<!--Estudio IMIBIC-->

R para Todos: El poder para explotar tus Datos {

<Por="Adrián Santiago Ortiz"/>

}

Equipo {

ADRIÁN SANTIAGO

BIOINFORMÁTICO

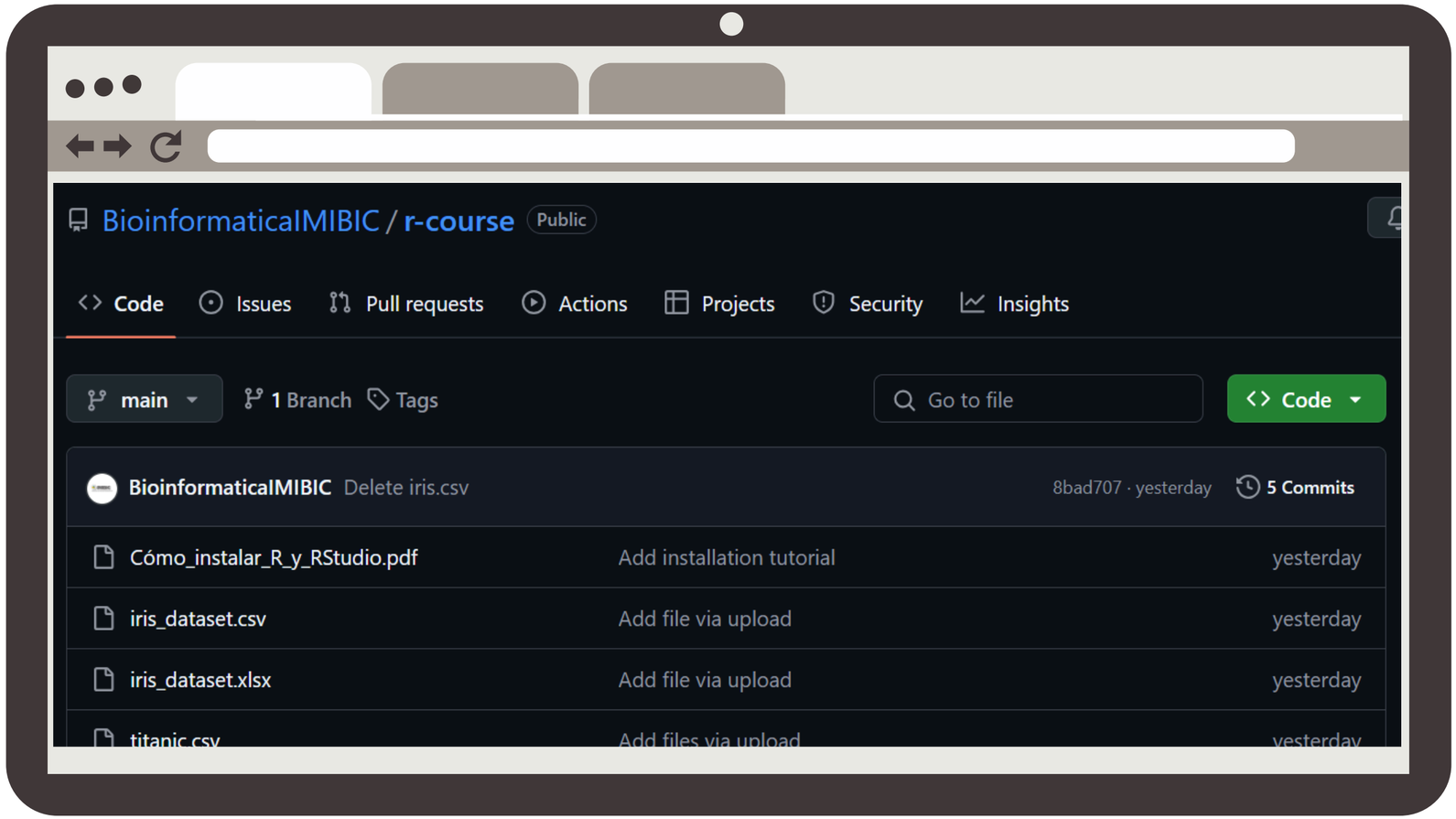
UCAIB BIOINFORMÁTICA Y BIOESTADÍSTICA (IMIBIC)

ALBERTO VELA

BIOESTADÍSTICO

}

Material disponible



https://github.com/BioinformaticaIMIBIC/r-course

Contenidos

Contenidos

01 Introducción a R y Rstudio

02 Tipos de datos

03 Operadores en R

04 Tipos de objetos y operaciones

Sesión 1

05 Estructuras de control y flujo de ejecución

06 Creación de funciones

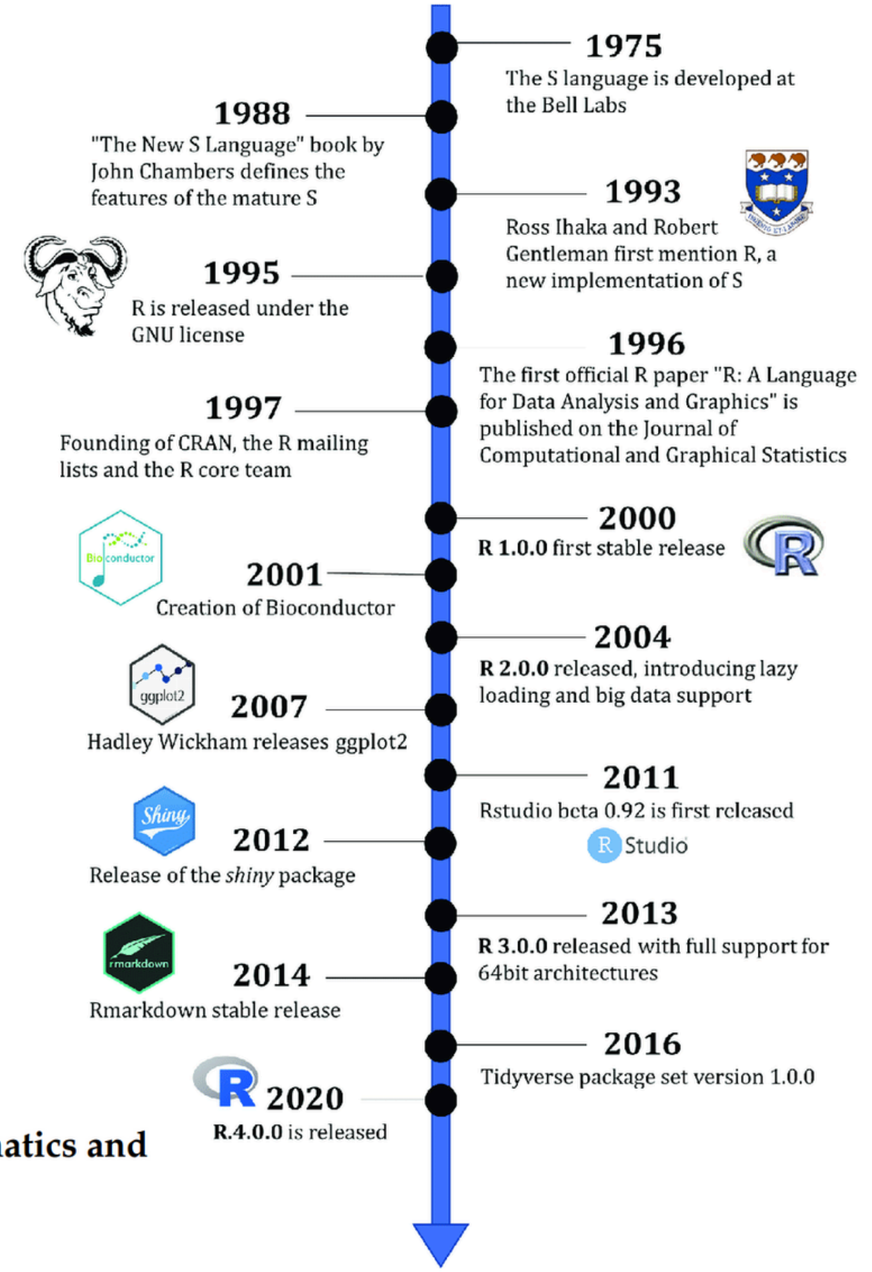
07 Visualización de resultados en gráficos

