

INTRODUCCIÓN A LA PROGRAMACIÓN EN R



Historia, conceptos y
conocimientos básicos

PAUTA DE CLASE

HISTORIA DE R

CONCEPTOS DE
PROGRAMACION

EJERCICIOS



R

- Ross Ihaka & Robert Gentleman (U. of Auckland)
- 1991
- software libre y parte del proyecto GNU desde (1996)
- Lenguaje de programacion centrado en estadisticas y computo numerico
- orientado a objetos
- practivo y facil de usar
- Altamente integrable con otros lenguajes de programacion con protagonismo en AD:
 - python, julia, c++, fortran

COMPUTACION AVANZADA

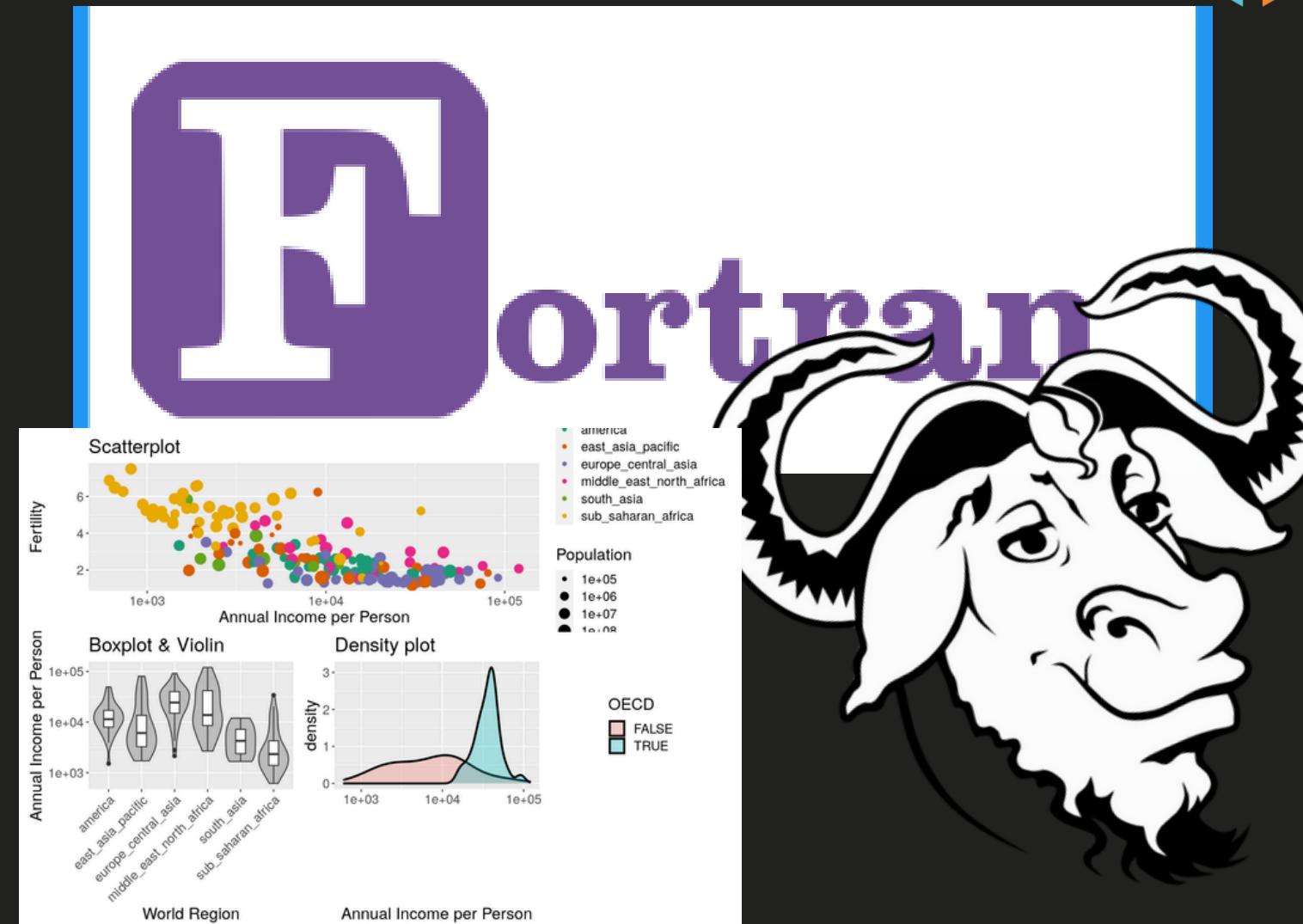
Se utiliza en tecnologias como Big Data, HPC, IA ,ciencias -ómicas

INTERPRETADO

se ejecuta línea a línea, ideal para exploración.

