

## [1,] 1 5 10

## [2,] 4 8 13

apply(df, 1, summary)

## [,1] [,2] [,3] [,4]

## Min. 1.000000 2.000000 3.000000 4.000000

## 1st Qu. 3.000000 4.000000 5.000000 6.000000

## Median 5.000000 6.000000 7.000000 8.000000

## Mean 5.333333 6.333333 7.333333 8.333333

## 3rd Qu. 7.500000 8.500000 9.500000 10.500000

## Max. 10.000000 11.000000 12.000000 13.000000

apply(df, 2, summary)

## x y z

## Min. 1.00 5.00 10.00

## 1st Qu. 1.75 5.75 10.75

## Median 2.50 6.50 11.50

## Mean 2.50 6.50 11.50

## 3rd Qu. 3.25 7.25 12.25

## Max. 4.00 8.00 13.00

ar <- array(data = 1:18, dim = c(3, 2, 3))

apply(ar, 3, sum)

## [1] 21 57 93

**EJEMPLO 2: tapply**

Esta función permite aplicar funciones a subconjuntos de datos de un data data frame o vectores. Su estructura esta dado por tapply(X, INDEX, FUN , …, simplify = TRUE). INDEX <- Es una lista de las categorías.

set.seed(2)

data\_set <- data.frame(precio = round(rnorm(25, sd = 10, mean = 30)),

tipo = sample(1:4, size = 25, replace = TRUE),

tienda = sample(paste("Tienda", 1:4),

size = 25, replace = TRUE))

head(data\_set)

## precio tipo tienda

## 1 21 2 Tienda 2

## 2 32 3 Tienda 3

## 3 46 4 Tienda 4

## 4 19 3 Tienda 4

## 5 29 1 Tienda 4

## 6 31 3 Tienda 4

precio <- data\_set$precio

tienda<- data\_set$tienda

tipo <- factor(data\_set$tipo,

labels = c("juguetes", "comida", "electrónica", "bebidas"))

precios\_medios <- tapply(precio, tipo, mean)

precios\_medios

## juguetes comida electrónica bebidas

## 39.50000 30.33333 32.20000 29.33333

class(precios\_medios)

## [1] "array"

precios\_medios[2]

## comida

## 30.33333

lista\_precios\_medios <- tapply(precio, tipo, mean, simplify = FALSE)

lista\_precios\_medios

## $juguetes

## [1] 39.5

##

## $comida

## [1] 30.33333

##

## $electrónica

## [1] 32.2

##

## $bebidas

## [1] 29.33333

lista\_precios\_medios$juguetes

## [1] 39.5

data\_set[1, 1] <- NA

data\_set[2, 3] <- NA

tapply(data\_set$precio, data\_set$tienda, mean)

## Tienda 1 Tienda 2 Tienda 3 Tienda 4

## 32.00000 NA 39.25000 33.14286

tapply(data\_set$precio, data\_set$tienda, mean, na.rm = TRUE)

## Tienda 1 Tienda 2 Tienda 3 Tienda 4

## 32.00000 33.50000 39.25000 33.14286

f <- **function**(x) mean(x, na.rm = TRUE)

tapply(data\_set$precio, data\_set$tienda, f)

## Tienda 1 Tienda 2 Tienda 3 Tienda 4

## 32.00000 33.50000 39.25000 33.14286

tapply(precio, list(tipo, tienda), mean)

## Tienda 1 Tienda 2 Tienda 3 Tienda 4

## juguetes 46 31.00000 49 36.66667

## comida 26 30.33333 39 NA

## electrónica 50 29.00000 32 25.00000

## bebidas 22 40.00000 20 36.00000

tapply(precio, list(tipo, tienda), mean, default = 0)

## Tienda 1 Tienda 2 Tienda 3 Tienda 4

## juguetes 46 31.00000 49 36.66667

## comida 26 30.33333 39 0.00000

## electrónica 50 29.00000 32 25.00000

## bebidas 22 40.00000 20 36.00000

**EJEMPLO 3: Sapply**

Esta función tiene un uso similar a la función lapply() ya que puede ser usada para data frame, vectores y listas pero se diferencia ya que esta imprime por consola un vector de valores y no una lista. La estructura está dada por sapply(x, FUN, …)

sapply(1:4, sqrt)

## [1] 1.000000 1.414214 1.732051 2.000000

sapply(1:4, **function**(i) sqrt(i))

## [1] 1.000000 1.414214 1.732051 2.000000

mi\_fun <- **function**(i) {

sqrt(i)

}

sapply(1:4, mi\_fun)

## [1] 1.000000 1.414214 1.732051 2.000000

out <- numeric(10)

