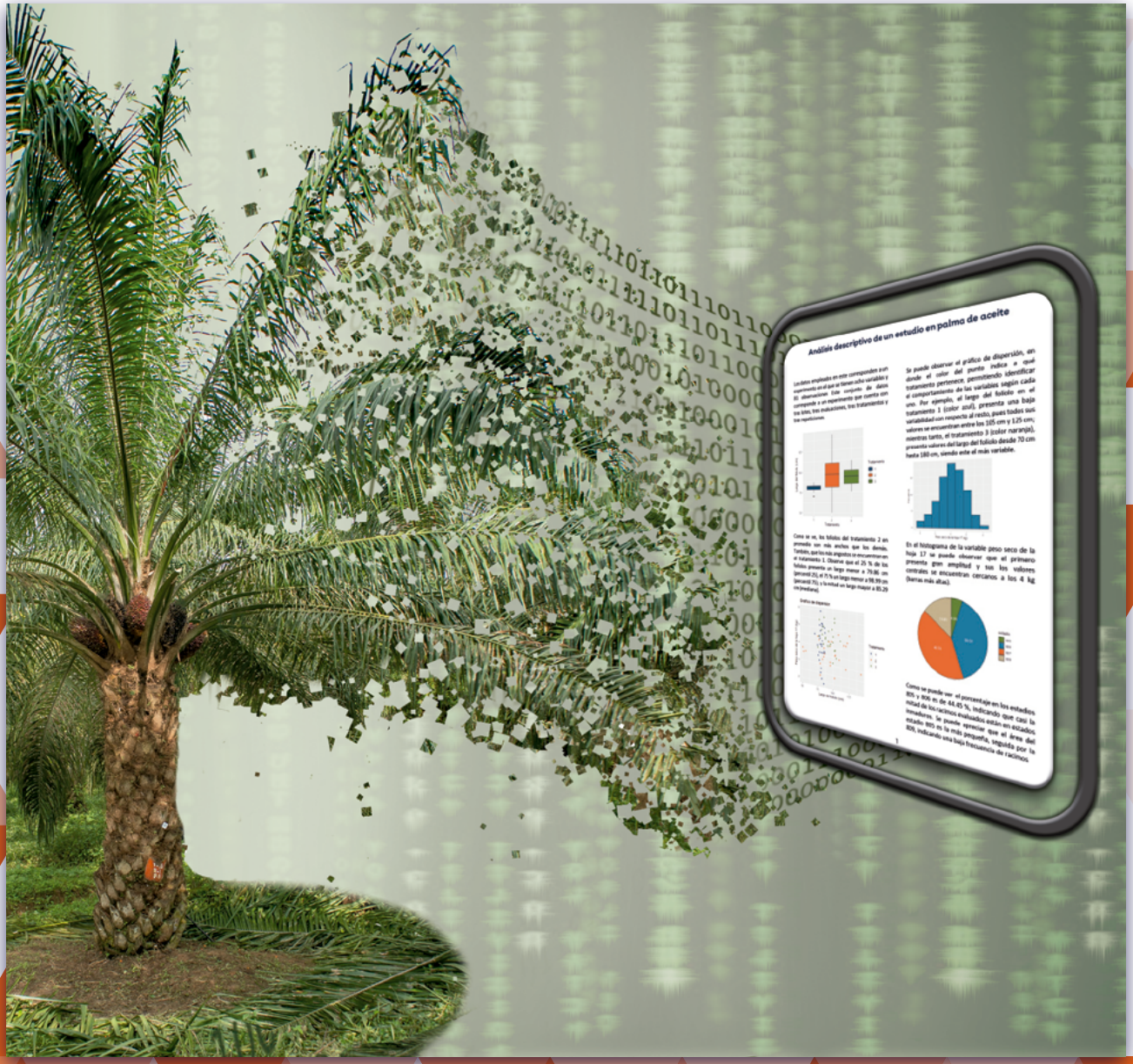

Introducción al análisis de datos con RStudio

Liseth Estefanía Vargas, Eloína Mesa Fuquen

cenipalma 30 años

Introducción al análisis de datos con RStudio

Introducción al análisis de datos con RStudio

Publicación de la Corporación Centro de Investigación en Palma de Aceite, Cenipalma, con el apoyo del Fondo de Fomento Palmero, administrado por Fedepalma.

Director General
Alexandre Patrick Cooman

Autores
Liseth Estefanía Vargas
Auxiliar de Investigación II
Eloína Mesa Fuquen
Investigadora Asociada

Cordinación editorial
Yolanda Moreno Muñoz

Diseño y diagramación
Fredy Johan Espitia B.

ISBN: 978-958-8360-83-6

Impresión
Estudio 45-8 S.A.S.

Cenipalma
Calle 98 No. 70-91, piso 14.
PBX: (57-601) 313 8600
Bogotá, D.C., Colombia
www.cenipalma.org

Septiembre de 2021

Contenido

Listado de figuras

Listado de tablas

Introducción

En el marco del análisis de datos es importante contar con una caja de herramientas, y su correcto uso aumenta las posibilidades de éxito en el análisis. Este manual tiene como objetivo, ofrecer a los usuarios una conceptualización básica de la estadística descriptiva, para emplearla en el análisis de datos con el *software* RStudio, explicando su utilidad de una forma sencilla.

Al tener la capacidad para realizar el análisis descriptivo de sus datos, identificando con claridad su estructura y las características más importantes de las variables bajo estudio, el usuario podrá entender el comportamiento de las observaciones y llegar a conclusiones acertadas.

En este documento se presentan los conceptos básicos para el análisis de datos, empleando el *software* estadístico R (R Core Team, 2020) versión 4.0.4. Se usa un conjunto de datos correspondiente a un experimento en un cultivo de palma de aceite, en el que se evaluaron algunas características vegetativas.

A continuación, se muestra una breve descripción de las temáticas abordadas en el manual.

La primera sección abarca una introducción al lenguaje de programación R y su interfaz RStudio. Se realiza un reconocimiento de las partes básicas del *software*, definiendo conceptos claves como consola, script, ambiente de trabajo, entre otros. La segunda, expone los pasos básicos en RStudio. Se indica cómo fijar un directorio de trabajo y crear una sección en el script, se presentan la sintaxis del lenguaje R y algunos de los operadores más conocidos. La tercera, aborda la conceptualización básica para el entendimiento de un conjunto de datos, la definición de variable, sus tipos y escalas de medición. Adicional a esto, se describe el paso a paso para la importación y manejo de un conjunto de datos en RStudio. En la cuarta, se habla de las medidas numéricas descriptivas, su cálculo en R y su interpretación. Se muestran tres: las de tendencia central, las de posición y las de variación. La quinta, cubre el tema de representación gráfica, aplicaciones e interpretación. Los gráficos abordados son: de dispersión, histograma, de cajas y bigotes, de líneas, de barras y circular. Para terminar, en los apéndices se expone el conjunto de datos que se trabaja a lo largo del manual, y la guía para instalar R y RStudio.

La estrategia para desarrollar esta guía de aprendizaje tiene varios componentes para tener en cuenta. Se utilizan códigos que el lector puede copiar y pegar en su consola de R, para obtener los resultados aquí presentados. Estos se destacan en una caja similar a la que aparece a continuación.

```
print ('Introducción al análisis de datos con R')
2 + 2
1:10
```

Los resultados obtenidos al ejecutar el código se muestran en rojo con dos símbolos de numeral (##) al inicio de cada línea o renglón; el usuario NO los debe copiar. Los resultados obtenidos luego de correr el código anterior son:

```
## [1] 'Introducción al análisis de datos con R'
## [1] 4
## [1] 1  2  3  4  5  6  7  8  9 10
```

Para resaltar algún aspecto importante, se emplean bloques informativos:

Nota

Advertencia

Introducción a R y RStudio

Foto: Zúñiga, F.

¿Qué es R?

R es un lenguaje de programación y se mantiene en un ambiente para cómputo estadístico y gráfico (Santana & Farfán, 2014). Permite al usuario (o programador) escribir una serie de instrucciones u órdenes de manera organizada, concentrándose en el manejo, análisis, procesamiento y visualización de datos.

Para comprender el origen del lenguaje de programación en R, es importante conocer algunos puntos históricos relevantes en su evolución. Fue creado en 1993 por los profesores e investigadores Robert Gentleman y Ross Ihaka (Usuga & Hernández, 2021). Sin embargo, sus inicios se remontan al lenguaje previo llamado S, de John Chambers y sus colaboradores en los Laboratorios Bell, durante la década de los setenta. Cabe señalar que desde 1995, el código fuente de R está disponible bajo licencia pública GNU para sistemas operativos Windows, MacOS y distribuciones Linux. Dicha licencia se mantiene con acceso libre por la Fundación R (*R Foundation*), y se puede ejecutar, copiar, distribuir, estudiar, modificar y mejorar sin restricción alguna (Gough, 2009).

La Fundación R es una organización sin fines de lucro, concebida por los miembros del *R Development Core Team*, cuyo objetivo es proporcionar un apoyo a R y las innovaciones en estadística computacional que requiera, asegurando un desarrollo continuo (R Core Team, 2020). En este manual se utiliza R para hablar del lenguaje de programación, más no la aplicación o paquete estadístico R. Esta aclaración tiene como excepción la descarga e instalación del programa con el mismo nombre.

Nota

Este manual no se concentra en el manejo del programa R específicamente. En su lugar se trabaja con RStudio, un entorno de desarrollo integrado que surgió de R, cuyas características lo hacen más fácil de usar.

¿Qué es RStudio?

RStudio es el principal entorno de desarrollo integrado o IDE[1] para R, que es un *software* libre disponible para los sistemas operativos Windows, MacOS, y Linux (RStudio Team, 2020). De manera sencilla, un IDE es un programa compuesto por cuatro herramientas:

- **Editor de texto:** permite escribir instrucciones u órdenes de forma organizada, sin ningún tipo de formato como negrilla, cursiva, texto centrado, entre otros (Figura 1). Es decir, deja crear archivos de texto plano.
- **Compilador:** es el encargado de traducir las instrucciones dadas a R en código de máquina, para que pueda ser interpretado y ejecutado. Para hacerlo, primero comprueba la sintaxis del código verificando que no se presenten errores, luego crea y ejecuta el código de máquina, y finalmente devuelve un resultado.
- **Depurador:** esta herramienta tiene la función de evaluar e identificar los errores en el código. Si existe uno, detiene el procesamiento, expone el error, su posición y la posible causa. En la Figura 2 se muestra un error presentado en la consola de R.

1 Sigla en inglés de *integrated development environment*.

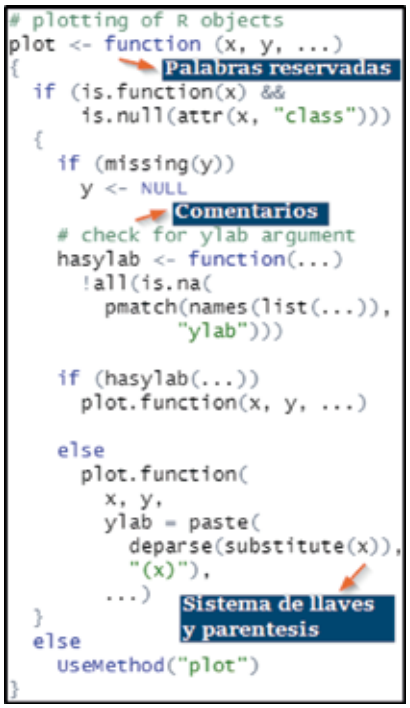

Figura 1. Editor de texto en RStudio

- **Interfaz gráfica:** permite tener versatilidad y comodidad, ya que muestra un entorno amable. Posibilita al usuario "interactuar" de una manera fácil con el computador, pues su entorno está constituido por una serie de menús e iconos, que representan las opciones que se pueden utilizar dentro del sistema. Por la importancia de su manejo para el desarrollo de este manual, a continuación se abordan sus características más relevantes.

Interfaz gráfica de RStudio

En esta sección se muestra detalladamente la interfaz gráfica de RStudio y sus partes principales. También se hace una introducción básica para su manejo. Para empezar, es importante mencionar que está compuesta por cuatro ventanas principales: el editor de texto o script, la consola, el entorno de trabajo y una de varios que abarca cinco pestañas distintas. Cuando se abre por primera vez, se observan solo tres de las cuatro ventanas, ya que el script aún no ha sido creado (Figura 3).

Cuenta con dos áreas principales: una de entrada para dar instrucciones al computador, y una de salida donde aparecen los resultados. Se debe tener en cuenta que RStudio proporciona una manera de "comunicarse" con el computador, mientras que R ofrece el lenguaje para hacerlo.

1. **Área de entrada.** Se compone de dos ventanas: la consola y el script. En estas se escriben las instrucciones que en adelante se denominarán códigos o comandos, para que el programa las ejecute.

   a. **La consola:** se visualiza en la parte izquierda del programa y su uso es simple. El código se puede escribir en la parte inferior y se presiona la tecla Enter para que sea ejecutado. Además, muestra todos los comandos que se han escrito, los resultados obtenidos y los errores o alertas que presente el código. Debido a estas características, es la ventana que ofrece una comunicación constante entre el usuario y el *software*.

