**Capítulo 13**

**ANÁLISIS CON R**

**Realizado por:**

Grace Viteri

Yoskira Cordero

**Objetivos**

Dominarás el uso básico del entorno R, conceptos esenciales de programación en R, tipos y estructuras de datos (vectores, *data frames*), operadores. Y generar visualizaciones para la toma de decisiones.

**Contexto y motivación**

Dirigido a lectores con experiencia mínima en el uso de R. Les permitirá generar productos para analizar, comunicar y tomar decisiones. Se transita en el origen y cualidades del lenguaje libre “R”, justificando su popularidad en el ámbito académico y profesional. Te dará independencia para poder manejar datos reales, fortaleciendo tu formación como analista de procedimientos estadísticos.

¡Empecemos!

**25.1. Breve historia de R como software estadístico**

R surge a principios de la década de 1990 como un proyecto académico en la Universidad de Auckland, Nueva Zelanda. Fue diseñado por Ross Ihaka y Robert Gentleman inspirados en el lenguaje S, desarrollado previamente en los laboratorios Bell de AT&T durante los años setenta. La intención original fue diseñar una herramienta de código abierto para la enseñanza de estadística que combinara flexibilidad, potencia analítica y facilidad de expansión (Ihaka & Gentleman, 1996).

Desde un principio, los autores del programa tomaron la decisión de publicar R bajo la licencia GNU GPL (software libre) en 1995, lo que facilitó su rápida adopción y el desarrollo de una extensa comunidad de usuarios y paquetes, lo que ha convertido a R en un ecosistema dinámico y popular para investigación y enseñanza. El proyecto evolucionó rápidamente con el apoyo de la comunidad científica internacional, que consolidó el R Core Team, encargado del mantenimiento del sistema base. Además, se creó el repositorio CRAN (Comprehensive R Archive Network), un sistema de distribución de paquetes que hoy contiene más de 20.000 extensiones para análisis en campos tan diversos como la educación, la sociología, la salud pública o la economía (Wickham & Grolemund, 2023).

Desde un punto de vista técnico y con aporte de Crawley (2012)., R se caracteriza por:

* Ser un lenguaje interpretado: los comandos se ejecutan directamente sin necesidad de compilación previa.
* Ser orientado a objetos, lo que permite manipular estructuras de datos como vectores, matrices, listas y *data frames*.
* Incluir un entorno gráfico poderoso, capaz de generar representaciones visuales de alta calidad.
* Ofrecer una amplia comunidad y documentación abundante, lo que facilita su aprendizaje colaborativo.

De acuerdo con Crawley (2012), R ha transformado la manera en que se conciben los análisis cuantitativos en el ámbito académico, debido a su naturaleza abierta, su comunidad activa y su orientación hacia la transparencia metodológica. En la enseñanza universitaria, su uso no sólo fortalece las competencias técnicas de los estudiantes, sino que también promueve una comprensión conceptual más profunda de los procedimientos estadísticos.

Tal como señala Contento (2019), el software R es uno de los más flexibles, potentes y profesionales que existen actualmente para realizar tareas estadísticas, desde las más elementales hasta las más avanzadas. Está desarrollado y soportado por una comunidad académica a nivel mundial, cuenta además con la ventaja de ser gratuito y su descarga e instalación sencilla[[1]](#footnote-1).

Además, R facilita el flujo de trabajo de data science desde la importación y limpieza hasta la visualización y modelado; en palabras de los autores, “R facilita transformar y visualizar datos para el análisis” (Wickham & Grolemund, 2023, p. 3).

Dos características explican su adopción generalizada en la academia y la industria: extensibilidad mediante paquetes (CRAN, Bioconductor, GitHub), que permite disponer de herramientas especializadas para campos concretos; y capacidad gráfica avanzada (base graphics, lattice, ggplot2) que posibilita tanto gráficos exploratorios como figuras de publicación.

**25.2. Diferencias de R con otros programas (SPSS, Stata, Python)**

En principio, es importante distinguir categorías: SPSS y Stata son entornos comerciales centrados en menús (aunque ambos poseen lenguajes de comandos), orientados hacia usuarios que requieren análisis estadísticos clásicos con una curva de aprendizaje relativamente suave para tareas estándar; en contraste, R es un lenguaje de programación estadística libre y abierto, con mayor énfasis en la reproducibilidad, la automatización y la extensibilidad mediante programación.

Al compararlo con SPSS, se puede señalar que el segundo suele ofrecer facilidad de uso en análisis descriptivos y pruebas estándar mediante interfaces gráficas y diálogos. Sin embargo, R amplía estas capacidades con un ecosistema de paquetes para métodos avanzados, gráficos de publicación y análisis reproducible mediante scripts y notebooks (Wickham & Grolemund, 2023).

Por otra parte, Stata combina un lenguaje de comandos con capacidades sólidas para análisis econométricos y de panel; además, Stata es software propietario con paquetes oficiales que facilitan ciertos análisis. En cambio, R ofrece opciones metodológicas a través de paquetes de la comunidad y mayor capacidad de programación funcional y orientada a objetos, aunque a costa de una curva de aprendizaje inicial mayor (Wickham & Grolemund, 2023).

En cuanto a Python, tenemos que este es un lenguaje de propósito general con bibliotecas potentes para ciencia de datos (pandas, NumPy, scikit-learn, matplotlib). En consecuencia, Python es más flexible fuera del ámbito estadístico y tiende a integrarse mejor en aplicaciones de producción; mientras tanto, R conserva ventajas para análisis estadístico especializado y visualización estadística a través de paquetes diseñados por estadísticos (McKinney, 2022).

Por tanto, la elección entre R, SPSS, Stata o Python depende del objetivo pedagógico o profesional: reproducibilidad y comunidad (R), facilidad de uso para análisis estándar (SPSS), flujo de trabajo econométrico y manejo de datos específicos (Stata) o integración con sistemas y producción (Python). En la Tabla 1, se resumen los principales puntos de comparación entre los diferentes programas.

**Tabla 1.** *Comparando los diferentes programas estadísticos*

| **Software** | **Tipo de licencia** | **Interfaz principal** | **Fortalezas** | **Limitaciones** |
| --- | --- | --- | --- | --- |
| **R** | Libre (GNU GPL) | Programación + entorno gráfico | Flexibilidad, reproducibilidad, comunidad amplia | Curva de aprendizaje inicial |
| **SPSS** | Comercial | Menús y ventanas | Facilidad de uso, soporte técnico | Escasa personalización |
| **Stata** | Comercial | Consola y menús | Sintaxis coherente, sólido en econometría | Licencia de pago, menos extensible |
| **Python** | Libre | Programación | Integración con IA y machine learning | No especializado en estadística clásica |

**25.3. Fundamentos de Programación con R**

El lenguaje R se ha consolidado como una de las herramientas más potentes para el análisis estadístico y la ciencia de datos, gracias a su flexibilidad y capacidad para manejar diversos tipos y estructuras de información. Comprender los tipos de datos, operadores, funciones y estructuras de control constituye la base para desarrollar programas eficientes y precisos. Este conocimiento permite al usuario transformar, analizar y visualizar información de manera estructurada, facilitando la toma de decisiones fundamentadas en los resultados obtenidos. Por tal razón en las siguientes subsecciones es importante abordar estos conceptos básicos.

* + 1. **Tipos de datos en R**

R maneja diversos tipos de datos fundamentales que permiten representar información numérica, textual o lógica. Entre los más comunes se encuentran en la Tabla 2:

**Tabla 2:** *Tipos de datos*

|  |  |  |
| --- | --- | --- |
| **Tipo de dato** | **Descripción** | **Ejemplo** |
| Numéricos | Representan valores enteros o decimales | x <- 3.14, y <- 10 |
| Caracteres | Cadenas de texto encerradas entre comillas | nombre <- “Grace”, ciudad <- “Guayaquil” |
| Lógicos | Expresan valores booleanos (TRUE o FALSE) | es\_mayor <- TRUE |
| Factores | Representan variables categóricas, muy útiles en análisis estadístico | genero <- factor(c(“Femenino”, “Masculino”, “Femenino”)) |

* + 1. **Estructuras de datos**

En R, los datos se organizan en estructuras que facilitan su manipulación. En la Tabla 3 se mencionan las estructuras que son la base del análisis de datos en R.