**for** (i **in** 1:10) {

out[i] <- i ^ 2

}

out

## [1] 1 4 9 16 25 36 49 64 81 100

sapply(1:10, **function**(i) i ^ 2)

## [1] 1 4 9 16 25 36 49 64 81 100

Lista <- list(A = 1:5, B = 6:20, C = 1)

sapply(Lista, length)

## A B C

## 5 15 1

**EJEMPLO 4: sapply vs lapply**

La función lapply permite trabajar con vectores, listas o data frame. Se pueden aplicar operaciones por elementos las cuales con retornadas como una lista. Su estructura está dada por lapply(x, simplify=FALSE, FUN,…).

simplify=FALSE <- Le indica a R que no retorne los valores como una lista.

sapply(c(3, 5, 7), exp)

## [1] 20.08554 148.41316 1096.63316

lapply(c(3, 5, 7), exp)

## [[1]]

## [1] 20.08554

##

## [[2]]

## [1] 148.4132

##

## [[3]]

## [1] 1096.633

sapply(c(3, 5, 7), exp, simplify = FALSE)

## [[1]]

## [1] 20.08554

##

## [[2]]

## [1] 148.4132

##

## [[3]]

## [1] 1096.633

as.list(sapply(c(3, 5, 7), exp))

## [[1]]

## [1] 20.08554

##

## [[2]]

## [1] 148.4132

##

## [[3]]

## [1] 1096.633

simplify2array(lapply(c(3, 5, 7), exp))

## [1] 20.08554 148.41316 1096.63316

unlist(lapply(c(3, 5, 7), exp))

## [1] 20.08554 148.41316 1096.63316

mi\_lista <- list(A = c(1, 4, 6), B = c(8, NA, 9 , 5))

sapply(mi\_lista, sum)

## A B

## 11 NA

sapply(mi\_lista, sum, na.rm = TRUE)

## A B

## 11 22

matrix(1, ncol = 3, nrow = 3)

## [,1] [,2] [,3]

## [1,] 1 1 1

## [2,] 1 1 1

## [3,] 1 1 1

sapply(1:3, **function**(i) matrix(i, ncol = 3, nrow = 3))

## [,1] [,2] [,3]

## [1,] 1 2 3

## [2,] 1 2 3

## [3,] 1 2 3

## [4,] 1 2 3

## [5,] 1 2 3

## [6,] 1 2 3

## [7,] 1 2 3

## [8,] 1 2 3

## [9,] 1 2 3

sapply(1:3, **function**(i) matrix(i, ncol = 3, nrow = 3), simplify = "array")

## , , 1

##

## [,1] [,2] [,3]

## [1,] 1 1 1

## [2,] 1 1 1

## [3,] 1 1 1

##

## , , 2

##

## [,1] [,2] [,3]

## [1,] 2 2 2

## [2,] 2 2 2

## [3,] 2 2 2

##

## , , 3

##

## [,1] [,2] [,3]

## [1,] 3 3 3

## [2,] 3 3 3

## [3,] 3 3 3

sapply(1:3, **function**(i) matrix(i, ncol = 3, nrow = 3), simplify = FALSE)

## [[1]]

## [,1] [,2] [,3]

## [1,] 1 1 1

## [2,] 1 1 1

## [3,] 1 1 1

##

## [[2]]

## [,1] [,2] [,3]

## [1,] 2 2 2

## [2,] 2 2 2

## [3,] 2 2 2

##

## [[3]]

## [,1] [,2] [,3]

## [1,] 3 3 3

## [2,] 3 3 3

## [3,] 3 3 3

df <- trees

res <- data.frame()

**for**(i **in** 1:ncol(df)) {

**for** (j **in** 1:nrow(df)) {

res[j, i] <- df[j, i] \* 2

}

}

sapply(1:ncol(df), **function**(i) {

sapply(1:nrow(df), **function**(j) {

df[j, i] \* 2

})

})