S

- R original
- desarrollado en Bell Labs (1976)
- John Chambers
- wrapper de fortran



CRAN

- Repositorio de paquetes
- ~20.000
- creado por comunidad
- de todo, sobre todo (bioinformatica, ecologia, etc)

PROGRAMACION

- traducir ideas a una forma ejecutable por una máquina.
- escribir un conjunto de instrucciones para ejecutar un circuito
- combina lógica, matemáticas y abstracción computacional para crear soluciones eficientes.
- Estructuras:
 - condicionales,
 - iteraciones
 - funciones
 - estructuras de datos



DATOS ATOMICOS

- unidad mínima de procesamiento
- Se consideran los componentes básicos de las estructuras más complejas



TIPOS DE DATOS ATOMICOS

Numeric (double)

- Representan números reales (decimales de precisión doble por defecto).

Integer

- Representan números enteros, explícitamente marcados con una L al final.

Character

- Cadenas de texto. Pueden ser una letra, una palabra, una frase o un texto largo.

Complex

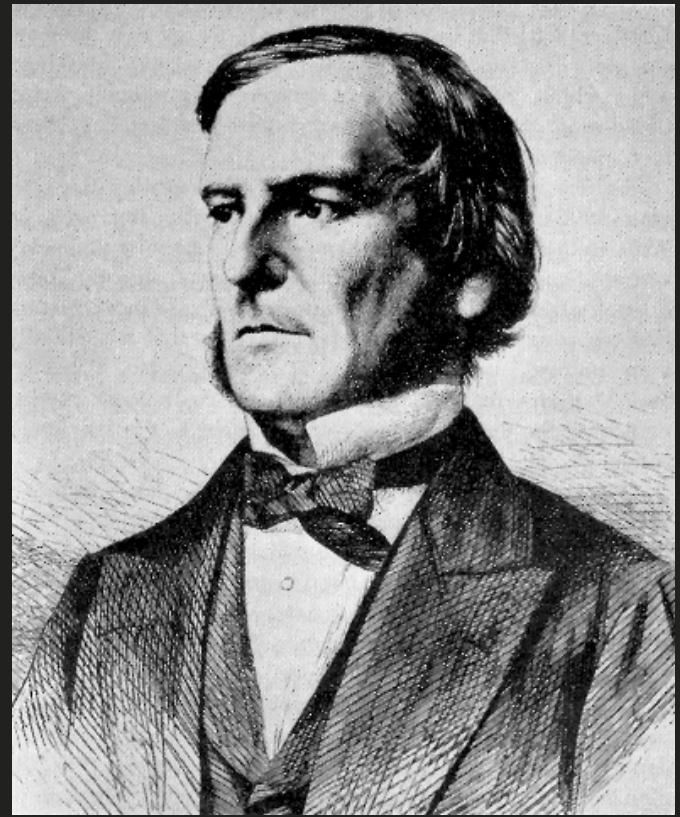
- Números complejos, con parte real e imaginaria.

Raw

- Datos en bruto, como bytes puros. Poco usado en análisis de datos, pero clave para manejo de bajo nivel

Datos nulos:

- NA



TIPOS DE DATOS ATOMICOS: BOOLEANO

- El boleano es un tipo de dato especial. Solo tiene 2 valores posibles TRUE OR FALSE
- Coercion numerica
 - TRUE = 1
 - FALSE = 0
- Componente fundamental de la computación
- Permiten tomar desiciones por medio de estructuras de control
-

OPERADORES CONDICIONALES

Son elementos que operan con los booleanos.
los evaluan, calculan expresiones y ejercen
acciones en base a eso.

Operadores Lógicos
Operadores Relacionales
Estructuras de Control

OPERADORES LÓGICOS

Son elementos que operan con los booleanos. Los evalúan, calculan expresiones y ejercen acciones en base a eso.

Permiten evaluar expresiones complejas, combinar booleanos o estructuras de booleanos.

Operador	Nombre	Evalúa	Uso común
&	AND	Elemento a elemento	$x \& y$
&&	AND (corto circuito)	Elemento a elemento	$x \&\& y$
	OR	Elemento a elemento	$x y$
	OR (corto circuito)	Elemento a elemento	$x y$
!	NOT (negación)	Vectorizado	$!x$

COMPARADORES RELACIONALES.

Son operadores que comparan valores numéricos y devuelven un resultado booleano (TRUE o FALSE). Son la base de toda evaluación condicional en R.

Se usan en:

- Control de flujo (if, while)
- Filtrado de datos (subset, dplyr, etc.)
- Operaciones vectorizadas lógicas

Operaciones vectorizadas
si se aplica con “NA”, da “NA”

Operador	Significado	Ejemplo	Resultado
<code>“==”</code>	Igual a	<code>5 == 5</code>	TRUE
<code>!=</code>	Distinto de	<code>5 != 3</code>	TRUE
<code><</code>	Menor que	<code>3 < 5</code>	TRUE
<code><=</code>	Menor o igual que	<code>5 <= 5</code>	TRUE
<code>></code>	Mayor que	<code>7 > 2</code>	TRUE

ESTRUCTURAS DE CONTROL DE FLUJO

Son estructuras que alteran el orden natural de ejecución del código según condiciones o repeticiones. Permiten construir lógica condicional, bucles y algoritmos.

