

# Programación y Uso de Librerías en R: Herramientas de Análisis y Visualización de Datos

Juan Luis Peñaloza Figueroa  
Universidad Complutense de Madrid

Milagros Dones Tacero  
Universidad Autónoma de Madrid

Carmen Gladys Vargas Pérez  
Universidad Complutense de Madrid

AÑO: 2025

## SCRIPT\_1: CAPÍTULO 3: PROGRAMACIÓN BÁSICA

### III.1.1. Operadores de asignación

```
x <- 15
25 -> x
x=5
x <<- 2
```

### III.1.2. Operadores relacionales

```
"Olmedo de Riobamba" > "Macará de Ambato" # La comparación se
realiza por orden alfabético
```

```
"Ana" > "Ariana" #Se comparan las 2das letras
```

# Operadores relacionales

Relacional	Descripción
<	Menor que
>	Mayor que
==	Igual
<=	Menor o igual que
>=	Mayor o igual que
!=	Diferente de
%in%	Pertenece al conjunto
is.na	Es NA
!is.na	No es NA

### III.1.3. Operadores lógicos (Logical Operators)

```
##Verificar dos condiciones a la vez: 'a' es menor que 10 y 'a' es menor que 2
```

```
a <- 5
(a < 10) & (a < 2) # Devuelve FALSO
```

```
##Cumple una condición u otra condición: 'a' es menor que 10 o 'a' es menor que 2
```

```
(a < 10) | (a < 2) #Devuelve VERDADERO
```

# Negación en R

```
> !a < 10
```

### III.1.4. Operador de igualdad y desigualdad

```
## Devuelve una respuesta lógica: TRUE or FALSE
```

```
5 > 10
5 > 5
```

```
5 >= 5
```

### III.1.5. Operador de secuencias

```
> x<-2:8
```

### III.1.6. La función de concatenación

```
> x<-c(1,3,5)
> y<-c(2,4,6)
> c(x,y)
```

### III.1.8. Operador pipe (%>%) o pipeline

```
##Paquete magrittr y dplyr
```

```
> first(x) %>% second(x) %>% third(x)
```

Ejemplo: Tormentas

```
> dstorms<-c("Alberta","Alex","Allison","Ana","Arlene",
"Arthur")
> viento<-c(110,45,65,40,50,45)
> presion<-c(1007,1009,1005,1013,1010,1010)
> fecha<-c("12-08-2000","30-07-1998","04-06-1995","01-07-
1997","13-06-1999","21-06-1995")
> dstorms<-cbind(tormenta,viento,presion,fecha)
> dstorms

#Transformando la BD en una Data-frame
> dstorms<-data.frame(cbind(tormenta,viento,presion,fecha))
> dstorms

# Uso directo de la función filter()
> filter(dstorms, wind >= 50)

# Usando el operador %in% con la función filter()
> df=data.frame(x=c(12,31,10,2,99),
+               y=c(22.1,44.5,6.1,10,99),
+               z=c("Manzana","Guayaba","Mango",
"Manzana","Mango"))
> filter(df, z %in% c("Manzana","Mango"))

# Operador %in% junto con la función which()
> which(seq(1:10) %in% seq(4:12))

# operador %>%
> library(magrittr)
> diris<-data("iris")
> iris %>% head()

# Sintaxis para contar el número de filas de una Base de datos: lenght ()
> length(which(iris$Sepal.Length >= 4.5))
```

## III.2. OPERACIONES MATEMÁTICAS BÁSICAS EN R

# División entera para encontrar el valor entero de un cociente

```
> 12%/5
```

# Matrices y Escalares

```
> z <- matrix (c(1,2,3,4), nrow=2, ncol=2, byrow=T, dimnames =
list(c("X1", "X2"), c("Y1", "Y2")))
> z
```

# Añadir o modificar el nombre de filas y columnas de una matriz

```

> m <- matrix (1:10, nrow=5)
> m
> colnames(m) <- c("Variable1", "Variable2")
> rownames(m) <- c("obs1", "obs2", "obs3", "obs4", "obs5")

# Sumar dos matrices de igual dimensión
> H1<-matrix(1:9,nrow=3)
> H1
> H2<-matrix(9:1,nrow=3)
> H2
> H<-H1+H2
> H
# Trasponer una matriz
> t(H1)
# Multiplicación por un escalar
> x<-matrix(1:16, nrow=4, ncol=4,byrow=F)
> x
> 3*x
> y <-matrix(11:26, nrow=4, ncol=4,byrow=F)
> x%%y
#Producto exterior de dos vectores o matrices
#outer(x, y, FUN = "*")
> x%o%y
#Calcular el cuadrado de la matriz x
> x%^2
> library(matrixcalc)
> matrix.power(x,2)
# Determinantes de una matrix
> A<-matrix(c(10, 8, 5, 12), ncol = 2, byrow = TRUE)
> A
> det(A)
# Inversa de una matriz
> M<-solve(A)
> M
> A%%M
#Matriz identidad
> diag(3)
# Función apply()
> df <- data.frame(x = 10:13, y = 15:18, z = 20:23)
> df
> sumaf<-apply(df,1,sum)
> sumaf
> sumac<-apply(df,2,sum)
> sumac
# Obtener la media de las columnas y y z
> lapply(df[,c(2,3)], mean)