## [,1] [,2] [,3]

## [1,] 16.6 140 20.6

## [2,] 17.2 130 20.6

## [3,] 17.6 126 20.4

## [4,] 21.0 144 32.8

## [5,] 21.4 162 37.6

## [6,] 21.6 166 39.4

## [7,] 22.0 132 31.2

## [8,] 22.0 150 36.4

## [9,] 22.2 160 45.2

## [10,] 22.4 150 39.8

## [11,] 22.6 158 48.4

## [12,] 22.8 152 42.0

## [13,] 22.8 152 42.8

## [14,] 23.4 138 42.6

## [15,] 24.0 150 38.2

## [16,] 25.8 148 44.4

## [17,] 25.8 170 67.6

## [18,] 26.6 172 54.8

## [19,] 27.4 142 51.4

## [20,] 27.6 128 49.8

## [21,] 28.0 156 69.0

## [22,] 28.4 160 63.4

## [23,] 29.0 148 72.6

## [24,] 32.0 144 76.6

## [25,] 32.6 154 85.2

## [26,] 34.6 162 110.8

## [27,] 35.0 164 111.4

## [28,] 35.8 160 116.6

## [29,] 36.0 160 103.0

## [30,] 36.0 160 102.0

## [31,] 41.2 174 154.0

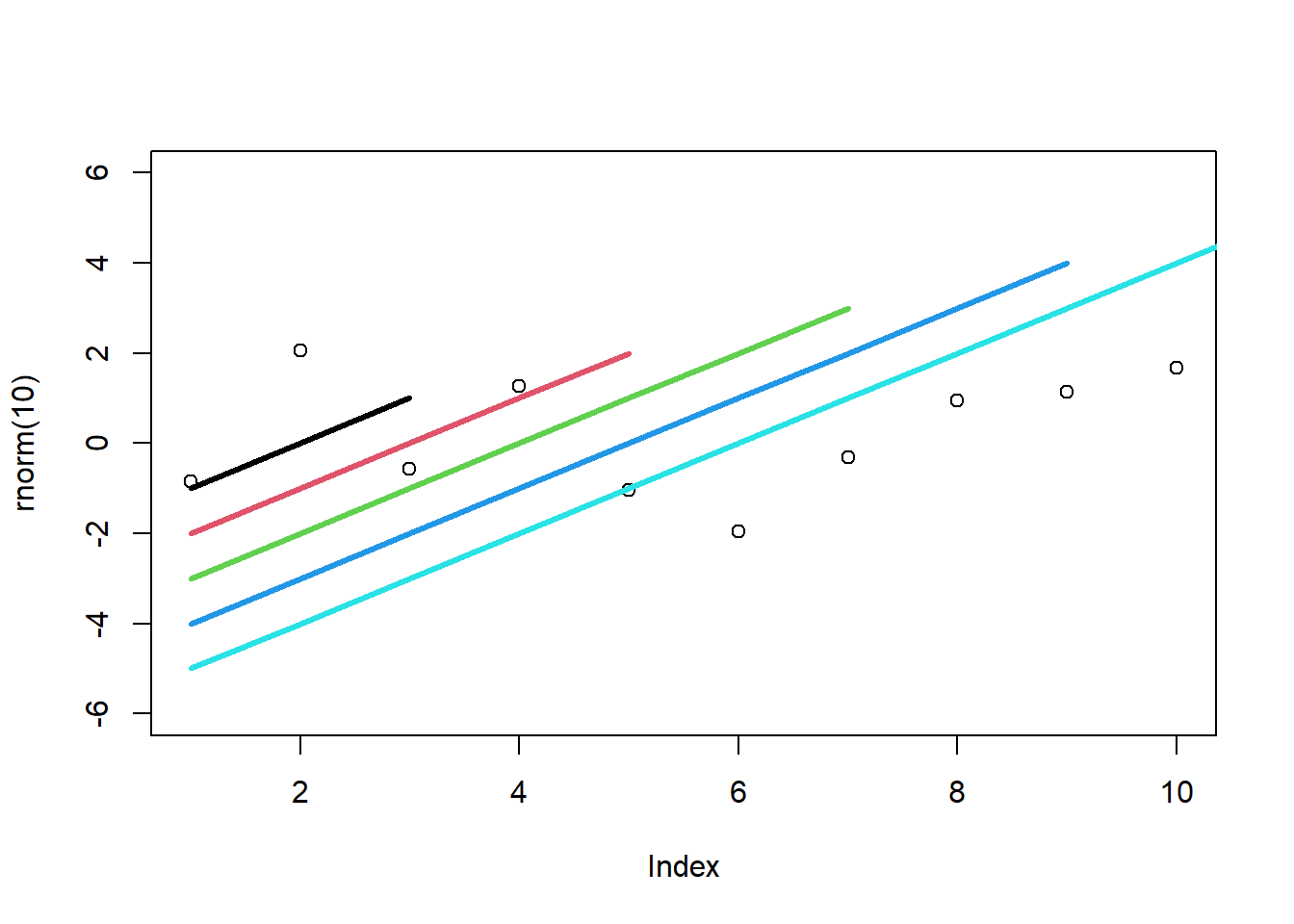
plot(rnorm(10), ylim = c(-6, 6))

nlines <- 5

**for** (i **in** 1:nlines) {

lines(-i:i, col = i, lwd = 3)

}



plot(rnorm(10), ylim = c(-6, 6))

nlines <- 5

invisible(sapply(1:nlines, **function**(i) lines(-i:i, col = i, lwd = 3)))

