ÍNDICE

Primer Bloque. Contextualización y Consideraciones Iniciales.. 13

Tema 1: De la Estadística a Data Science y Big Data 13

1. Métodos Estadísticos 14
1.1. Definición y clasificación de la Estadística 14
1.2. Conceptos estadísticos fundamentales 15
1.3. La Estadística oficial en España y Europa 17

2. Data Science y Big Data. La Nueva Realidad 19
2.1. Conceptos clave 22
2.2. Minería de Datos o Data Mining 26
2.3. Modelos SEMMA y CRISP-DM 26
2.3.1. Modelo SEMMA 27
2.3.2. Modelo CRISP-DM 28
2.3.3. Diferencias entre SEMMA y CRISP-DM 29
2.4. Principales métodos y algoritmos en la Minería de Datos 31

3. Big Data 34
3.1. Desafíos 34
3.1.1. Open Data 35
3.1.2. Small Data y Smart Data 35
3.1.3. No es oro todo lo que reluce 36
3.1.4. Consideraciones legales básicas 37
3.2. Aplicaciones 38
3.3. Principales herramientas 40
3.3.1. Hadoop y MapReduce 40
3.3.2. Spark 41

4. Programas de Software más Utilizados 42
4.1. R y RStudio 43
4.1.1. Consola R-Studio 44
4.1.2. R Markdown 45
4.1.3. Librerías en el programa estadístico R 48
4.2. Weka 53
4.3. Introducción al lenguaje de programación Python 60
4.4. El programa IBM SPSS Statistics: conexión con R y Python 64
4.5. Otros softwares: Julia y Scala 67
4.5.1. Julia 67
4.5.2. Scala 68

Tema 2: Introducción al Lenguaje R 69

1. Introducción a R y Ayuda en Línea 70

2. Objetos en R 71
2.1. Vectores 72
2.1.1. Crear 72
2.1.2. Seleccionar elementos 74
2.1.3. Trabajar con vectores 75

2.2. Matrices ... 76
2.2.1. Crear ... 76
2.2.2. Seleccionar elementos ... 79
2.2.3. Trabajar con matrices ... 80
2.3. Listas ... 80
2.3.1. Crear ... 81
2.3.2. Seleccionar elementos ... 81
2.3.3. Manipular elementos ... 82
2.3.4. Unir listas ... 83
2.3.5. Convertir lista en vector ... 83
2.4. Factores ... 83
2.5. Dataframes ... 84
2.5.1. Crear ... 84
2.5.2. Seleccionar elementos ... 86
2.5.3. Manipular dataframes ... 86
2.5.4. Importar/Exportar datos externos ... 88

3. Realizar Consultas ... 90
3.1. Condiciones simples ... 90
3.2. Condiciones múltiples ... 92
3.3. Filtrar dataframes ... 93

4. Funciones ... 94
4.1. Definir una función ... 94
4.2. Funciones incorporadas ... 95
4.3. Funciones definidas por el usuario ... 96

5. Estructuras de Control ... 97
5.1. Sentencias condicionales ... 97
5.1.1. Sentencia if ... 97
5.1.2. Sentencia if…else ... 97
5.1.3. Sentencias anidadas ... 98
5.2. Bucles ... 99
5.2.1. Bucle For ... 99
5.2.2. Bucle While ... 100

6. La Familia de Funciones Apply ... 101
6.1. Las funciones apply vs. bucles ... 103

7. Gráficos ... 104

Segundo Bloque. Métodos Estadísticos Multivariantes ... **115**

Tema 3: Modelo Lineal General y Modelo Lineal Generalizado ... 115

1. Modelo Lineal General ... 116
1.1. Modelo de Regresión Lineal ... 116
1.1.1. Introducción ... 116
1.1.2. Modelo de Regresión Lineal Simple ... 116
1.1.3. Modelo de Regresión Lineal Múltiple ... 118
1.1.4. Propiedades estadísticas del estimador MCO ... 120
1.1.5. Coeficiente de determinación ... 121
1.1.6. Inferencia acerca de los estimadores ... 122

1.1.7. Predicción 125
1.1.8. Estimación del modelo de regresión con R 126
1.2. Extensiones al Modelo de Regresión Lineal 129
1.2.1. Introducción 129
1.2.2. Heterocedasticidad 131
1.2.3. Autocorrelación 133
1.2.4. Deficiencias muestrales 135
1.2.5. Errores de especificación 136
1.2.6. Métodos de selección de variables en el modelo lineal general 136
1.3. Modelos con variables cualitativas explicativas 138
1.3.1. Introducción 138
1.3.2. Modelos ANOVA: efectos fijos 139
1.3.3. Modelos de componentes de la varianza: efectos aleatorios 160
1.3.4. Modelos anidados o jerárquicos 163
1.3.5. Modelos ANCOVA 168
1.4. Modelos con variable dependiente multivariante: MANOVA y MANCOVA 172
1.4.1. Definición del contraste 172
1.4.2. Supuestos para su aplicación 172
1.4.3. Estadísticos 173
1.4.4. Interpretación del test 173
1.4.5. Cálculo en R 173
1.5. Estimación por Máxima Verosimilitud Restringida (REML) en modelos mixtos 175
1.6. Ajuste de modelos mixtos con R 176
1.6.1. Función lme() del paquete nlme 176
1.6.2. Función lmer() del paquete lme4 177
1.6.3. Ejemplos de modelos con R 177