```

### III.6. OPERACIONES MATEMÁTICAS COMPLEJAS

```

# Derivadas
> install.packages("Deriv")
> library(Deriv)

```

```

> D <- function(x) x^2 + 3*x + 2
> derivada <- Deriv(D)
> print(derivada)

# Integrales
> install.packages("pracma")
> library(pracma)
> integral <- integral(fun = function(x) x^2, xmin = 0, xmax = 1)
> print(integral)

```

### III.6.2. PROGRAMACIÓN LÍNEAL

```

> install.packages("lpSolve")
> library(lpSolve)
#Función objetivo
Obj_fun<-c(3,2,5)

#Restricciones
rest<-matrix(c(1,2,1,3,0,2,1,4,0),nrow=3,byrow=T)
rest
direct<-c("<=", "<=", "<=")
rhs<-c(430,460,420)

## Solución
sol<-lp("max",Obj_fun, rest,direct, rhs,compute.sens=T)
sol
sol$status # 0 = éxitos; 2 = solución no factible
sol$duals
#Resumen
sol$objval
sol$objective
sol$solution
#Solución óptima
solopt<-sol$solution
names(solopt)<-c("x1", "x2", "x3")
solopt

# Programación entera
> solucion <- lp('max', coef, A, dir, b, all.int=TRUE)
> solucion$objval
> solucion$solution

```

```

# Problema de asignación
> cost <- matrix(c(15, 9, 10, 10, 15, 12, 9, 10, 8), nrow=3)
> lp.assign(cost)
> lp.assign(cost)$solution

```

### III.6.4. OPTIMIZACIÓN MATEMÁTICA

```

# Minimización de una función cuadrática con restricciones lineales
> fun_objetivo <- function(params)
{
  x <- params[1]  y <-
params[2]
  return((x - 3)^2 + (y - 4)^2)
}

```

```
# Restricciones lineales
A <- matrix(c(1, 1, -1, 0, 0, -1), nrow = 3, byrow = TRUE)
b <- c(10, 0, 0)
# Punto inicial (dentro de la región factible)
> punto_inicial <- c(0, 0)
# Optimización con restricciones
> result <- constrOptim(punto_inicial, funcion_objetivo, NULL,
ui = A, ci = b)
# Mostrar el resultado
> print(resultado$par)
```

### III.6.5. ECUACIONES DIFERENCIALES ORDINARIAS

```
> install.packages("deSolve")
> library(deSolve)

# Procedimiento de construcción del modelo
> modelo <- function(t, y, parms) {
list(-0.1 * y)
}
> solucion <- ode(y = 100, times = seq(0, 100, by = 1), func =
modelo, parms = NULL)
> plot(solucion)
```

### III.6.6. AUTOVALORES Y AUTOVECTORES EN R

```
> A<-matrix(c(10, 8, 5, 12), ncol = 2, byrow = TRUE)
> A
> eigen(A)$value
> eigen(A)$vectors
```

### III.6.7. DESCOMPOSICIÓN SINGULAR, QR Y DE CHOLESKY

```
#Descomposición en valor singular de una matriz
> MH1<-1/cbind(seq(1,3),seq(2,4),seq(3,5))
> MH1
# Descomposición en valor singular
> MH_svd<-svd(MH)
> MH_svd
# Descomposición QR
> MH_qr<-qr(MH)
> MH_qr
# Descomposición de Cholesky
> MH_chol<-chol(MH) #Esta es la matriz U, la matriz triangular superior
> MH_chol
```

### III.7. OPERACIONES MATEMÁTICAS AVANZADAS

```
> log(x = 8, base=2)
> log(x=3, base=10)
```

### III.8. OPERACIONES TRIGONOMÉTRICAS

```
# Medidas trigonométricas de vectores
> angulos <- c(0, pi/2, pi)
> sin(angulos)
```

```
#Teoría de conjuntos
> nombres <- sort(nombres)
> nombres
> n <- length(nombres)
> set.seed(123)
> futbol <- sort(sample(nombres, 10))
> baloncesto <- sort(sample(nombres, 8))
> voleybol <- sort(sample(nombres, 11))
> atletismo <- sort(sample(nombres, 7))
> ajedrez <- sort(sample(nombres, 10))

# Suma de conjuntos
> jueganFB <- union(futbol, baloncesto)
> jueganFBV <- union(jueganFB, voleybol)
```

----0000----