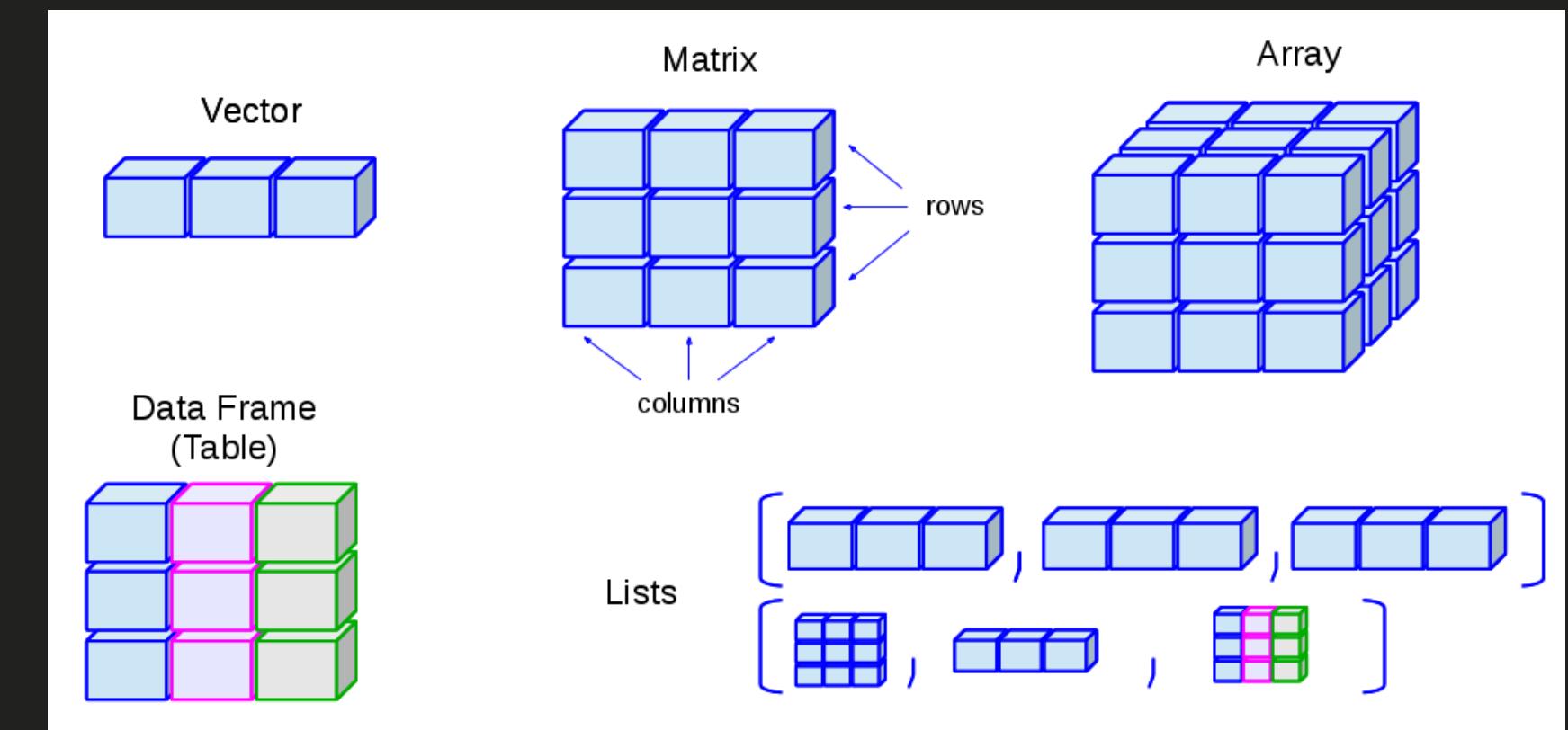
- Tomar decisiones
- Saltar pasos
- Salir de estructuras

Expr	Acción	Contexto válido
if/else	Evaluuar condiciones	Cualquier bloque
switch	Selección múltiple por valor	Cualquier bloque
break	Salir de un bucle	Dentro de bucles
next	Saltar a siguiente iteración	Dentro de bucles
return	Finalizar ejecución con valor	Dentro de funciones

tipo de input	output	tipo de operador
booleano	booleano	Operador Logico
cualquiera (no booleano)	booleano	comparador relacional
booleano	acción	estructura de control

