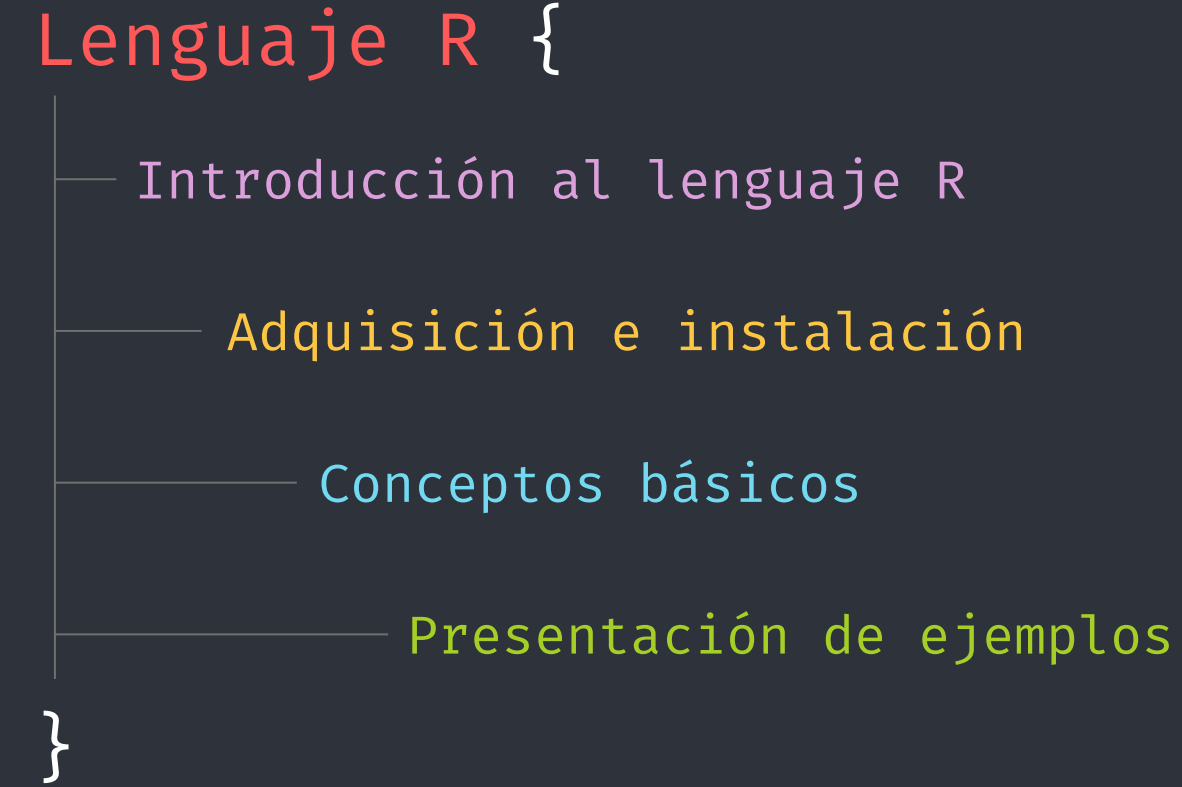
```
1
2  Programming_Language ← “R”
3
4
5
6
7
8
9  Print(‘COMP 3800’)
10 #Profesora: Elizabeth Maldonado
11
12 Integrantes ← list (Génesis Ojeda, Adier Maldonado)
13
14
```

Objetivos


Lenguaje R {
Introducción al lenguaje R
Adquisición e instalación
Conceptos básicos
Presentación de ejemplos
}

Historia

R, creado en 1993 por Ross Ihaka y Robert Gentleman, fue inspirado por el lenguaje S de los Laboratorios AT&T Bell y diseñado específicamente para facilitar el análisis estadístico y la manipulación de datos. La primera versión oficial de R se lanzó en 1995. Como software de código abierto, R atrajo una comunidad de desarrolladores que, mediante colaboraciones, expandieron su funcionalidad con una extensa variedad de paquetes que lo convirtieron en una herramienta poderosa y versátil. La organización R-Project.org gestiona su desarrollo y estandarización, haciendo de R un recurso clave en el ámbito académico y en sectores empresariales enfocados en análisis de datos y estadísticas.