**Tabla 3:** *Tipos de estructuras*

|  |  |  |
| --- | --- | --- |
| **Tipo de estructura** | **Descripción** | **Ejemplo** |
| Vectores | Conjunto unidimensional de elementos del mismo tipo. | edades <- c(20, 25, 30, 35) |
| Matrices | Arreglos bidimensionales de datos del mismo tipo. | matriz <- matrix(1:9, nrow=3, ncol=3) |
| Data Frames | Tablas de datos donde cada columna puede tener un tipo distinto (similar a una hoja de cálculo). | datos <- data.frame(  nombre = c(“Ana”, “Luis”, “Grace”),  edad = c(25, 30, 35),  ciudad = c(“Quito”, “Guayaquil”, “Manta”)) |
| Listas | Estructuras que pueden contener elementos de diferentes tipos y tamaños. | lista <- list(nombre= “Proyecto”, año=2025, resultados=c(0.8, 0.9)) |

* + 1. **Operadores aritméticos, relacionales, lógicos y de asignación.**

Los operadores son esenciales para la toma de decisiones dentro del flujo del programa. R incluye una amplia gama de operadores que facilitan los cálculos matemáticos y las comparaciones lógicas.

Los operadores aritméticos permiten realizar cálculos matemáticos sobre datos numéricos. Son los más comunes y se utilizan para sumar, restar, multiplicar, dividir valores entre otras operaciones.

**Tabla 4:** *Operadores aritméticos*

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Descripción** | **Ejemplo** | **Resultado** |
| + | Suma | 5 + 3 | 8 |
| - | Resta | 10 - 4 | 6 |
| \* | Multiplicación | 6 \* 2 | 12 |
| / | División | 8 / 2 | 4 |
| ^ o \*\* | Exponenciación | 3 ^ 2 | 9 |
| %% | Módulo (residuo de división) | 10 %% 3 | 1 |
| %/% | División entera | 10 %/% 3 | 3 |

Los operadores relacionales se utilizan para comparar valores y devuelven un resultado de tipo booleano (TRUE o FALSE). Sirven para evaluar condiciones dentro de estructuras como if, while o for.

**Tabla 5:** *Operadores relacionales*

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Descripción** | **Ejemplo** | **Resultado** |
| == | Igual a | 5 == 5 | TRUE |
| != | Diferente de | 5 != 3 | TRUE |
| > | Mayor que | 10 > 7 | TRUE |
| < | Menor que | 2 < 8 | TRUE |
| >= | Mayor o igual que | 4 >= 4 | TRUE |
| <= | Menor o igual que | 3 <= 5 | TRUE |

Los operadores lógicos permiten combinar o negar expresiones lógicas. Se utilizan principalmente para evaluar condiciones múltiples dentro de estructuras de control.

**Tabla 6:** *Operadores lógicos*

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Descripción** | **Ejemplo** | **Resultado** |
| & | Y lógico (AND) | TRUE & FALSE | FALSE |
| ` | ` | O lógico (OR) | `TRUE |
| ! | Negación lógica (NOT) | !TRUE | FALSE |

La asignación de valores es un concepto fundamental en todo lenguaje de programación, y R no es la excepción. La asignación consiste en guardar un dato (número, texto, resultado de una operación, etc.) dentro de un objeto o variable para poder usarlo más adelante.

**Tabla 7:** *Operadores de asignación*

|  |  |  |  |
| --- | --- | --- | --- |
| **Símbolo** | **Nombre** | **Uso común** | **Ejemplo** |
| <- | Asignación hacia la izquierda | Es la forma más tradicional y recomendada. | x <- 10 |
| -> | Asignación hacia la derecha | Menos frecuente, se usa en sentido inverso. | 10 -> x |
| = | Asignación directa | También válida, pero menos usada en programación avanzada. | x = 10 |

* + 1. **Funciones básicas y creación de funciones propias**

Una función puede definirse como un conjunto de instrucciones o procedimientos agrupados bajo un nombre específico. Estas funciones reciben argumentos, es decir, parámetros que determinan la forma en que deben ejecutarse las operaciones internas. Al invocar una función, R ejecuta las acciones definidas en su estructura utilizando los argumentos proporcionados por el usuario. Es igual como cuando usamos las funciones predefinidas de Excel.

En el lenguaje R, las funciones se identifican mediante la notación *nombre\_de\_la\_función(),* donde los paréntesis indican que puede contener uno o varios argumentos. Algunos ejemplos de funciones comunes en R se describen en la Tabla 8.

**Tabla 8:** *Funciones predefinidas*

|  |  |  |
| --- | --- | --- |
| **Sintaxis** | **Descripción** | **Ejemplo** |
| c(valor1, valor2, valor3, ...) | La función c() (abreviatura de combine) sirve para crear vectores. Permite agrupar múltiples valores en una sola variable. | nombres <- c(“Ana”, “Luis”, “Rebecca”, “María”)  print(nombres) |
| mean(x)  *donde x es un vector o conjunto de valores numéricos.* | Se utiliza para calcular el promedio o media aritmética de un conjunto de datos numéricos. Es una herramienta fundamental para obtener medidas de tendencia central. | valores <- c(10, 15, 20, 25, 30)  media <- mean(valores)  print(media) |
| quantile(x, probs)  *donde x es el conjunto de datos y probs indica las probabilidades deseadas.* | La función quantile() se emplea para determinar los cuantiles de una distribución de datos, es decir, los puntos que dividen un conjunto ordenado en partes iguales. | edades <- c(18, 22, 25, 29, 35, 40, 50)  cuartiles <- quantile(edades, probs = c(0.25, 0.5, 0.75))  print(cuartiles) |
| summary(x) | La función summary() ofrece un resumen estadístico general de los datos, mostrando medidas como el mínimo, primer cuartil, mediana, media, tercer cuartil y máximo. | temperaturas <- c(22, 24, 25, 23, 28, 30, 27)  resumen <- summary(temperaturas)  print(resumen) |
| density(x) | La función density() calcula una estimación de la densidad de probabilidad de una variable numérica continua. | valores <- c(4, 5, 6, 6, 7, 8, 9, 10)  densidad <- density(valores)  plot(densidad, main = “Densidad de probabilidad de los valores”) |
| length(x) | Devuelve el número total de elementos contenidos en un vector, lista o cualquier otro objeto similar. | numeros <- c(5, 10, 15, 20)  length(numeros) |
| str(x) | Muestra una descripción compacta de la estructura interna de un objeto, como su tipo, tamaño y contenido. | datos <- data.frame(  nombre = c(“Ana”, “Luis”, “Rebecca”),  edad = c(25, 30, 35))  str(datos) |

Además, es posible crear funciones personalizadas por el programador para automatizar tareas repetitivas. El siguiente ejemplo, implementa en una función el cálculo del área de una figura geométrica en la que se solicita como parámetro la base y la altura:

calcular\_area <- function(base, altura) {

area <- (base \* altura) / 2

return(area)

}

calcular\_area(10, 5)

* + 1. **Control de flujo y estructuras condicionales**

Un aspecto clave para comprender la lógica de programación en R es el control de flujo, que permite ejecutar distintas acciones según condiciones específicas. A continuación, se describen las estructuras condicionales y los bucles:

* **Estructuras condicionales**

Las estructuras condicionales permiten controlar el flujo de un programa según si una o varias condiciones se cumplen o no. Entre ellas tenemos a: if, if ... else, ifelse(),

switch() que se detallan en la Tabla 9.