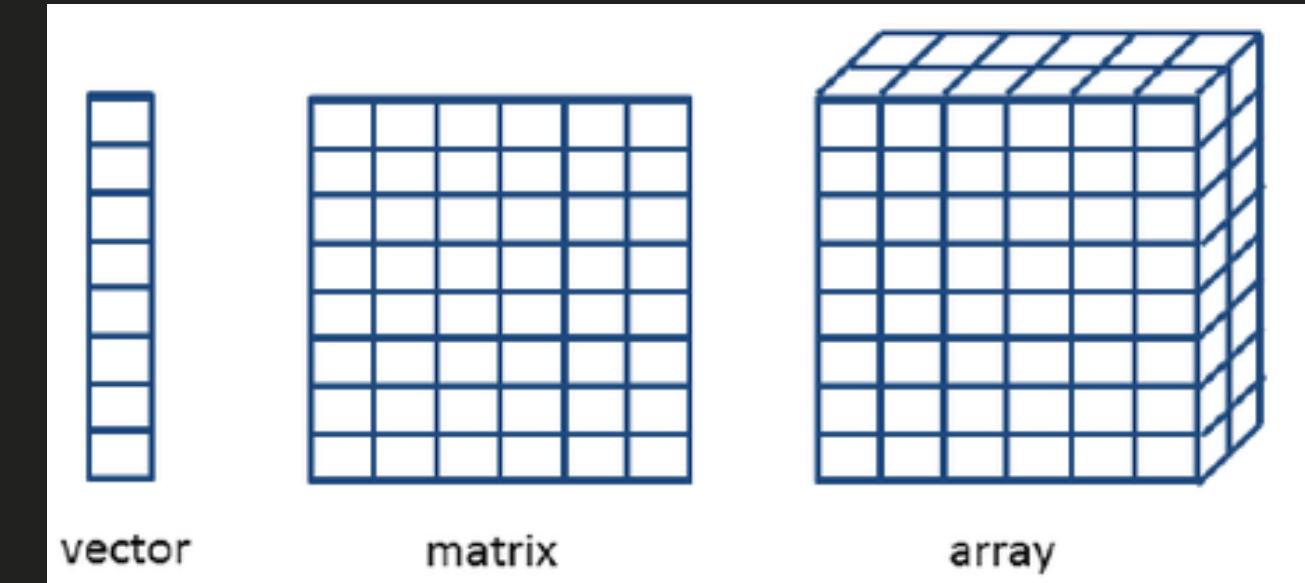
ESTRUCTURAS DE DATOS

- Es una forma organizada de almacenar datos atómicos para facilitar su acceso y uso.
- representar colecciones de elementos relacionados
- conjunto de datos junto con las operaciones permitidas sobre ellos
- elección adecuada de la estructura de datos es crucial para la eficiencia algorítmica



VECTORES

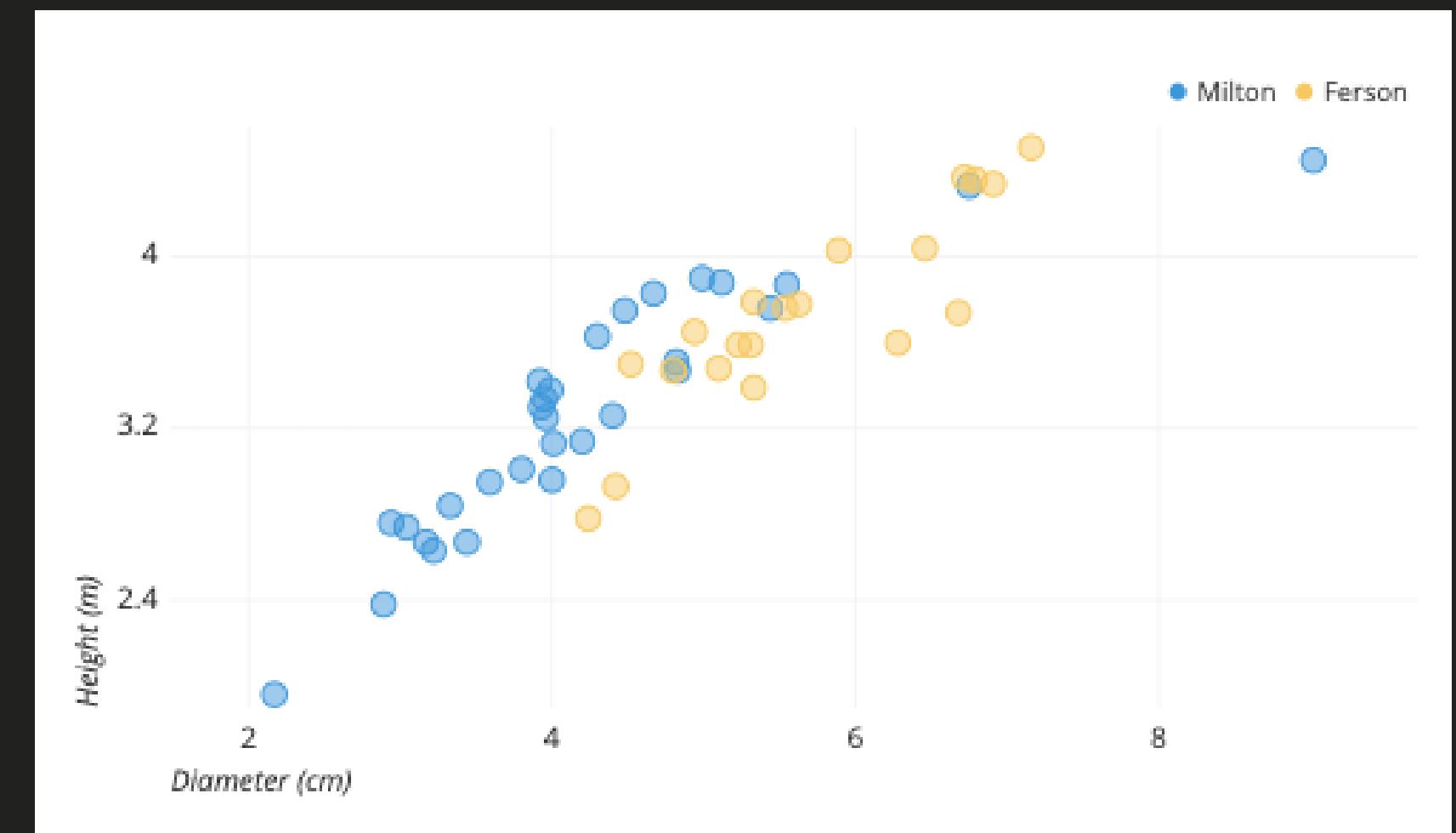
- Vectores
 - La estructura más básica.
 - Contienen elementos del mismo tipo (numérico, carácter, lógico, etc.).
 - Base para construir estructuras más complejas.
- Matrices
 - Vectores con dimensiones (2D).
 - homogéneos
 - Acceso mediante filas y columnas.
- Arrays
 - Generalización de matrices a más dimensiones (3D, 4D, etc.).
 - homogéneos (mismo tipo).
 - Útiles para datos multidimensionales como imágenes o series temporales multidimensionales.