2. Modelo Lineal Generalizado 193
2.1. Formulación general 193
2.2. Modelos con variables cualitativas endógenas 199
2.2.1. Modelo probabilístico lineal 199
2.2.2. Modelo Logit 200
2.2.3. Modelo Probit 204
2.2.4. Modelo Logit vs Modelo Probit 207
2.3. Modelo Tobit 209

3. Evaluación de Modelos 210
3.1. Devianza. Estadístico G2 de Wilks de razón de verosimilitudes 211
3.2. Estadístico $\chi 2$ de Pearson 212
3.3. Criterio de Información de Akaike (AIC) y Criterio de Información Bayesiano (BIC) 213
3.4. Prueba de Hosmer-Lemeshaw 216
3.5. Medidas tipo $R2$ 217
3.5.1. Pseudo R^2 de McFadden (McFadden, 1974) 217
3.5.2. Pseudo R^2 de Cox-Snell (Cox & Snell, 1989) 217
3.5.3. Pseudo R^2 de Nagelkerke (Nagelkerke, 1991) 218
3.6. Métodos específicos para modelos de clasificación 219
3.6.1. Métodos basados en métricas 219
3.6.2. Métodos basados en la curva ROC 220
3.6.3. Métodos basados en una matriz de costes 221

Tema 4: Métodos Estadísticos de Reducción de Dimensiones 225

1. Análisis Factorial y Componentes Principales 226
1.1. Introducción 226
1.2. Análisis Factorial vs Componentes Principales 227
1.3. Análisis de Componentes Principales 228
1.4. Análisis Factorial 233
1.4.1. Planteamiento 234
1.4.2. Hipótesis en el Modelo Factorial 235
1.4.3. Comunalidad y especificidad (unicidad) 235
1.4.4. Diseño del análisis 235
1.4.5. Extracción de los factores 237
1.4.6. La matriz factorial o de componentes 238
1.4.7. Autovalores o valores propios 238
1.4.8. Número de factores a conservar 239
1.4.9. Rotación de los factores 240
1.4.10. Puntuaciones factoriales 242
1.4.11. Interpretación de los factores 242
1.4.12. Casos de Heywood y otras anomalías sobre estimaciones de comunalidad 243
1.4.13. Ejemplos con R 244

2. Análisis de Correspondencias 249
2.1. Introducción 249
2.2. Objetivo 250
2.3. Análisis de Correspondencias Simple 250
2.3.1. Planteamiento 250
2.3.2. Definición de perfiles 251
2.3.3. Medida de distancia utilizada 252
2.3.4. Extracción de las dimensiones o espacios factoriales 253
2.3.5. Método de normalización 255
2.3.6. Interpretación de resultados 255
2.4. Análisis de Correspondencias Múltiple 257
2.4.1. Planteamiento 257
2.4.2. Nube de puntos, perfiles 259
2.4.3. Inercia 259
2.4.4. Solución del Análisis de Correspondencias 260
2.4.5. Interpretación de los resultados 260
2.5. Ejemplo de Análisis de Correspondencias Simple con el software R 260
2.5.1. Análisis exploratorio 261
2.5.2. Estimación del modelo 262
2.5.3. Valores propios 263
2.5.4. Biplot simétrico 264
2.5.5. Análisis de perfiles fila 264
2.5.6. Análisis de perfiles columna 271
2.5.7. Biplots asimétricos 273
2.5.8. Biplot de contribución 275
2.5.9. Descripción de la dimensión 276
2.6. Ejemplo de Análisis de Correspondencias Múltiple en R 277
2.6.1. Análisis exploratorio 277
2.6.2. Estimación del modelo 278
2.6.3. Valores propios 279
2.6.4. Biplot simétrico 280

2.6.5. Análisis de las variables 280
2.6.6. Análisis de los individuos 287
2.6.7. Coloreando individuos por grupos 288
2.6.8. Descripción de la dimensión 290
2.6.9. Individuos y variables suplementarias 291
2.6.10. Filtrado de resultados 294

Tema 5: Medidas de Distancias y Agrupamiento 297

1. Medidas de distancia/proximidad 298
1.1. Medidas de distancia o disimilaridad 298
1.1.1. Escala de intervalo 298
1.1.2. Frecuencias 299
1.1.3. Datos binarios 299
1.2. Medidas de proximidad o similaridad 300
1.2.1. Escala de intervalo 301
1.2.2. Datos binarios 301
1.3. Distancia de Mahalanobis 304
1.3.1. Distancia euclídea normalizada 304
1.3.2. Definición y propiedades de la distancia de Mahalanobis 304
1.3.3. Distancias singulares 305