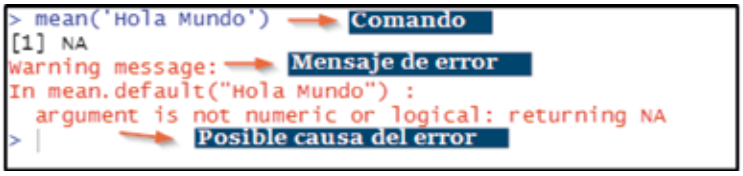

Figura 2. Mensaje de error en la consola de RStudio

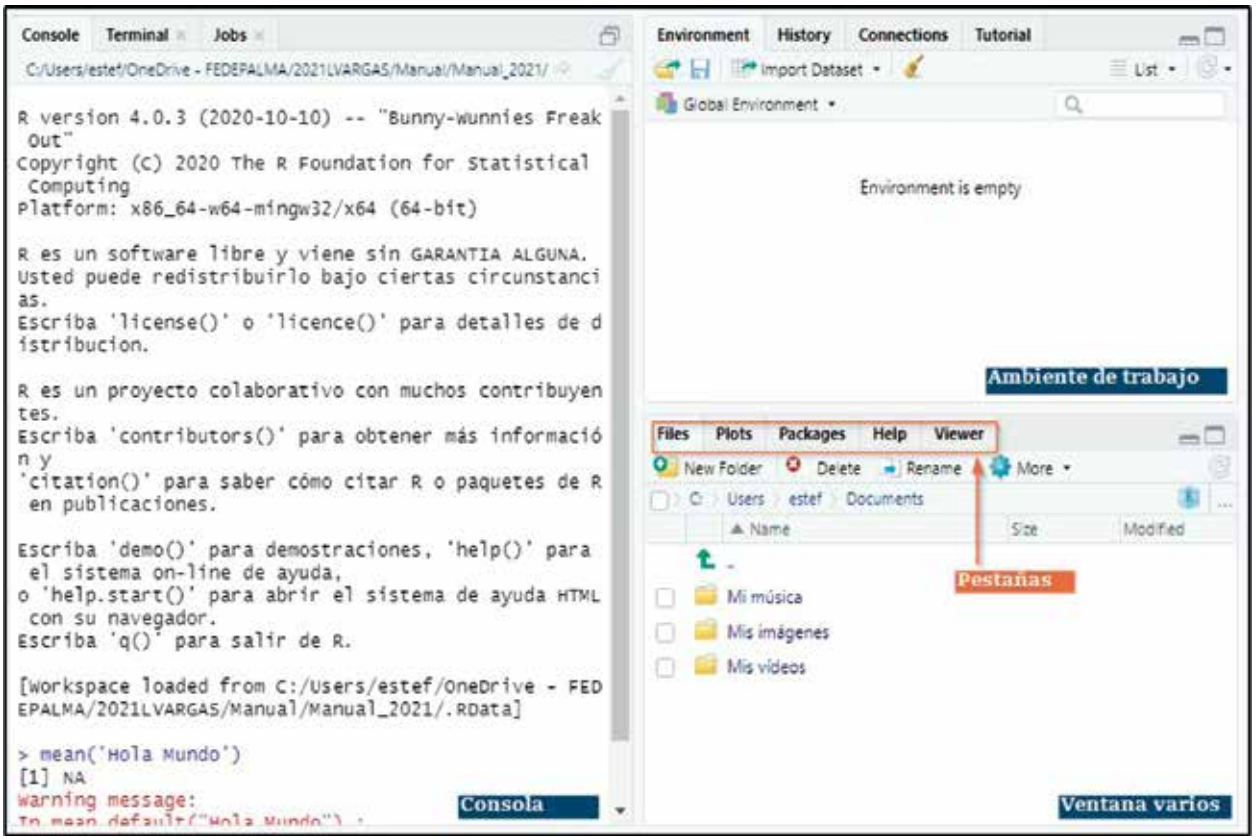

Figura 3. Interfaz gráfica de RStudio

Advertencia

Al escribir el código directamente en la consola, este no se guarda en ninguna parte. Por esto es recomendable anotarlo en el script, del que se habla a continuación.

b. El script: editor de texto de RStudio, que permite escribir y ejecutar el código. A diferencia de la consola, el script deja guardar dicho código para utilizarlo en futuras ocasiones. Al abrir RStudio por primera vez no se observa y es necesario crearlo. Para esto, se hace clic en el botón New file que se encuentra en la esquina superior izquierda (Figura 4), y se selecciona R Script. También es posible hacerlo con el atajo de teclado Ctrl + Shift + N.

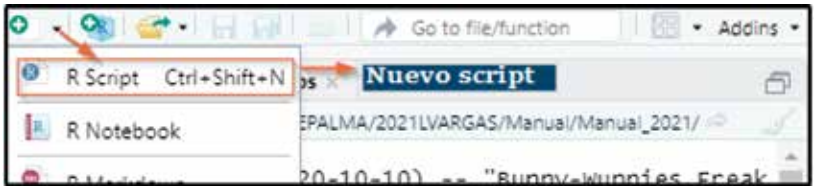

Figura 4. Nuevo script en RStudio

Una vez se tiene el script, emerge una ventana en la parte superior izquierda, justo encima de la consola. Esta se asemeja a un bloc de notas usual. Sin embargo, en la parte superior cuenta con diferentes opciones tales como guardar, ejecutar todo el código o una línea de comando, entre otras (Figura 5).

Una vez creado el script, es fundamental guardarlo: primero se hace clic en el botón Save document que se encuentra en la parte superior izquierda, o se usa el atajo de teclado Ctrl + S. Posterior a esto, aparece una ventana emergente que permite conservar el archivo en cualquier carpeta del computador.

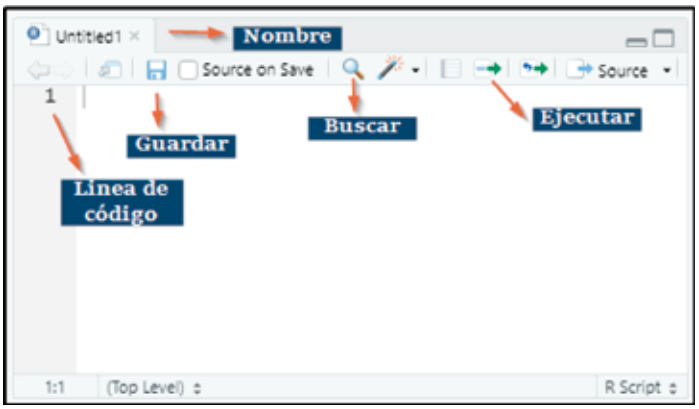

Figura 5. Editor de texto (script) de RStudio

Nota

Se recomienda crear una carpeta llamada "Manual" en el escritorio del equipo, para almacenar allí todos los archivos. El script se caracteriza por tener una extensión .R, su nombre no debe contener espacios ni caracteres alfanuméricos (puntos, tildes, letra ñ, especiales como @, &, *), ya que puede presentar problemas al abrir el archivo.

Advertencia

No es suficiente escribir el código, se debe ejecutar para que RStudio lo reconozca y realice la orden indicada. Correr un comando en R es solicitar que cumpla la instrucción dada. Para hacerlo en la consola, se presiona la tecla Enter mientras que en el script se oprimen Ctrl + Enter, o se hace clic en el botón RUN que se encuentra en la parte superior derecha del script.

2. **Área de salida.** Cuenta con tres partes: el ambiente de trabajo, la ventana varios y la consola. Esta área se encarga de mostrar resultados, visualizar gráficos, presentar ayudas y manuales para el manejo de R y RStudio.
   a. **Ambiente o entorno de trabajo:** se visualiza en la parte superior derecha de la interfaz. Aquí se muestra el conjunto de datos que se está trabajando, los resultados obtenidos al ejecutar el código, funciones, variables y constantes empleadas (Figura 6). Al cerrar RStudio estos se pierden, por lo que es recomendable guardar el ambiente de trabajo en un archivo con extensión .Rdata.

Nota

En R los datos trabajados se guardan como un objeto y se les asigna un nombre para identificarlos. Algunos de estos objetos son constantes, variables, tablas de datos, entre otros.

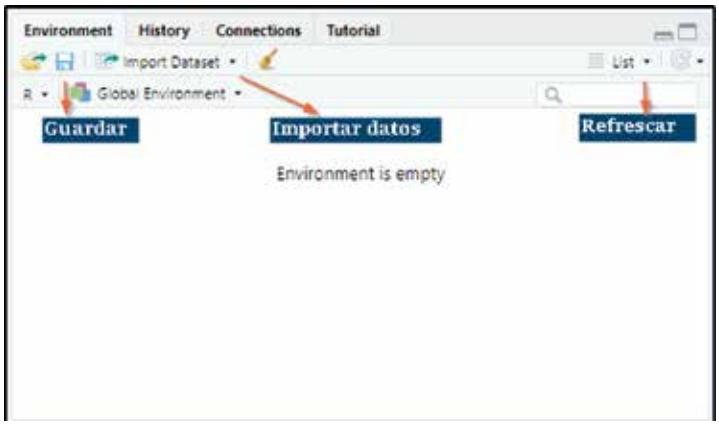

Figura 6. Ambiente de trabajo (Environment) de RStudio

b. **Ventana varios:** se encuentra en la parte inferior derecha y se compone por cinco pestañas: Files, Plots, Packages, Help y Viewer. Las primeras cuatro se describen a continuación.

- **Files:** aquí se muestran las carpetas del computador y por defecto se presenta la de documentos. Para cambiarla, se debe hacer clic en los tres puntos que están en el costado superior derecho (Figura 7), y en la ventana emergente se escoge, en este caso, la carpeta "Manual" que se encuentra en el escritorio.
- **Plots:** permite visualizar todos los gráficos generados (haciendo clic en las flechas de la parte superior izquierda), y los almacena temporalmente (Figura 8). También es posible exportarlos como imágenes (png, jpg, eps) o como archivos pdf.

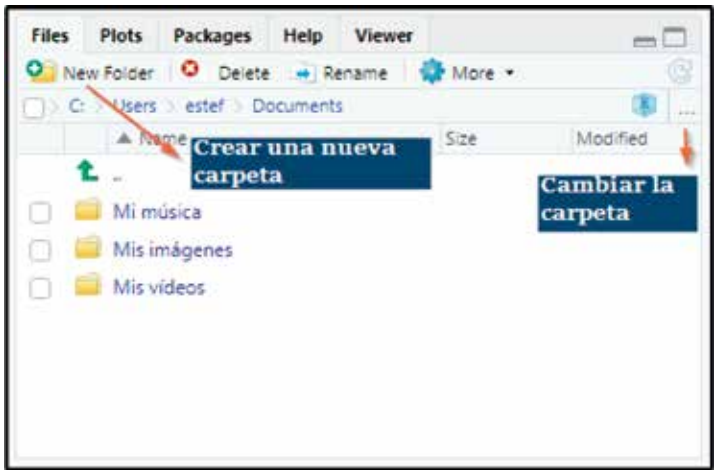

Figura 7. Pestaña de documentos (Files) en RStudio

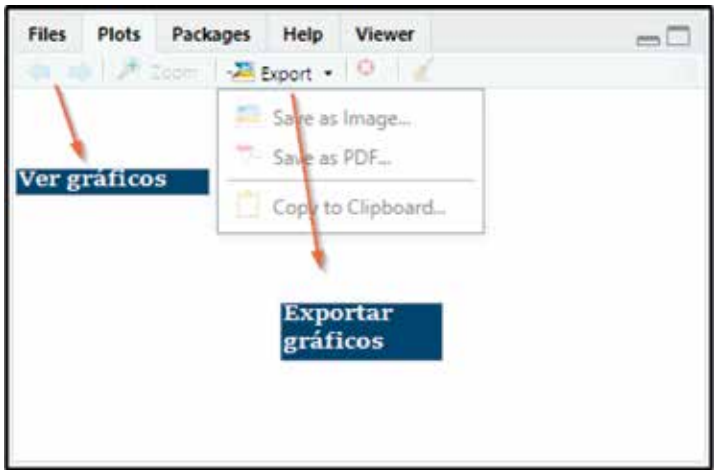

Figura 8. Pestaña de gráficos (Plots) en RStudio

- **Packages:** en esta se encuentran los paquetes de R, que almacenan todas las funciones para que sea posible emplear el *software*. Existen paquetes para diversas tareas como elaborar gráficos, importar conjuntos de datos, entre otros. Para utilizar algunos es necesario instalarlos primero, y luego se deben cargar, es decir, dejarlos listos para su ejecución (se explica en la página 25-Paquetes). Sin embargo, la instalación se realiza una sola vez en el computador, y la acción de cargar cada vez que se inicia una sesión de trabajo en R. Los paquetes se encuentran alojados en CRAN[2] (red de servidores web), así que pasan por un control riguroso antes de estar disponibles para su uso. La Figura 9 muestra esta pestaña con sus partes.

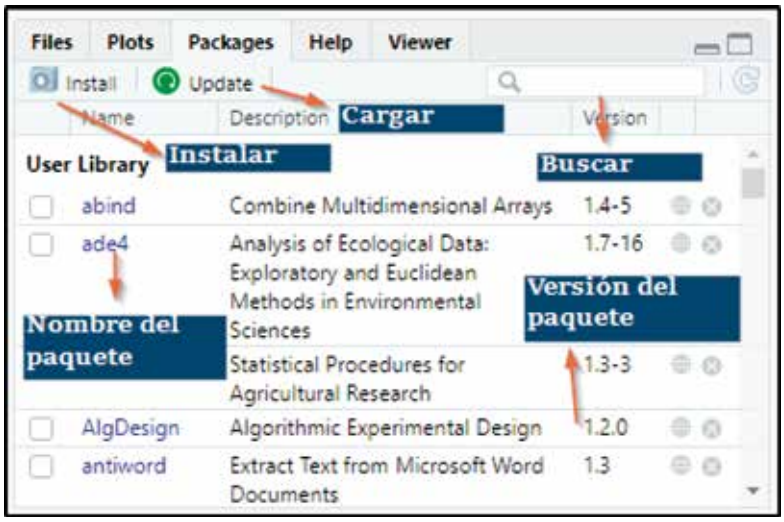

Figura 9. Pestaña de paquetes (Packages) en RStudio

2 Por su sigla en inglés: Comprehensive R Archive Network. Su página web es https://cran.r-project.org/

- **Help:** comprende todo el soporte y la documentación de RStudio. Permite acceder al CRAN, que ofrece diferentes recursos para el programador, tales como manuales para el usuario, cursos en línea, guías, entre otros. Además, presenta información de los paquetes instalados y sus funciones, como se observa en la Figura 10.

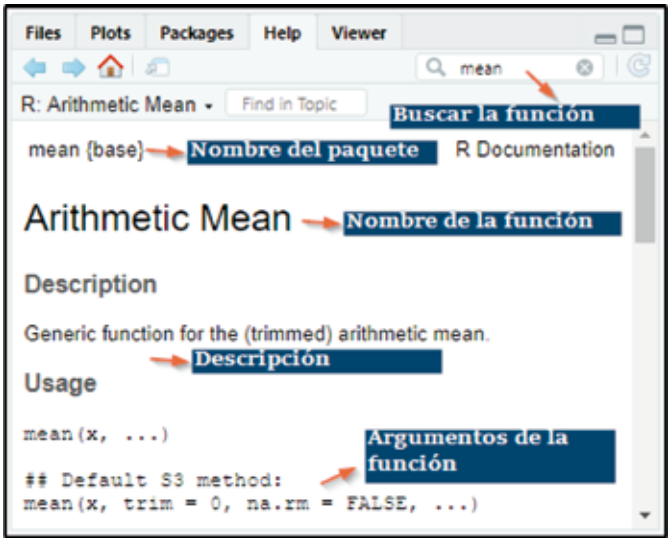

Figura 10. Pestaña de ayudas (Help) en RStudio

Las pestañas y conceptos se irán describiendo a lo largo de este manual. Sin embargo, es fundamental que el usuario explore la interfaz de RStudio, pues aquí únicamente se exponen los conceptos principales y las pestañas más utilizadas.

Uso básico de RStudio

Para ilustrar mejor las características y ventajas que ofrece RStudio, a continuación se explican algunos conceptos fundamentales para emplearlo.

Directorio de trabajo

Hace referencia a la carpeta donde se encuentran los archivos con los que se va a trabajar en RStudio. En este lugar, R busca archivos para importarlos y allí mismo exporta otros como gráficos, tablas, documentos, etc. Para definir el directorio de trabajo en R se tienen diversas opciones:

- Se ejecuta en la consola el comando setwd ("/directorio de trabajo/")

```
#Directorio de trabajo
-----------------------------------------------
setwd ('C:/Usuario/Desktop/Manual')
```

Advertencia

Para que el código anterior funcione, cada usuario deberá escribir su propia ruta pues esta cambia según el computador.

- En la parte superior de la pantalla se encuentra el botón Session. Al desplegarlo, se debe seleccionar la quinta opción denominada Set Working Directory. Esta a su vez muestra tres posibles alternativas (Figura 11):
  - Fijar la ubicación donde está guardado el script (To Source File Location).
  - Establecer la localización de la carpeta de documentos (Files) que se encuentra en la pestaña de ayudas (To Files Pane Location).
  - Elegir cualquier carpeta del computador (Choosse Directory).