**Tabla 9:** *Estructuras condicionales*

|  |  |  |
| --- | --- | --- |
| **Sintaxis** | **Descripción** | **Ejemplo** |
| *if (condición) {*  *# código que se ejecuta si la condición es verdadera*  *}* | La instrucción if evalúa una condición lógica y ejecuta un bloque de código solo si esa condición es verdadera (TRUE). | x <- 10  if (x > 5) {  print("El valor de x es mayor que 5")  } |
| *if (condición) {*  *# se ejecuta si la condición es verdadera*  *} else {*  *# se ejecuta si la condición es falsa*  *}* | Permite ejecutar un bloque de código cuando la condición es verdadera y otro bloque diferente cuando es falsa. | edad <- 17  if (edad >= 18) {  print("Eres mayor de edad")  } else {  print("Eres menor de edad")  } |
| *ifelse(condición, valor\_si\_verdadero, valor\_si\_falso)* | A diferencia de if y else, la función ifelse() evalúa condiciones de manera vectorizada, es decir, puede aplicarse a todos los elementos de un vector al mismo tiempo. | notas <- c(8, 5, 9, 4)  resultado <- ifelse(notas >= 7, "Aprobado", "Reprobado")  print(resultado) |
| *switch(opción,*  *caso1 = expresión1,*  *caso2 = expresión2,*  *caso3 = expresión3)* | switch() se utiliza cuando se deben evaluar múltiples opciones. Es más eficiente que usar varios if ... else if .... | opcion <- "B"  resultado <- switch(opcion,  "A" = "Seleccionaste la opción A",  "B" = "Seleccionaste la opción B",  "C" = "Seleccionaste la opción C",  "Opción no válida")  print(resultado) |

* **Estructuras repetitivas**

Las estructuras repetitivas, también llamadas bucles o ciclos, permiten ejecutar una o varias instrucciones de forma repetida mientras se cumpla una condición o hasta recorrer todos los elementos de un conjunto de datos. En R existen tres estructuras de repetición principales: for, while, repeat.

**Tabla 10:** *Estructuras repetitivas*

|  |  |  |
| --- | --- | --- |
| **Sintaxis** | **Descripción** | **Ejemplo** |
| for (variable in conjunto) {  # código que se ejecuta en cada iteración  } | El bucle for se utiliza cuando se conoce de antemano cuántas veces se debe repetir una acción. Es ideal para recorrer los elementos de un vector, lista o conjunto de datos. | # Imprimir los números del 1 al 5  for (i in 1:5) {  print(paste("Iteración número:", i))  } |
| *while (condición) {*  *# código a ejecutar*  *}* | El bucle while ejecuta un bloque de código mientras una condición sea verdadera (TRUE). Se utiliza cuando no se conoce exactamente cuántas veces se repetirá la instrucción, pero sí la condición que controla el ciclo. | x <- 1  while (x <= 5) {  print(paste("El valor de x es:", x))  x <- x + 1 # Incremento  } |
| *repeat {*  *if (condición) {*  *break # detiene el bucle*  *}*  *}* | El bucle repeat ejecuta indefinidamente un bloque de código hasta que se cumpla una condición de salida, la cual se define mediante la instrucción break. Se usa cuando se requiere una repetición infinita o controlada internamente. | contador <- 1  repeat {  print(paste("Contador:", contador))  contador <- contador + 1    if (contador > 3) {  break  }  } |

**25.4. Estadística descriptiva en R**

**25.4.1. Medidas de tendencia central y dispersión**

En primer lugar, cabe definir que la estadística descriptiva se orienta a “organizar, resumir y presentar los datos de forma que puedan interpretarse fácilmente” (Salazar, 2015, p. 14). Dentro de este ámbito, las medidas de tendencia central y las medidas de dispersión constituyen herramientas fundamentales para describir el comportamiento de una variable cuantitativa, y en algunas de ellas también se aplican a las cualitativas.

**Medidas de tendencia central**

Las comunes son: la media aritmética, la mediana y la moda.

La media aritmética, se define como la suma de los valores dividida entre el número de valores; es muy sensible a valores extremos.

La mediana, es el valor que organiza los datos en dos mitades iguales; resulta más robusta frente a valores atípicos.

La moda, es el valor que aparece con mayor frecuencia en la muestra; útil particularmente en variables cualitativas o discretas.

En R, la media se calcula con mean(x), la mediana con median(x). Y la moda requiere una función personalizada (ya que R no incluye una función de moda por defecto), usando la función table().

**Medidas de dispersión**

La dispersión describe qué tan esparcidos o concentrados están los datos respecto de la tendencia central. Entre las más utilizadas están: el rango (máximo − mínimo), la varianza, la desviación estándar y los cuartiles/intercuartílico. Salazar (2015) precisa que la estadística descriptiva incluye las “medidas de posición, de dispersión y de forma” (p. 11).

La varianza se define, como el promedio del cuadrado de las desviaciones respecto de la media; su raíz cuadrada es la desviación estándar, que se expresa en las mismas unidades de la variable original. En R, se usan var(x) y sd(x).

Además, es pertinente calcular los cuartiles con quantile(x, probs = c(0.25,0.5,0.75)) y el rango con range(x) o directamente max(x) - min(x).

El conocimiento de estas medidas es esencial en ciencias sociales y educación porque, como señalan Hernández (2015), permiten “describir un conjunto de datos en investigación educativa y analizar la variabilidad de los resultados de un instrumento o encuesta” (p. 133).

**25.4.2. Tablas de frecuencias y resúmenes estadísticos**

**Tablas de frecuencias**

Son una herramienta estadística fundamental para resumir y organizar cualquier conjunto de datos. Cuando se trabaja con variables cualitativas, un paso inicial es su codificación. En el cual se asigna un valor numérico a cada categoría, luego se puede registrar como frecuencia absoluta (el conteo de la ocurrencia de cada categoría), y la frecuencia relativa (su proporción o porcentaje). Y la frecuencia acumulada puede ser útil interpretarla si tiene un orden lógico.

Y las variables cuantitativas (obtenidas bien se por conteo o medición) también es pertinente utilizar la tabla de frecuencia para ayudarse a analizar variables discretas y continuas. Por ejemplo, Hernández (2015) considera la tabla de frecuencia como parte del “ordenamiento de datos” en estadística descriptiva (p. 117).

En R, una tabla de frecuencias se puede generar con table(x) para la frecuencia absoluta, y para la relativa con prop.table(table(x)) \* 100.

Para variables cuantitativas, es habitual presentar un resumen con la media, mediana, desviación estándar, cuartiles, valores mínimo y máximo y número de observaciones (N). En R, la función summary(x) ofrece ese resumen básico.

Por ejemplo, si x es la variable “edad” de una encuesta, summary(x) muestra el mínimo, primer cuartil, mediana, media, tercer cuartil y máximo.

Este tipo de resumen permite al investigador educativo verificar si las variables tienen distribución sesgada, presencia de valores extremos o variabilidad elevada.

**Tablas de frecuencias para variables cuantitativas agrupadas**

En ocasiones conviene agrupar variables cuantitativas en intervalos (por ejemplo, grupos de edad 18 - 25, 26 - 35, etc.) y luego construir una tabla de frecuencias de esos intervalos. En R se puede usar la función cut() para agrupar y luego table() o prop.table() para obtener las frecuencias. Esta práctica es útil cuando se deben presentar resultados en informes educativos o institucionales.

**25.4.3. Ejemplo: análisis descriptivo de una encuesta**

A continuación, se presenta un ejemplo orientado a Ciencias Sociales / Educación que ilustra el uso de R (y ggplot2) para realizar un análisis de estadística descriptiva de una encuesta.