Contenidos

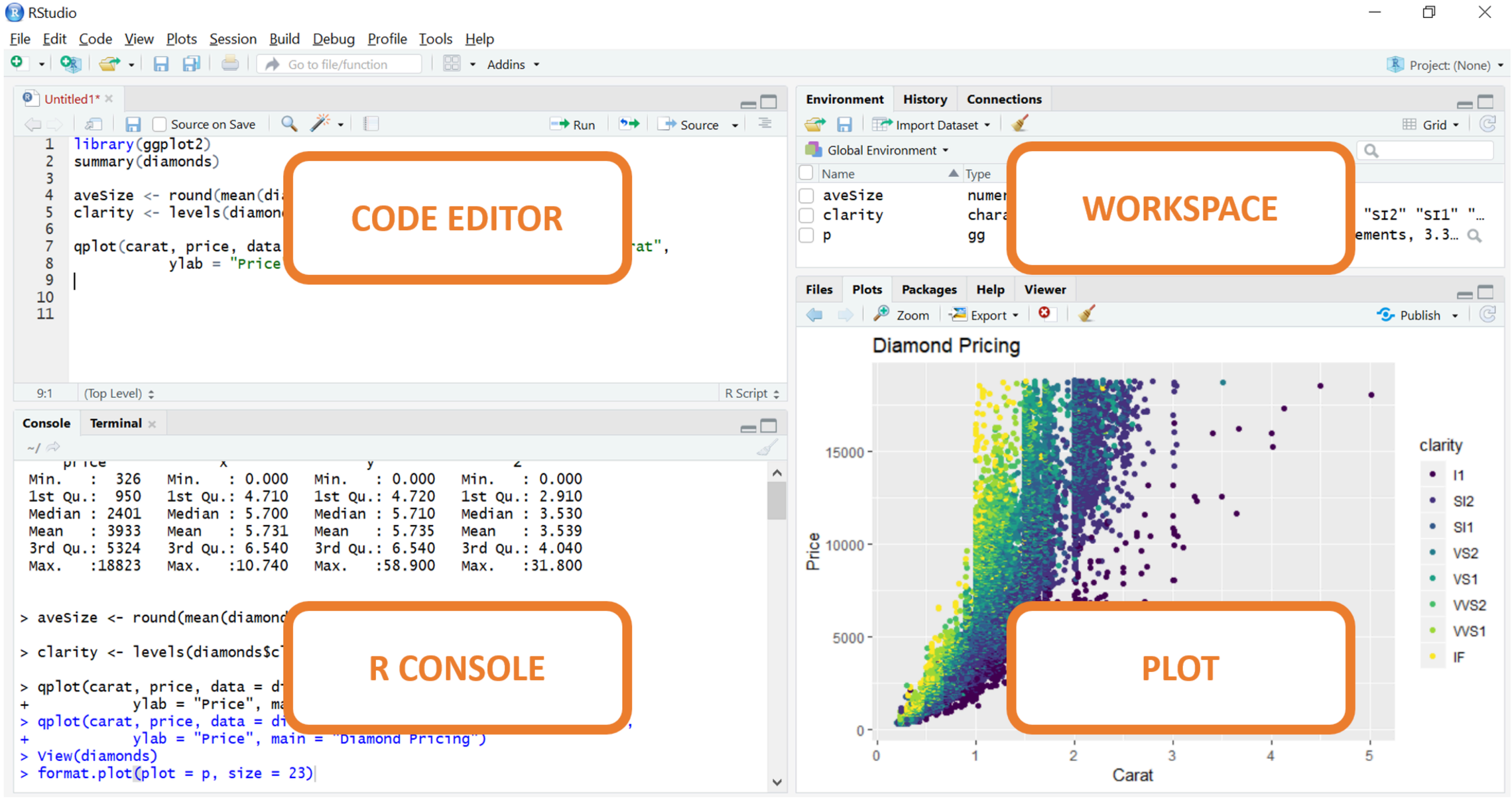
Review

The R Language: An Engine for Bioinformatics and Data Science

Federico M. Giorgi [1,*], Carmine Ceraolo [1,2] and Daniele Mercatelli [1]



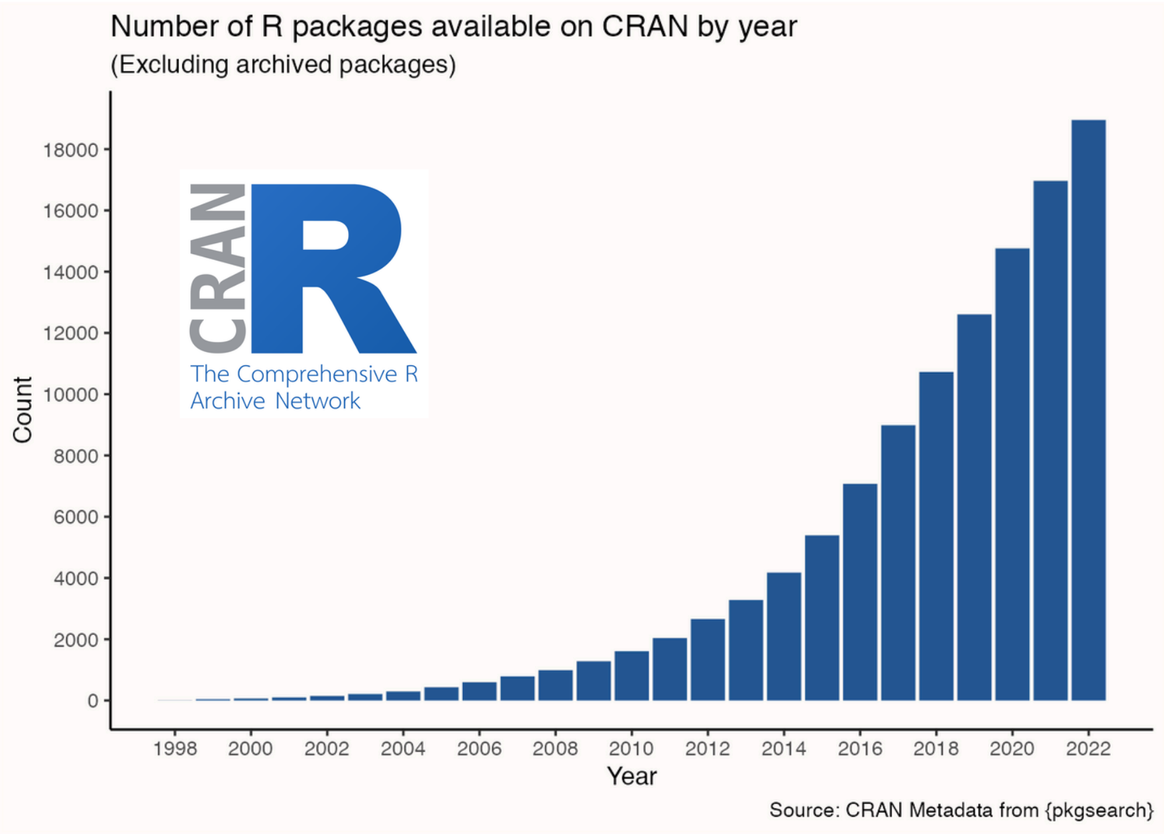
Un vistazo a RStudio®



Librerías e importación de datos

```
install.packages("package1")
library(package1)
```