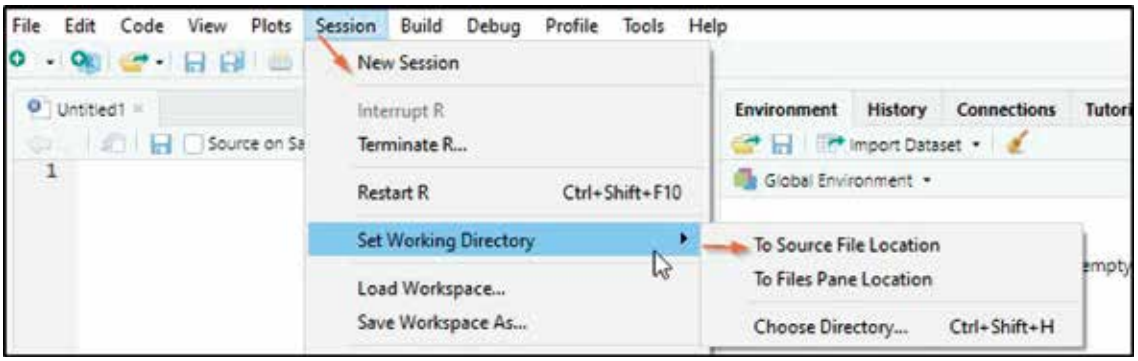

Figura 11. Fijar el directorio de trabajo en RStudio

El directorio de trabajo que se va a manejar es la carpeta "Manual", ubicada en el escritorio del computador. Como se observa en la Figura 12, una vez fijado, el comando empleado aparece en la consola y se sugiere escribirlo en el script para ejecutarlo en futuras ocasiones.

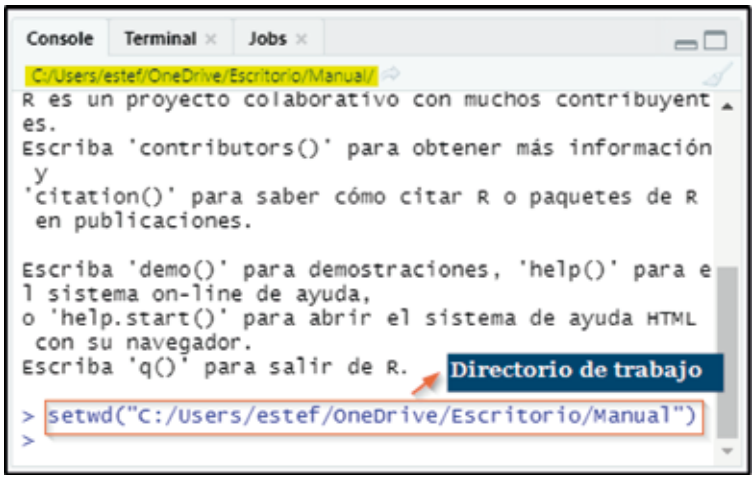

Figura 12. Salida en consola al fijar el directorio de trabajo

Nota

Si se está ejecutando un comando, el programa muestra un signo STOP en la esquina superior derecha de la consola (Figura 13). Si se presiona, se cancelará la operación en curso.

Figura 13. Símbolo STOP indicando que la consola está en ejecución

Sintaxis y estilo

Como en todo lenguaje, existen reglas ortográficas que se recomienda considerar. R tiene un conjunto de códigos que siguen un patrón lógico y cuentan con la siguiente estructura básica:

- **Asignación:** consiste en crear un "objeto" y asignarle un valor.[3] Suponga que quiere dar el valor de 5 al objeto **x**. Para hacerlo se sugiere emplear el símbolo **<-**. Es importante tener presente que R asume diferente las letras mayúsculas y minúsculas.[4] A continuación, se muestran algunos ejemplos.

```
#Asignar los valores 5, 9.5 y 10 a las variables 'x', 'y' y 'X'
x <- 5      #Asigna el valor 5 a la letra x
y <- 9.5    #Asigna el valor 9.5 a la letra y
X <- 10     #Asigna el valor 10 a la letra X (Mayúscula)
```

3 En R la cifra decimal se marca con punto y no con coma, siguiendo la notación norteamericana.
4 Observe que en el código presentado se asigna 5 a x (minúscula) y 10 a X mayúscula.

Cuando se genera una asignación, el objeto creado se muestra en el ambiente de trabajo (Figura 14).

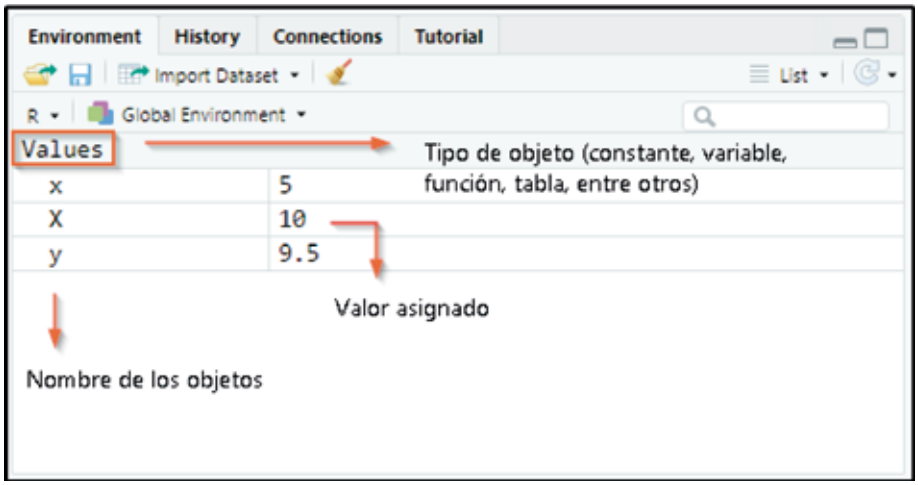

Figura 14. Ambiente de trabajo al crear objetos

A continuación, se exponen dos maneras para visualizar o imprimir el valor de una variable en consola: la primera consiste en llamar la variable de manera explícita por medio de la función print(), como sigue:

```
print(x)
## [1] 5
```

También se puede de manera implícita, simplemente escribiendo el nombre de la variable en la consola y ejecutándola.

```
x
## [1] 5
```

- **Comentarios:** RStudio admite escribir comentarios en el código. Para ello se debe digitar el símbolo numeral (#) antes del comentario, y automáticamente este cambia a color verde. Si se ubica el cursor en una línea o se seleccionan varias líneas de código, y se presiona la combinación de teclas Ctrl + Shift + C, las líneas quedan comentadas.

```
#Este es un comentario; Empieza con el
#carácter '#' y tiene un color
#verde
```

- **Nombres de los objetos:** se sugiere usar el símbolo _ dentro de los nombres. Por ejemplo, se puede escribir "peso_Racimo" en lugar de "peso Racimo", pues dejar espacios genera un error (Figura 15). Adicional a esto, no es posible empezar el nombre de un objeto con un número, ni emplear palabras reservadas del sistema como TRUE o FALSE.

```
peso_Racimo <- 16 #Nombre correcto
peso Racimo <- 16 #Nombre incorrecto
```

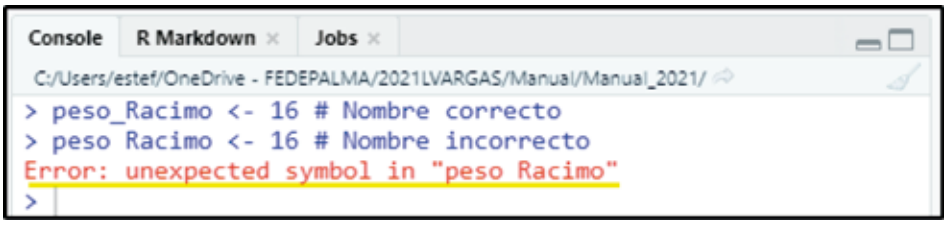

Figura 15. Salida en consola al escribir un nombre con espacio

- **Operadores básicos:** en RStudio se reconoce a los operadores binarios con los siguientes símbolos:
  - + Suma
  - - Resta
  - * Multiplicación
  - / División
  - ^ Potencia
  - %/% Cociente entero de una división
  - %% Residuo de una división

```
#Operaciones básicas
x+y # Sumar
```

```
## [1] 14.5
```

```
y-x #Restar
```

```
## [1] 4.5
```

```
x*y #Multiplicar
```

```
## [1] 47.5
```

```
x/y #Dividir
```

```
## [1] 0.5263158
```

```
x^3 #Elevar x a la potencia 3
```

```
## [1] 125
```

- **Pruebas lógicas:** R permite verificar si un objeto cumple una condición dada.
  - < indica si un número es menor que otro
  - > si es mayor que otro
  - == si es igual que otro
  - <= si es menor o igual que otro
  - >= si es mayor o igual que otro

```
#Pruebas lógicas
1 > 5
```

```
## [1] FALSE
```

```
x == y
```

```
## [1] FALSE
```

```
3 < 20
```

```
## [1] TRUE
```

```
x == X #Observe que aquí se diferencia entre
#mayúsculas y minúsculas
```

```
## [1] FALSE
```

```
x == x
```

```
## [1] TRUE
```

- **Operadores lógicos:** se emplean en conjuntos como la negación (!), intersección (&) y disyunción (|).

Funciones

Para definir el concepto de función, imagine que se tiene una máquina que necesita un insumo para generar un producto. Por ejemplo, una bombilla requiere energía eléctrica para producir luz (Figura 16).

Figura 16. Partes de una función

Una función tiene este mismo principio: precisa de un insumo, realiza un proceso interno y devuelve un resultado. Las funciones en R llevan un nombre corto y dan idea de lo que hacen. Usualmente, la mayoría de los comandos son funciones predeterminadas, pero también es posible crearlas. Sus partes son:

- **Entradas o argumentos:** sirven para ingresar los datos necesarios para que la función realice su procedimiento.
- **Cuerpo:** es el proceso interno que transforma los argumentos de entrada en objetos de salida o resultados.
- **Salidas:** son los resultados de una función, y toda función debe tener al menos uno.

En la Figura 17 se muestra la estructura general de una función en R. A continuación, se presenta el código para elaborar la función suma entre dos constantes.

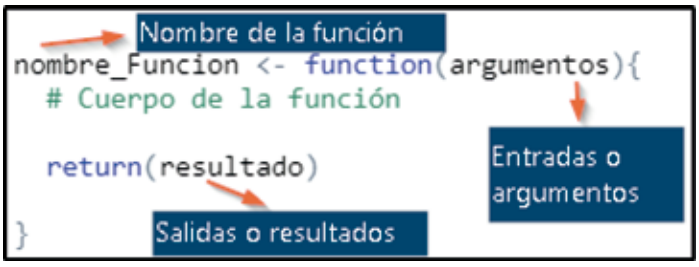

Figura 17. Definición de una función en RStudio

```
suma <- function (x,y) {
  s = x + y
  return (s)
}
```

```
suma (2,2)
## [1] 4
suma (3,4)
## [1] 7
```

Para ejecutar la función se debe escribir el nombre, en este caso suma, y entre paréntesis los argumentos necesarios como se expone a continuación:

Observe que al ejecutarla, en el ambiente de trabajo se crea una sección para las funciones (Figura 18). Esto permite identificar fácilmente los objetos como tablas, variables o constantes.

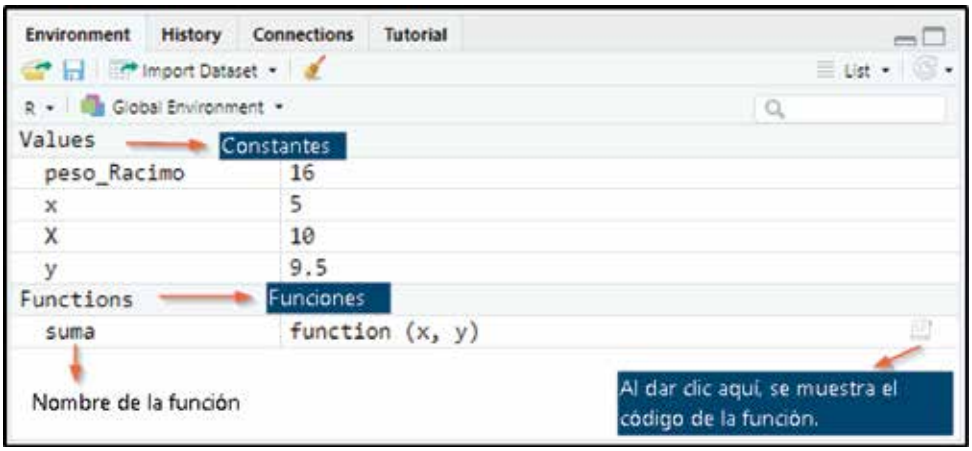

Figura 18. Ambiente de trabajo al crear una función

Secciones

Cuando se maneja un *software*, es importante mantener organizado el ambiente de trabajo. Para esto, R cuenta con una opción llamada sección que permite dividir el script en varias partes. Los pasos para crearla se presentan en la Figura 19.

Se debe ir al botón Code ubicado en la parte superior izquierda del menú, y seleccionar la opción Insert section (también se puede emplear el atajo de teclado Ctrl + Shift + R). Luego sale una ventana emergente donde se asigna el nombre. Finalmente, la sección aparece en el script con un color verde, y la línea de comando inicia con un #. A lo largo de este documento se irán generando secciones por cada tema presentado, las cuales se pueden observar en la parte inferior del script. Desde allí también es posible cambiar de sección y ver las funciones creadas.

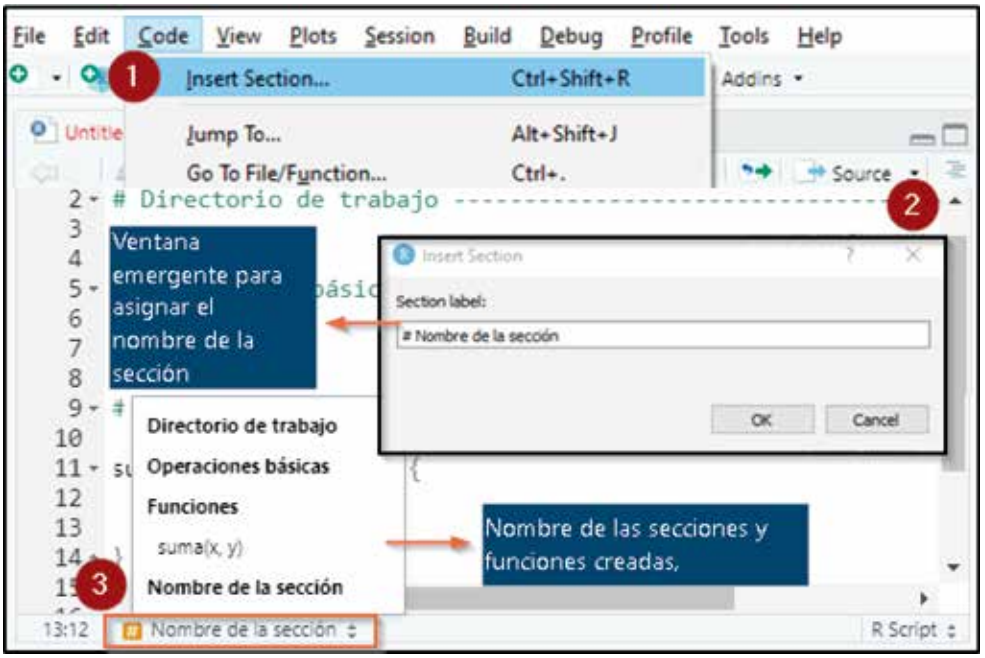

Figura 19. Crear una sección en RStudio

Paquetes

Para que el procesamiento de R sea eficiente, todas la funciones y conjunto de datos están guardados en paquetes. Existen unos especializados para hacer gráficos, importar y exportar datos, manejar archivos, entre otros. El sistema de R cuenta por defecto con el paquete base, que contiene la mayoría de las funciones elementales, y permite ejecutar el programa. A continuación, se muestra cómo instalar y cargar un paquete.