**Contexto del estudio:**

Se administró una encuesta a 120 estudiantes universitarios sobre su percepción de la modalidad de enseñanza en línea. Las variables consideradas son:

* Edad: edad en años (cuantitativa)
* Género: “Femenino” / “Masculino” (cualitativa)
* Satisfacción: escala de 1 a 5 sobre satisfacción con la modalidad en línea (cualitativa, ordinal)
* Frecuencia de uso: número de días por semana que se conectan a la plataforma (cuantitativa)
* Preferencia estrategia: categoría “Colaborativa”, “Expositiva”, “Mixta” (cualitativa)

**Paso 1:** Ingreso de datos (simulado para efectos del ejemplo)

set.seed(2025)

n <- 120

encuesta <- data.frame(

Edad = round(rnorm(n, mean = 21, sd = 2)),

Genero = sample(c("Femenino","Masculino"), n, replace = TRUE, prob = c(0.6, 0.4)),

Satisfaccion = sample(1:5, n, replace = TRUE, prob = c(0.1,0.2,0.3,0.25,0.15)),

Frecuencia\_uso = sample(0:7, n, replace = TRUE),

Preferencia\_estrategia = sample(c("Colaborativa","Expositiva","Mixta"), n, replace = TRUE)

)

**Paso 2:** Resumen estadístico y medidas de tendencia central y dispersión

summary(encuesta)

mean(encuesta$Edad)

sd(encuesta$Edad)

table(encuesta$Preferencia\_estrategia)

prop.table(table(encuesta$Preferencia\_estrategia))\*100

**Interpretación:** Si, por ejemplo, la media de edad es 21.3 años (sd = 2.1), se concluye que la mayoría de estudiantes tienen edades alrededor de los 21 años con poca variabilidad. Si la estrategia “Mixta” representa el 45 % de los casos, se puede inferir una fuerte preferencia por formas híbridas.

**Paso 3:** Visualización gráfica

**Histograma de edad:**

library(ggplot2)

ggplot(encuesta, aes(x = Edad)) +

geom\_histogram(binwidth = 1, fill = "skyblue", color = "white") +

labs(title = "Distribución de edades de los estudiantes",

x = "Edad (años)", y = "Frecuencia") +

theme\_minimal()

**Barplot de frecuencia de estrategia**

ggplot(encuesta, aes(x = Preferencia\_estrategia, fill = Preferencia\_estrategia)) +

geom\_bar() +

labs(title = "Preferencia de estrategia pedagógica",

x = "Estrategia", y = "Número de estudiantes") +

theme\_minimal()

**Boxplot de satisfacción por género**

ggplot(encuesta, aes(x = Genero, y = Satisfaccion, fill = Genero)) +

geom\_boxplot() +

labs(title = "Satisfacción con la modalidad en línea según género",

x = "Género", y = "Satisfacción (1-5)") +

theme\_minimal()

**Paso 4: Interpretación de resultados**

La forma del histograma de edades permitirá verificar si la variable tiene distribución aproximadamente normal o sesgada.

El barplot revela claramente cuál estrategia pedagógica es más frecuente entre los estudiantes.

El boxplot permite comparar medianas, dispersión y posibles valores atípicos de satisfacción según género; por ejemplo, si el género “Femenino” muestra mediana más alta que “Masculino”, se podría inferir una diferencia en la percepción de la modalidad en-línea.

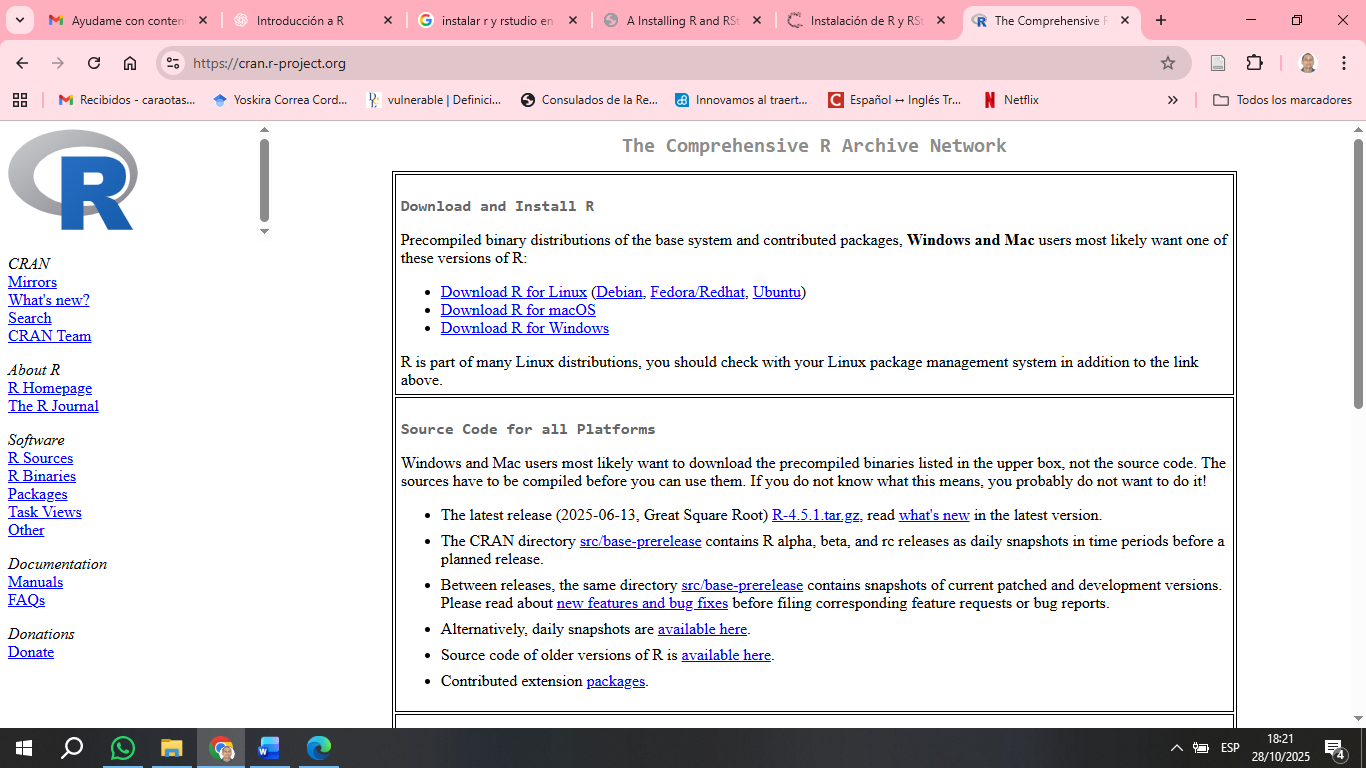
**Paso 5: Conclusiones didácticas**

La presentación sistemática de medidas numéricas y gráficas facilita que los estudiantes comprendan no sólo qué promedios y dispersión hay, sino cómo se distribuyen los datos y cómo varían entre subgrupos. Esta dualidad (numérica + gráfica) es especialmente valiosa en contextos educativos donde la interpretación y comunicación de resultados es tan relevante como el propio cálculo.