2. Agrupamiento de la Información 306
2.1. Análisis Discriminante 306
2.1.1. Clasificación con dos grupos 306
2.1.2. Clasificación con más de dos grupos 310
2.1.3. Ejemplos con el software R 310
2.2. Análisis Clúster 312
2.2.1. Introducción 312
2.2.2. Etapas a seguir en el desarrollo del Análisis Clúster 313
2.2.3. Modelos jerárquicos 314
2.2.4. Modelos no jerárquicos 320
2.3. Escalamiento Multidimensional 322
2.3.1. Modelo general o método clásico 322
2.3.2. Otros modelos de escalamiento 325
2.3.3. Relación con otras técnicas multivariantes 331
2.4. Análisis de Correlación Canónica 331
2.4.1. Introducción 331
2.4.2. Modelo 331
2.4.3. Interpretación de resultados 333
2.4.4. Ejemplo en R 334

Tercer Bloque. Introducción al Machine Learning 339

Tema 6: Regresión y Clasificación: Árboles de Decisión y Redes Neuronales 339

1. Uso de Muestras para el Entrenamiento, Validación y Test 340
1.1. Muestras de entrenamiento, validación y test 340
1.2. Validación cruzada 341

2. Árboles de Decisión y Clasificación 342
2.1. Introducción 342
2.2. Aplicabilidad de los árboles de decisión para clasificación 345
2.3. Características de los algoritmos de clasificación 345
2.3.1. Particiones posibles y criterios de selección 346
2.3.2. Ganancia de información 347
2.3.3. El criterio de proporción de ganancia 347
2.3.4. Índice de diversidad de Gini 347
2.3.5. Otros criterios de selección 348
2.3.6. Poda en Árboles de clasificación 349
2.4. Árbol CHAID (CHi-square Automatic Interaction Detection) y CHAID exhaustivo. 351
2.5. Árbol CRT (Classification and Regression Trees) 356
2.6. Árbol QUEST (Quick, Unbiased, Efficient Statistical Tree) 357
2.7. Árbol C5.0 360
2.8. Otros algoritmos de clasificación 362
2.8.1. Algoritmo de construcción de árboles consolidados 362
2.8.2. Random Forest 362
2.8.3. Decision Stum 362
2.9. Árboles de decisión con R 363
2.9.1. Conditional Inference Tree 364
2.9.2. Recursive Partitioning and Regression Trees 365
2.9.3. CHAID (CHi-square Automatic Interaction Detection) 365
2.9.4. Árbol C5.0 de Quinlan 366
2.9.5. Random Forest 370
2.9.6. Árboles con caret (ejemplo de Random Forest) 370
2.9.7. Árboles con Rweka 371

3. Redes Neuronales Artificiales 375
3.1. Introducción 375
3.2. Tipos de modelos de redes neuronales 376
3.3. Unidades de procesamiento de la información 378
3.4. Propiedades de los sistemas neuronales 380
3.5. Perceptrón multicapa 381
3.5.1. Etapa de funcionamiento 381
3.5.2. Etapa de aprendizaje 382
3.5.3. Metodología de aplicación de un perceptrón multicapa 385
3.5.4. Evaluación del rendimiento del modelo 386
3.6. Funciones de base radial 387
3.7. Comparación entre las Funciones de Base Radial y el Perceptrón Multicapa 390
3.8. Análisis de sensibilidad e interpretación de los pesos de la red 390
3.8.1. Análisis basado en la magnitud de los pesos de la red 391
3.8.2. Análisis de sensibilidad 391
3.9. Redes neuronales y modelos estadísticos clásicos 395
3.10. Otras arquitecturas de redes neuronales 397
3.11. Librería R Weka con redes neuronales 398
3.11.1. Análisis con base de datos German Credit 400
3.11.2. Análisis con base de datos Boston Housing 401

Tema 7: Exploración y Preprocesado de los Datos 403

1. Introducción: Fases Metodológicas de un Proceso de Data Science 404

2. Imputación de Datos Ausentes 406

3. Filtrado y Eliminación de Valores Extremos u Outlier 417

4. Transformación de la Base de Datos 422

4.1. Discretización de variables 423

5. Balanceo de las Clases 426

6. Reducción de Variables o de la Dimensionalidad 428

6.1. Aproximación indirecta o filter 429
6.2. Aproximación directa o wrapper (envoltura) 430
6.3. Selección de variables con la librería caret de R 431
6.4. Selección de variables con el programa WEKA 433