1. **Instalar paquetes:** se utiliza la función install.packages() dando como argumento el nombre del paquete entre comillas. Para descargarlo es necesario tener acceso a internet.
2. **Cargar paquetes:** una vez instalado, es preciso cargarlo con library() para poder emplear todas sus funciones.

Nota

Los paquetes se deben instalar una única vez en R. Sin embargo, se debe cargar siempre que se utilice el programa.

Para el desarrollo de este manual se instalarán dos paquetes: dplyr que proporciona una forma fácil, ágil y eficaz para manejar datos, y ggplot2 que se enfoca en la visualización de datos. El código necesario para la instalación se presenta a continuación.

```
#Instalar paquetes ----------------------------------------

install.packages ('dplyr') #Este código se ejecuta una única vez
install.packages ('ggplot2')

#Cargar paquetes ----------------------------------------

library ('dplyr') #Se cargan siempre que se necesiten
library ('ggplot2')
library ('readxl') #Este paquete permite importar archivos a R
```

Observe que el paquete readxl no se instala, ya que este se encuentra por defecto en R; solo se debe cargar para poder emplear sus funciones.

Entendimiento de los datos

Foto: Zúñiga, F.

Un **conjunto de datos** está organizado usualmente en tablas u hojas de cálculo (Excel). En cada celda se encuentra un **dato,** que es un registro de una medición hecha sobre una variable de interés. En las columnas se hallan las **variables,** que son atributos medibles y observables de personas o cosas que toman diferentes valores, y en las filas se ubican las **observaciones,** uno o varios datos, dependiendo de la cantidad de columnas que se tengan. Algunas variables están conformadas con números, como la altura de la palma en metros o el número de racimos por palma; mientras que otras lo hacen por categorías, como el cultivar de un lote, el grado de infestación de una plaga o la clase de suelo en un área. Los términos **variable cuantitativa** y **variable cualitativa** se emplean para distinguir ambos tipos.

Tipos de variables

- **Variable cuantitativa o numérica** consiste en cifras que representan conteos o mediciones. En R se identifican como numeric o num.

Nota

Para el manejo de las variables cuantitativas, es importante tener en cuenta su unidad de medida, el peso (gramos, libras, kilogramos, toneladas), las distancias (centímetros, metros, kilómetros), el tiempo (segundos, minutos, horas, días, años), entre otros.

Esta variable se divide a su vez en **discreta** y **continua**.

- **La variable discreta** generalmente proviene de un proceso de conteo. Por ejemplo: número de racimos por palma, de insectos en una hoja o de palmas en un lote.
- **La variable continua** puede tomar cualquier valor dentro de un intervalo de infinitos números reales. Por ejemplo: el peso del racimo, la altura de la palma, la acidez, la pluviosidad, la distancia, la temperatura, entre otros.

Para identificar el tipo de variable se emplea la función class() como sigue:

```
#Mostrar cuál es la clase o tipo de X
class(X)
```

```
## [1] "numeric"
```

```
#Mostrar cuál es la clase para x/y
class(x/y)
```

```
## [1] "numeric"
```

```
#Crear una variable de números llamada pesos
pesos <- c(12, 16, 9, 6, 5, 7)
#Mostrar el tipo de la variable pesos
class(pesos)
```

```
## [1] "numeric"
```

```
#Para identificar si una variable es numérica
is.numeric(x)
```

```
## [1] TRUE
```

```
is.numeric(y)
```

```
## [1] TRUE
```

```
#Identificar si la palabra 'Hola' es numérica.
is.numeric('Hola')
```

```
## [1] FALSE
```

```
#Convertir una variable a numérica
#(mientras sea posible)
as.numeric()
#Convertir la palabra 'Hola' a numérica no es
#posible y se genera un
#dato faltante (NA)
as.numeric('Hola')
```

```
## [1] NA
```

- **Variable cualitativa o categórica** representa cualidades o atributos por medio de nombres o etiquetas. En R, tienen diversas formas de identificarse.
  - El texto aparece como character o chr, y se reconoce fácilmente ya que está rodeado de comillas simples o dobles. Estos datos son los más flexibles de R, pues un texto puede contener letras, números, espacios, entre otros.
  - También se representan por medio del comando factor, que permite a R tener en cuenta las categorías de la variable.

Para ilustrar este tipo de variables se crea una vector con los nombres de algunos cosecheros, y otra paralela a la primera, con los lotes donde cada uno se encuentra.

```
#Crear las variables cosecheros y lotes
cosecheros <- c('Andrés', 'Juan', 'Pedro',
'Diana', 'María', 'Luis', 'Carlos', 'Raquel')
lotes <- c('lote 1', 'lote 3', 'lote 2', 'lote 1',
'lote 2', 'lote 3', 'lote 1', 'lote 3')

cosecheros
## [1] "Andres" "Juan"  "Pedro" "Diana" "Maria"
"Luis"  "Carlos" "Raquel"
lotes
## [1] "lote 1" "lote 3" "lote 2" "lote 1" "lote 2"
"lote 3" "lote 1" "lote 3"
#Identificar el tipo de las variables cosechero
#y lotes
class(cosecheros)
## [1] "character"
class(lotes)
## [1] "character"
#Para identificar si una variable o un dato es
#categórico
is.character(cosecheros)
## [1] TRUE
#Si se quiere tener en cuenta las categorías de
#la variable lotes, se puede emplear la función
as.factor()
#Para convertir una variable o un dato a
#categórico (mientras sea posible)
lotes = as.factor(lotes)
class(lotes)
## [1] "factor"
#Note que al ejecutar lotes, en la consola se
#muestra el vector y sus
#categorías (levels)
lotes
## [1] lote 1 lote 3 lote 2 lote 1 lote 2 lote 3
lote 1 lote 3
## Levels: lote 1 lote 2 lote 3
```

Advertencia

En ocasiones, los datos categóricos se identifican con números que reemplazan los nombres. Por ejemplo, se puede escribir 1 en lugar de "primero". Aunque parecerían cuantitativos, en realidad son cualitativos.

Escalas de medición

Una vez definidas las variables de interés en una investigación o estudio, los análisis que se realicen dependen de la manera en que estas se clasifiquen. Una forma común es empleando las escalas de medición (Clifford Blair & Taylor, 2008) con cuatro niveles diferentes: nominal, ordinal, de intervalo y de razón. Dada la variable a evaluar, se define la escala.

Escalas para variables cualitativas

Las variables cualitativas cuentan con dos escalas de medición: **nominales** y **ordinales**, que hacen referencia a datos que cuentan con categorías o atributos.

- **Los valores de las variables con escala nominal** se caracterizan por ser etiquetas, categorías o clases. Adicional a esto, "los datos carecen de cualquier orden o significancia numérica, es decir, ninguna de las categorías tiene mayor jerarquía que otra" (Triola, 2018). Ejemplos comunes son el tipo de cultivar sembrado, el nombre de las zonas palmeras, las plantaciones que participan de cierto estudio, etc.
- **Los valores de las variables con escala ordinal** consisten exclusivamente en nombres, etiquetas o categorías. Sin embargo, a diferencia de la nominal, las clasificaciones producidas cuentan con cierto orden (Clifford Blair & Taylor, 2008). Ejemplos comunes son el nivel de infestación (alto, medio o bajo) o los rangos de un examen (excelente, sobresaliente, aceptable e insuficiente).

Escalas para variables cuantitativas

Las variables cuantitativas tienen escalas de medición basadas en **intervalos** y en **razones**, refiriéndose a datos de una serie numérica ordenada.

- **Por intervalo** "es una escala numérica que indica el orden y la distancia con respecto a un cero arbitrario" (Isaza, 2012). Este concepto se refiere a que el punto cero en la escala, no representa ausencia de la característica medida (Clifford Blair & Taylor, 2008). El ejemplo más frecuente es el de la temperatura, en donde los 0 °C no indican la falta de temperatura, sino otro punto posible en la escala. Es por esto que no se puede afirmar que 36 °C representa dos veces más temperatura que 18 °C.
- **"Los datos provenientes de una escala de razón** indican tanto el orden como la distancia a un ***cero natural***" (Isaza, 2012). Es decir, esta escala cuenta con un sistema numérico en donde el cero expresa la ausencia de la característica medida. Esto permite que los cocientes y comparaciones entre los valores tengan significado. Por ejemplo, una palma que mide 4 m de altura es el doble de grande que una de 2 m; un racimo de 8 kg, pesa la mitad que uno de 16 kg.

Advertencia

Recuerde que el análisis estadístico se realiza sobre los datos, y esto genera la información con la que se toman decisiones. De aquí que la frase "el análisis de la información" tiende a generar confusiones, lo correcto sería "el análisis de los datos arroja la siguiente información ..."

Importación de archivos

Cuando se tienen datos propios provenientes de una investigación o estudio, el primer paso para empezar su exploración y análisis en R, es realizar una importación adecuada. R puede importar una gran variedad de archivos de datos (.txt, .xlsx, .csv) ofreciendo flexibilidad para su manejo.

Los datos empleados en este manual se presentan en el Apéndice A, y corresponden a un experimento en el que se tienen ocho variables y 81 observaciones. Se sugiere copiar la tabla de datos en un archivo Excel, empezando en la parte superior izquierda sin dejar filas ni columnas vacías, y guardar el archivo en la carpeta Manual que se fijó previamente como directorio de trabajo. El nombre recomendado es "datos1.xlsx". Sin embargo, se puede asignar cualquiera recordando no emplear espacios, signos de puntuación o caracteres especiales.

La importación de archivos se realiza por medio de readxl (Wickham & Bryan, 2019), uno de los principales paquetes para leer archivos de datos provenientes de Excel. Para cargarlo, se emplea la función library como se presentó en la página 25 (Paquetes).

```
#El paquete debe cargarse siempre que se
#inicie una nueva sesión en R
library ('readxl')
```

En este manual se emplea principalmente la función read_excel, que permite importar archivos desde Excel (.xlsy .xlsx) y tiene los siguientes argumentos principales:

- **path:** ruta del archivo a importar. Si no se especifica completamente y solo se indica el nombre del archivo (con su extensión), este será buscado en el directorio de trabajo fijado (ver página 19 Directorio de trabajo).
- **sheet:** nombre de la hoja a importar. Si no se detalla, se leerá la primera pestaña del Excel.
- **range:** rango de celdas que se quiere leer. Por defecto se leen las columnas que tengan contenido.
- **col_names:** permite definir si la pestaña a importar tiene encabezados para usar como nombres de columna. Su valor por defecto es TRUE.

Si se quieren consultar todos los argumentos se ejecuta la ayuda de la función.

```
help(read_excel)
#Una forma abreviada sería
?read_excel
```

Existen diferentes maneras de importar un archivo a RStudio por medio de read_excel(), siendo las siguientes las más usuales:[5]

- Si el directorio de trabajo no se ha fijado, se debe escribir toda la dirección del archivo.

```
datos1 <- read_excel(path = 'C:/Usuario/
Desktop/Manual/datos1.xlsx')
```

- Si ya se fijó, únicamente se debe poner el nombre del archivo.

```
datos1 <- read_excel(path = 'datos1.xlsx')
```

- Si no se conoce la dirección del archivo, se emplea la función file.choose, que genera una ventana emergente donde se puede buscar.

```
datos1 <- read_excel(path = file.choose())
```

Observe que el nombre asignado para los datos es datos1. Una vez ejecutado el comando en el ambiente de trabajo, aparece el conjunto de datos importados. Para identificar el número de observaciones y de variables que tiene el archivo (dimensiones del archivo), se emplea el comando dim() que los devuelve.

```
dim(datos1)
```

```
## [1] 81 8
```

La salida anterior indica que el archivo cuenta con 81 observaciones y ocho variables. Es decir, automáticamente el *software* reconoce las filas como observaciones y las columnas como variables. Esta estructura en R se conoce como data.frame.

5 El usuario debe escribir la ubicación de la carpeta en su computador. Si la copia directamente va a generar un error.

```
#El comando class permite identificar la
#estructura de los datos
class(datos1)
```

```
## [1] "tbl_df"     "tbl"      "data.frame"
```

Para conocer el nombre de las variables se emplea el comando names().

```
#El comando names muestra el nombre de las
#columnas de un data.frame
names(datos1)
```

```
## [1]    "lote"     "eval"     "trat"     "rep"
"ancho_fol"
## [7] "peso_seco" "estadio"  "largo_fol"
```

Este conjunto de datos corresponde a un experimento que cuenta con tres lotes (lote), tres evaluaciones (eval), tres tratamientos (trat) y tres repeticiones (rep). Las características medidas fueron: ancho del folíolo (ancho_fol), peso seco de la hoja 17 (peso_seco), estadio del racimo cosechado (estadio) y largo del folíolo (largo_fol).

Ahora se debe verificar que las variables se estén leyendo de manera adecuada. En otras palabras, que se esté asignando el tipo correcto a cada variable, pues de esto depende toda la descripción y análisis estadístico. Los valores de las variables (lote, eval, trat y rep), representan etiquetas o categorías por medio de números, y R incorrectamente las asume como variables cuantitativas cuando en realidad son cualitativas. En este estudio se tomaron mediciones en el tiempo: a los 6, a los 12 y a los 18 meses. Estos datos se encuentran en la variable evaluación (eval) que tiene los tres valores. Se puede observar que el estadio se lee correctamente, pues aunque se representa con valores numéricos, estos están rodeados por comillas indicando que son un character. Lo anterior se puede verificar por medio del comando str(), que muestra toda la estructura de la tabla de datos.

```
str(datos1)
```

```
## tibble [81 x 8] (S3: tbl_df/tbl/data.frame)
##  $ lote       : num [1:81] 1 1 1 1 1 1 1 1 1 ...
##  $ eval       : num [1:81] 6 6 6 6 6 6 6 6 12 ...
##  $ trat       : num [1:81] 1 1 1 2 2 2 3 3 1 ...
##  $ rep        : num [1:81] 1 2 3 1 2 3 1 2 3 1 ...
##  $ ancho_fol: num [1:81] 5.12 4.94 5.24 8.1 7.52 8.95 5.81 5.3 6.86 5.08 ...
##  $ peso_seco: num [1:81] 2.48 3.61 3.96 3.39 2.34 4.19 4.99 4.31 4.42 3.98 ...
##  $ estadio    : chr [1:81] "805" "806" "806" "806" ...
##  $ largo_fol  : num [1:81] 80.2 78 81.7 109.8 102.7 ...
```

Para asignar el tipo correcto a cada variable, y empezar con un manejo básico de los datos, se emplea el paquete dplyr (Wickham *et al.*, 2020), que cuenta con una gran variedad de funciones altamente optimizadas para el manejo de data.frames.