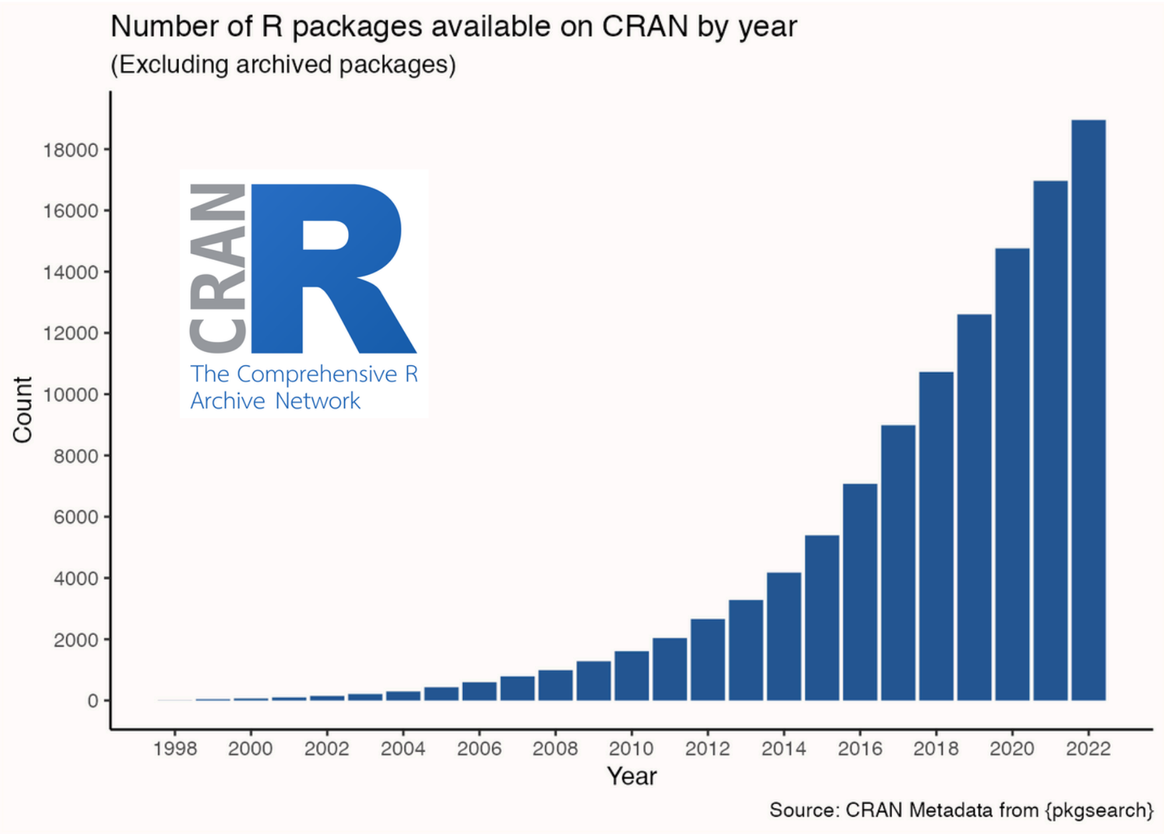
Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS



Librerías e importación de datos

```
install.packages("package1")
library(package1)
```

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS



Librerías e importación de datos

```
read.csv("file.csv", ...)
```


```
read.table("file.txt", ...)
```


```
install.packages("readxl")
library(readxl)

read_excel("file.xlsx", ...)
```

Contenidos

Operadores aritméticos

Operador	Descripción	Ejemplo	Resultado
+	Suma	23 + 16	
–	Resta	12 - 7	
*	Multiplicación	8 * 6	
/	División	5 / 2	
^ ó **	Potenciación	10 ** 3	
%%	Módulo	5 %% 3	
%/%	División entera	8 %/% 6	

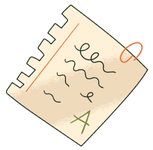
ORDEN	OPERADORES
1	^
2	* /
3	+ -
4	< > <= >= == !=
5	!
6	&
7	|
8	<-

Operadores aritméticos

Operador	Descripción	Ejemplo	Resultado
+	Suma	23 + 16	39
–	Resta	12 - 7	5
*	Multiplicación	8 * 6	48
/	División	5 / 2	2.5
^ ó **	Potenciación	10 ** 3	1000
%%	Módulo	5 %% 3	2
%/%	División entera	8 %/% 6	1

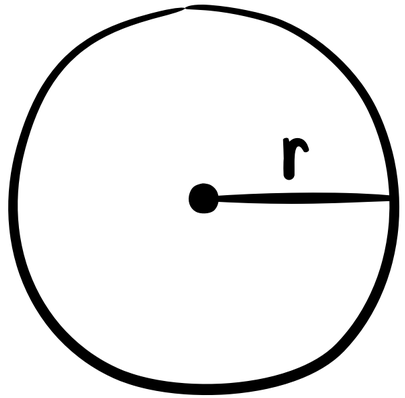
ORDEN	OPERADORES
1	^
2	* /
3	+ -
4	< > <= >= == !=
5	!
6	&
7	|
8	<-

Ejercicios

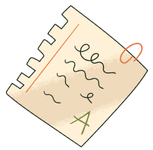


Calcula el área de un círculo con radio 5

?Constants



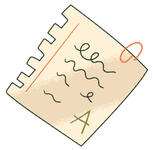
$$A = \pi r^2$$



Contenido de Guanina-Citosina (GC) en una secuencia de ADN

```
seq <- “GCCATGAGGGTC”
```

Ejercicios

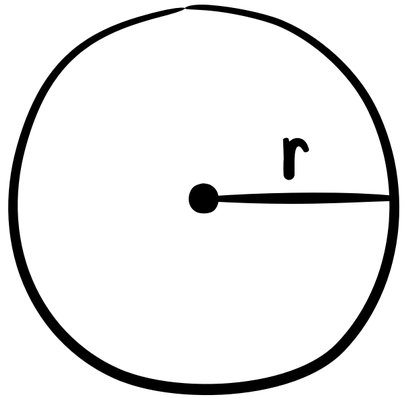


Calcula el área de un círculo con radio 5

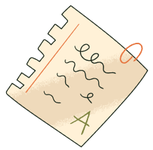
```
3.1416 * 5^2
[1] 78.54
```

```
pi * 5^2
[1] 78.53982
```

?Constants



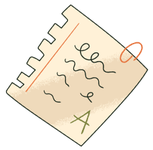
$$A = \pi r^2$$



Contenido de Guanina-Citosina (GC) en una secuencia de ADN

```
seq <- "GCCATGAGGGTC"
```

Ejercicios



Calcula el área de un círculo con radio 5

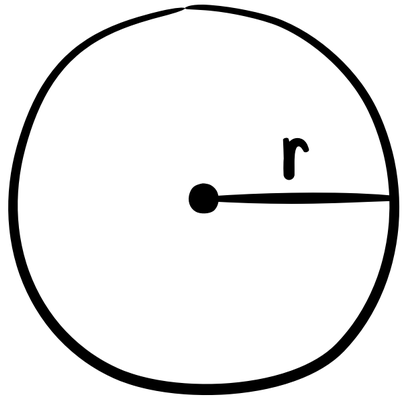
```
3.1416 * 5^2
```

```
[1] 78.54
```

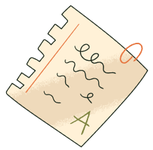
```
pi * 5^2
```

```
[1] 78.53982
```

?Constants



$$A = \pi r^2$$



Contenido de Guanina-Citosina (GC) en una secuencia de ADN

```
seq <- "GCCATGAGGGTC"
content_GC <- 8 * 100 / 12
```

```
[1] 66.66667
```