Cortesía de RPubs:
https://rpubs.com/Alex-key/838868

Características

El lenguaje **R** es especialmente útil en el ámbito de la estadística, el análisis de datos y la ciencia de datos. Permite realizar cálculos complejos, manipular grandes volúmenes de datos y generar gráficos de alta calidad. Sus principales usuarios incluyen:

- **Estadísticos y científicos de datos** que necesitan realizar análisis cuantitativos avanzados.
- **Investigadores** de disciplinas como biología, economía y ciencias sociales, donde el análisis y visualización de datos son esenciales.
- **Profesionales del análisis de datos en empresas**, que buscan obtener valor de grandes conjuntos de datos.

R se distingue por su capacidad para realizar cálculos estadísticos avanzados y por la calidad de sus gráficos, lo que lo convierte en una herramienta fundamental para el análisis de datos y la visualización.

Ventajas

Gran comunidad
Amplia comunidad que contribuye con herramientas, bibliotecas y proporciona apoyo.

Personalizable
Altamente personalizable e integrable con otros lenguajes y entornos.

Gran compatibilidad
Compatible con otros lenguajes, entornos y herramientas.

Paquetes potentes
Gran biblioteca de paquetes para manipulación de datos, modelado estadístico y machine learning.

Grandes capacidades gráficas
Capacidades gráficas para crear visualizaciones de alta calidad.

Adier Maldonado

Desventajas

Curva de aprendizaje pronunciada
Curva de aprendizaje difícil para principiantes sin experiencia en programación.

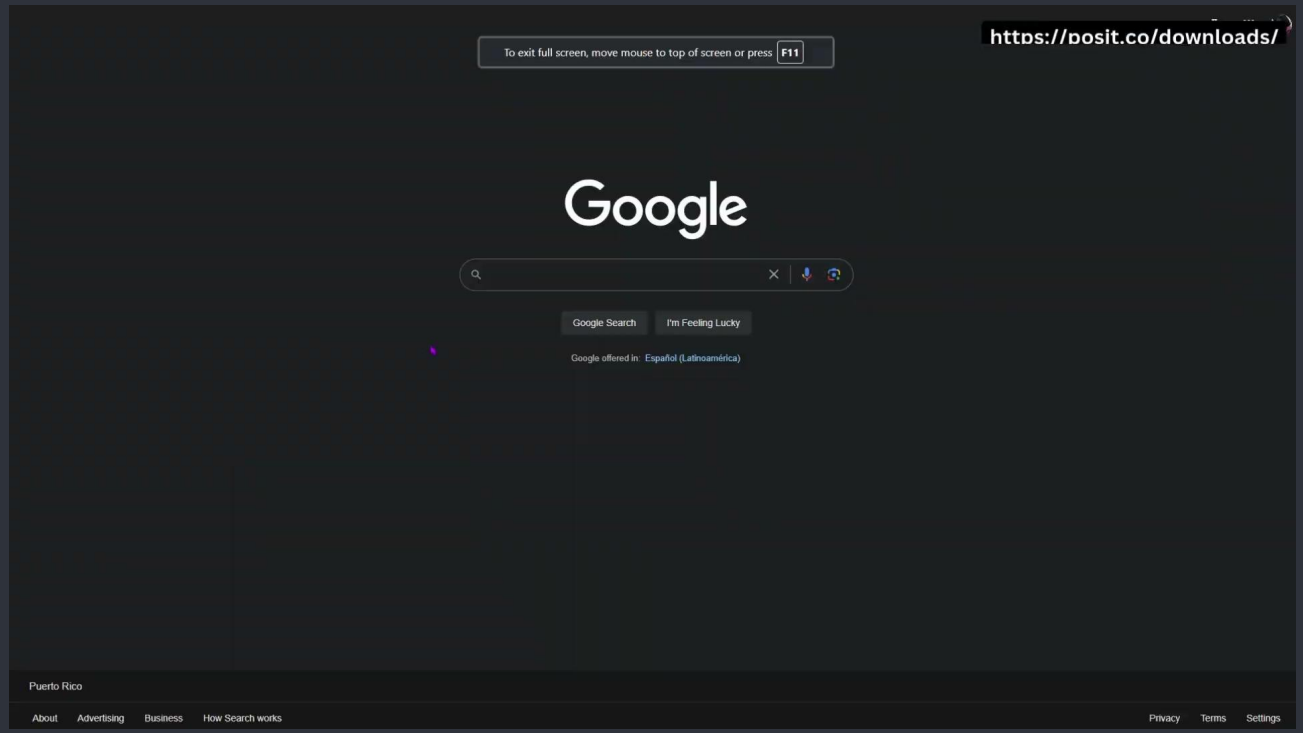
Velocidad lenta
Más lento en comparación con lenguajes compilados como C++ o Python.