El paquete dplyr es muy útil, ya que proporciona funciones fáciles de entender y manejar. Además son muy rápidas, puesto que están implementadas en el lenguaje C++ (Peng, 2016). Algunas de las principales son:

- **select:** devuelve un conjunto de variables.
- **filter:** devuelve un conjunto de filas según cierta condición lógica.
- **arrange:** reordena las filas de acuerdo con una variable.
- **rename:** renombra las variables.
- **mutate:** añade nuevas columnas o transforma las existentes.
- **group_by:** agrupa los datos según las características de una variable cualitativa.
- **summarise:** genera resúmenes estadísticos a variables numéricas, usualmente agrupadas por las categorías de una característica cualitativa.
- **%>%:** el operador *pipe* se emplea para conectar las funciones que se van a utilizar.

Recuerde que debe instalar el paquete y posteriormente cargarlo en la sesión de R.

```
install.packages('dplyr')
library('dplyr')
```

Es usual que al cargar el paquete en la consola se muestren una serie de *warnings*, debido a que tiene funciones con el mismo nombre que en otros paquetes.

Para asignar el tipo correcto a cada una de las variables se emplea la función mutate(), que permite modificar una columna. Se toman lote, eval, trat, rep y estadio, y se convierten en variables categóricas por medio de la función as.factor(). Al realizar este cambio se crea una nueva tabla llamada datos, con la que se va a trabajar a lo largo de este manual.

```
datos <- datos1 %>%
  mutate(lote = as.factor(lote),
         eval = as.factor(eval),
         trat = as.factor(trat),
         rep = as.factor(rep),
         estadio = as.factor(estadio))
```

Una vez que el comando es ejecutado debidamente, en el ambiente de trabajo se muestra la tabla datos, que cuenta con el mismo número de observaciones y de variables que datos1. Sin embargo, en esta nueva tabla todas las variables se leen correctamente.

```
str(datos)
```

```
## tibble [81 x 8] (S3: tbl_df/tbl/data.frame)
##  $ lote       : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ eval       : Factor w/ 3 levels "6","12","18": 1 1 1 1 1 1 1 1 1 2 ...
##  $ trat       : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 3 3 3 1 ...
##  $ rep        : Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 1 2 3 1 ...
##  $ ancho_fol: num [1:81] 5.12 4.94 5.24 8.1 7.52 8.95 5.81 5.3 6.86 ...
##  $ peso_seco: num [1:81] 2.48 3.61 3.96 3.39 2.34 4.19 4.99 4.31  ...
##  $ estadio    : Factor w/ 4 levels "805","806","807",..: 1 2 2 2 2  ...
##  $ largo_fol  : num [1:81] 80.2 78 81.7 109.8 102.7 ...
```

Una vez importados los datos a R y corroborado que se están leyendo correctamente, se procede a realizar su descripción y análisis básico.

Medidas numéricas descriptivas

En esta sección se presentan metodologías para organizar un conjunto de datos, por medio de resúmenes estadísticos, de los que se obtiene una información abreviada y una idea clara de su comportamiento. El objetivo principal es que el usuario adquiera habilidad en el manejo de los datos, identificando sus características más importantes y analizando correctamente los resultados. Se abordan tres medidas numéricas de interés para cualquier conjunto: las de tendencia central, las de posición y las de variabilidad o dispersión.

Medidas de tendencia central

Representan un valor alrededor del cual tienden a concentrarse las observaciones (Isaza, 2012). Las más utilizadas son la media aritmética o promedio, la mediana y la moda.

Media aritmética o promedio

Esta medida describe un conjunto de observaciones mediante un valor que representa el centro de los datos, y es la suma de todas las observaciones dividida entre el número total de datos.

Su cálculo en R se realiza mediante la función mean, que cuenta con dos argumentos básicos:

- **x:** variable cuantitativa sobre la que se va a realizar el cálculo.
- **na.rm:** indica si se deben excluir los datos faltantes (valores NA), en caso de que la variable de interés los tenga. Por defecto: na.rm = FALSE

Si quiere consultar la información de esta función, puede ir a la ayuda ejecutando el comando:

```
?mean
help(mean)
```

Para ilustrar el cálculo de la media aritmética o promedio, se emplea la variable numérica ancho_fol. Primero se debe seleccionar la variable de interés del conjunto de datos por medio del símbolo $. Se escribe el nombre de la tabla (data.frame), seguido por $ y el nombre de la variable. Por ejemplo, si se elige ancho_fol de la tabla datos, el código utilizado es datos$ancho_fol. Finalmente, se emplea la función mean como se muestra a continuación.

```
mean(datos$ancho_fol)
```

```
## [1] 5.851852
```

Se puede observar que en promedio, el ancho del folíolo en este conjunto de datos es de 5.85 cm. Es decir, aunque hay algunos con una medida menor y mayor a este, se espera que la mayoría de los folíolos del conjunto presente un ancho alrededor de esa cifra.

Advertencia

La media aritmética es la medida más apropiada de tendencia central para muchos conjuntos de datos, pero se deja influenciar por valores extremos. Así, si un conjunto presenta valores demasiado grandes o pequeños, el promedio puede que no represente el centro de los datos y genere resultados sesgados. Esto se debe a que la medida depende de igual manera de todas las observaciones, incluyendo los datos extremos.

Ahora bien, para entender mejor el comportamiento del ancho del folíolo, se puede calcular su promedio por tratamiento. Esto se realiza por medio de dos funciones del paquete dplyr:

- **group_by:** agrupa un conjunto de observaciones, de acuerdo con las categorías de una variable cualitativa.
- **summarise:** permite calcular algunas medidas numéricas descriptivas (media, mediana, varianza, entre otras) a una variable numérica, agrupada por categorías de una variable cualitativa.

Entonces, empleando la función group_by(), las observaciones se agrupan según las categorías de la variable tratamiento. Luego, utilizando el comando summarise(), se calcula el ancho promedio del folíolo en cada tratamiento. Las funciones empleadas se unen por medio del comando %>%, y el resultado es una nueva tabla llamada anchoPromedio, que tiene solo dos columnas: tratamiento y ancho promedio del folíolo (ancho_promedio). A continuación se muestra el código empleado.

Como se puede observar en la Tabla 1, los folíolos del tratamiento 2, en promedio son más anchos que los demás. También, que los más angostos se encuentran en el tratamiento 1.

```
anchoPromedio <- datos %>% #Se asigna el nombre a la nueva tabla de datos
  group_by(trat) %>% #Se agrupan las observaciones según el tratamiento
  summarise(ancho_promedio = mean(ancho_fol))
#Se crea una nueva columna llamada ancho_promedio y se calcula el
#ancho promedio del folíolo
```

Tabla 1. Ancho promedio del folíolo por tratamiento

Tratamiento (trat)	Ancho promedio del folíolo (ancho_promedio)
1	5.15
2	6.27
3	6.14

Mediana

Esta medida se encuentra en la posición central de un conjunto de datos ordenados. Las observaciones se organizan de manera creciente (de menor a mayor). Si el conjunto de datos contiene un número impar de valores, la mediana es simplemente el valor que se ubica en el medio de los datos ordenados. Si es una cifra par, resulta del promedio de los dos números que se hallan en el medio de los datos. Se dice que la mitad de las observaciones es menor o igual al valor de la mediana (Isaza, 2012).

Nota

En comparación con la media aritmética, la mediana no se deja afectar por datos con valores extremos, de modo que usualmente es una medida más informativa con respecto al centro de un conjunto de datos sesgados (con valores extremos).

En R, se emplea la función median indicando la variable de interés sobre la que se va a realizar el cálculo, acompañada del argumento

na.rm si la variable presenta valores faltantes. A continuación se calcula la mediana para la variable ancho del folíolo (ancho_fol).

```
median(datos$ancho_fol)
```

```
## [1] 5.52
```

Se puede ver que el 50 % de los folíolos tienen un ancho menor a los 5.52 cm, y el otro 50 % uno mayor a 5.52 cm. Como se determinó anteriormente, el ancho promedio del folíolo es de 5.85 cm, muy cercano al valor de la mediana, indicando que no se presentan valores extremos y que los datos se están representando de manera adecuada por estas dos medidas.

Para calcular la mediana del ancho del folíolo por tratamiento, se emplean las funciones group_by y summarise como ya se explicó. La tabla de datos resultante se llama anchoMediano y el código empleado se muestra a continuación.

```
anchoMediano <- datos %>%
  group_by(trat) %>%
  summarise(ancho_mediano = median(ancho_fol))
#Se crea una nueva columna llamada ancho_mediano y se calcula la
#mediana del ancho del folíolo
```

La Tabla 2 muestra la mediana del ancho del folíolo por tratamiento. Se observa que el 3 presenta un ancho mayor con 6.21 cm. Si se compara este resultado con el promedio de los folíolos (Tabla 1), se puede ver que aunque el tratamiento 1 abarca los más angostos, hay diferencia entre la media y la mediana de los otros dos tratamientos. Esto indica que los datos tienen un comportamiento característico dentro de cada tratamiento, y se hace necesario apoyar los análisis con medidas de variación para entender mejor la distribución de los datos.

Tabla 2. Mediana del ancho del folíolo por tratamiento

Tratamiento (trat)	Ancho mediano del folíolo (ancho_mediano)
1	5.12
2	6.03
3	6.21

Moda

La moda es el valor que más veces se repite en un conjunto de datos, y junto al promedio y la mediana, permite caracterizar el comportamiento de las observaciones. Se determina al contar el número de veces que cada valor ocurre dentro de un conjunto de datos.

Nota

A diferencia de las otras medidas de centralidad, la moda se puede calcular tanto para variables cuantitativas como para cualitativas.

Si se trabaja con una variable cuantitativa es posible que ningún valor se repita, por lo que no existe la moda. Por otro lado, si el número máximo de repeticiones se encuentra en dos o más valores, se dice que los datos tienen una distribución multimodal, es decir, con más de una moda.

En el paquete base de R no hay una función directa para calcular la moda. Sin embargo, el comando table permite contar el número de las veces que se repiten los valores de un conjunto de datos, y así identificar la observación más frecuente. A continuación, se presenta el código empleado para calcular las frecuencias para los valores de la variable ancho del folíolo.

```
table(datos$ancho_fol)
```

```
##
## 2.09 2.88 2.99 4.15 4.45 4.72 4.81 4.84 4.89 4.9 4.92 4.94 4.97 5.01
##    1    1    1    1    1    1    1    1    1    1    1    2    1    1
## 5.06 5.08  5.1 5.11 5.12 5.15 5.19 5.21 5.22 5.24 5.25 5.28  5.3 5.32
##    1    1    2    2    2    1    1    2    1    1    1    2    2    1
## 5.39 5.43  5.5 5.51 5.52 5.53 5.64 5.68 5.69  5.7 5.72 5.81 5.82 5.85
##    1    2    1    1    1    1    1    1    2    1    1    1    1    1
##  5.9 6.03 6.12 6.21 6.24 6.33 6.41 6.46 6.47 6.56 6.65 6.72 6.85 6.86
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## 6.98 7.03 7.08  7.1 7.12 7.13  7.2 7.49 7.52 7.71  8.1 8.33 8.85 8.95
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## 9.12 9.32
##    1    1
```

Como se puede ver, hay más de dos observaciones con una frecuencia de 2, por lo que es multimodal. Para que sea más fácil identificar los datos que se repiten, se puede organizar el resultado con la función sort().

```
#De mayor a menor
sort(table(datos$ancho_fol), decreasing =
TRUE)
```

Si se quiere calcular de manera directa la moda de una variable, el paquete modest cuenta con la función mfv(), que identifica de forma automática el o los valores más frecuentes de un conjunto de observaciones. Para emplearla se debe primero instalar y cargar el paquete mencionado.

```
#install.packages('modeest')
library(modeest)
mfv(datos$ancho_fol)
```

```
## [1] 4.94 5.10 5.11 5.12 5.21 5.28 5.30 5.43
5.69
```

La salida de esta función presenta los valores que más se repiten, pero no su frecuencia. En este caso, la variable ancho del folíolo tiene nueve observaciones que se repiten dos veces.

Para ilustrar el cálculo de la moda en una variable cualitativa, se emplea la variable estadio, que mide el estado de maduración de un racimo al ser cosechado. A continuación se muestra el código empleado para este cálculo.

```
table(datos$estadio)
```

```
## 805 806 807 809
##  4  32  33  12
```

Se evidencia que la moda de esta variable es la categoría 807, con una frecuencia de 33 racimos. Adicionalmente, el estadio 806 tiene una alta frecuencia (32 racimos) muy cercana a la moda. Solamente cuatro racimos se cosecharon en un estado inmaduro (estadio 805) y 12 en sobremaduro (estadio 809).

Medidas de posición

Son valores que indican la ubicación de los datos en relación con el resto de las observaciones ordenadas, y resultan muy útiles para identificar el comportamiento de un conjunto de datos. Las medidas de posición se llaman generalmente cuantiles, y se pueden clasificar en tres grupos: cuartiles, deciles y percentiles.

- Los cuartiles denominados Q_1, Q_2 y Q_3, dividen el conjunto de datos en cuatro grupos iguales, en donde cada uno contiene un 25 % de los datos.
- Los deciles son nueve valores que fraccionan el conjunto de datos en 10 partes iguales, y se denominan D_1, D_2, ..., D_9
- Los percentiles expresados como P_1, P_2, ..., P_{99}, dividen los valores de la variable en 100 grupos.

Nota

El percentil 25 coincide con el primer cuartil, el 50 es la mediana o segundo cuartil, y el 75 es el tercer cuartil.

La función quantile() permite calcular los cuantiles de un conjunto, y su argumento principal es la variable numérica sobre la que se va a realizar el cálculo. El siguiente es el código para estimar estas medidas de posición a la variable largo_fol.

```
quantile(datos$largo_fol)
```

```
##    0%   25%   50%   75%  100%
## 50.49 79.86 85.29 98.99 143.37
```

Observe que la función calcula el valor mínimo 50.49 cm; el 79.86 cm corresponde al primer cuartil o percentil 25; la mediana calculada es 85.29 cm; el percentil 75 o tercer cuartil es de 98.99 cm, y el valor máximo 143.37 cm. La interpretación de estos datos es la siguiente:

- El 25 % de los folíolos presenta un largo menor o igual a 79.86 cm (percentil 25).
- El 75 % un largo menor o igual a 98.99 cm (percentil 75).
- La mitad, un largo mayor a 85.29 cm (mediana).

Medidas de dispersión

Permiten cuantificar la variabilidad de una característica de interés. La variación puede definirse, como la diferencia que existe entre las unidades de estudio respecto a la variable analizada (Isaza, 2012). Entre las medidas de dispersión se encuentran: el rango, el rango intercuartil, la varianza, la desviación estándar y el coeficiente de variación.