Bibliografía 439

TEMA 2: INTRODUCCIÓN AL LENGUAJE R

ÍNDICE

1. Introducción a R y Ayuda en Línea 70

2. Objetos en R 71
2.1. Vectores 72
2.2. Matrices 76
2.3. Listas 80
2.4. Factores 83
2.5. Dataframes 84

3. Realizar Consultas 90
3.1. Condiciones Simples 90
3.2. Condiciones Múltiples 92
3.3. Filtrar dataframes 93

4. Funciones 94
4.1. Definir una función 94
4.2. Funciones incorporadas 95
4.3. Funciones definidas por el usuario 96

5. Estructuras de Control 97
5.1. Sentencias condicionales 97
5.2. Bucles 99

6. La Familia de Funciones Apply 101
6.1. Las funciones apply vs. bucles 103

7. Gráficos 104

OBJETIVOS		PALABRAS CLAVE
· Introducción y operativa básica con R · Trabajar con los distintos tipos de objetos en R. · Realizar consultas. · Utilizar las funciones incorporadas en R y definir funciones propias. · Programación de sentencias condicionales y bucles. · Realizar una estadística descriptiva de nuestros datos a partir de las funciones apply y de la generación de gráficos.		· R · Librerías · Funciones · Vectores, matrices, listas, factores, dataframes · Estructuras de control

1. INTRODUCCIÓN A R Y AYUDA EN LÍNEA

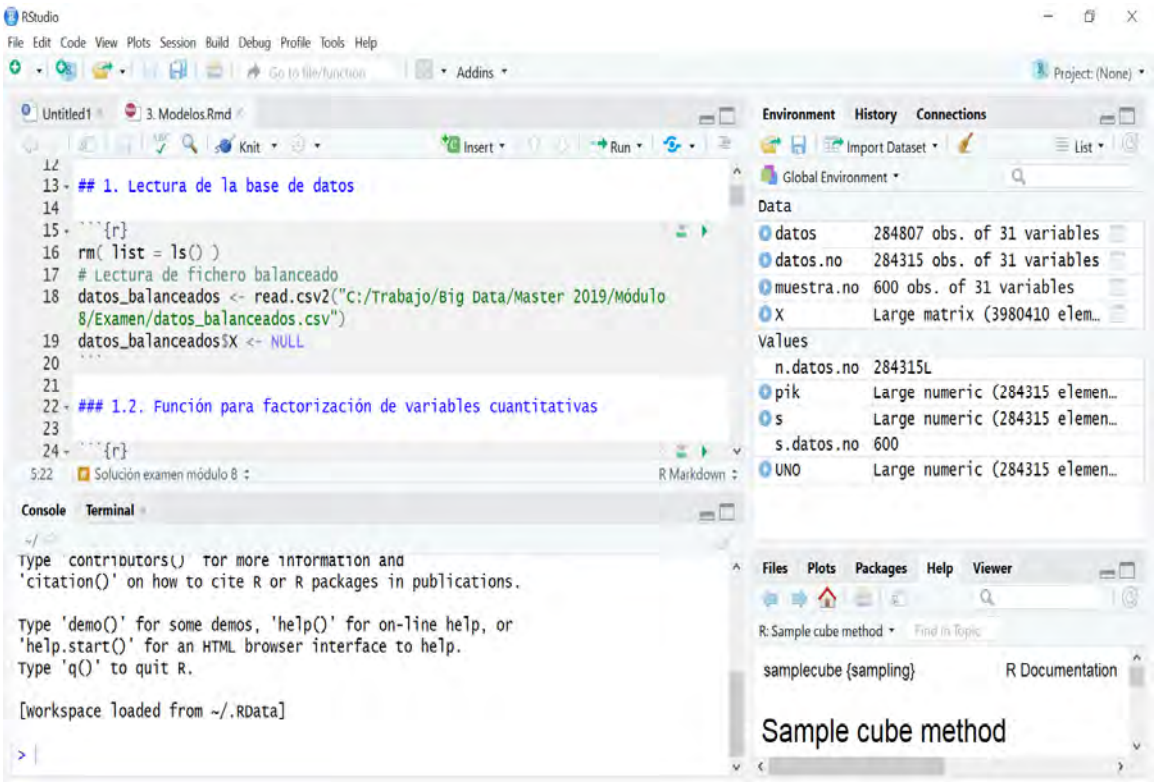
R es un lenguaje de programación y un entorno de software orientado al análisis estadístico, además del cálculo numérico, cabe destacar su potencial para la representación gráfica y la creación de informes.

R es gratuito y se distribuye bajo la Licencia Pública General de GNU. Está disponible para los sistemas operativos: Windows, Mac y Linux.

La interfaz propia de R es poco amigable, una simple consola para escribir y ejecutar código. La mejor plataforma para utilizar R es RStudio.

RStudio es un IDE muy popular, que ofrece un entorno amigable para trabajar en R. Un IDE (Integrated Development Environment) es un entorno de desarrollo integrado es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

FIGURA 1. CONSOLA RSTUDIO



Fuente: RStudio IDE

A grandes rasgos, RStudio se compone de cuatro secciones:

1. Izquierda-Arriba (esta sección se abrirá cuando sea necesario):
   - Editor de código: Para escribir y guardar scripts de R. También se puede editar/guardar cualquier archivo de texto (.csv, .txt), HTML, etc.
   - Visor de objetos: Para consultar el contenido de ciertos objetos R.
2. Izquierda-Abajo: La Consola, donde se escriben y ejecutan los comandos de R.
3. Derecha-Arriba:

- Environment: Muestra el entorno de trabajo, en el que iremos viendo los objetos R (variables y funciones) que vayamos creando, cargando,… Obsérvese que esta pestaña contiene ciertos iconos que permiten guardar el contenido de la memoria, cargar el contenido de la memoria de una sesión de trabajo anterior, importar archivos de datos (CSV, Excel, SPSS,…) y limpiar el contenido de la memoria.
- History: Guarda un historial de comandos R según se van introduciendo en la consola.