Falta de soporte para procesamiento paralelo
Sin soporte nativo robusto para el procesamiento paralelo, limitado con grandes volúmenes de datos.

Gestión de la memoria
Problemas de rendimiento al trabajar con grandes conjuntos de datos.

Dependencia de paquetes
Dependencia de paquetes externos puede causar problemas de compatibilidad y versiones.

Instalación y adquisición



https://drive.google.com/file/d/10ebpkd7t0DWljmL9p7w7CiOZmnapbhZY/view?usp=sharing

RStudio
https://posit.co/

- Desktop Open source
- Desktop Pro

Rstudio Cloud
https://posit.cloud/

- Gratuita
- Paga

Instalación y adquisición

Apps:

1. Rstudio: https://cloud.rstudio.com

Opción portable:

1. Jupyter Notebook with R: https://jupyter.org/install

Servicios en la nube:

1. Rstudio cloud: https://posit.cloud/
2. Google collab with R: https://colab.research.google.com/gist/xiaonilee/cfd263ebef98b1d60aa57e1ebc0ec29d/rincolab.ipynb

Recursos para aprender R

Aplicaciones:

1. Learn R programming:
   https://play.google.com/store/apps/details?id=rprogramming.r.programming.coding.learn.analytics.data.analyst.bi.rpa.dataanalytics&hl=en-US

Massive open online course (MOOC):

1. Data science: R (Harvard):
   https://pll.harvard.edu/course/data-science-r-basics
2. R programming (Coursera):
   https://www.coursera.org/learn/r-programming
3. Introduction R (DataCamp):
   https://www.datacamp.com/courses/free-introduction-to-r

Génesis Ojeda

Recursos para aprender R

Libro:

1. An introduction on R: Notes on R: A programming Environment for Data Analysis and Graphics

   ISBN: 0954161742

   Versión: 4.4.1

   Lanzamiento: Junio 14, 2024

Páginas web:

1. W3schools - R: https://www.w3schools.com/r/default.asp

Génesis Ojeda

Tipos de datos básicos

Tipo de Dato	Descripción	Declaración de Variable	Ejemplo
Numeric	Números reales o decimales	var <- 2.59	pi_value <- 3.14159
Integer	Números enteros (se usa L)	var <- 10L	edad <- 25L
Character	Cadenas de texto (string)	var <- "Hola"	nombre <- "R"
Logical	Valores booleanos: TRUE o FALSE	var <- TRUE	is_active <- FALSE
Complex	Números complejos	var <- 2 + 3i	z <- 1 + 4i

Génesis Ojeda

Manejo de datos en R

Concepto	Descripción	Declaración de Variable	Ejemplo
Comentarios	Texto que no es ejecutado, usado para describir el código	`# comentario`	`# Esto es un comentario`
Variables	Almacenamiento de datos que pueden cambiar	`variable <- valor`	`x <- 10`
Constantes	Valores fijos que no cambian	`CONST <- valor`	`PI <- 3.14159`

R es un lenguaje Case sensitive, por lo tanto ‘variable y ‘Variable’ no son lo mismo.