Rango

Es la diferencia entre el valor más grande y el mínimo de una variable numérica, y permite cuantificar la máxima dispersión de los datos. Aunque el cálculo es sencillo, su debilidad se debe a que no tiene en cuenta el comportamiento de los datos entre los valores mínimo y máximo. Por esta razón, si las observaciones presentan niveles extremadamente grandes o pequeños, el rango se deja influenciar por estos y se debe tener cuidado con su interpretación.

Para calcular el rango en R, primero se debe identificar el valor máximo y mínimo de la variable de interés, con las funciones max() y min(), y posteriormente restarlos. El argumento de estas funciones es la variable numérica sobre la que se va a realizar el cálculo, y si se presentan valores faltantes, na.rm permite no tenerlos en cuenta. Para ilustrar este cálculo se emplea la variable peso seco de la hoja 17 (peso_seco).

```
max(datos$peso_seco) - min(datos$peso_seco)
```

```
## [1] 4.04
```

Se observa que el rango del peso seco es de 4.04 kg, lo que indica que el cambio más grande entre los valores en este conjunto de datos es cercano a los 4 kg, pues aunque se tienen hojas con un peso de 1.89 kg (peso seco mínimo de la hoja) también hay unas con 5.93 kg (peso seco máximo).

Rango intercuartil

También llamado dispersión media, es la diferencia entre el tercer y el primer cuartil de un conjunto de datos (ver página 38 Medidas de posición). Esta medida tiene en cuenta el 50 % de los valores centrales de un conjunto de datos, lo que indica que los valores extremos no influyen en ella.

A diferencia del rango, existe la función IQR() que estima directamente el rango intercuartil, y su argumento principal es la variable numérica sobre la que se va a realizar la medición. A continuación, se muestra el código para calcular esta medida de variación a la variable peso_seco.

```
IQR(datos$peso_seco)
```

```
## [1] 1.04
```

El rango intercuartil para el peso seco es de 1.04 kg.

Varianza y desviación estándar

Como se vio anteriormente, el rango mide la dispersión total y el rango intercuartil la dispersión media. Sin embargo, ninguna considera el comportamiento de todos los datos. La varianza, por otro lado, "es una medida de variación que teniendo en cuenta la distribución de todo el conjunto de datos, permite cuantificar el grado de dispersión de las observaciones alrededor de su media" (Isaza, 2012).

La varianza se define como el promedio de las diferencias al cuadrado entre cada dato y su media, de aquí que puede ser confuso su análisis. Por esta razón es más frecuente presentar su raíz cuadrada, conocida como la desviación estándar, con la que se realiza un análisis más intuitivo al utilizar las mismas unidades que las de la variable bajo estudio. La desviación estándar es la medida de dispersión más común, y cuantifica qué tan separados se encuentran los datos con respecto a su promedio: a mayor desviación estándar mayor dispersión.

El cálculo de la varianza y la desviación estándar en R, se realiza mediante las funciones var y sd, respectivamente. El argumento básico es la variable cuantitativa de interés. Para ilustrar el cálculo de estas medidas de dispersión,

se emplea la variable peso_seco como se muestra a continuación.

```
#Para calcular la varianza
var(datos$peso_seco)
#Para calcular la desviación estándar
sd(datos$peso_seco)
```

```
## [1] 0.7144512
## [1] 0.8452522
```

La varianza del peso seco de la hoja corresponde a 0.714 kg^2, pero no se tiene una fácil interpretación. Sin embargo, la desviación estándar para esta variable es de 0.85 kg, es decir esta es aproximadamente la distancia de los datos con respecto a su valor promedio 4.06 kg.

```
#Para calcular la media
mean(datos$peso_seco)
```

```
## [1] 4.060123
```

Coeficiente de variación

Es una medida de variabilidad relativa, que se calcula al dividir la desviación estándar sobre la media o promedio, y usualmente se presenta como un porcentaje (multiplicado por 100). Es muy útil para comparar la variabilidad de conjuntos de datos con medias diferentes.

```
sd(datos$peso_seco)/mean(datos$peso_seco)
```

```
## [1] 0.2081839
```

Se observa que el coeficiente de variación del peso seco es del 20 %, indicando que este conjunto no presenta una dispersión alta, y que el promedio representa correctamente los datos. Ahora bien, para entender mejor el comportamiento del peso seco, se puede calcular su promedio, la mediana, la desviación estándar y el coeficiente de variación por tratamiento. Esto se realiza por medio de las funciones group_by y summarise, como se mostró anteriormente.

```
pesoResumen <- datos %>% #Tabla de interés
  group_by(trat) %>%  #Se agrupan según el tratamiento
  summarise(peso_promedio = mean(peso_seco), #Una columna para la media
    peso_mediano = median(peso_seco), #Una columna para la mediana
    desviacion = sd(peso_seco), #Una columna para la desviación
    coeficiente_v = (desviacion/ peso_promedio)*100
    #Una columna para el CV
    )
```

Como se puede apreciar en la Tabla 3, el peso seco promedio de las hojas en el tratamiento 3 es superior al de los otros. Además, produce observaciones menos variables, con una desviación estándar de 0.63 kg y un coeficiente de variación del 14.06 %.

Tabla 3. Resumen del peso seco de la hoja 17 por tratamiento

Tratamiento (trat)	**Peso medio** (peso_promedio)	**Peso mediano** (peso_mediano)	**Desviación estándar** (desviacion)	**C. de variación %** (coeficiente_v)
1	4.04	4.05	0.99	24.60
2	3.66	3.57	0.68	18.48
3	4.49	4.43	0.63	14.06

Exploración gráfica

En esta sección se muestran gráficos exploratorios para resumir datos y obtener una descripción clara y precisa para su interpretación, teniendo en cuenta la conceptualización estadística necesaria. Es importante mencionar que si estos no se presentan de manera correcta, pueden generar conclusiones erróneas. Así, se recomienda no omitir información pertinente, especificar siempre el nombre de los ejes y considerar la escala empleada.

Para crear los gráficos, se emplea ggplot2 (Wickham, 2016), que es un paquete de R que permite hacerlos de una manera potente y flexible, ofreciendo al usuario ilimitadas posibilidades de edición novedosas, que se adaptan a un problema específico. Cuenta con funciones especializadas, y debido a esto, el nivel de trabajo de ciertos gráficos puede ser elevado. Sin embargo, el objetivo de esta sección es presentar los elementos básicos para la elaboración de los gráficos descriptivos usuales. Recuerde que debe instalar y cargar el paquete en la sesión de trabajo.

```
install.packages('ggplot2')
library('ggplot2')
```

Gráficos de dispersión

Es una representación de dos **variables numéricas** por medio de una nube de puntos, en donde una se ubica en el eje x y la otra en el eje y, y se dibujan los puntos (x,y) para cada observación. En este gráfico se puede observar el comportamiento de dos variables al mismo tiempo, e identificar los puntos alejados de la nube para revisarlos y sacar conclusiones acerca de ellos. También, se emplea para establecer posibles asociaciones lineales y no lineales entre las variables de interés.

Para ilustrar cómo crear un diagrama de dispersión se seleccionan las variables largo_fol y peso_seco. Al emplear el paquete ggplot2, primero se debe generar un gráfico básico en donde se define la tabla de datos a usar, y las variables que van en cada uno de los ejes. Esto se realiza por medio de la función ggplot(). Posteriormente, para generar el diagrama de dispersión, se emplea la función geom_point() y se une al gráfico básico por medio del símbolo +, como se muestra a continuación.

```
ggplot(datos, aes(x = largo_fol, y = peso_seco)) +
#gráfico básico
geom_point() #Nube de puntos
```

Para cambiar el nombre de los ejes y agregar títulos y subtítulos, se utiliza la función labs().

```
ggplot(datos, aes(x = largo_fol, y = peso_seco)) +
geom_point() +
labs(title = 'Gráfico de dispersión',
  #Permite agregar un título
  x = 'Largo del folíolo (cm)',
  #Cambia el nombre del eje x
  y = 'Peso seco de la hoja 17 (kg)')
  #Cambia el nombre del eje y
```

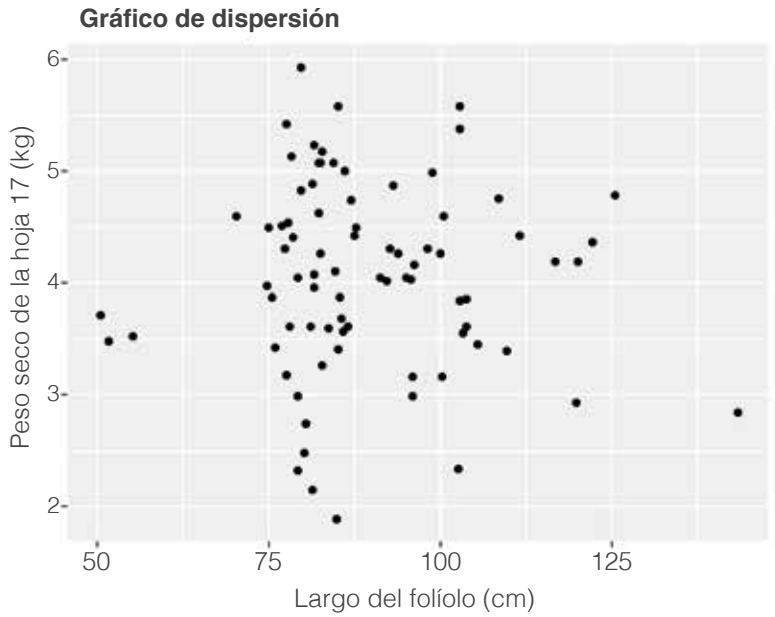

Figura 20. Gráfico de dispersión con ggplot

En la Figura 20 se ilustra el comportamiento de la variable largo del folíolo y el peso seco de la hoja 17. Se observan tres puntos alejados de los demás datos, que deben revisarse para identificar si presentan algún error de digitación y corregirse. Si son comportamientos inusuales, es preciso establecer su posible causa y describirla. En este caso, los datos no presentan ningún error de digitación y su comportamiento se debe al tratamiento al que pertenecen. Adicional a esto, no se observa ninguna asociación entre las dos variables, ya que la nube de puntos no muestra ningún comportamiento específico.

Si se quiere asignar un color a la nube de puntos se emplea el comando col, que permite cambiar el tono de los puntos según diferentes características. R cuenta con una amplia gama que se puede ver al ejecutar el comando colors() en la consola. Sin embargo, ggplot2 tiene una paleta de tonalidades preestablecida. Por ejemplo, si se desea que los colores de los puntos hagan referencia a los tratamientos, se utiliza el comando col = trat como se muestra a continuación.

```
ggplot(datos, aes(x = largo_fol, y = peso_seco,
col = trat)) +
 geom_point() +
 labs(title = 'Gráfico de dispersión',
    x = 'Largo del folíolo (cm)',
    y = 'Peso seco de la hoja 17 (kg)')
```

Como se mencionó, ggplot2() permite editar los gráficos de una manera sencilla. Para modificar los colores de los tratamientos se puede emplear el comando scale_color_manual(), y para el título de las categorías la función labs().

```
ggplot(datos,
   aes(x = largo_fol, y = peso_seco, col = trat)) +
 geom_point() +
 labs(title = 'Gráfico de dispersión',
    x = 'Largo del folíolo (cm)',
    y = 'Peso seco de la hoja 17 (kg)',
   col = 'Tratamiento') +
 scale_color_manual(
  values = c('#19446E','#F1592A','#638239'))
```

En la Figura 21 se puede observar el gráfico de dispersión, en donde el color del punto indica a qué tratamiento pertenece, permitiendo identificar el comportamiento de las variables según cada uno. Por ejemplo, el largo del folíolo en el tratamiento 1 (color azul), presenta una baja variabilidad con respecto al resto, pues todos sus valores se encuentran entre los 105 cm y 125 cm; mientras tanto, el tratamiento 3 (color naranja), presenta valores del largo del folíolo desde 70 cm hasta 180 cm, siendo este el más variable.

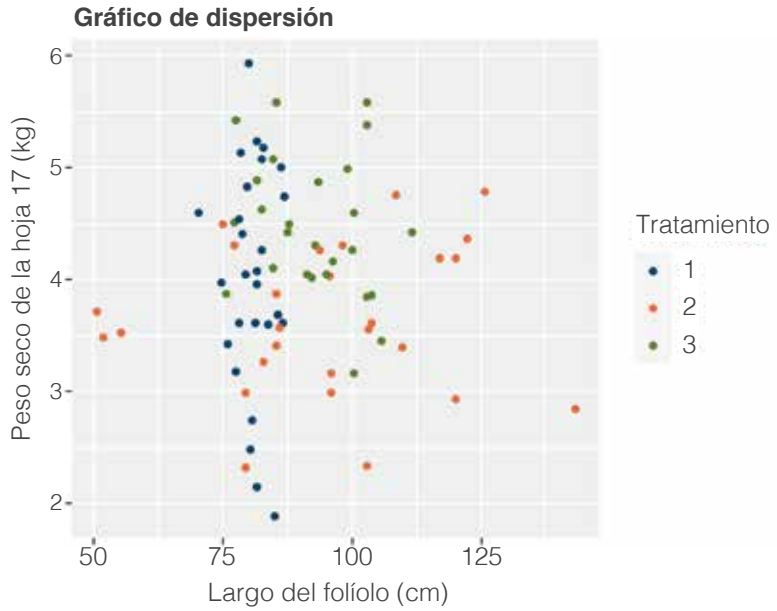

Figura 21. Gráfico de dispersión teniendo en cuenta el tratamiento

Gráfico de líneas

El gráfico de líneas se emplea para identificar el comportamiento de una variable usualmente medida en el tiempo, y comparar visualmente patrones. Para crear este tipo de gráficos, la variable tiempo se ubica en el eje x y la característica numérica de interés en el eje y. Dentro del esquema se dibujan puntos (x,y) por cada observación, y se unen por medio de líneas.

Para ilustrar la elaboración del gráfico, se usa evaluación (eval) como la variable de tiempo (que se realizaron cada seis meses) y el largo del folíolo (largo_fol), con el fin de identificar su comportamiento en el tiempo, según el tratamiento.