?nchar

Operadores relacionales y lógicos

TRUE or FALSE?

```
vector <- c(1,3,5,7,9)
```

Operador	Descripción	Ejemplo	Resultado
`< y >`	menor/mayor a	vector > 3	
`<= y >=`	menor/mayor o igual a	vector >= 3	
`==`	exactamente igual a	vector == 3	
`!=`	no es igual a	vector != 3	
`x|y`	x OR y	vector [vector >= 5 | vector < 3]	
`x & y`	x AND y	vector [vector >=3 & vector < 7]	
`x %in% y`	x IN y	vector [vector %in% c(3,7)]	

Operadores relacionales y lógicos

TRUE or FALSE?

```
vector <- c(1,3,5,7,9)
```

Operador	Descripción	Ejemplo	Resultado
`< y >`	menor/mayor a	vector > 3	FALSE, FALSE, TRUE, TRUE, TRUE
`<= y >=`	menor/mayor o igual a	vector >= 3	FALSE, TRUE, TRUE, TRUE, TRUE,
`==`	exactamente igual a	vector == 3	FALSE, TRUE, FALSE, FALSE, FALSE
`!=`	no es igual a	vector != 3	TRUE, FALSE, TRUE, TRUE, TRUE
`x|y`	x OR y	vector [vector >= 5 | vector < 3]	1, 5, 7, 9
`x & y`	x AND y	vector [vector >=3 & vector < 7]	3, 5
`x %in% y`	x IN y	vector [vector %in% c(3,7)]	3, 7

Contenidos

OOP

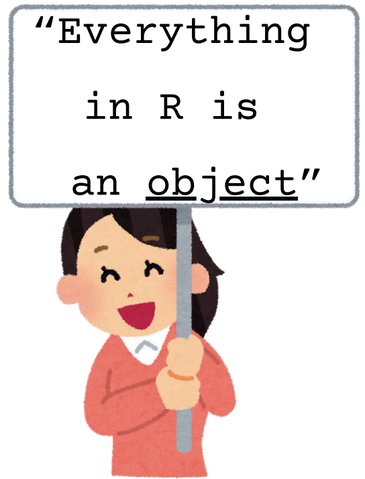
Object-Oriented Programming

Clases: Naturaleza de un objeto

Métodos: funciones ya definidas para clases específicas

Objetos

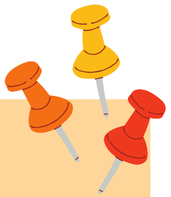
- **Vector**
- Matriz
- **Data.frame**
- Array
- List



Otro vistazo a RStudio®

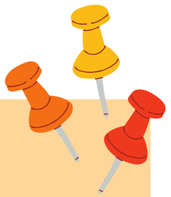
	Dimensions	Mode (data "type")	Example
Vector	1 	Identical	`c(10,0.2,34,48,53)`

Tips para operar con vectores



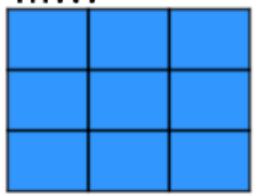
Función	Ejemplo	Resultado
c(a, b, ...)	c(1, 3, 5, 7, 9)	
a:b	1:10	
seq(from, to, by, length.out)	seq(from = 10, to = 50, by = 5)	
rep(x, times, each, length.out)	rep(c(4, 5), times = 3, each = 2)	

Tips para operar con vectores



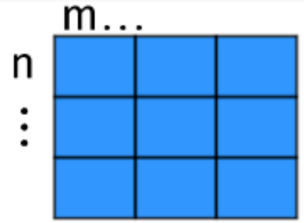
Función	Ejemplo	Resultado
c(a, b, ...)	c(1, 3, 5, 7, 9)	1, 3, 5, 7, 9
a:b	1:10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
seq(from, to, by, length.out)	seq(from = 10, to = 50, by = 5)	10, 15, 20, 25, 30, 35, 40, 45, 50
rep(x, times, each, length.out)	rep(c(4, 5), times = 3, each = 2)	4, 4, 5, 5, 4, 4, 5, 5, 4, 4, 5, 5

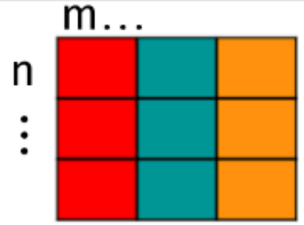
Otro vistazo a RStudio®

	Dimensions	Mode (data "type")	Example
Vector	1 	Identical	`c(10,0.2,34,48,53)`
Matrix	n ⋮ 	Identical	`matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3)`

Otro vistazo a RStudio®

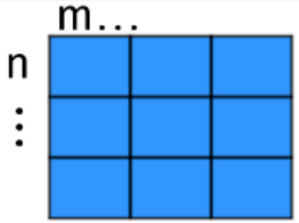
	Dimensions	Mode (data "type")	Example
Vector		Identical	`c(10,0.2,34,48,53)`

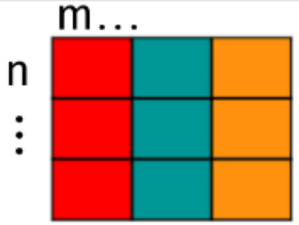
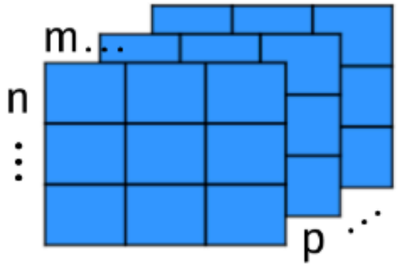
Matrix		Identical	`matrix(c(1,2,3,`
			`11,12,13),`
			`nrow = 2,`
			`ncol = 3)`

Data frame		Can be different	`data.frame(x = 1:3,` `y = 5:7)`

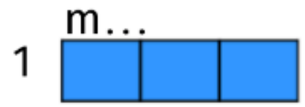
Otro vistazo a RStudio®

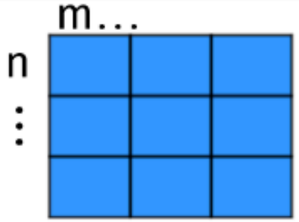
	Dimensions	Mode (data "type")	Example
Vector		Identical	`c(10,0.2,34,48,53)`

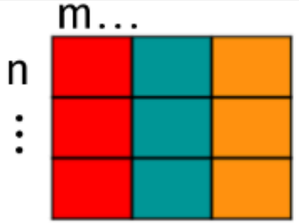
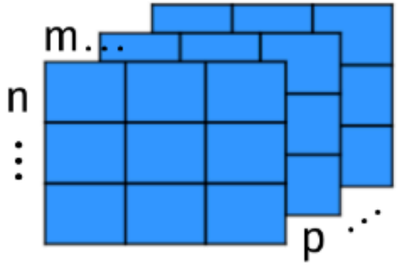
Matrix		Identical	`matrix(c(1,2,3,`
			`11,12,13),`
			`nrow = 2,`
			`ncol = 3)`

Data frame		Can be different	`data.frame(x = 1:3,` `y = 5:7)`
Array		Identical	`array(data = 1:3,` `dim = c(2,4,2))`

Otro vistazo a RStudio®

	Dimensions	Mode (data "type")	Example
Vector		Identical	`c(10,0.2,34,48,53)`

Matrix		Identical	`matrix(c(1,2,3,`
			`11,12,13),`
			`nrow = 2,`
			`ncol = 3)`