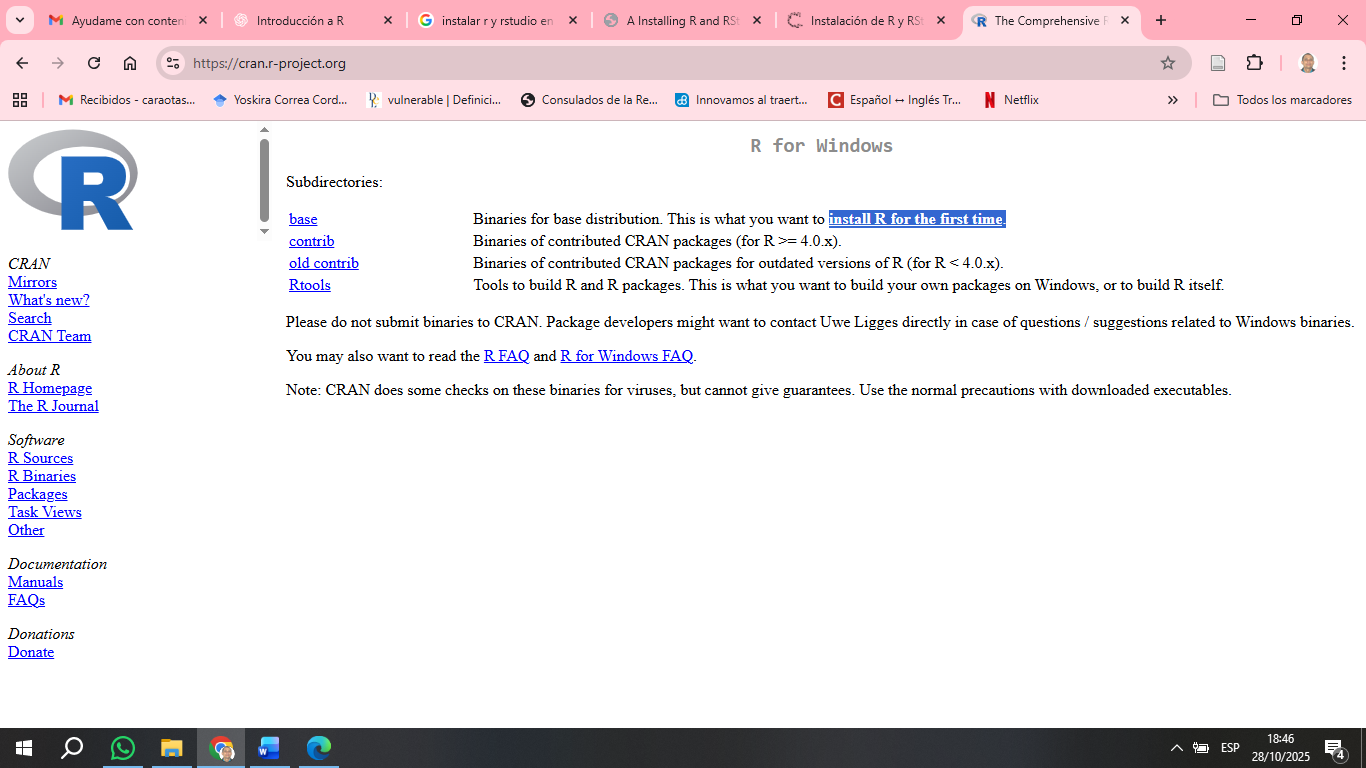
* 1. **Instalación de R**

La descarga e instalación de R es sencilla y gratuita, similar a cualquier otro programa informático. A continuación, se muestra los pasos para Windows (siendo similar para cualquier otro sistema operativo).

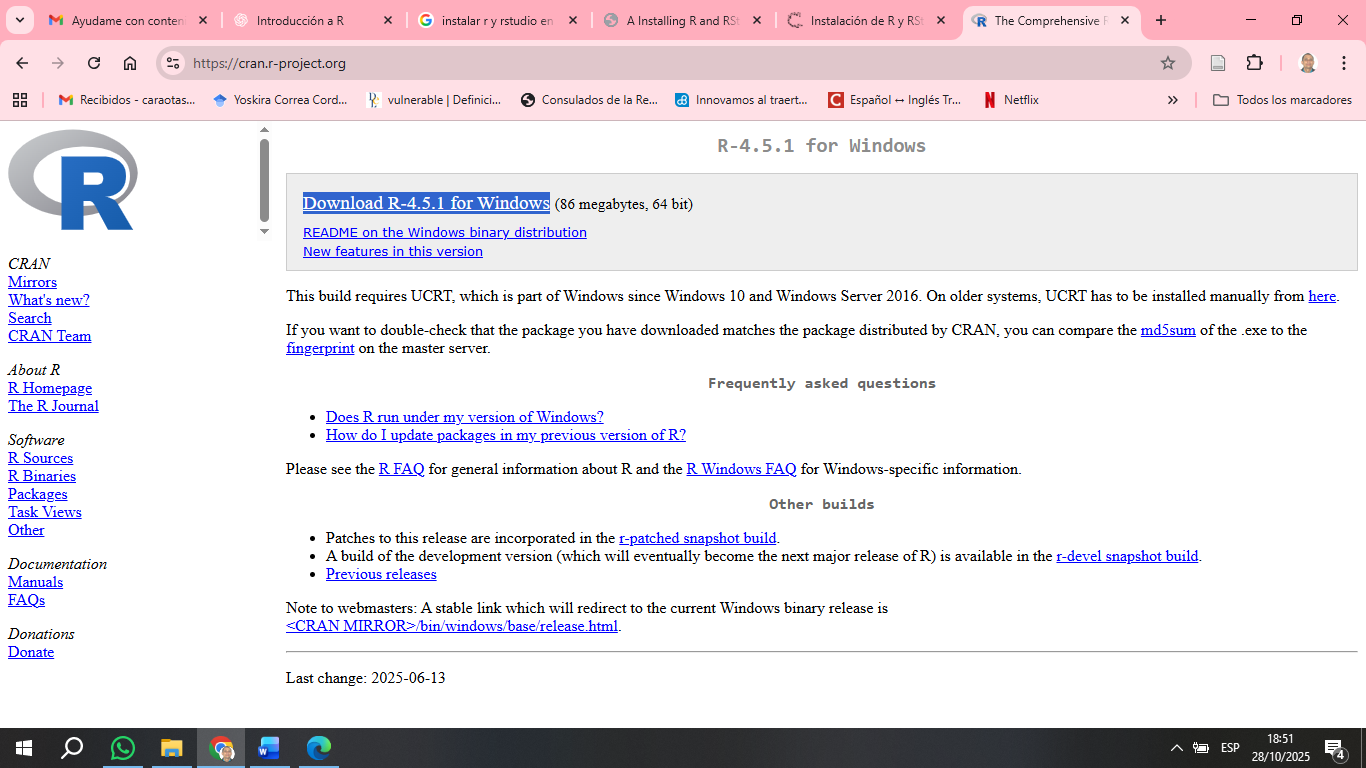
**Paso 1.** Acceder a CRAN (Comprehensive R Archive Network) que es el repositorio central de software de R (http://cran.r-project.org/). CRAN está compuesto por una serie de servidores en todo el mundo también conocidos como sitios espejo que son utilizados para distribuir tanto R como sus paquetes.



**Paso 2.** Seleccionar "install R for the first time".



**Paso 3.** Seleccionar “Download R 4.0.3 for Windows”.



**Paso 4.** Guardar el archivo descargado en la computadora y luego dar doble click para ejecutarlo.

**Paso 5**. Buscar el icono de R en el menú Inicio para abrir el programa.



**Problemas frecuentes durante la instalación de R**

Si existen problemas al instalar o iniciar R, se sugiere las siguientes recomendaciones:

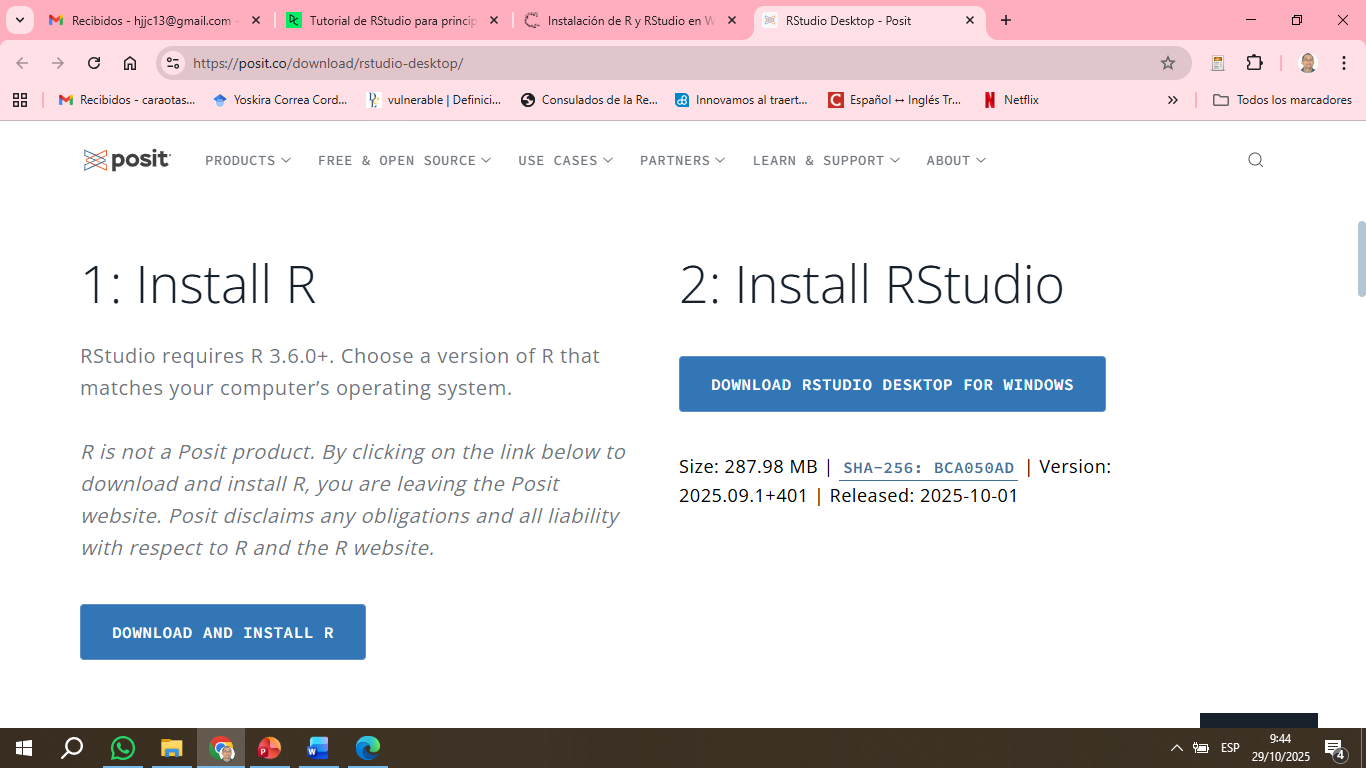
* Desinstalar R, elegir manualmente una ubicación CRAN cercana a tu lugar de residencia y descargar R de nuevo.
* Actualizaciones recientes de Windows que no se han instalado completamente pueden causar problemas de instalación. Reiniciar el ordenador e instalar R de nuevo.
* La instalación puede fallar si hay algún programa de antivirus en funcionamiento. Desactivar este programa e instalar R otra vez.
  1. **RStudio**

RStudio (actualmente Posit RStudio) es un entorno integrado de desarrollo (IDE por sus siglas en inglés). Si bien, podemos utilizar R directamente desde documentos de texto plano, se recomienda instalar y usar un IDE, ya que es una manera más práctica para trabajar y desarrollar proyectos. Esto se debe a que facilita tareas como administrar archivos, emplear herramientas para escribir y revisar código, generar gráficos; además, brinda fluidez para trabajar en el entorno local del ordenador. Existen varias opciones de entornos integrados de desarrollo para R, pero RStudio es un IDE de código abierto y por ende está en constante desarrollo.

* + 1. **Instalación de RStudio en Windows**

**Paso 1**. Instalar primero R (paso anterior)

**Paso 2.** Abrir página Restudio[[2]](#footnote-2) y pulsar click en Download Restudio Desktock for Windows



**Paso 3**. Guardar el archivo descargado en la computadora y luego pulsar doble click para ejecutarlo. Al terminar el proceso, aparecerá una ventana con el siguiente mensaje: “Completando el Asistente de Instalación de RStudio”. Por último, haz clic en “Terminar” y se habrá instalado RStudio en la computadora.

**Paso 4.** Cálculo de medidas básicas

Al abrir RStudio por primera vez, se recomienda ejecutar un pequeño ejercicio para confirmar el correcto funcionamiento del entorno. Por ejemplo:

# Cálculo de medidas básicas

x <- c(2, 5, 7, 9, 10)

mean(x)

sd(x)

El resultado mostrará la media y la desviación estándar de la variable x, confirmando que el entorno está listo para el análisis estadístico.

**Paso 5.** Configuración inicial y directorio de trabajo

El directorio de trabajo es la carpeta donde R guarda y busca los archivos. Se puede definir con:

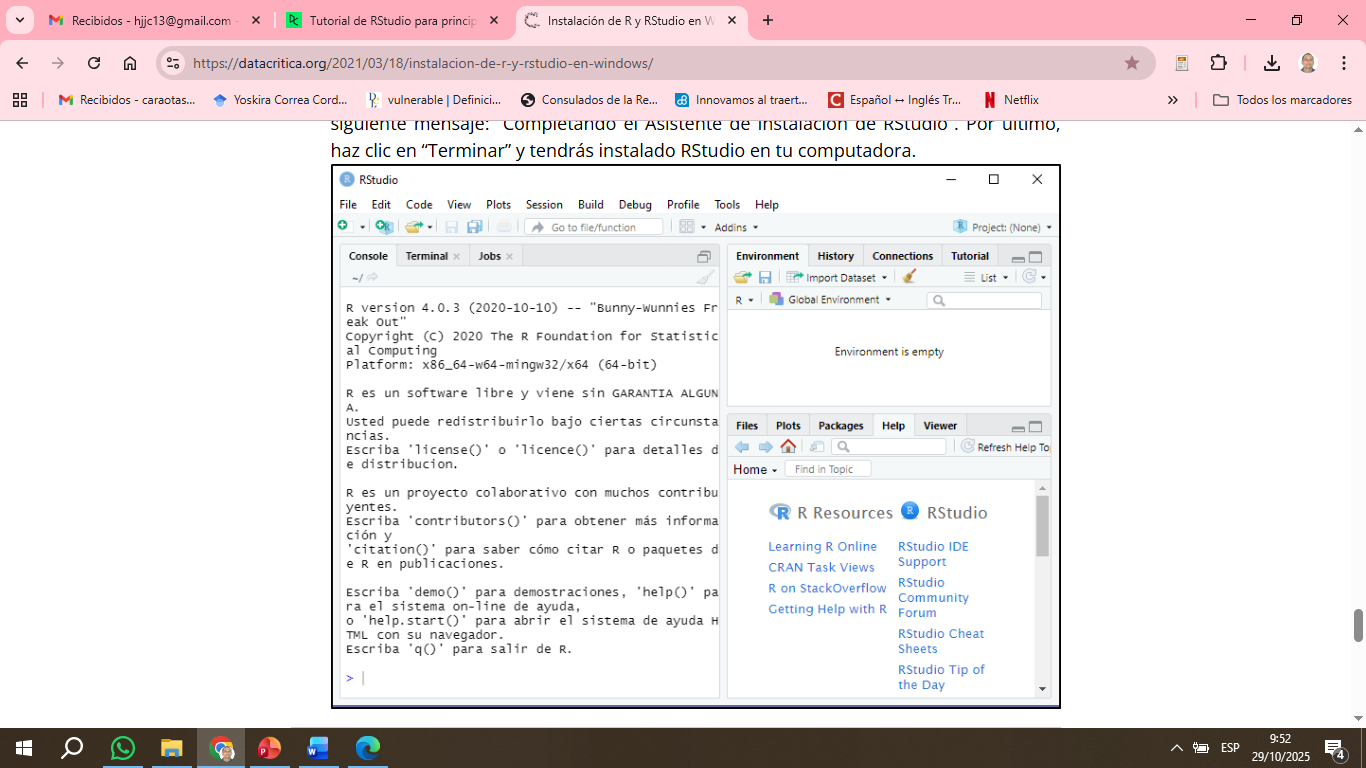
setwd("C:/Usuarios/Estudiante/Documentos/Curso\_R")

getwd()

Es recomendable crear una carpeta específica para cada proyecto o asignatura, promoviendo así una organización sistemática y reproducible de los análisis.