Primero se calcula el largo del folíolo promedio según la evaluación. Por medio de la función group_by(), las observaciones se agrupan de acuerdo con las categorías de la variable evaluación. Luego, utilizando el comando summarise(), se estima el largo del folíolo promedio por evaluación. Las funciones empleadas se unen por medio del comando %>%, y el resultado es una nueva tabla llamada largo_eval, que tiene solo dos columnas (Tabla 4). A continuación se muestra el código usado.

```
largo_eval <- datos %>%
#Se asigna el nombre a la nueva tabla de datos
 group_by(eval) %>%
#Se agrupan las observaciones según la evaluación
summarise(largo_promedio = mean(largo_fol))
#Se crea una nueva columna llamada largo_
#promedio y se calcula el
#largo del folíolo promedio
```

Tabla 4. Largo del folíolo promedio por evaluación

Evaluación (eval)	Largo del folíolo promedio (largo_promedio)
6	91.07
12	84.45
18	92.77

Para elaborar el gráfico de líneas, se crea el esquema básico con la función ggplot(), cuyo primer argumento es la tabla largo_eval. Después, en los ejes x y y se ubican las variables eval y largo_promedio, respectivamente. Como no se tienen otras adicionales, se asigna 1 al comando group(). Por medio del geom_line(), se origina una línea que presenta el comportamiento del largo promedio del folíolo en el tiempo, y con la función geom_point() se presentan los puntos en el gráfico.

```
ggplot(largo_eval, #El primer argumento es la tabla de interés
  aes(x = eval, y = largo_promedio, group=1)) + #se definen los ejes
    geom_line() + #Se agrega la línea
    geom_point() + #Se agregan los puntos
    labs(x = 'Evaluación (meses)', #Se cambia en nombre al eje x
    y = 'Largo promedio del folíolo (cm)') #Se cambia el nombre al eje y
```

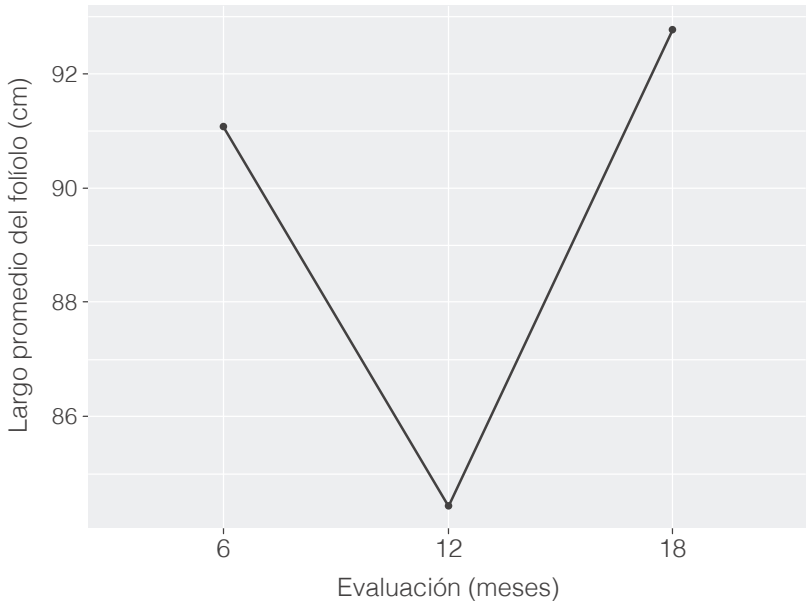

Figura 22. Gráfico de líneas con ggplot

En la Figura 22 se presenta el comportamiento para la variable largo del folíolo a través del tiempo. Se observa un descenso en la segunda evaluación, pasando de 91.07 cm a 84.45 cm, pero en la tercera volvió a aumentar a 92.76 cm.

Para entender mejor estos resultados, se emplea la función group_by(), esta vez agrupando las variables categóricas eval y trat. La nueva tabla se llama largo_eval_trat, y el código empleado es el siguiente:

```
largo_eval_trat <- datos %>% #Se asigna el nombre a la nueva tabla de datos
  group_by(eval,trat) %>%
  #Se agrupan las observaciones según la evaluación y el tratamiento
   summarise(largo_promedio = mean(largo_fol))
  #Se crea una nueva columna llamada largo_promedio y se calcula el
  #largo promedio del folíolo
```

Tabla 5. Largo promedio del folíolo por evaluación y tratamiento

Evaluación (eval)	**Tratamiento** (trat)	**Largo del folíolo** (largo_promedio)
6	1	81.1
	2	98.76
	3	93.35
12	1	78.9
	2	85.06
	3	89.4
18	1	82.42
	2	101.15
	3	94.74

En la Tabla 5 se presenta el largo promedio del folíolo por evaluación y por tratamiento. Sin embargo, no es fácil identificar el comportamiento de la variable a través del tiempo. Al realizar el gráfico de líneas, es posible entender y comparar visualmente la distribución de los datos en las evaluaciones, teniendo en cuenta el tratamiento.

Para crear el gráfico se emplea un código muy similar al anterior. La tabla de interés es largo_eval_trat, y en los ejes x y y se ubican las variables eval y largo_promedio. Para trazar una línea por tratamiento se usa el argumento group = trat, y para colorearlas se utiliza el argumento col y scale_color_manual() como se mostró anteriormente.

```
ggplot(largo_eval_trat, #Primer argumento es la tabla largo_eval_trat
  aes(x = eval, y = largo_promedio,group = trat, col =trat)) +
  #Se crea una línea por tratamiento empleando el comando group = trat
  geom_line(size = 1)+
  #El argumento size permite asignar el tamaño a la línea
  geom_point()+
  #Se cambian manualmente los colores
  scale_color_manual(values = c('#19446E', '#F1592A', '#638239'))+
  labs(x = 'Evaluación (meses)',
       y = 'Largo promedio del folíolo (cm)',
       col = 'Tratamiento')
```

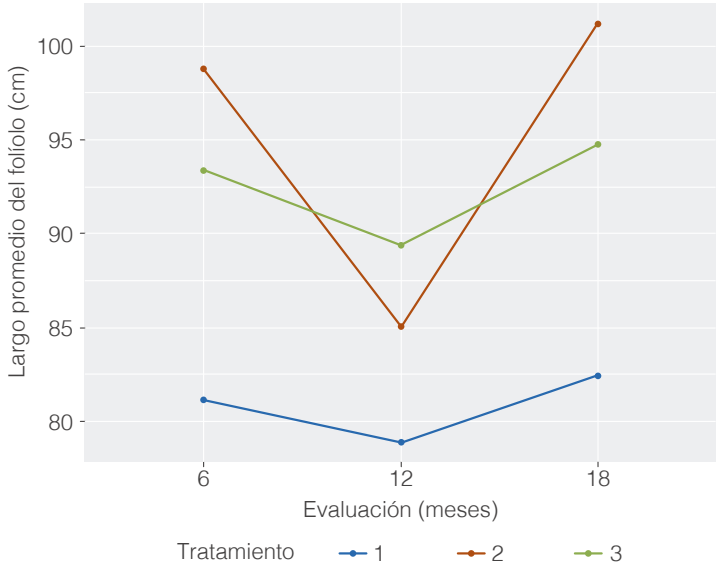

Figura 23. Gráfico de líneas teniendo en cuenta el tratamiento

En la Figura 23 se observa que el tratamiento que produce menores largos de folíolo es el 1 (color azul). También, que al año de empezar este experimento (en la segunda evaluación) hay un descenso promedio en todos los tratamientos, resaltando el 2 (color naranja).

Gráfico de cajas y bigotes

El gráfico de cajas y bigotes también conocido como boxplot, se emplea para ver el comportamiento de una variable cuantitativa, y es muy útil para comparar las distribuciones entre varios grupos de datos con la misma característica evaluada. Su principal atributo es que realiza un resumen de los datos mediante cinco números: la mediana o percentil 50 (P_{50}), el percentil 25 (P_{25}), el percentil 75 (P_{70}) y los límites superior e inferior de las líneas del gráfico, llamados bigotes.

Para entenderlo, en la Figura 24 se muestran dos gráficos de cajas y bigotes con sus partes. El ancho de la caja representa la dispersión que tienen los datos, y sus límites corresponden a los percentiles 25 y 75, cuyo cálculo e

interpretación se explicó en la página 38 (Medidas de posición). Si la caja es ancha, los datos presentan una alta dispersión; si es angosta es baja. Dentro de ella se encuentra la mediana (P_{50}), lo que hace sencillo analizar la centralidad de los datos. Además, si estos se están comparando en diferentes categorías, dicha mediana permite estudiarlos rápidamente en términos de su tendencia central. Los bigotes dejan ver el comportamiento de los datos en los extremos; las observaciones que se encuentren fuera son consideradas lejanas o atípicas.

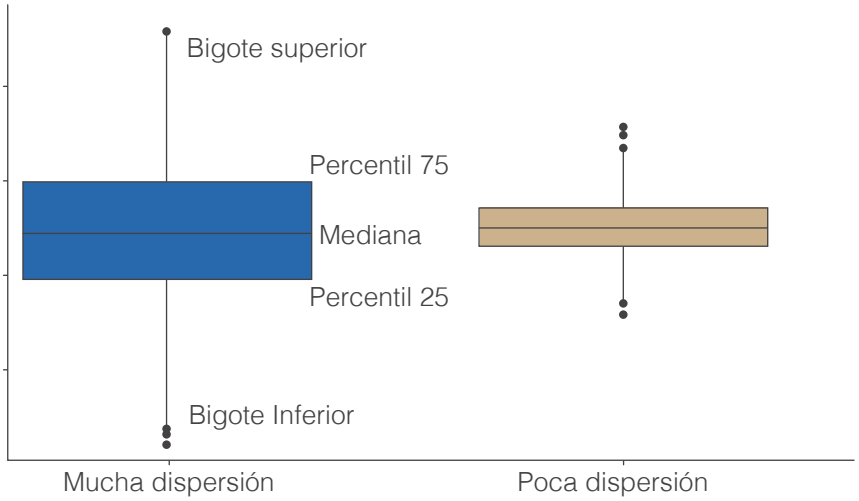

Figura 24. Gráfico de cajas y bigotes con sus partes

El paquete ggplot() permite elaborar el gráfico de cajas y bigotes por medio de la función geom_boxplot(). Se debe crear primero el básico con la función ggplot(), en donde se definen la tabla de datos y sus ejes. Si se quiere ver el comportamiento del largo del folíolo según el tratamiento, en el eje x se ubica la variable trat y en el eje y el largo_fol. El color del boxplot() se determina por medio del comando fill; en este caso el tono representa el tratamiento. Una vez definidas estas características, se procesa el gráfico con la función geom_boxplot(). El siguiente es el código empleado:

```
ggplot(datos, aes(x = trat, y = largo_fol, fill = trat)) +
 geom_boxplot()+
 scale_fill_manual(values = c('#19446E', '#F1592A', '#638239'))+
 labs(x = 'Tratamiento',
      y = 'Largo del folíolo (cm)',
      fill = 'Tratamiento')
```

Lo más relevante de la Figura 25 es que el segundo tratamiento presenta una mayor dispersión en el largo del folíolo comparándola con los otros dos, pues su caja es mucho más ancha. Se puede ver que la mediana en el primer tratamiento es más baja, aunque es el que arroja

resultados menos variables. Adicional a esto, el primer tratamiento muestra un valor fuera de los bigotes, lo que indica que hay un folíolo con un largo extrañamente menor. Las medianas del segundo y tercero son muy similares, pero la dispersión de este último es mucho más baja y sus bigotes son cortos, expresando que no se presentan datos muy lejanos de la mediana.

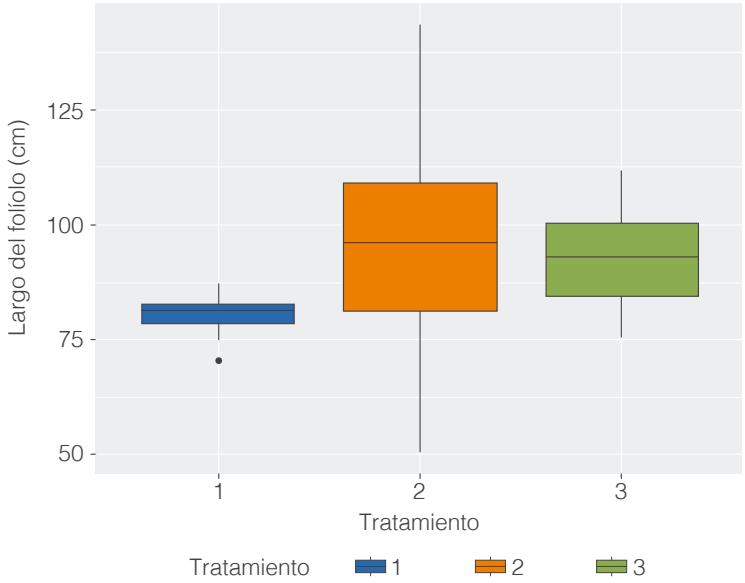

Figura 25. Gráfico de cajas y bigotes con ggplot

Histograma

El histograma o histograma de frecuencias, es una gráfica muy útil para describir el comportamiento de una variable cuantitativa o numérica. Consiste en barras adyacentes de igual ancho, que representan la frecuencia de los valores de la variable, permitiendo identificar los valores centrales, la amplitud de los datos, la dispersión y la distribución del conjunto de observaciones.

También, ayuda a determinar fácilmente dónde se encuentra el centro de los datos, pues la barra más alta contiene el valor en el que se concentran la mayoría de las observaciones, y da una rápida imagen de la dispersión de estos, ya que el ancho del histograma representa la variación.

Para ilustrar la creación de este gráfico, en la Figura 26 se emplea la variable numérica peso seco de la hoja (peso_seco). El paquete ggplot2 cuenta con la función geom_histogram(), que permite graficar el histograma. Uno de los argumentos más importantes de esta es binds, que define el número de barras. Por defecto es igual a 30, pero si se tienen pocos datos, el histograma no se muestra de manera adecuada. La cantidad de barras depende de las observaciones de la variable, y existen diversas maneras de calcularla. La más común es la raíz cuadrada del número de observaciones.

```
ggplot(datos, aes(x = peso_seco))+
  geom_histogram(bins = 9, fill = '#0578A0', col = 'black')+
  labs(y = 'Frecuencia', x = 'Peso seco de la hoja 17 (kg)')
```

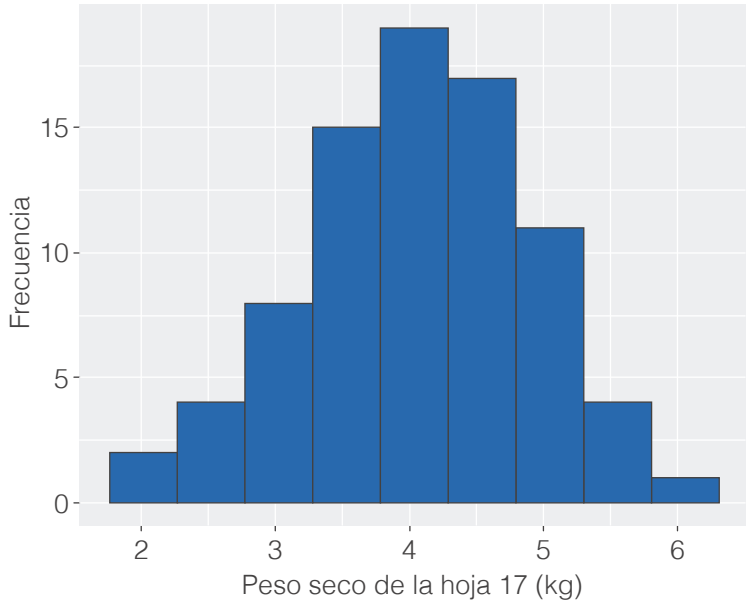

Figura 26. Histograma con ggplot

Una manera rápida de crear un histograma por tratamiento, es empleando la función facet_wrap(). También es posible cambiar la posición de la leyenda por medio del comando legend.position = 'bottom', como se muestra a continuación.

```
ggplot(datos, aes(x = peso_seco, fill = trat))+
  geom_histogram(bins = 10, col = 'black')+
  facet_wrap(~trat)+
  scale_fill_manual(values = c('#19446E', '#F1592A', '#638239'))+
  labs(y = 'Frecuencia', x = 'Peso seco de la hoja 17 (kg)') +
  theme( legend.position = 'bottom')
```

En la Figura 27 se exponen los histogramas de la variable peso seco de la hoja 17, según el tratamiento. Se puede observar que el primero presenta una amplitud mayor que los otros dos. Además, los valores centrales de los tres tratamientos se encuentran cercanos a los 4 kg (barras más altas).