4. Derecha-Abajo:
   - Files: Explorador de archivos. Por defecto, el directorio actual es home.
   - Plots: Se mostrarán los gráficos que generemos en R.
   - Packages: Podemos ver qué paquetes tenemos instalados (un paquete es una colección de funciones que aumenta la funcionalidad de R). También nos permite descargar e instalar nuevos paquetes, y borrar paquetes instalados.
   - Help: Permite acceder a la ayuda de R.
   - Viewer: Muestra contenido web local.

Por otra parte, la ayuda en línea de R proporciona información muy útil de cómo utilizar las funciones. La ayuda se encuentra disponible directamente para una función dada. Por ejemplo:

```
?lm
  starting httpd help server ... done
```

El comando help(lm) o help("lm") tiene el mismo efecto. Esta última función se debe usar para acceder a la ayuda con caracteres no-convencionales:

```
help("!")
```

El mismo resultado se obtiene en el cajetín de la pestaña "help" de la ventana que se despliega en el cuadrante derecho-abajo de la consola R-Studio.

2. OBJETOS EN R

En cualquier lenguaje de programación es necesario usar *variables* para almacenar información. Las variables no son más que ubicaciones de memoria reservadas para almacenar valores. Los datos a almacenar pueden ser de varios *tipos*, como: carácter, numérico (entero o coma flotante), lógico, etc.

A diferencia de otros lenguajes de programación (como *C* y *Java*), en *R* las variables no se declaran como un tipo de datos. Las variables se asignan con *objetos R*, y el *tipo de datos* del objeto R, se convierte en el tipo de datos de la variable. Hay muchas clases de *objetos en R*, las más comunes para almacenar datos son:

- Vectores.
- Matrices.
- Listas.
- Factores.
- Dataframes.

Durante una sesión de *R*, todos los objetos estarán en memoria y se pueden guardar en disco para futuras sesiones.

2.1. VECTORES

Los vectores son los *objetos de datos* en *R* más básicos (estructura de datos *unidimensional*).

2.1.1. Crear

Para *asignar* a una variable un valor determinado, se suele utilizar el operador `<-`. También se puede utilizar el operador `=`. La función `class` permite conocer la *clase* del objeto. La función `typeof` permite conocer el *tipo* del objeto, cómo se almacena el objeto en memoria. El carácter `#` se utiliza para introducir un *comentario*.

A continuación, se explica cómo *crear* vectores.

Vectores de un solo elemento

```
variable <- 3
class(variable)
   [1] "numeric"
typeof(variable)
   [1] "double"

variable <- 3L  # Sufijo "L": el número debe ser almacenado como un entero
class(variable)
   [1] "integer"
typeof(variable)
   [1] "integer"

variable <- 3.5
class(variable)
   [1] "numeric"
typeof(variable)
   [1] "double"

variable <- "manzana"
class(variable)
   [1] "character"
typeof(variable)
   [1] "character"

variable <- "3"
class(variable)
   [1] "character"
typeof(variable)
   [1] "character"

variable <- TRUE
class(variable)
   [1] "logical"
typeof(variable)
   [1] "logical"

variable <- FALSE
class(variable)
   [1] "logical"
typeof(variable)
   [1] "logical"
```

Vectores de varios elementos

En los ejemplos anteriores, hemos creado vectores de un solo elemento. Se pueden crear vectores de *varios* elementos utilizando:

- El operador : genera una secuencia de números (con un incremento de 1 o -1) para crear un vector.
- La función c reúne varios elementos para formar un vector.
- La función seq genera una secuencia de números para crear un vector, se puede especificar el incremento o el número de elementos.
- La función rep replica los elementos de un vector.

Otras funciones de interés:

- La función length muestra el número de elementos de un objeto (longitud).
- La función str muestra la estructura de un objeto.
- La función summary muestra un resumen estadístico de un objeto.

```
x <- 1:5  # Genera una secuencia de números del 1 al 5
x
   [1] 1 2 3 4 5

class(x)
   [1] "integer"

typeof(x)
   [1] "integer"

length(x)
   [1] 5

str(x)
   int [1:5] 1 2 3 4 5

summary(x)
   Min.  1st Qu.  Median  Mean  3rd Qu.  Max.
      1        2       3     3        4     5

x <- 9:5  # Genera una secuencia de números del 9 al 5
x
   [1] 9 8 7 6 5

x <- c(1, 4, 3, 5)
x
   [1] 1 4 3 5

class(x)
   [1] "numeric"

typeof(x)
   [1] "double"

x <- seq(1, 9, 2)  # Genera una secuencia de números del 1 al 9, con un incremento de
2
x
   [1] 1 3 5 7 9

seq(1, 9, length = 5)  # Genera una secuencia de 5 números, del 1 al 9
   [1] 1 3 5 7 9

seq(1, 9, length = 6)  # Genera una secuencia de 6 números, del 1 al 9
   [1] 1.0 2.6 4.2 5.8 7.4 9.0

rep(2, 4)   # Repite "2" cuatro veces
   [1] 2 2 2 2

rep(1:4, 3) # Repite "1,2,3,4" tres veces
```

```
   [1] 1 2 3 4 1 2 3 4 1 2 3 4

x <- c("manzana", "pera", "naranja")
x <- c(TRUE, FALSE, TRUE)
```

Si un vector es de una clase o tipo concreto (carácter, entero, numérico de doble precisión, lógico, ...), ¿qué pasaría si mezclásemos elementos de diferentes tipos?

```
x <- c(3, TRUE, "naranja")
x
   [1] "3" "TRUE" "naranja"

class(x)
   [1] "character"

x <- c(4.25, FALSE, 12L, TRUE)
x
   4.25 0.00 12.00 1.00

class(x)
   [1] "numeric"
```

2.1.2. Seleccionar elementos

Los elementos de un vector se seleccionan mediante el *índice*, es decir, la posición que ocupa un elemento en el vector. Para la indexación se utilizan los *corchetes* [ ]. La indexación comienza en la posición *1*. Si se asigna un valor *negativo* al índice, se eliminará ese elemento del resultado. TRUE y FALSE, también se pueden usar en la indexación.

```
x <- c("lunes","martes","miércoles","jueves","viernes","sábado","domingo")
z <- x[c(2,3,6)]  # Elementos 2º, 3º y 6º del vector
z
   [1] "martes" "miércoles" "sábado"

z <- x[-7]
z
   [1] "lunes" "martes" "miércoles" "jueves" "viernes" "sábado"

z <- x[c(-2,-5)]
z
   [1] "lunes" "miércoles" "jueves" "sábado" "domingo"

z <- x[c(TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE)]
z
   [1] "lunes" "sábado"

z <- x[c(F,T,F,F,T,F,F)]
z
   [1] "martes" "viernes"

# Actualizar un elemento del vector
x[2] <- "MARTES"
x
   [1] "lunes" "MARTES" "miércoles" "jueves" "viernes" "sábado" "domingo"

v <- seq(1,2,.1)
v
   [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0

# Añadir un elemento al final del vector
v <- c(v,2.1)
v
```

```
   [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1

# Añadir un elemento entre los elementos 4º y 5º
v <- c(v[1:4], 1.35, v[5:length(v)])
v
   [1] 1.00 1.10 1.20 1.30 1.35 1.40 1.50 1.60 1.70 1.80 1.90 2.00 2.10
```

También podemos asignar *nombres* a los elementos de un vector, mediante la función `names`.

```
precios <- c(0.23, 0.35, 0.14, 0.2, 0.23)
names(precios) <- c("Manzana", "Uva", "Pera", "Naranja", "Cereza")
precios
   Manzana   Uva  Pera  Naranja  Cereza
      0.23  0.35  0.14     0.20    0.23

# Seleccionar un elemento utilizando el atributo "nombre"
precios["Naranja"]
   Naranja
       0.2

# Seleccionar varios elementos mediante el atributo "nombre"
precios[c("Manzana","Uva")]
   Manzana   Uva
      0.23  0.35

# Eliminar los nombres asignados
names(precios) <- NULL
```

2.1.3. Trabajar con vectores

Operaciones aritméticas

```
v1 <- c(5:1, 6:8, 12:9)
v1
   [1] 5 4 3 2 1 6 7 8 12 11 10 9

# Operar con un vector
v1 + 6
   [1] 11 10 9 8 7 12 13 14 18 17 16 15

(v1 - 5)^2
   [1] 0 1 4 9 16 1 4 9 49 36 25 16
```

Dos vectores de la misma longitud se pueden sumar, restar, multiplicar o dividir, obteniendo un vector resultante.

```
# Creamos dos vectores
v1 <- c(4,9,5,6,0,12)
v2 <- c(5,12,0,9,2,3)

# Sumar vectores
resultado <- v1 + v2
resultado
   [1] 9 21 5 15 2 15

# Restar vectores
resultado <- v1 - v2
resultado
   [1] -1 -3 5 -3 -2 9

# Multiplicar vectores
resultado <- v1 * v2
```

```
resultado
   [1] 20 108 0 54 0 36

# Dividir vectores
resultado <- v1 / v2
resultado
   [1] 0.8000000 0.7500000 Inf 0.6666667 0.0000000 4.0000000
```

NA (Not Available/Missing Value): No disponible. Constante lógica que indica un valor perdido.
Inf: Infinito positivo y negativo. Por ejemplo: 1/0, log(0).