Génesis Ojeda

Manejo de datos en R

Concepto	Descripción	Declaración de Variable	Ejemplo	Resultado
Inputs	Captura de datos del usuario	`variable <- readline(prompt = "mensaje")`	`nombre <- readline(prompt = "Ingrese su nombre: ")`	(Depende del input del usuario)
Outputs (print)	Muestra de datos al usuario	`print(variable)`	`print("Hola, Mundo!")`	Hola, Mundo!
Outputs (cat)	Muestra de datos concatenados al usuario	`num <- 24.5` `cat(variable, "\n")`	`cat("Resultado:", num, "\n")`	Resultado: 24.5
Outputs (paste)	Muestra de datos concatenados con separador	`num <- 5` `paste(string, variable, string)`	`paste("Tengo", num, "manzanas")`	Tengo 5 manzanas

Génesis Ojeda

Operadores lógicos

Operador	Descripción
&	**Operador lógico AND elemento por elemento**: Devuelve TRUE si ambos elementos son TRUE.
&&	**Operador lógico AND**: Devuelve TRUE si ambas expresiones son TRUE.
|	**Operador lógico OR elemento por elemento**: Devuelve TRUE si una de las expresiones es TRUE.
||	**Operador lógico OR**: Devuelve TRUE si una de las expresiones es TRUE.
!	**Operador lógico NOT**: Devuelve FALSE si la expresión es TRUE.

Operadores matemáticos

Operador	Nombre	Ejemplo
+	Suma	x + y
-	Resta	x - y
*	Multiplicación	x * y
/	División	x / y
^	Exponente	x ^ y
%%	Módulo (Resto de la división)	x %% y
%/%	División entera	x %/% y

If statement

Un **if statement sencillo** evalúa una sola condición. Si la condición es verdadera (**TRUE**), el código dentro del bloque **if** se ejecuta. Si es falsa, no se ejecuta.

```
x ← 5
if(x > 0){
print("Positive number")
}
```

If-Else statements

Este es un **if** básico con una opción **else**. Si la condición del **if** es falsa, se ejecuta el código dentro del bloque **else**.

```
x ← 3

if(x > 5){
print("x es mayor que 5")
} else {
    print(“no es mayor que 5”)
}
```

Nested If statement

Un **nested if** ocurre cuando tienes un **if** dentro de otro **if**. Esto significa que una condición se evalúa solo si la primera condición es verdadera.

```
x ← 10
y ← 20

if(x > 5){
    if(y > 15) {
        print(“x es mayor que 5 y y es mayor que 15”)
    }
}
```

Adier Maldonado

Double If statement

El **else if** te permite evaluar múltiples condiciones. Si la primera condición es falsa, se evalúa la siguiente condición. Puedes usar varios **else if** seguidos.

```
x ← 10

if(x > 15) {
    print(“x es mayor que 15”)
}   else if (x == 10) {
    print(“x es mayor que 10”)
}   else {
    print(“x es menor que 10”)
}
```

Resumen de los statements

- **if statement**: Evalúa una condición, ejecuta el código si es verdadera.
- **if-else**: Evalúa una condición, ejecuta un bloque de código si es verdadera y otro si es falsa.
- **nested if**: Un **if** dentro de otro, donde se evalúan condiciones de forma jerárquica.
- **else-if (double if)**: Permite evaluar múltiples condiciones, ejecutando el código correspondiente a la primera condición verdadera.