Data frame		Can be different	`data.frame(x = 1:3,` `y = 5:7)`
Array		Identical	`array(data = 1:3,` `dim = c(2,4,2))`
List	Vector Matrix Data frame Array	Can be different	`list(x = cars[,1],` `y = cars[,2])`

Fuente: https://r.qcbs.ca/workshop01/pres-en/workshop01-pres-en.html#100

<!--Estudio IMIBIC-->

Enhorabuena {

<"Has superado la primera sesión"/>

}

UCAIB BIOINFORMÁTICA Y BIOESTADÍSTICA (IMIBIC)

<!--Estudio IMIBIC-->

R para Todos: El poder para explotar tus Datos {

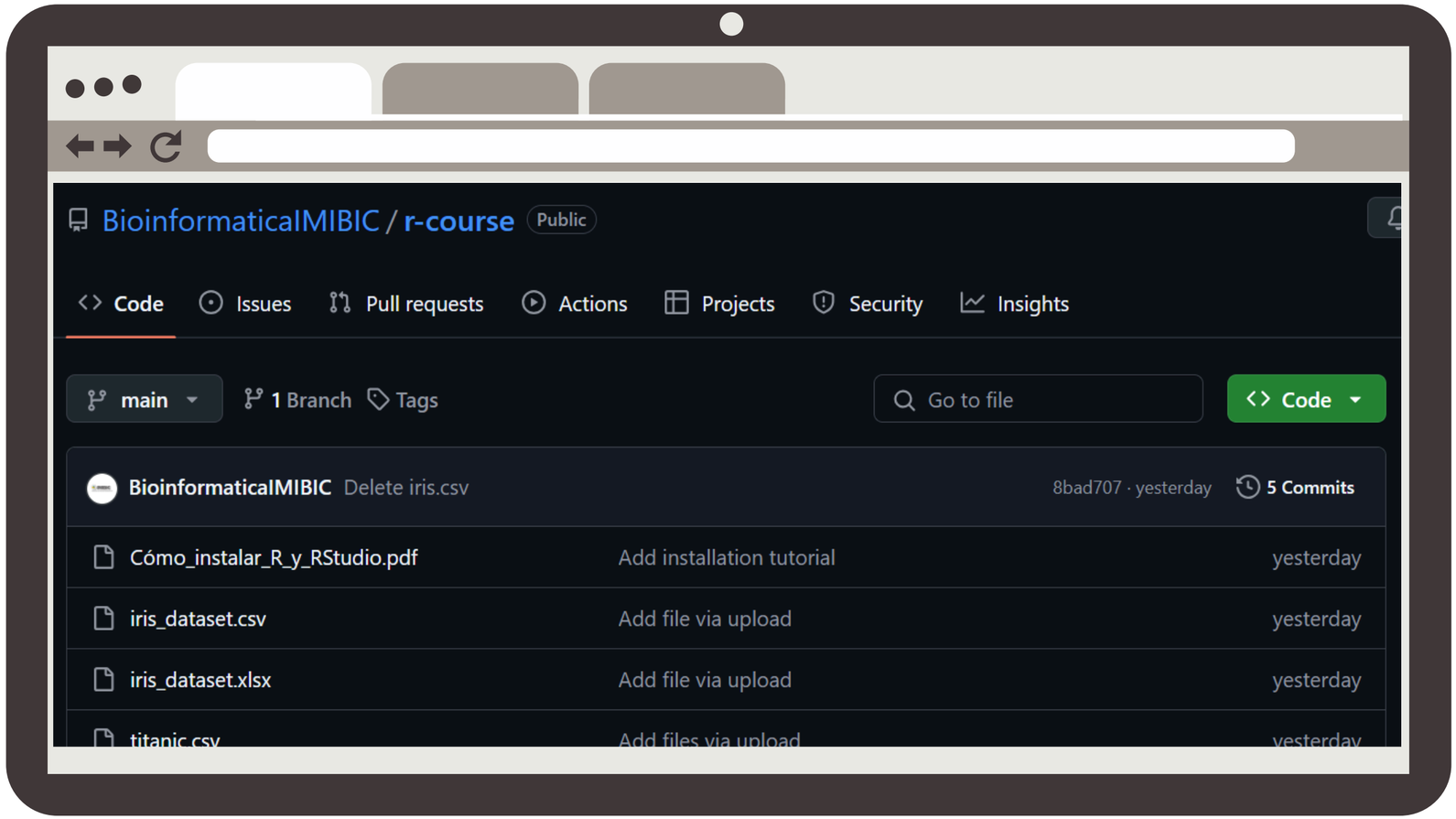
<Por="Adrián Santiago Ortiz"/>

}

Contenidos

Sesión 2

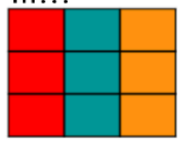
Material disponible



https://github.com/BioinformaticaIMIBIC/r-course

Repaso de la sesión 1

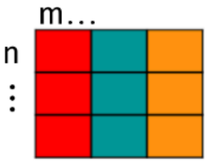
R es un lenguaje orientado a objetos

	Dimensions	Mode (data "type")	Example
Vector	1 m… 	Identical	`c(10,0.2,34,48,53)`
Data frame	n ⋮ m… 	Can be different	`data.frame(x = 1:3, y = 5:7)`

Repaso de la sesión 1

R es un lenguaje orientado a objetos

	Dimensions	Mode (data "type")	Example
Vector	1 	Identical	`c(10,0.2,34,48,53)`

	Dimensions	Mode (data "type")	Example
Data frame	n 	Can be different	`data.frame(x = 1:3, y = 5:7)`

Instalar paquetes y cargar librerías

CRAN R
The Comprehensive R Archive Network

```
install.packages("package1")
library(package1)
```

Repaso de la sesión 1

R es un lenguaje orientado a objetos

	Dimensions	Mode (data "type")	Example
Vector	1 	Identical	`c(10,0.2,34,48,53)`

	Dimensions	Mode (data "type")	Example
Data frame	n ⋮ 	Can be different	`data.frame(x = 1:3, y = 5:7)`

Instalar paquetes y cargar librerías

```
install.packages("package1")
library(package1)
```

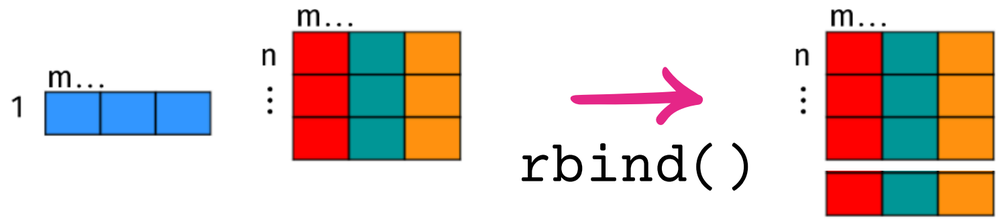
Cargar nuestros datos (tablas)

```
library(readxl)
read_excel("file.xlsx", ...)
```

```
read.table("file.txt", ...)
read.csv("file.csv", ...)
```