Acción / Función	Vector	Matriz	Array
Obtener tamaño	<code>length()</code>	<ul style="list-style-type: none"> • <code>dim()</code> • <code>nrow()</code> • <code>ncol()</code> 	<code>dim()</code>
Crear estructura	<ul style="list-style-type: none"> • <code>c()</code> • <code>seq()</code> • <code>rep()</code> 	<code>matrix()</code>	<code>array()</code>
Estadísticas básicas	<ul style="list-style-type: none"> • <code>sum()</code> • <code>mean()</code>, • <code>median()</code> • <code>sd()</code> 	<ul style="list-style-type: none"> • <code>rowSums()</code> • <code>colSums()</code> • <code>rowMeans()</code> • <code>colMeans()</code> 	Uso de <code>apply()</code>
Aplicar función	No habitual (vectorizado)	<code>apply()</code> (filas/columnas)	<code>apply()</code> (dimensiones/má)

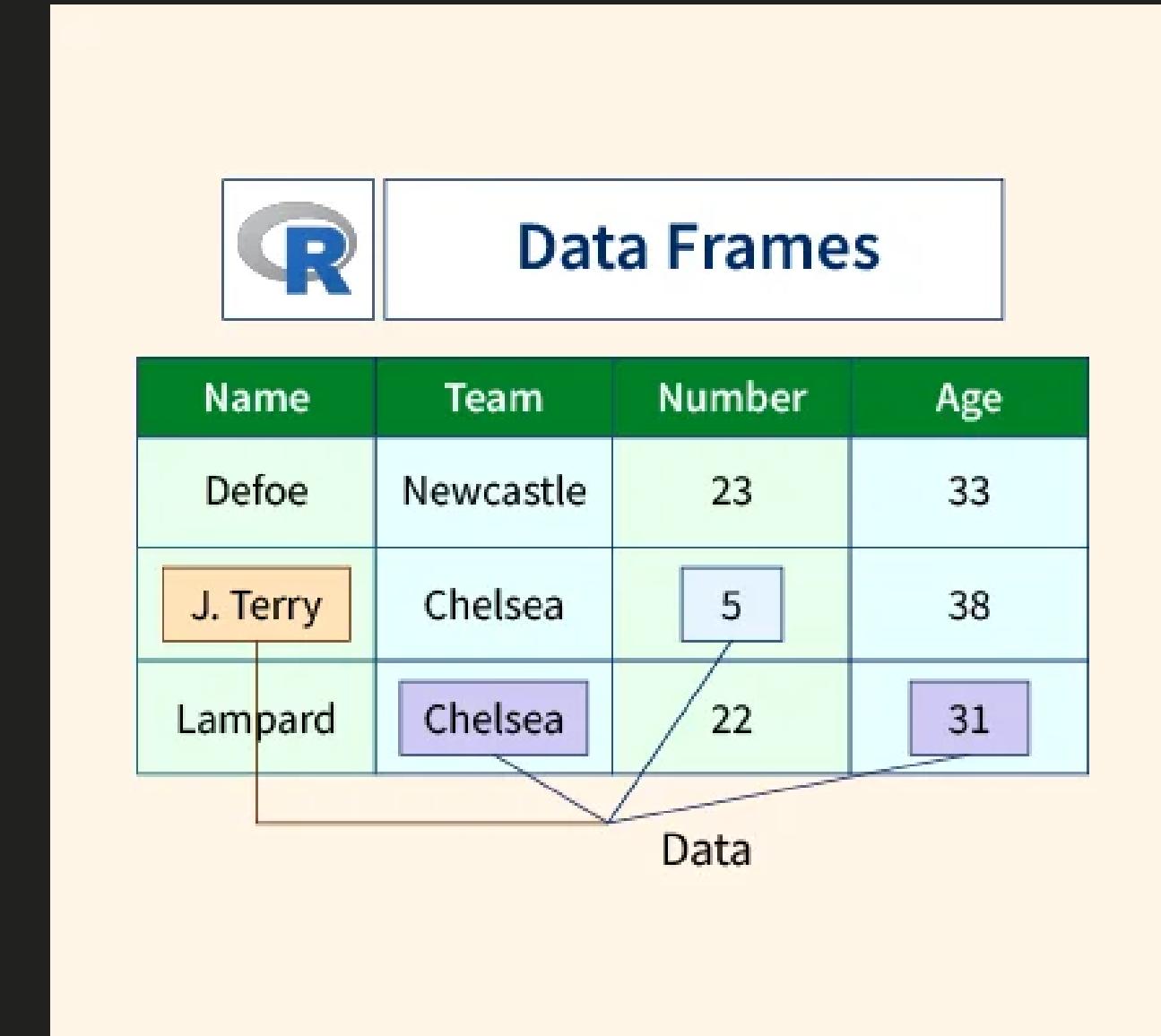
FACTORES

- Estructura que representa variables categóricas (cualitativas).
- Permite definir categorías con orden (factores ordenados) o sin orden.
- Las variables categóricas suelen ser variables independientes o de agrupamiento en pruebas estadísticas.
- Los factores organizan y codifican estas categorías para análisis.
- Facilitan el computo para pruebas específicas



DATOS TABULARES

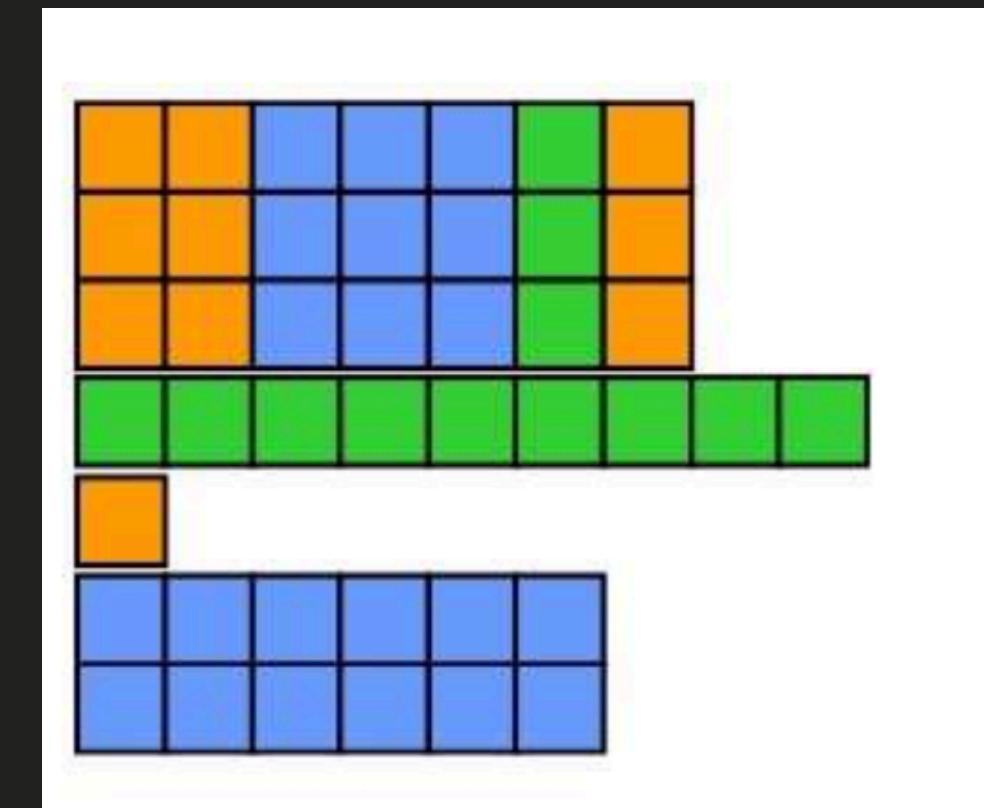
- Data Frame
 - estructura base tabular
 - columnas heterogéneas.
- Data Table (data.table):
 - extensión optimizada de data frame
 - más eficiente para grandes datos
 - operaciones rápidas.
- Tibble
 - variante moderna de data frame (tidyverse)
 - comportamiento más predecible y complejo.



