

Bloque 7: DataFrames y Análisis de Datos

El data frame es la estructura de datos más usada en análisis con R. Equivale a una tabla con filas (observaciones) y columnas (variables). Cada columna puede ser de un tipo diferente.

7.1 Crear y manipular data frames

```
# Crear un data frame
empleados <- data.frame(
  nombre = c("Ana", "Luis", "Maria", "Pedro"),
  edad   = c(28, 35, 42, 31),
  salario = c(28000, 45000, 55000, 38000),
  depto  = c("IT", "Ventas", "IT", "RRHH")
)

# Inspeccion basica
nrow(empleados)      # numero de filas
ncol(empleados)      # numero de columnas
dim(empleados)       # dimensiones
str(empleados)       # estructura
summary(empleados)   # estadisticas
head(empleados, 2)   # primeras 2 filas
```

7.2 Filtrado y selección

```
# Seleccionar columna
empleados$nombre
empleados[["salario"]]

# Filtrar filas
empleados[empleados$depto == "IT", ]
empleados[empleados$salario > 40000, ]

# Filtrar con multiple condicion
empleados[empleados$edad > 30 & empleados$depto == "IT", ]

# Seleccionar columnas especificas
empleados[, c("nombre", "salario")]
```

Ejercicio 20: Análisis de ventas con aggregate

A partir de un data frame de ventas por región y mes, calcula el total de ventas agrupando de distintas formas.

```
ventas <- data.frame(  
  Region = c("A", "B", "A", "B", "A"),  
  Mes     = c("Enero", "Enero", "Febrero", "Febrero", "Febrero"),  
  Ventas  = c(100, 150, 200, 120, 100)  
)  
  
# Total por mes  
aggregate(Ventas ~ Mes, data = ventas, FUN = sum)  
  
# Total por region y mes  
aggregate(Ventas ~ Region + Mes, data = ventas, FUN = sum)  
  
# Media por region  
aggregate(Ventas ~ Region, data = ventas, FUN = mean)
```

Explicación: aggregate() es el equivalente a GROUP BY de SQL. La fórmula Ventas ~ Region significa 'Ventas según Region'. FUN acepta cualquier función: sum, mean, max, sd...

Ejercicio 21: DataFame con datos de videojuegos

Carga un CSV de videojuegos y responde: cuantos juegos hay de GameCube o PSP, y cuantos juegos de Switch tienen meta_score entre 50 y 70.

```
# Cargar el archivo  
ruta <- file.path("datos", "videojuegos.csv")  
df <- read.csv(ruta)  
  
# Cuantos juegos hay de GameCube o PSP  
nrow(df[df$platform %in% c("GameCube", "PSP"), ])  
  
# Juegos de Switch con meta_score entre 50 y 70  
nrow(df[  
  df$platform == "Switch" &  
  df$meta_score >= 50 &  
  df$meta_score <= 70,  
)  
  
# Ver los nombres de los juegos que cumplen  
df[df$platform == "Switch" & df$meta_score >= 50 &  
  df$meta_score <= 70, "name"]
```

Explicación: read.csv() importa ficheros CSV. %in% equivale a múltiples OR: x %in% c('A','B') es mas legible que x=='A' | x=='B'. nrow() cuenta las filas del filtro.

Ejercicio 22: DataFame avanzado: ordenar y exportar

Genera un data frame aleatorio de 100 ventas, calcula totales por región y mes, ordena los resultados y exporta a CSV.

```
set.seed(10)
regiones <- c("Norte", "Sur", "Este", "Oeste")
meses    <- c("Enero", "Febrero", "Marzo", "Abril", "Mayo")

datos <- data.frame(
  Region = sample(regiones, 100, replace = TRUE),
  Mes    = sample(meses, 100, replace = TRUE),
  Ventas = sample(100:500, 100, replace = TRUE)
)

# Factor con orden logico de meses
datos$Mes <- factor(datos$Mes, levels = meses)

# Total por region y mes, ordenado
resumen <- aggregate(Ventas ~ Region + Mes,
                      data = datos, FUN = sum)
resumen <- resumen[order(resumen$Region, resumen$Mes), ]

print(resumen)

# Exportar a CSV
write.csv(resumen, "resumen_ventas.csv", row.names = FALSE)
```

Explicación: factor() con levels= establece el orden lógico (meses calendario). order() devuelve los índices ordenados. write.csv() exporta el resultado para usar en Excel u otras herramientas.

Bloque 8: Visualización con Gráficos

R incluye un potente sistema de gráficos base (funciones plot, pie, barplot, hist...) y el paquete ggplot2 para visualizaciones más avanzadas y elegantes.