* + 1. **Interfaz básica de RStudio (scripts, consola, entorno, gráficos).**

Su interfaz es una ventana con tres secciones.



A grandes rasgos, podemos dividir la ventana de trabajo en tres zonas:

* Área izquierda: incluye las pestañas Consola, Terminal y Trabajos en segundo plano.
* Zona superior derecha: incluye las pestañas Entorno, Historia, Conexiones y Tutorial.
* Zona inferior derecha: incluye las pestañas Archivos, Parcelas, Paquetes, Ayuda, Visor y Presentación.

**A) Pestañas esenciales**

**Consola:** en esta pestaña, se ve primero la información sobre la versión de R en uso y también algunos comandos básicos para probar. Al final de esas descripciones, se puede escribir el código R, pulsar Intro y obtener el resultado debajo de la línea de código (por ejemplo, ejecutar 2\*2). Se puede hacer, por ejemplo:

* Instalación y carga de paquetes R
* Realizar operaciones matemáticas simples o complejas
* Asignar el resultado de una operación a una variable
* Importación de datos
* Creación de tipos comunes de objetos R, como vectores, matrices o dataframes
* Exploración de datos
* Análisis estadístico
* Creación de visualizaciones de datos

**Environment**

Cada vez que se define una nueva variable o se reasigna una existente en RStudio, se almacena como un objeto en el área de trabajo y se muestra, junto con su valor, en la pestaña “Environment” (Entorno, Ambiente), en la parte superior derecha de la ventana de RStudio.

**B) Otras pestañas importantes**

**Terminal:** para ejecutar comandos desde el terminal

**Historial:** para realizar un seguimiento del historial de todas las operaciones realizadas durante la sesión actual de RStudio.

**Archivos:** para ver la estructura de la carpeta de trabajo, restablecer la carpeta de trabajo, navegar entre carpetas, etc.

**Plots:** para previsualizar y exportar las visualizaciones de datos creadas

**Paquetes:** para comprobar qué paquetes se han cargado y cargar o descargar paquetes (activando/desactivando la casilla situada a la izquierda del nombre de un paquete).

**Scripts de R en RStudio**

En el entorno RStudio, un script de R es un archivo de texto con extensión “.R” que contiene secuencias de instrucciones escritas en lenguaje R. Su función principal es almacenar, documentar y ejecutar código de manera reproducible, lo que resulta esencial para el trabajo académico y científico. A diferencia de la consola, donde los comandos se ejecutan de forma inmediata y efímera, el script constituye un registro permanente del proceso analítico, permitiendo replicar los resultados o compartirlos con otros investigadores. Los scripts en R operan como cuadernos de laboratorio digitales, donde el investigador puede registrar cada paso analítico y comentar el propósito de cada bloque de código.

**Creación de un script en RStudio**

1. Para crear un nuevo script en RStudio, se siguen los siguientes pasos:
2. En la barra superior, seleccionar File → New File → R Script.
3. Aparecerá una nueva pestaña en el panel superior izquierdo, donde se puede escribir el código.
4. Guardar el archivo con un nombre significativo, mediante mediante el comando File → Save As o presionando Ctrl + S.
5. Al guardar el archivo, RStudio agrega automáticamente la extensión R, la cual indica que se trata de un script ejecutable dentro del entorno R.

**Estructura básica de un script**

Un buen script para ejecutar comandos, consiste en documentar el proceso analítico de manera clara. Por ello, se recomienda seguir una estructura ordenada. Por ejemplo:

# 1. Cargar o crear datos

# En este ejemplo se crea un conjunto de datos simple sobre estudiantes

# que incluye edad, promedio académico y género.

estudiantes <- data.frame(

nombre = c("Ana", "Luis", "María", "Carlos", "Elena", "José"),

edad = c(20, 22, 19, 21, 20, 23),

promedio = c(8.5, 7.8, 9.1, 7.5, 8.9, 8.0),

genero = c("Femenino", "Masculino", "Femenino", "Masculino", "Femenino", "Masculino")

)

# 2. Exploración inicial de los datos

# Mostrar las primeras filas y un resumen estadístico

head(estudiantes)

summary(estudiantes)

# 3. Cálculos básicos

# Calcular la media del promedio académico y la edad promedio

mean(estudiantes$promedio)

mean(estudiantes$edad)

# 4. Visualización básica

# Graficar la distribución de promedios por género

barplot(

tapply(estudiantes$promedio, estudiantes$genero, mean),

main = "Promedio académico según género",

ylab = "Promedio",

col = c("lightblue", "lightgreen")

)

En este ejemplo, los comentarios (precedidos por #) sirven para explicar cada bloque de código, facilitando la comprensión y posterior revisión del análisis.

**Ejecución de comandos desde el script**

Una vez escrito el código, se puede ejecutar parcial o totalmente de varias formas:

* Para ejecutar una línea específica: ubicar el cursor en la línea y presionar Ctrl + Enter.
* Para ejecutar un bloque de líneas: seleccionar el bloque y presionar la misma combinación.
* Para ejecutar todo el script: usar el menú Code → Run Region → Run All.

RStudio enviará los comandos a la consola, donde se mostrarán los resultados o gráficos generados.

Esta interacción entre script y consola materializa el principio de “programación interactiva”, permitiendo al estudiante experimentar con modificaciones, observar los efectos y desarrollar un pensamiento estadístico activo.

**Integración de scripts con proyectos de RStudio**

RStudio permite crear proyectos (Projects) que funcionan como carpetas de trabajo organizadas. Cada proyecto puede contener varios scripts, archivos de datos y resultados. Esta estructura favorece la organización modular del trabajo estadístico.

Pasos para crear un proyecto:

Menú File → New Project → New Directory → New Project.

Asignar un nombre (por ejemplo, “AnálisisEducativo”).

Guardar en una ubicación deseada del disco.

Una vez creado el proyecto, cualquier script o conjunto de datos guardado dentro de esa carpeta será fácilmente accesible, sin necesidad de definir rutas largas.

* 1. **Gráficos en R. Interpretación y presentación de resultados**

La visualización de datos constituye una de las fases relevantes del análisis estadístico, ya que permite comprender patrones, relaciones y tendencias que no siempre son evidentes mediante los cálculos numéricos. Como sostiene Healy (2018): “la visualización es una extensión del pensamiento estadístico, pues traduce la complejidad de los datos en estructuras cognitivamente accesibles” (p. 9). En R, la generación de gráficos es una de sus mayores fortalezas, tanto por la versatilidad de las funciones base como por el ecosistema de paquetes dedicados a la representación visual (Wickham, 2016).

R ofrece un sistema gráfico flexible que se organiza en dos niveles principales:

* **Gráficos base:** Son las funciones incluidas en el propio lenguaje R (por ejemplo, plot(), hist(), barplot(), boxplot()), que permiten crear representaciones simples y rápidas.
* **Gráficos avanzados:** Se basan en librerías externas como ggplot2, lattice o plotly, que implementan modelos más elaborados de visualización interactiva o multivariante.

Como explica Wickham (2016), “la gramática de los gráficos en ggplot2 se inspira en la idea de que cada visualización puede entenderse como la suma estructurada de capas” (p. 3). Esta lógica modular posibilita un diseño progresivo y personalizable de las representaciones gráficas.