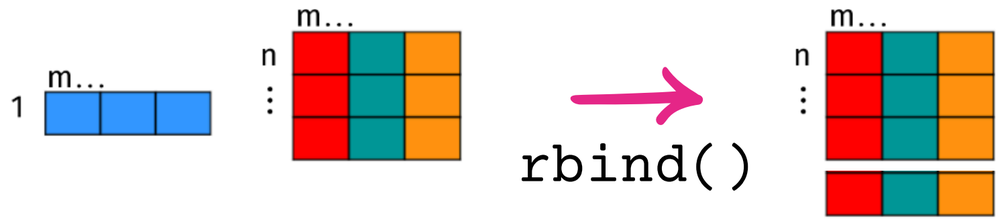
Repaso de la sesión 1

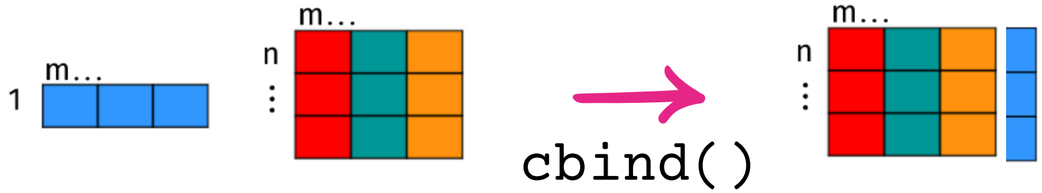
Añadir filas/columnas de un data.frame



Repaso de la sesión 1

Añadir filas/columnas de un data.frame





Petal.Width	Species
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa

Petal.Width	Species	Country
0.2	setosa	Spain
0.2	setosa	France
0.2	setosa	Italy
0.2	setosa	France

```
cbind(df, c("Spain", "France",
            "Italy", "France"))
```

```
df$Country <- c("Spain", "France",
                "Italy", "France")
```

Contenidos

Sesión 2

Contenidos

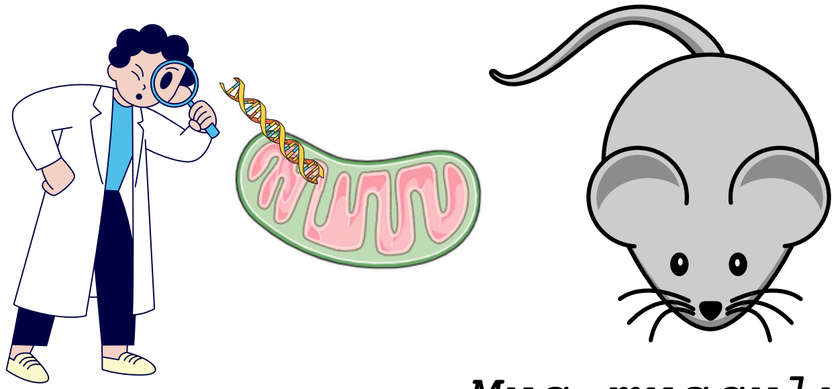
Sesión 2

Ejemplo final (resuelto)

¿Cuántas A, T, G y C hay en el genoma mitocondrial del ratón?
¿Cuál es su contenido GC?

Archivo fasta:

```
>MT dna_rm:chromosome chromosome:GRCm39:MT:1:16299:1 REF
GTTAATGTAGCTTAATAACAAAGCAAAGCACTGAAAATGCTTAGATGGATAATTGTATCC
CATAAACACAAAGGTTTGGTCCTGGCCTTATAATTAATTAGAGGTAAAATTACACATGCA
AACCTCCATAGACCGGTGTAAAATCCCTTAAACATTTACTTAAAATTTAAGGAGAGGGTA
TCAAGCACATTAAAATAGCTTAAGACACCTTGCCTAGCCACACCCCCACGGGACTCAGCA
GTGATAAATATTAAGCAATAAACGAAAGTTTGACTAAGTTATACCTCTTAGGGTTGGTAA
ATTTCGTGCCAGCCACCGCGGTCATACGATTAACCCAAACTAATTATCTTCGGCGTAAAA
CGTGTCAACTATAAATAAATAAATAGAATTAAAATCCAACTTATATGTGAAAATTCATTG
TTAGGACCTAAACTCAATAACGAAAGTAATTCTAGTCATTTATAATACACGACAGCTAAG
ACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCATAAACCTAAATAATTAAATTTA
ACAAAACTATTTGCCAGAGAACTACTAGCCATAGCTTAAAACTCAAAGGACTTGGCGGTA
CTTTATATCCATCTAGAGGAGCCTGTTCTATAATCGATAAACCCCGCTCTACCTCACCAT
CTCTTGCTAATTCAGCCTATATACCGCCATCTTCAGCAAACCCTAAAAAGGTATTAAAGT
AAGCAAAAGAATCAAACATAAAAACGTTAGGTCAAGGTGTAGCCAATGAAATGGGAAGAA
ATGGGCTACATTTTCTTATAAAAGAACATTACTATACCCTTTATGAAACTAAAGGACTAA
GGAGGATTTAGTAGTAAATTAAGAATAGAGAGCTTAATTGAATTGAGCAATGAAGTACGC
ACACACCGCCCGTCACCCTCCTCAAATTAAATTAAACTTAACATAATTAATTTCTAGACA
TCCGTTTATGAGAGGAGATAAGTCGTAACAAGGTAAGCATACTGGAAAGTGTGCTTGGAA
TAATCATAGTGTAGCTTAATATTAAAGCATCTGGCCTACACCCAGAAGATTTCATGACCA
ATGAACACTCTGAACTAATCCTAGCCCTAGCCCTACACAAATATAATTATACTATTATAT
AAATCAAAACATTTATCCTACTAAAAGTATTGGAGAAAGAAATTCGTACATCTAGGAGCT
ATAGAACTAGTACCGCAAGGGAAAGATGAAAGACTAATTAAAAGTAAGAACAAGCAAAGA
TTAAACCTTGTACCTTTTGCATAATGAACTAACTAGAAAACTTCTAACTAAAAGAATTAC
AGCTAGAAACCCCGAAACCAAACGAGCTACCTAAAAACAATTTTATGAATCAACTCGTCT
ATGTGGCAAAATAGTGAGAAGATTTTTAGGTAGAGGTGAAAAGCCTAACGAGCTTGGTGA
TAGCTGGTTACCCAAAAAATGAATTTAAGTTCAATTTTAAACTTGCTAAAAAAACAACAA
AATCAAAAAGTAAGTTTAGATTATAGCCAAAAGAGGGACAGCTCTTCTGGAACGGAAAAA
ACCTTTAATAGTGAATAATTAACAAAACAGCTTTTAACCATTGTAGGCCTAAAAGCAGCC
ACCAATAAAGAAAGCGTTCAAGCTCAACATAAAATTTCAATTAATTCCATAATTTACACC
AACTTCCTAAACTTAAAATTGGGTTAATCTATAACTTTATAGATGCAACACTGTTAGTAT
GAGTAACAAGAATTCCAATTCTCCAGGCATACGCGTATAACAACTCGGATAACCATTGTT
AGTTAATCAGACTATAGGCAATAATCACACTATAAATAATCCACCTATAACTTCTCTGTT
AACCCAACACCGGAATGCCTAAAGGAAAGATCCAAAAAGATAAAAGGAACTCGGCAAACA
AGAACCCCGCCTGTTTACCAAAAACATCACCTCTAGCATTACAAGTATTAGAGGCACTGC
```