Ciclos

Concepto	Descripción	Ejemplo de Declaración	Ejemplo de Código	Resultado
For	Iteración sobre una secuencia.	`for (variable in sequence){ #código a ejecutar }`	`for (i in 1:3) { print(paste("Número:", i)) }`	"Número: 1" "Número: 2" "Número: 3"
While	Ejecución de una condición mientras sea cierta.	`while (condition){ #código a ejecutar }`	`x <- 1 while (x <= 3) { print(paste("Iteración", x)) x <- x + 1 }`	"Iteración 1" "Iteración 2" "Iteración 3"

Génesis Ojeda

Ciclos

Concepto	Descripción	Ejemplo de Declaración	Ejemplo de Código	Resultado

Break	Salir de un bucle inmediatamente	`for (variable in sequence){`
		`#código a ejecutar`
		`break`
		`}`
			`for (i in 1:5) {`
			`if (i == 3) break`
			`print(i)`
			`}`
				1
				2
				4
				5

Next	Saltar a la siguiente iteración de un bucle	`while (condition){`
		`#código a ejecutar`
		`next`
		`}`
			`x <- 1`
			`while (x <= 5) {`
			`if (x == 3) {`
			`x <- x + 1`
			`next`
			`}`
			`print(x)`
			`x <- x + 1`
			`}`
				1
				2
				4
				5

Génesis Ojeda

Funciones creadas por usuario

Son bloques de código que el propio programador define para realizar tareas específicas. Son útiles cuando necesitas realizar una operación varias veces o cuando el código es muy complejo. Estas funciones permiten reutilizar el código.

```
sumar ← function(a, b) {
    return(a + b)
}

resultado ← sumar(3, 5)
print(resultado)  # Imprime 8
```

Funciones integradas

Son funciones que ya vienen definidas en el lenguaje y están listas para usar. Permiten realizar tareas comunes, como cálculos matemáticos, estadísticos o manipulación de datos, sin necesidad de escribir el código desde cero.

```r
vector ← c(10, 20, 30, 40, 50)
length(vector) # Resultado: 5
```

Funciones Integradas

Función	Descripción	Ejemplo de uso
min()	Calcula el valor mínimo de un vector numérico.	min(c(3, 5, 2, 8)) # Resultado: 2
max()	Calcula el valor máximo de un vector numérico.	max(c(3, 5, 2, 8)) # Resultado: 8
sum()	Calcula la suma de los elementos de un vector numérico.	sum(c(3, 5, 2, 8)) # Resultado: 18
mean()	Calcula la media (promedio) de los elementos de un vector numérico.	mean(c(3, 5, 2, 8)) # Resultado: 4.5
range()	Calcula los valores mínimo y máximo de un vector numérico.	range(c(3, 5, 2, 8)) # Resultado: 2, 8
str()	Muestra la estructura de un objeto en R, como su tipo y tamaño.	str(mtcars) # Muestra estructura del dataset mtcars
ncol()	Devuelve el número de columnas de una matriz o marco de datos.	ncol(matrix(1:9, nrow = 3)) # Resultado: 3
length()	Devuelve el número de elementos de un objeto R.	length(c(3, 5, 2, 8)) # Resultado: 4

Adier Maldonado

Arrays

01

Crear vector

c (1:24)

02

Definir dimensiones

dim = c (4,3,2)

#4 filas, 3 columnas y 2 capas

03

Crear array

my_array <- array(c (1:24), dim= c (4,3,2))

vector

dimensiones

Vector: Colección de elementos de un mismo tipo.

Al definir dimensiones se definen las filas, columnas y las capas.
Solo pueden tener un tipo de dato.