8.1 Gráficos base

```
# Histograma
set.seed(1)
datos <- rnorm(500)
hist(datos, main = "Distribucion Normal",
      xlab = "Valor", col = "steelblue", border = "white")

# Grafico de lineas
x <- 1:10
y <- x^2
plot(x, y, type = "l", col = "red", lwd = 2,
     main = "y = x^2", xlab = "x", ylab = "y")

# Diagrama de dispersion
plot(cars$speed, cars$dist, pch = 19, col = "blue",
     main = "Velocidad vs Distancia de frenado")
abline(lm(dist ~ speed, data = cars), col = "red")
```

Ejercicio 23: Grafico de sectores (pie)

Crea un grafico de sectores con los goles de tres equipos de futbol. Incluye porcentajes y leyenda.

```
equipos <- c("Cadiz", "Ponferrada", "Alcorcon")
goles <- c(10, 5, 2)

# Calculamos porcentajes
pct <- round(goles / sum(goles) * 100, 1)
etiquetas <- paste0(equipos, "\n", pct, "%")

# Grafico
pie(goles,
    labels = etiquetas,
    col = rainbow(length(goles)),
    main = "Distribucion de Goles",
    cex = 1.2)

legend("topright", legend = equipos,
      fill = rainbow(length(goles)))
```

Explicación: pie() acepta un vector de valores numéricos. Labels son las etiquetas. rainbow(n) genera n colores del arco iris. legend() anade la leyenda en la posición indicada.

Ejercicio 24: Gráfico de barras comparativo

Gráfico de barras agrupadas comparando ventas mensuales de dos regiones.

```
meses    <- c("Ene", "Feb", "Mar", "Abr", "May")
region_a <- c(120, 150, 180, 130, 200)
region_b <- c(90, 170, 140, 160, 210)

# Matriz: filas = regiones, columnas = meses
datos <- rbind(region_a, region_b)
colnames(datos) <- meses

barplot(datos,
        beside = TRUE,
        col    = c("steelblue", "coral"),
        main   = "Ventas mensuales por Region",
        ylab   = "Ventas (euros)",
        legend  = c("Region A", "Region B"),
        args.legend = list(x = "topright"))
```

Explicación: `rbind()` combina vectores como filas de una matriz. `barplot()` con `beside=TRUE` pone las barras una al lado de la otra en vez de apiladas. `legend=` dentro de `barplot()` lo gestiona automáticamente.

Ejercicio 25: Dashboard de análisis con gráficos múltiples

Crea un panel de 4 gráficos (2x2) que analicen en conjunto un conjunto de datos de ventas.

```
set.seed(42)
n      <- 200
ventas <- rnorm(n, 5000, 1000)
region <- sample(c("Norte", "Sur", "Este", "Oeste"), n, TRUE)
df      <- data.frame(ventas, region)

# Panel 2x2
par(mfrow = c(2, 2), mar = c(4, 4, 3, 1))

# 1. Histograma
hist(df$ventas, col = "steelblue", border = "white",
     main = "Distribucion de Ventas", xlab = "Euros")

# 2. Boxplot por region
boxplot(ventas ~ region, data = df,
       col = rainbow(4), main = "Ventas por Region")

# 3. Grafico de medias
medias <- tapply(df$ventas, df$region, mean)
barplot(medias, col = "coral", main = "Media por Region",
       ylab = "Media ventas")

# 4. Pie de distribucion de registros
conteo <- table(df$region)
pie(conteo, col = rainbow(4),
   main = "Registros por Region")

par(mfrow = c(1, 1)) # restaurar layout
```

Explicación: `par(mfrow=c(filas, columnas))` divide la ventana grafica en una cuadrícula. `tapply()` es como `aggregate()` pero devuelve un array. `table()` cuenta frecuencias. Recuerda restaurar `par(mfrow=c(1,1))` al terminar.

Resumen: Funciones Clave de R

Categoría	Funciones	Para que sirven
Vectores	<code>c()</code> , <code>seq()</code> , <code>rep()</code> , <code>1:n</code>	Crear y generar vectores
Estadística	<code>mean()</code> , <code>median()</code> , <code>sd()</code> , <code>var()</code>	Medidas de centralización y dispersión
Texto	<code>nchar()</code> , <code>substr()</code> , <code>paste()</code> , <code>gsub()</code>	Manipular cadenas de texto
Lógica	<code>any()</code> , <code>all()</code> , <code>which()</code> , <code>%in%</code>	Condiciones sobre vectores
DataFrames	<code>read.csv()</code> , <code>aggregate()</code> , <code>order()</code>	Importar y resumir datos
Funciones	<code>function()</code> , <code>return()</code> , <code>Vectorize()</code>	Definir y vectorizar funciones
Gráficos	<code>plot()</code> , <code>hist()</code> , <code>pie()</code> , <code>barplot()</code>	Visualización básica
Aleatorio	<code>rnorm()</code> , <code>runif()</code> , <code>sample()</code> , <code>set.seed()</code>	Generar datos aleatorios