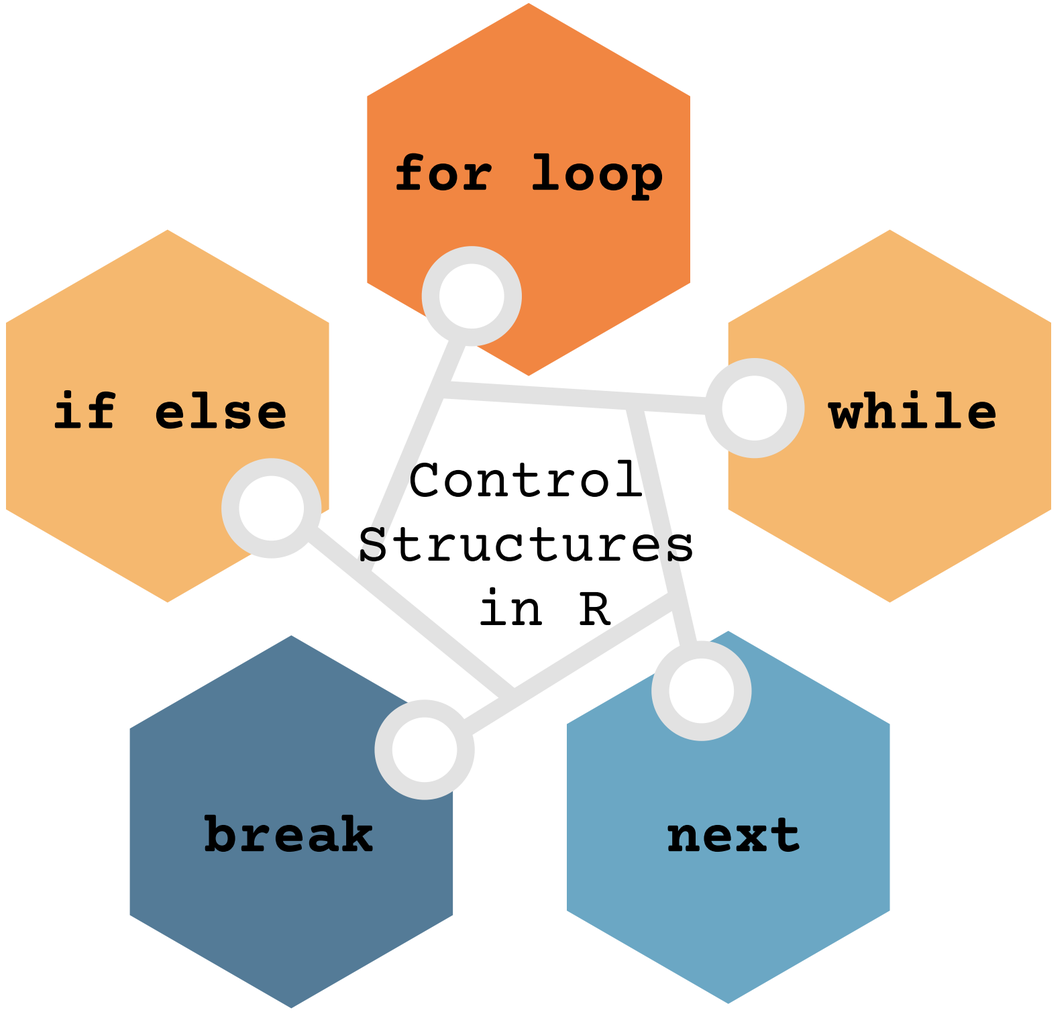


Mus musculus

Las herramientas que vamos a ir viendo nos ayudarán.

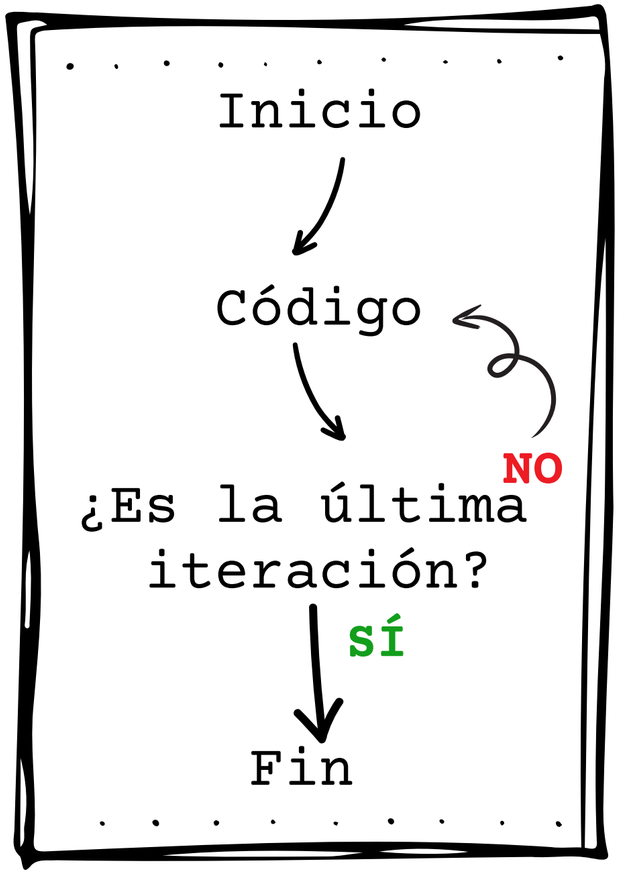
¿Por qué utilizarlas?

- Automatización
- Manejo de grandes datos
- Flexibilidad
- Código eficiente



Sintaxis del bucle for

se utiliza para **repetir** un bloque de código un número específico de veces



Sintaxis del bucle for

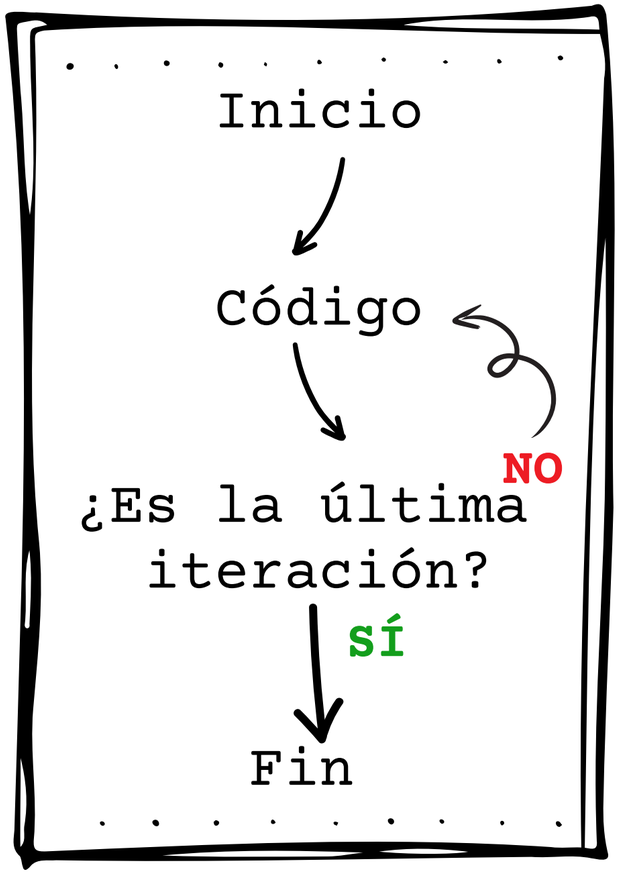
se utiliza para **repetir** un bloque de código un número específico de veces

var
Se toman elementos de iterable

iterable
Objeto R
(**vector**, lista, etc)

```
for (var in iterable) {
    statment
    statment
    ...
}
```

cuerpo
líneas de código a ejecutar



Ejercicio

Muestra los números del 1 al 10 por pantalla

Usa el comando **print** para mostrar un valor por pantalla

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

var
Se toman elementos de iterable

iterable
Objeto R
(**vector**, lista, etc)

```
for (var in iterable) {
    statment
    statment
    ...
}
```

cuerpo
líneas de código a ejecutar

Ejercicio

Muestra los números del 1 al 10 por pantalla

i valdrá 1,2,3,...,10 en cada iteración

Usa el comando **print** para mostrar un valor por pantalla

```
for (i in 1:10) {
   print (i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

var
Se toman elementos de iterable

iterable
Objeto R
(**vector**, lista, etc)

```
for (var in iterable) {
    statment
    statment
    ...
}
```

cuerpo
líneas de código a ejecutar

IF / ELSE / ELSE IF /

Para ejecutar un bloque de código **solo si** se cumple una condición específica

condition
evalúa si es
TRUE o FALSE