Génesis Ojeda

Arrays

```r
# Crear un array de 2x3x2
my_array <- array(1:12, dim = c(2, 3, 2))

# Imprimir el array
print(my_array)
```

```
#output
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

Al definir dimensiones se definen las filas, columnas y las capas.
Solo pueden tener un tipo de dato.

Arrays

Concepto	Descripción	Ejemplo	Resultado
Un elemento	Acceder a un elemento específico del array usando fila, columna y capa	`nombre_del_array[1, 2, 3]`	Valor en la posición (1, 2, 3)
Filas o columnas	Una coma (,) significa todo de algo, dependiendo de la posición	`nombre_del_array[, 2, 1]`	Todas las filas de la columna 2 en la capa 1

```
mi_array <- array(1:4, dim = c(1, 2, 2))
print(mi_array)
```

```
mi_array[1, 2, 1]
```

```
mi_array[1, 1, 2]
```

```
mi_array[, , 2]
```

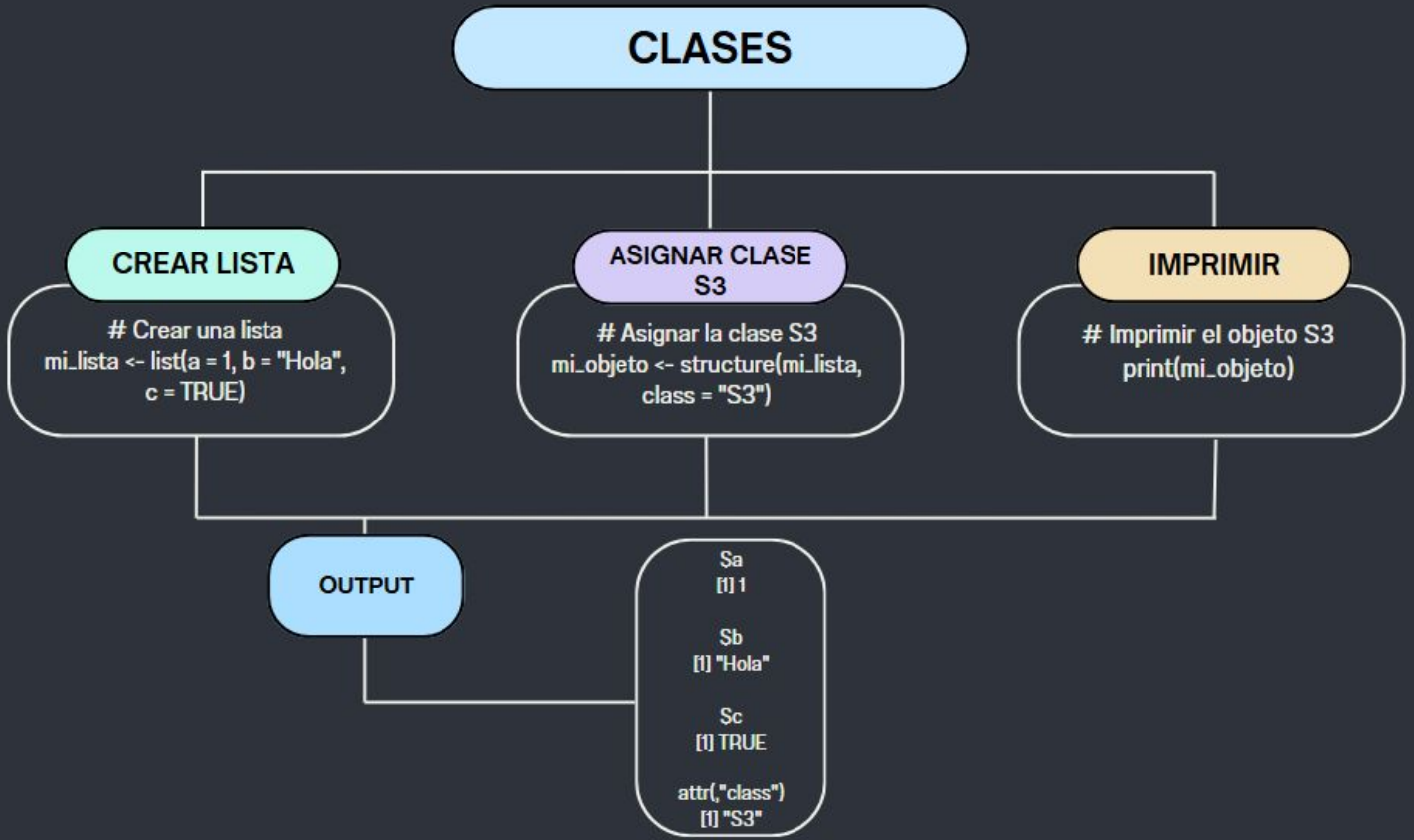
```
, , 1

     [,1] [,2]
[1,]    1    2

, , 2

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

Génesis Ojeda

Arrays

Concepto	Descripción	Ejemplo	Resultado
`length()`	Devuelve la longitud (número de elementos) de un objeto	`arr <- array(1:12, dim = c(2, 3, 2)); length(arr)`	1 2
`dim()`	Devuelve las dimensiones de un array	`arr <- array(1:12, dim = c(2, 3, 2)); dim(arr)`	2 3 2
`%in%`	Verifica si un valor está presente en un array	`arr <- array(1:12, dim = c(2, 3, 2)); 5 %in% arr`	TRUE

Clases



Génesis Ojeda

Script

¿Qué es un script?