NaN (Not a Number): No es un número. Expresa un resultado imposible de calcular, como es el caso de las indeterminaciones, la raíz cuadrada de un número negativo, el logaritmo de un número negativo, etc. Por ejemplo: 0/0, sqrt(-1), log(-1). Para poder controlar estas situaciones, *R* dispone de las siguientes funciones: is.na, is.finite, is.infinite, is.nan.

Si realizamos operaciones aritméticas con dos vectores de diferente longitud, los elementos del vector más corto se repetirían hasta tener la misma longitud que el otro. Veamos un ejemplo.

```
v1 <- c(4,9,5,6,0,12)
v2 <- c(5,12)
# v2 se convierte en c(5,12,5,12,5,12)

v1 + v2
   [1] 9 21 10 18 5 24
```

Ordenar vectores

Los elementos de un vector se pueden ordenar mediante la función sort.

```
v <- c(8,4,-5,12,0,7)
v <- sort(v)
v
   [1] -5 0 4 7 8 12
sort(v, decreasing = TRUE)  # orden decreciente
   [1] 12 8 7 4 0 -5

v <- c("rojo","amarillo","verde","azul")
sort(v)
   [1] "amarillo" "azul" "rojo" "verde"
sort(v, decreasing = TRUE)  # orden decreciente
   [1] "verde" "rojo" "azul" "amarillo"
```

2.2. MATRICES

En las matrices los elementos están dispuestos en una estructura de *dos dimensiones* (filas y columnas). Al igual que los vectores, todos los elementos de una matriz serán del *mismo tipo*. Aunque podemos crear una matriz que contenga solo caracteres o solo valores lógicos, no es de mucha utilidad. Generalmente, se utilizan matrices que contengan *números*, para realizar cálculos matemáticos.

2.2.1. Crear

Una matriz se puede crear mediante la función matrix. Veamos la sintaxis de dicha función.

```
matrix(data, nrow, ncol, byrow, dimnames)
```

Descripción de los *argumentos* de la función:

- *data*: vector de entrada, desde el cual se obtendrán los elementos de la matriz.
- *nrow*: número de filas que se crearán.
- *ncol*: número de columnas que se crearán.
- *byrow*: si es TRUE, los elementos del vector de entrada se rellenan por fila.
- *dimnames*: nombres asignados a las filas y columnas.

```
# Los elementos se colocan secuencialmente por fila
m <- matrix(1:12, nrow = 4, byrow = TRUE)
m
        [,1] [,2] [,3]
   [1,]    1    2    3
   [2,]    4    5    6
   [3,]    7    8    9
   [4,]   10   11   12

# Los elementos se colocan secuencialmente por columna
n <- matrix(1:12, nrow = 4)  # Por defecto, byrow = FALSE
n
        [,1] [,2] [,3]
   [1,]    1    5    9
   [2,]    2    6   10
   [3,]    3    7   11
   [4,]    4    8   12

# Definir nombres de columnas y filas
nombresfilas <- c("row1", "row2", "row3", "row4")
nombrescolumnas <- c("col1", "col2", "col3")
m <- matrix(1:12, nrow = 4, byrow = TRUE, dimnames = list(nombresfilas, nombrescolumn
as))
m
        col1 col2 col3
   row1    1    2    3
   row2    4    5    6
   row3    7    8    9
   row4   10   11   12
```

Función `dim`: dimensión del objeto (filas/columnas).

Función `nrow`: número de filas del objeto.

Función `ncol`: número de columnas del objeto.

Función `rownames`: nombres de las filas del objeto.

Función `colnames`: nombres de las columnas del objeto.

```
class(m)
   [1] "matrix"

typeof(m)
   [1] "integer"

length(m)
   [1] 12

dim(m)
   [1] 4 3
```

```
nrow(m)
   [1] 4

ncol(m)
   [1] 3

rownames(m)
   [1] "row1" "row2" "row3" "row4"

colnames(m)
   [1] "col1" "col2" "col3"

colnames(m) <- c("a", "b", "c")
colnames(m)
   [1] "a" "b" "c"
```

También podemos crear matrices a partir de vectores, mediante las funciones `rbind` y `cbind`.

```
x <- 1:10
y <- 11:20
z <- 21:30

m <- cbind(x, y, z)  # Por columnas
m
          x  y  z
    [1,]  1 11 21
    [2,]  2 12 22
    [3,]  3 13 23
    [4,]  4 14 24
    [5,]  5 15 25
    [6,]  6 16 26
    [7,]  7 17 27
    [8,]  8 18 28
    [9,]  9 19 29
   [10,] 10 20 30

n <- rbind(x, y, z)  # Por filas
n
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
   x    1    2    3    4    5    6    7    8    9    10
   y   11   12   13   14   15   16   17   18   19    20
   z   21   22   23   24   25   26   27   28   29    30

n <- cbind(m, 31:40) # Añadir una columna a una matriz
n
          x  y  z
    [1,]  1 11 21 31
    [2,]  2 12 22 32
    [3,]  3 13 23 33
    [4,]  4 14 24 34
    [5,]  5 15 25 35
    [6,]  6 16 26 36
    [7,]  7 17 27 37
    [8,]  8 18 28 38
    [9,]  9 19 29 39
   [10,] 10 20 30 40

colnames(n) <- NULL  # Eliminar los nombres de las columnas
n
```

```
      [,1] [,2] [,3] [,4]
 [1,]    1   11   21   31
 [2,]    2   12   22   32
 [3,]    3   13   23   33
 [4,]    4   14   24   34
 [5,]    5   15   25   35
 [6,]    6   16   26   36
 [7,]    7   17   27   37
 [8,]    8   18   28   38
 [9,]    9   19   29   39
[10,]   10   20   30   40
```