```
if (condition) {
    statment
    statment
    ...
}
```

cuerpo
se ejecuta si
condition es TRUE

TRUE or FALSE?

Operador	Descripción
< y >	menor/mayor a
<= y >=	menor/mayor o igual a
==	exactamente igual a
!=	no es igual a
x|y	x OR y
x & y	x AND y
x %in% y	x IN y

IF / ELSE / ELSE IF /

Para ejecutar un bloque de código **solo si** se cumple una condición específica

condition
evalúa si es
TRUE o FALSE

```
x <- 10

if (x > 5) {
  print("x es mayor que 5")
}
```

cuerpo
se ejecuta si
condition es TRUE

TRUE or FALSE?

Operador	Descripción
< y >	menor/mayor a
<= y >=	menor/mayor o igual a
==	exactamente igual a
!=	no es igual a
x|y	x OR y
x & y	x AND y
x %in% y	x IN y

Ejercicio

Muestra los números del 1 y el 10 diferentes a 5

```
for (                ) {


    }
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

!=	no es igual a

IF / **ELSE** / ELSE IF /

condition
evalúa si es
TRUE o FALSE

```
if (condition) {
    statment
    statment
    ...
}
```

Rama TRUE
Se ejecuta si
condition es TRUE

IF / **ELSE** / ELSE IF /

condition
evalúa si es
TRUE o FALSE

```
if (condition) {
    statment
    statment
    ...
} else {
    statment
    statment
    ...
}
```

Rama TRUE
Se ejecuta si
condition es TRUE

Rama FALSE
Se ejecuta si
condition es FALSE

Ejercicio

Evalua con if/else si un número es mayor o menor a cero

```
# Define un número


# Usa if / else para evaluar el número
```

IF / ELSE / **ELSE IF** /

condition
evalúa si es
TRUE o FALSE

```
if (condition) {
    statment
    statment
    ...
}
```

1ª condición
Se ejecuta si
condition es TRUE

IF / ELSE / **ELSE IF** /

condition
evalúa si es
TRUE o FALSE

```
if (condition) {
    statment
    statment
    ...
} else if (condition2) {
    statment
    statment
    ...
}
```

1ª condición
Se ejecuta si
condition es TRUE

Nueva condición
Se ejecuta si
la previa es **FALSE**

IF / ELSE / **ELSE IF** /

condition
evalúa si es
TRUE o FALSE

```
if (condition) {
    statment
    statment
    ...
} else if (condition2) {
    statment
    statment
    ...
} else {
    statment
    statment
    ...
}
```

1ª condición
Se ejecuta si
condition es TRUE

Nueva condición
Se ejecuta si
la previa es **FALSE**

Rama FALSE
Se ejecuta si
ninguna es TRUE

Ejercicios

Evalua con if/else if / else si un número es mayor, menor o igual a 0

```
# Define un número
```

Ejercicios

Evalua con if/else if / else si un número es mayor, menor o igual a 0

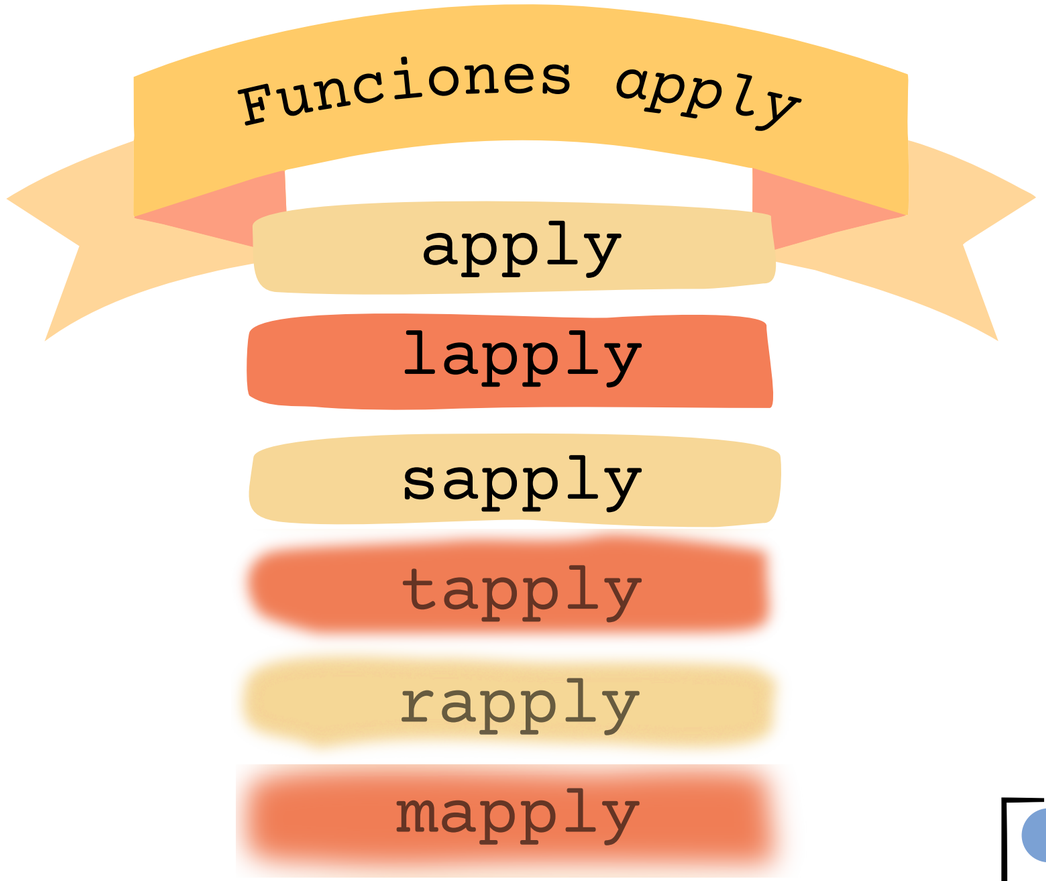
```
# Define un número
num <- -5

if (num > 0) {
    print("El número es positivo")

} else if (num < 0) {
      print("El número es negativo")

} else {
      print("El número es cero")
}
```

Optimización de bucles

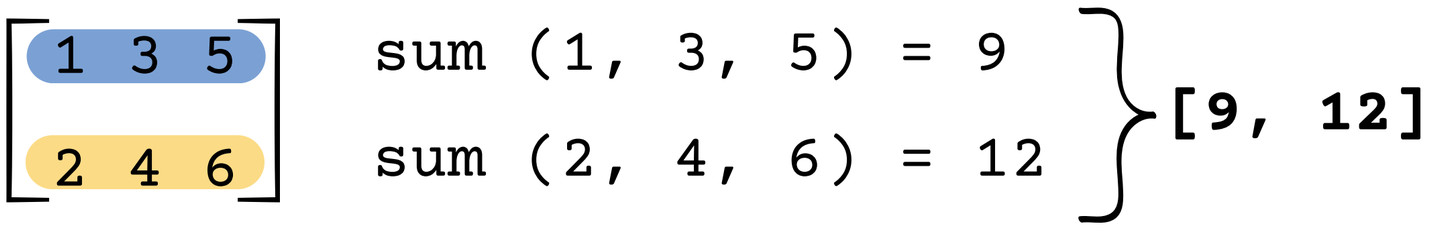


Por fila (1)

apply(m, 1, sum)

objeto: m

función: sumar



<!--Estudio IMIBIC-->

Enhorabuena {

<"Has superado la segunda sesión"/>

}

UCAIB BIOINFORMÁTICA Y BIOESTADÍSTICA (IMIBIC)