Un script es simplemente un archivo de texto que contiene un conjunto de comandos y comentarios. El script se puede guardar y utilizar más tarde para volver a ejecutar los comandos guardados. El script también se puede editar para que puedas ejecutar una versión modificada de los comandos. Creación de un script de R.

```r
# Declaración de variables y operaciones
x ← 34
y ← 16
z ← x + y    # Suma
w ← y / x    # División

# Mostrar resultados
x; y; z; w

# Cambiar valor de x y mostrar de nuevo
x ← "texto"
x; y; z; w
```

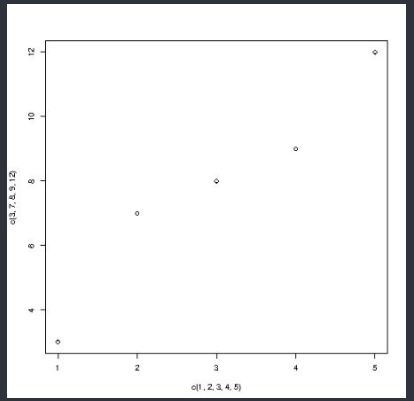
Gráficas

Gráfica	Descripción	Sintaxis	Ejemplo
Plot	Un gráfico de dispersión simple entre dos variables.	`plot(x, y)`	`R x <- 1:10 y <- x^2 plot(x, y)`
Líneas	Un gráfico de líneas que conectan puntos de datos.	`plot(x, y, type="l")`	R x <- 1:10 y <- x^2 plot(x, y, type="l")
Barra	Un gráfico de barras que muestra la distribución de datos categóricos.	`barplot(height)`	R heights <- c(2, 3, 5, 7, 11) barplot(heights)
Scatterplot	Un gráfico que muestra la relación entre dos conjuntos de datos.	`plot(x, y)`	R x <- rnorm(50) y <- rnorm(50) plot(x, y, pch=19, col="blue")
Pie chart	Un gráfico circular que muestra proporciones de un todo.	`pie(x)`	R slices <- c(10, 20, 30, 40) pie(slices)

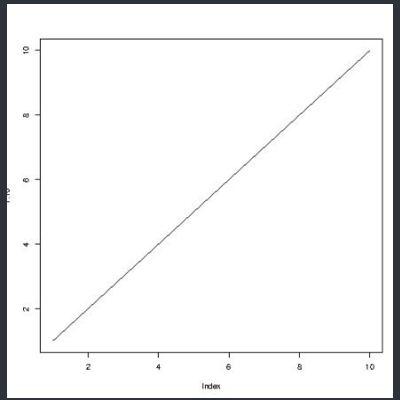
Ejemplo de plot y line

```
1
2   #puntos o coordenadas
3
4   x <- c(1, 2, 3, 4, 5)
5   y <- c(3, 7, 8, 9, 12)
6
7   plot(x, y)
8
9
10  #Crear una línea
11
12  plot(1:10, type="l")
13
14
```