2.2.2. Seleccionar elementos

Los elementos de una matriz se seleccionan mediante el *índice de la fila* y el *índice de la columna*. También se pueden seleccionar elementos utilizando el *nombre* de la fila o de la columna.

Para la indexación se utilizan los *corchetes*, con *dos partes* separadas por una "coma": la de la izquierda se refiere a las filas y la de la derecha a las columnas [filas, columnas].

```
# El elemento 1ª fila y 3ª columna
m[1,3]
   z
   21

# Todos los elementos de la 1ª fila
v <- m[1,]
v
   x  y  z
   1 11 21
is.vector(v)
   [1] TRUE

# Todos los elementos de la 3ª columna
v <- m[,3]
v
   [1] 21 22 23 24 25 26 27 28 29 30
is.vector(v)
   [1] TRUE

# Tres primeras filas. La 1ª y 3ª columna
n <- m[1:3,c(1,3)]
n
      x  z
   [1,] 1 21
   [2,] 2 22
   [3,] 3 23
is.matrix(n)
   [1] TRUE

# Indicar las columnas por nombre
m[1:3,c("x","z")]
      x  z
   [1,] 1 21
   [2,] 2 22
   [3,] 3 23

# Actualizar un elemento de la matriz
m[2,3] <- 50
m[2,]
```

```
   x  y  z
   2 12 50

# Actualizar toda la 1ª columna de la matriz
m[,1] <- 31:40
```

2.2.3. Trabajar con matrices

Operaciones aritméticas

Diversas operaciones matemáticas se pueden realizar con las matrices. El resultado de la operación también será una matriz.

Para poder operar, las matrices deben tener la misma dimensión, es decir, el mismo número de filas y de columnas.

```
m <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
m
        [,1] [,2] [,3]
   [1,]    3   -1    2
   [2,]    9    4    6

n <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
n
        [,1] [,2] [,3]
   [1,]    5    0    3
   [2,]    2    9    4

# Sumar matrices
x <- m + n
x
        [,1] [,2] [,3]
   [1,]    8   -1    5
   [2,]   11   13   10

# Restar matrices
x <- m - n
x
        [,1] [,2] [,3]
   [1,]   -2   -1   -1
   [2,]    7   -5    2

# Multiplicar matrices
x <- m * n
x
        [,1] [,2] [,3]
   [1,]   15    0    6
   [2,]   18   36   24

# Dividir matrices
x <- m / n
x
        [,1]      [,2]      [,3]
   [1,]  0.6      -Inf 0.6666667
   [2,]  4.5 0.4444444 1.5000000
```

2.3. LISTAS

Las listas son *objetos R* que contienen elementos de *diferentes tipos*, como: números, cadenas de texto, vectores, matrices, etc. Las listas son estructuras de datos que encapsulan toda esa

información heterogénea. Una lista también puede contener otras listas que, a su vez, pudieran contener otras.

2.3.1. Crear

Una lista se crea utilizando la función list.

A continuación, creamos una lista que contiene: cadenas de texto, números, vectores y valores lógicos.

```
lista <- list("rojo", "azul", c(15,28,9), TRUE, 23.14, 108.1)
lista
   [[1]]
   [1] "rojo"

   [[2]]
   [1] "azul"

   [[3]]
   [1] 15 28  9

   [[4]]
   [1] TRUE

   [[5]]
   [1] 23.14

   [[6]]
   [1] 108.1

class(lista)
   [1] "list"

length(lista)
   [1] 6
```

2.3.2. Seleccionar elementos

Se puede acceder a los elementos de una lista, mediante el *índice* del elemento en la lista. Las listas disponen de un operador para extraer elementos: los dobles corchetes [[ ]].

```
# Acceder al primer elemento de la lista
lista[1]
   [[1]]
   [1] "rojo"

lista[[1]]
   [1] "rojo"

length(lista[[3]])
   [1] 3

lista[[3]][2]
   [1] 28
```

Es posible asignar nombres a los elementos de una lista y se puede acceder a ellos usando el *nombre*, mediante el operador $.

```
# Creamos una lista que contiene: un vector, una matriz y una lista
lista <- list(c("Enero","Febrero","Marzo"), matrix(c(3,9,5,1,-2,8), nrow=2),
              list("verde",12.3))
```