Acción	Data Frame	Data Table	Tibble
Filtrar filas	subset(), [i,]	[i,]	filter()
Seleccionar columnas	[,j], \$	[, j]	select()
Ordenar	order() sort()	setorder()	arrange()
Agregar columnas	\$ o cbind.data.frame()	:=	mutate()
Resumen / agrupación	aggregate() tapply()	.SD + by	summarise()
Unión (join)	merge()	merge()	left_join() inner_join()

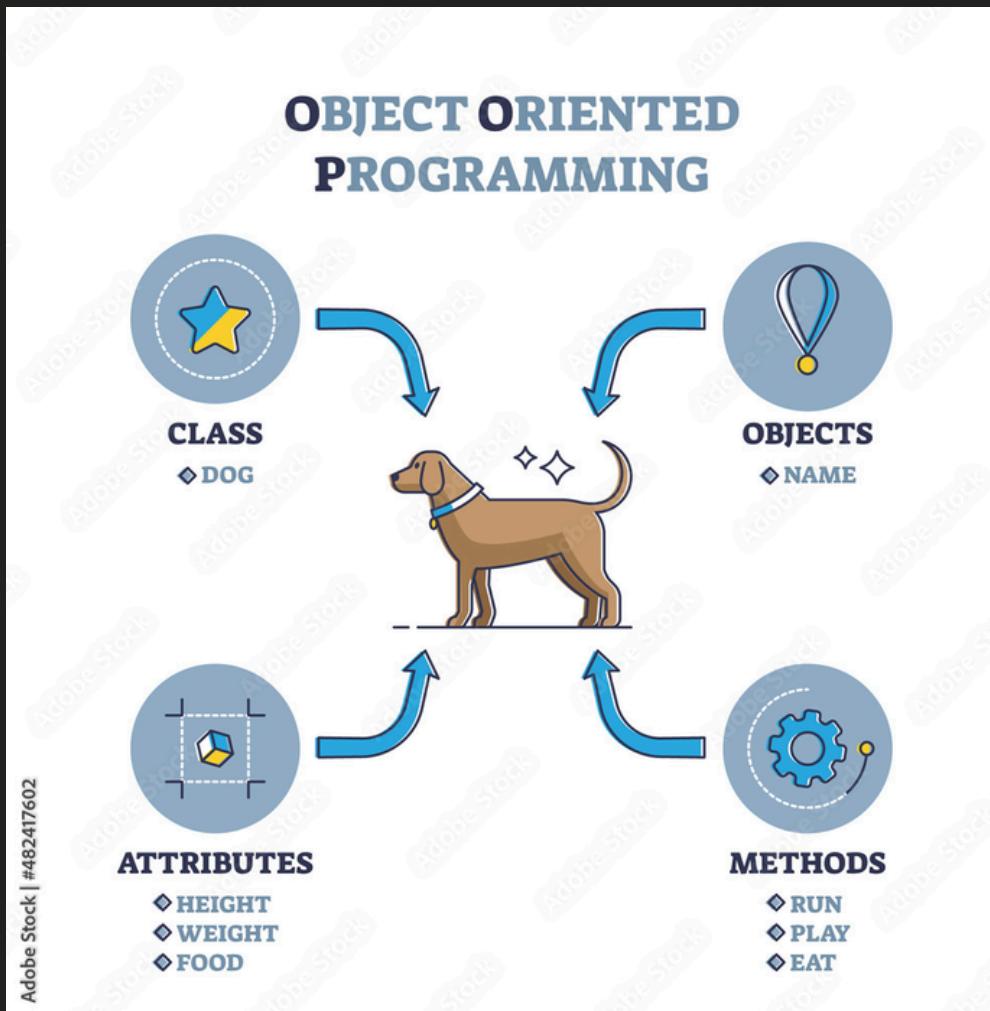
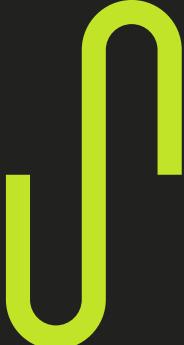
LISTAS

- Es una estructura de datos heterogénea y flexible
- Se accede a los elementos por posición o nombre
- Permiten construir objetos complejos y jerárquicos.
- Son fundamentales para la programación orientada a objetos en R.
- Permiten definir estructuras de datos personalizadas.
- Facilitan la extensión de R con nuevos tipos de objetos.
- Son fáciles de crear y manipular, ideales para programación flexible.



Método	Descripción
<code>list()</code>	Crea una lista
<code>length()</code>	Devuelve la cantidad de elementos de la lista
<code>names()</code>	Obtiene o asigna nombres a los elementos
<code>\$</code>	Accede a un elemento por nombre (estilo <code>\$nombre</code>)
<code>[[]]</code>	Accede a un elemento específico (por índice o nombre)
<code>unlist()</code>	Aplana una lista a vector (si es posible)
<code>str()</code>	Estructura de la lista (estructura interna)
<code>lapply()</code>	Aplica una función a cada elemento y retorna lista
<code>names<-</code>	Asigna nombres a los elementos

PROGRAMACION ORIENTADA A OBJETOS



KEY POINTS:

TODO ES UN OBJETO

TIPO BASE, ATRIBUTOS,
CLASES Y MÉTODOS

PROPIEDADES ESPECÍFICAS CON
COMPORTAMIENTOS ESPECÍFICOS

Un objeto es una especie de "caja digital" que:

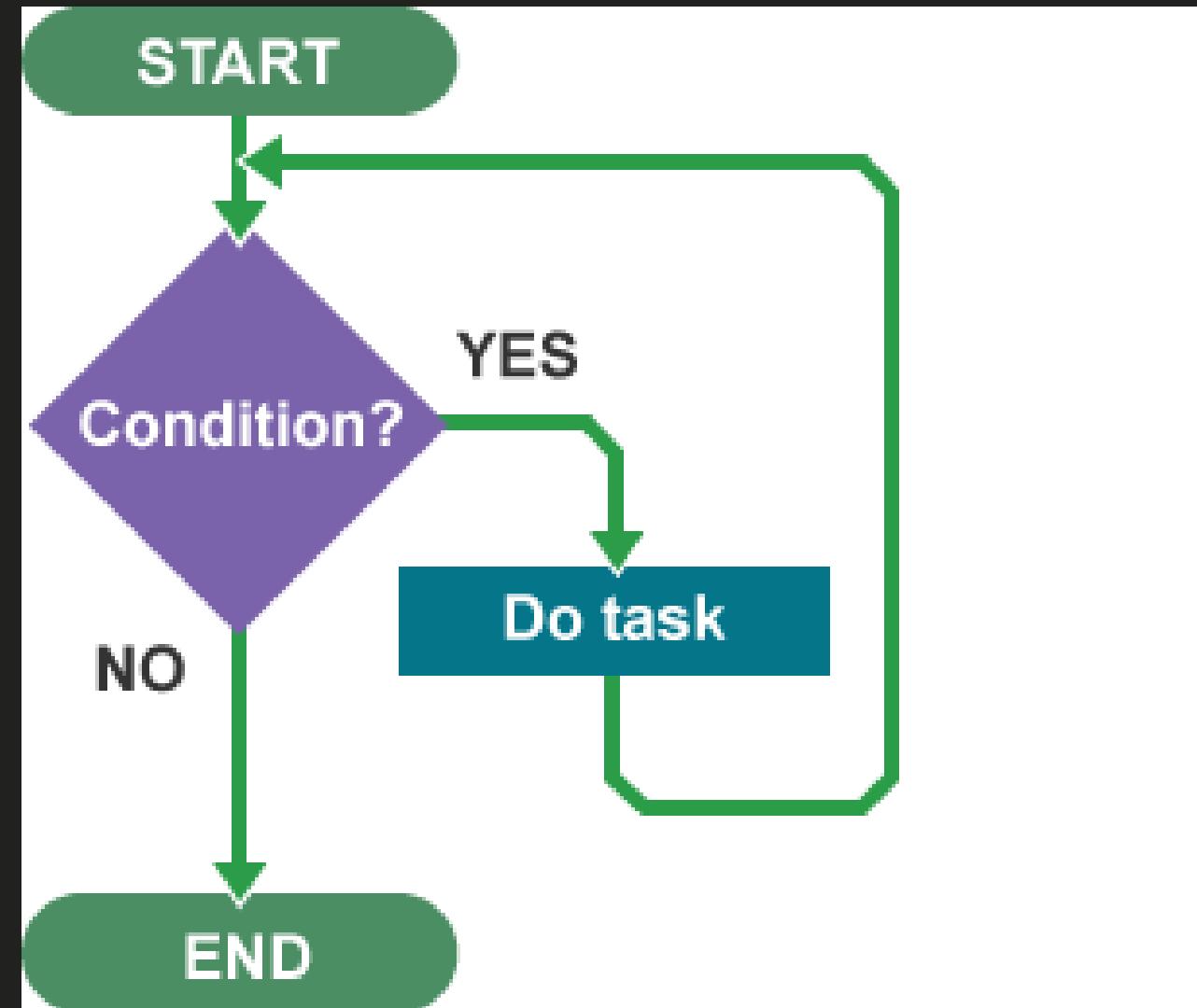
- Tiene datos
- Y acciones que puede hacer

Encapsulación: Usar herramientas sin entenderlas

ITERACIÓN

un proceso computacional que permite repetir una o varias instrucciones dentro de elementos de estructuras de datos

Automatiza acciones repetitivas o que se requieren hacer en varios objetos
Manejo de grandes volúmenes



LOOPS

Los loops permiten ejecutar un bloque de código múltiples veces. Son herramientas fundamentales para automatizar tareas repetitivas y procesar datos secuencialmente.

```
# FOR loop con 'next'  
for (i in 1:5) {  
    if (i == 3) {  
        next # Salta la iteración cuando i == 3  
    }  
    print(i)  
}  
  
# WHILE loop  
j <- 1  
while (j <= 5) {  
    print(j)  
    j <- j + 1  
}  
  
# REPEAT loop  
k <- 1  
repeat {  
    print(k)  
    k <- k + 1  
    if (k > 5) break  
}
```

FUNCIONES

Una función es un objeto de R que encapsula un conjunto de instrucciones. Permite modularizar el código, reducir la repetición y organizar la lógica en unidades reutilizables.

- Las funciones son objetos de primera clase: pueden pasarse como argumentos, almacenarse en listas, retornarse, etc.
- `return()` es opcional: R devuelve automáticamente la última expresión evaluada.

```
10 #This is the function with the new
shorthand
11 x <- 3
12 y <- 4
13 adding <- \((x,y){
14   z<-x+y
15   print(z)
16 }
17 adding(x,y)
```

EJERCICIOS.