https://www.w3schools.com/r/tryr.asp?filename=demo_graph_plot3



https://www.w3schools.com/r/tryr.asp?filename=demo_graph_plot_line

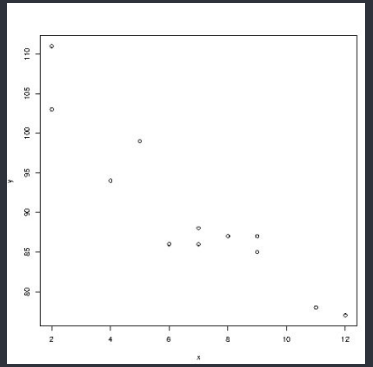


Génesis Ojeda

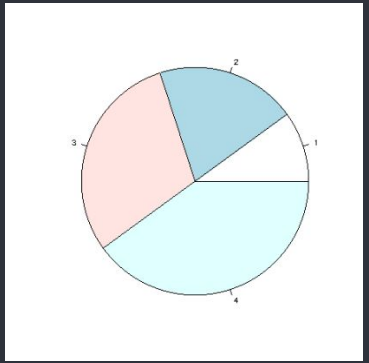
Ejemplo de scatterplot y pie chart

```
1
2  x <- c(5,7,8,7,2,2,9,4,11,12,9,6)
3  y <- c(99,86,87,88,111,103,87,94,78,77,85,86)
4
5  plot(x, y)
6
7
8  # Create a vector of pies
9  x <- c(10,20,30,40)
10
11 # Display the pie chart
12 pie(x)
13
14
```

https://www.w3schools.com/r/r_graph_scatterplot.asp



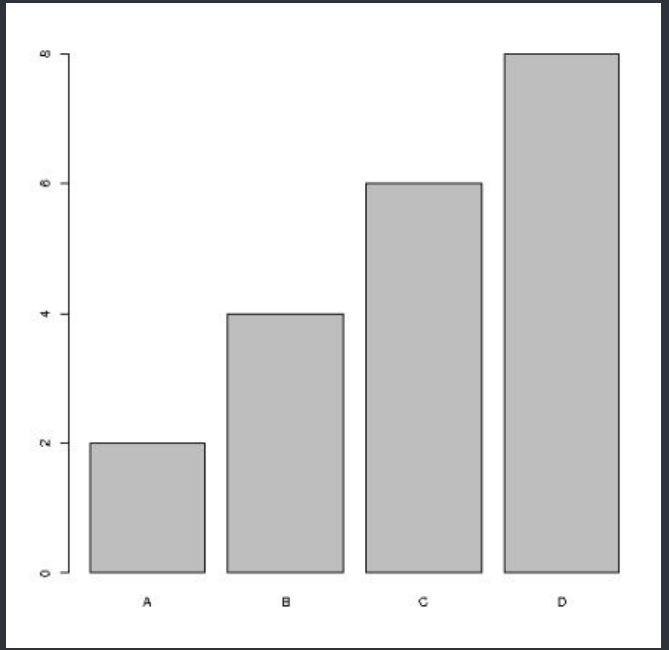
https://www.w3schools.com/r/tryr.asp?filename=demo_graph_pie



Génesis Ojeda

Ejemplo de barras

```
1
2   x <- c("A", "B", "C", "D")
3
4   # y-axis values
    y <- c(2, 4, 6, 8)
5
6   barplot(y, names.arg = x)
7
8
9
10
11
12
13
14
```



https://www.w3schools.com/r/tryr.asp?filename=demo_graph_bar

Génesis Ojeda

Ejemplos de programación

Link a Google collab:
https://colab.research.google.com/drive/1CROH071TJTEbUCeaQCoi1k9JORgKBoNM?usp=sharing

Génesis Ojeda y Adier Maldonado

Crédito de diseño de slides