**25.7.1 Tipos de gráficos básicos en R**

El entorno R incorpora un conjunto de funciones para la elaboración de gráficos que pueden clasificarse según el tipo de variable que se desea representar.

**a. Histogramas**

Representan la distribución de una variable numérica continua, mostrando la frecuencia con que los datos se agrupan en intervalos.

# Ejemplo: distribución de edades de estudiantes

edades <- c(18, 19, 20, 21, 22, 20, 19, 21, 22, 23, 18, 20)

hist(edades,

main = "Distribución de edades de los estudiantes",

xlab = "Edad",

ylab = "Frecuencia",

col = "lightblue",

border = "white")

El histograma permite visualizar la forma de la distribución (simétrica, sesgada, unimodal o multimodal), aspecto fundamental para seleccionar los métodos estadísticos adecuados (Crawley, 2015).

**b. Diagramas de barras**

Los diagramas de barras se utilizan para variables categóricas. En investigación educativa, por ejemplo, pueden mostrar la frecuencia de respuestas según el género o la preferencia por estrategias pedagógicas.

# Ejemplo: preferencia por estrategias didácticas

estrategia <- c("Colaborativa", "Individual", "Mixta", "Colaborativa", "Mixta", "Colaborativa")

barplot(

table(estrategia),

main = "Preferencia por tipo de estrategia didáctica",

ylab = "Frecuencia",

col = "lightgreen"

)

Los diagramas de barras facilitan la comparación entre categorías y la identificación de diferencias en proporciones.

**c. Diagramas de dispersión**

Estos gráficos representan la relación entre dos variables numéricas, permitiendo observar tendencias o correlaciones.

# Ejemplo: relación entre horas de estudio y calificación

horas <- c(2, 3, 4, 5, 6, 7, 8, 4, 6, 5)

nota <- c(6.5, 7.0, 7.5, 8.2, 8.7, 9.0, 9.5, 7.2, 8.4, 8.0)

plot(horas, nota,

main = "Relación entre horas de estudio y calificación",

xlab = "Horas de estudio semanales",

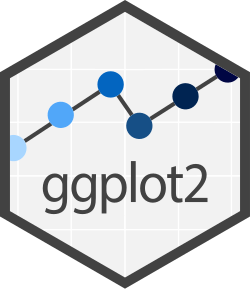
ylab = "Calificación promedio",

pch = 19,

col = "blue")

La pendiente visual del conjunto de puntos permite inferir si existe una relación positiva o negativa entre las variables.

**25.7.2. Visualización avanzada con ggplot2**



El paquete ggplot2 se ha convertido en el estándar moderno de la visualización en R, gracias a su estructura declarativa y la alta calidad de sus resultados.

Ejemplo:

library(ggplot2)

# Datos simulados sobre satisfacción estudiantil

datos <- data.frame(

carrera = c("Educación", "Psicología", "Derecho", "Ingeniería", "Medicina"),

satisfaccion = c(8.4, 7.9, 8.1, 7.5, 8.6)

)

# Gráfico con ggplot2

ggplot(datos, aes(x = carrera, y = satisfaccion, fill = carrera)) +

geom\_col() +

theme\_minimal() +

labs(

title = "Nivel promedio de satisfacción estudiantil por carrera",

x = "Carrera universitaria",

y = "Satisfacción (escala 1-10)"

)

El resultado es un gráfico de barras elegante, con colores automáticos, etiquetas legibles y estilo profesional, ideal para informes académicos y artículos científicos.

Además, ggplot2 permite añadir capas como líneas de tendencia (geom\_smooth()), etiquetas (geom\_text()) o facetas (facet\_wrap()), adaptándose a las necesidades de cada investigación.

**Consideraciones estéticas y comunicativas**

La función principal de un gráfico no es adornar, sino comunicar evidencia empírica de forma clara y honesta. Una visualización eficaz combina precisión analítica con expresividad narrativa, evitando distorsiones perceptivas o exageraciones visuales.

Por ello, se recomienda seguir ciertas pautas:

* Usar escalas y proporciones adecuadas.
* Evitar colores que dificulten la lectura o confundan categorías.
* Incluir títulos, etiquetas y leyendas completas.
* Favorecer la sobriedad estética sobre el exceso decorativo.

**Exportación de gráficos**

Los gráficos generados pueden exportarse desde el panel Plots o mediante código, lo que permite integrarlos en informes y presentaciones.

* 1. **Síntesis del capítulo**

Finalmente, destacamos que en este Capítulo 13 se introduce al R como un lenguaje de programación de estadística libre y abierto, desarrollado por Ross Ihaka y Robert Gentleman en los años 90. Se posiciona frente a alternativas como SPSS, Stata y Python, por su flexibilidad, reproducibilidad y su fortaleza en la capacidad gráfica. Se mencionan las bases de programación esenciales, de tipos de datos (Numéricos, Caracteres, Lógicos, Factores) y las estructuras clave como Vectores, Matrices y, fundamentalmente, *Data Frames*. Además, se resaltan los operadores lógicos y aritméticos, el control de flujo y la creación de funciones personalizadas para automatizar tareas.

En la segunda parte nos referimos a la Estadística Descriptiva en R, presentando las medidas de tendencia central (Media, Mediana, Moda) y las de dispersión (Varianza, Desviación Estándar), y sus respectivas funciones en el *software* (mean(), median(), table(), var(), sd()). Se detalla la construcción de tablas de frecuencias, diferenciando el tratamiento de variables cualitativas (a través de la codificación) y cuantitativas (discretas y continúas agrupadas).

Finalmente, el capítulo guía la instalación de R y RStudio y demuestra la importancia de la visualización, mostrando la generación de gráficos base e introduciendo el paquete ggplot2 para obtener figuras de alta calidad profesional.

**Referencias**

Contento, R. (2019). Estadística con aplicaciones en R. Bogotá: Universidad de Bogotá Jorge Tadeo Lozano. 412 pp.

Crawley, M. J. (2012). The R Book (2ª ed.). John Wiley & Sons. https://livresbioapp.wordpress.com/wp-content/uploads/2015/07/crawley-the\_r\_book.pdf

Crawley, M. J. (2015). *Statistics: An introduction using R* (2nd ed.). Wiley. https://minerva.it.manchester.ac.uk/~saralees/statbook4.pdf

Healy, K. (2018). Data visualization: A practical introduction. Princeton University Press. https://jwmason.org/wp-content/uploads/2021/08/Kieran-Healy-Data-Visualization\_-A-Practical-Introduction-Princeton-University-Press-2019.pdf

Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5(3), 299–314. https://bookdown.org/rdpeng/rprogdatascience/history-and-overview-of-r.html

McKinney, W. (2022). Python for Data Analysis (3rd ed.). O’Reilly Media. URL: https://wesmckinney.com/book/

Vicuña A. & Ponce J. (2022). Estadística Descriptiva con R. Universidad Técnica Estatal de Quevedo, Ecuador. 252 pp.

Wickham, H., & Grolemund, G. (2023). R para Ciencia de Datos. https://es.r4ds.hadley.nz/

Wickham, H. (2016). ggplot2: Elegant graphics for data analysis (2nd ed.). Springer. https://ms.mcmaster.ca/~bolker/misc/ggb.pdf

Hernández, G. J. P. (2015). Elementos básicos de estadística descriptiva para el análisis de datos. FUNLAM. https://www.funlam.edu.co/uploads/fondoeditorial/120\_Ebook-elementos\_basicos.pdf

Salazar, C. (2015). Fundamentos básicos de estadística. (Edición digital). https://gc.scalahed.com/recursos/files/r161r/w24899w/Trabajo%20Final/Fundamentos\_Basicos\_de\_Estadistica.pdf

1. Puede hacerse desde la página http://cran.r-project.org/. [↑](#footnote-ref-1)
2. Link: <https://posit.co/download/rstudio-desktop/> [↑](#footnote-ref-2)