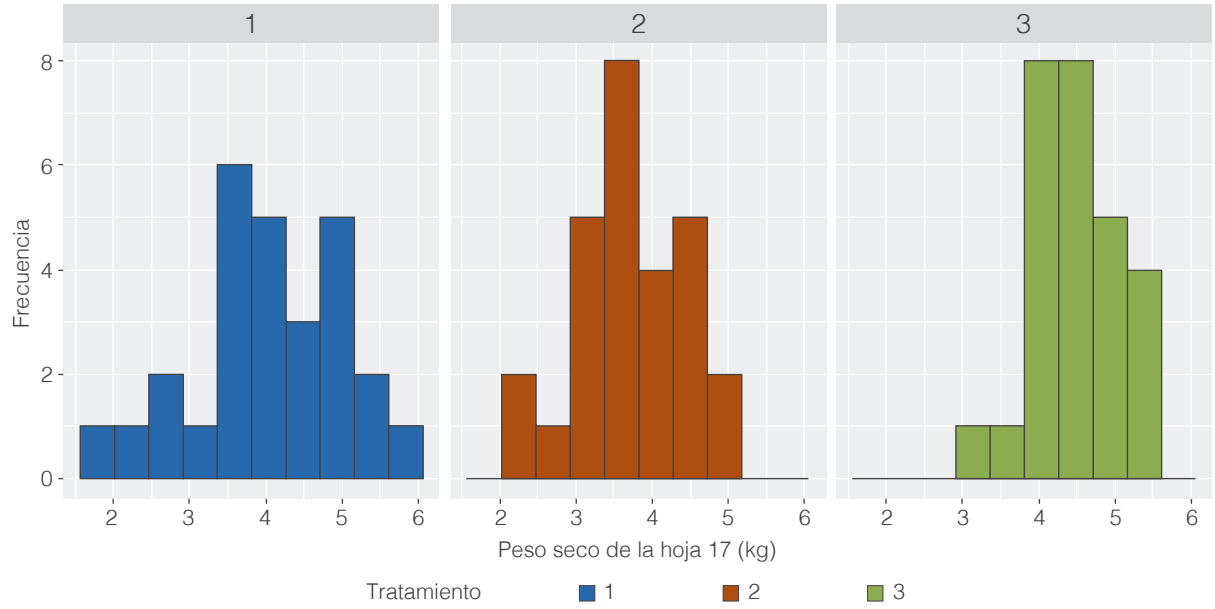

Figura 27. Histograma de frecuencias por tratamiento

Advertencia

Es muy común confundir el histograma con el gráfico de barras. Para tener claridad, el histograma calcula la frecuencia de los valores de una variable numérica y permite ver la distribución de los datos. El gráfico de barras se emplea para comparar las categorías de una variable cualitativa, como se explica a continuación.

Gráfico de barras

En este gráfico cada barra representa una categoría, y su altura indica la frecuencia. Es muy útil para resumir y entender el comportamiento de una variable categórica. Para ilustrarlo se utiliza la variable categórica estadio. Primero se emplea la función básica ggplot(), donde se define la tabla de datos y la variable categórica de interés. Luego, por medio de geom_bar() se crea el gráfico, se asigna un color a las barras por medio de fill y uno a los contornos con el comando col.

```
ggplot(datos, aes(x=estadio)) +
geom_bar(fill = '#C3AF8D', col = '#19446E') +
labs(x = 'Estadio',
     y = 'Racimos')
```

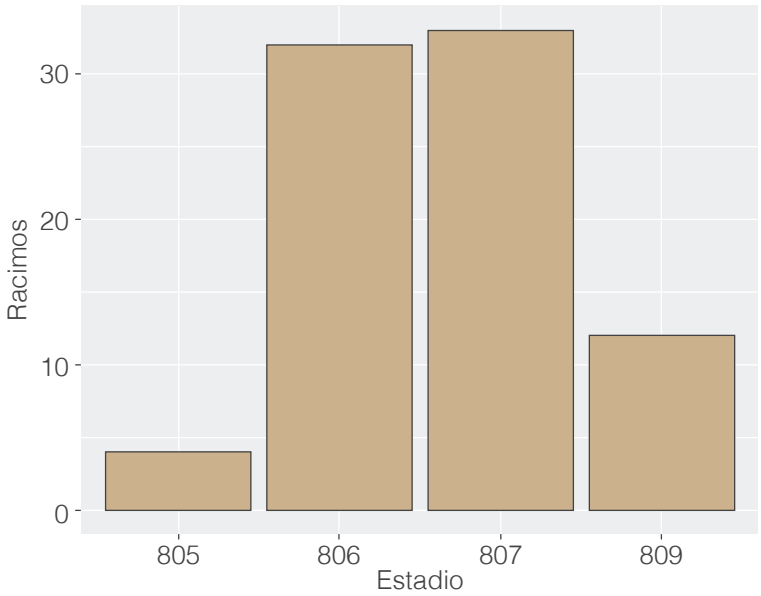

Figura 28. Gráfico de barras con ggplot

En la Figura 28 se muestra la frecuencia de cada estadio, en donde los 806 y 807 presentan un mayor número de racimos, que automáticamente calcula la función geom_bar(). Si se quiere determinar el porcentaje de la frecuencia de cada categoría, se debe crear una tabla que contenga este resumen. Se emplea la función group_by() para agrupar las observaciones según el estadio, y summarise() para establecer las medidas de interés. El comando n() permite realizar el conteo de observaciones dentro de cada estadio, y a partir de este se calcula la proporción, dividiendo la frecuencia por el número total de datos. Para calcular el porcentaje, la proporción se multiplica por 100 y por medio de round() se redondea a dos decimales. A la tabla resultante se le asigna el nombre de estadio_resumen (Tabla 6). El código empleado se muestra a continuación.

```
#Número de observaciones
n = 81

estadio_resumen <- datos %>%
group_by(estadio) %>% #Se agrupa por estadio
summarise(Racimos = n(), #Se calcula la frecuencia
          Proporcion = Racimos/n, #Se divide por el total de datos
          Porcentaje = round(Proporcion*100,2))
#Se multiplica por 100 la proporción y se redondea
#para tener solo dos decimales.
```

Tabla 6. Número de racimos por estadio

Estadio (estadio)	**Núm. racimos** (Racimos)	**Proporción** (Proporcion)	**Porcentaje** (Porcentaje)
805	4	0.0493	4.94
806	32	0.3950	39.51
807	33	0.4074	40.74
809	12	0.1481	14.81

Ahora, la tabla estadio_resumen contiene la información necesaria para realizar el gráfico de barras. En el eje x se ubica la variable estadio, y en el eje y la proporción de racimos por estadio. Para que este cuente con una escala en porcentajes, se emplea la función scale_y_continuous().

```
ggplot(estadio_resumen, aes(x=estadio, y= Proporcion)) +
  geom_bar(stat = 'identity', fill = '#C3AF8D', col = '#19446E') +
  scale_y_continuous(labels=scales::percent)
```

Para crear una gráfica más completa, se puede mostrar el porcentaje correspondiente a cada barra, por medio de la función geom_text(), que tiene como argumento principal el valor a exponer (en este caso el porcentaje). Como se mencionó anteriormente, los gráficos presentan una gran variedad de argumentos para su edición, como el tamaño de la letra (size) o la posición del texto (vjust), entre otros.

```
ggplot(estadio_resumen, aes(x=estadio, y= Proporcion)) +
 geom_bar(stat = 'identity', fill = '#C3AF8D', col = '#19446E') +
 scale_y_continuous(labels=scales::percent) +
 geom_text(aes(label =Porcentaje), size = 3, vjust = -0.4)+
 labs(x = 'Estadio',
      y = 'Porcentaje')
```

En la Figura 29 aparece el porcentaje de racimos por estadio. Cada barra cuenta con su respectiva cifra, permitiendo un análisis más sencillo. Se observa que el 80.25 % de los racimos se encuentra en los estadios 806 y 807, indicando que la mayoría de los racimos cosechados están maduros.

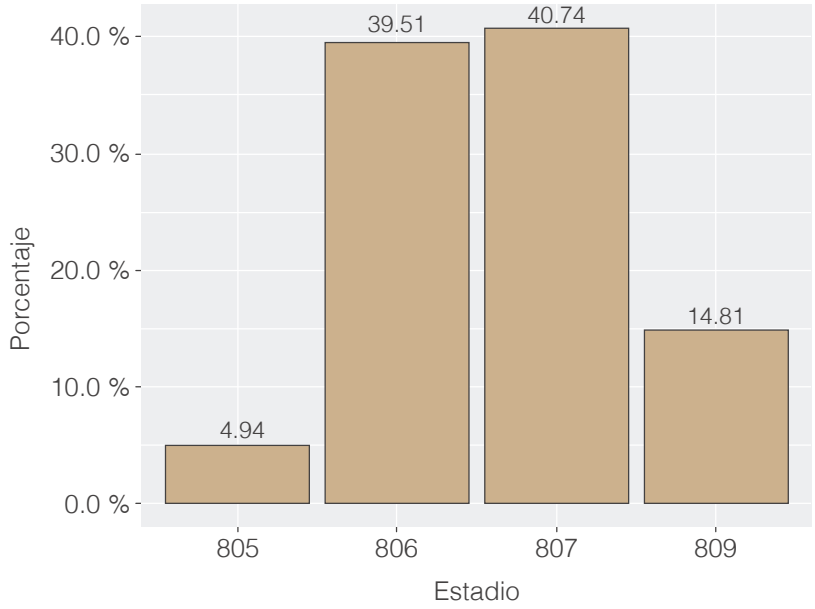

Figura 29. Gráfico de barras teniendo en cuenta porcentajes

Gráfico circular

Al igual que el gráfico de barras, el circular permite presentar visualmente el comportamiento de una variable cualitativa. Sin embargo, se recomienda utilizarlo solo cuando se tienen pocas categorías.

Es un círculo dividido en diferentes áreas, y cada una de ellas representa la frecuencia de cada etiqueta.

Para crearlo en R se emplean las funciones geom_bar() y coord_polar(). En el eje y se ubica el porcentaje de racimos por estadio, y por medio del argumento fill se asigna el color del área de acuerdo con el mismo. A diferencia de otros gráficos, el eje x se deja vacío. Primero es necesario usar geom_bar() para calcular el área correspondiente a cada categoría, y posterior a esto coord_polar() para dibujar el círculo característico del gráfico circular.

```
ggplot(estadio_resumen,aes(x='',y=Porcentaje, fill=estadio))+
  geom_bar(stat = 'identity', color='black')+
  coord_polar(theta = 'y',direction = -1)+
  geom_text(aes(label=Porcentaje),
    position=position_stack(vjust=0.5),color='white',size=4)+
  theme_void()+
  scale_fill_manual(values = c('#638239', '#0578A0', '#F1592A', '#C3AF8D'))
```

Como se observa en la Figura 30, el porcentaje en los estadios 805 y 806 es de 44.45 %, indicando que casi la mitad de los racimos evaluados están en estados inmaduros. Se puede apreciar que el área del estadio 805 es la más pequeña, seguida por la 809, indicando una baja frecuencia de racimos.

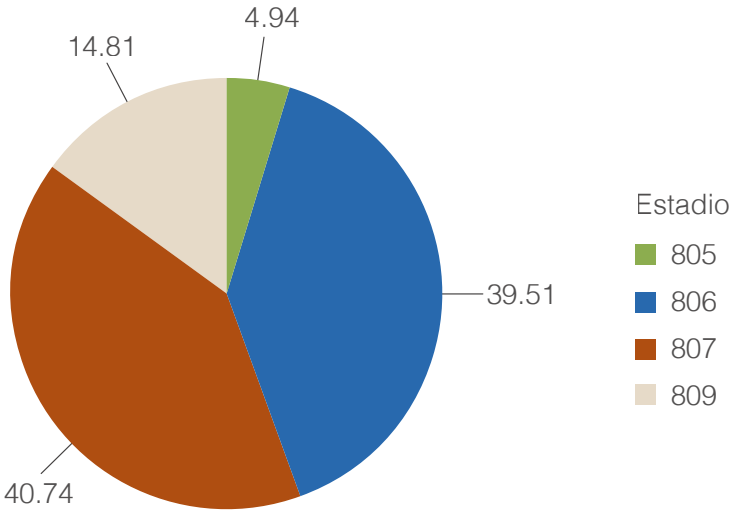

Figura 30. Gráfico circular con ggplot

Apéndices

A. Conjunto de datos empleado

Tabla 7. Conjunto de datos empleado[6]

lote	eval	trat	rep	ancho_fol	peso_seco	estadio	largo_fol
1	6	1	1	5,12	2,48	805	80,23
1	6	1	2	4,94	3,61	806	78,04
1	6	1	3	5,24	3,96	806	81,67
1	6	2	1	8,1	3,39	806	109,83
1	6	2	2	7,52	2,34	806	102,74
1	6	2	3	8,95	4,19	806	120,24
1	6	3	1	5,81	4,99	806	98,99
1	6	3	2	5,3	4,31	807	92,77
1	6	3	3	6,86	4,42	806	111,75
1	12	1	1	5,08	3,98	809	74,79
1	12	1	2	5,64	4,07	806	81,61
1	12	1	3	4,72	4,6	807	70,43
1	12	2	1	7,12	3,62	805	103,91
1	12	2	2	6,46	4,04	806	95,78
1	12	2	3	6,47	3,17	806	95,98
1	12	3	1	6,72	4,05	806	95,02
1	12	3	2	5,7	4,63	807	82,5
1	12	3	3	5,25	4,51	807	77,06
1	18	1	1	5,21	3,62	806	81,2
1	18	1	2	5,69	4,75	806	87,03
1	18	1	3	5,1	5,93	807	79,86
1	18	2	1	5,82	3,88	807	85,42
1	18	2	2	9,12	4,79	807	125,65
1	18	2	3	8,85	4,37	806	122,34
1	18	3	1	7,03	3,84	809	102,97
1	18	3	2	6,24	4,87	807	93,34
1	18	3	3	5,53	5,08	807	84,62

6 Descargue el archivo de datos en este vínculo: https://fedepalma.org/wp-content/uploads/2023/07/datos1.xlsx

lote	eval	trat	rep	ancho_fol	peso_seco	estadio	largo_fol
2	6	1	1	5,19	4,26	807	82,53
2	6	1	2	5,52	3,62	807	86,56
2	6	1	3	5,11	5,23	807	81,62
2	6	2	1	5,3	2,99	806	79,24
2	6	2	2	5,85	3,57	806	85,89
2	6	2	3	7,71	4,76	807	108,61
2	6	3	1	5,12	5,43	805	77,62
2	6	3	2	5,69	4,1	806	84,66
2	6	3	3	5,51	5,07	809	82,37
2	12	1	1	4,9	4,05	807	79,27
2	12	1	2	5,28	3,6	809	83,92
2	12	1	3	5,43	3,69	809	85,78
2	12	2	1	7,1	4,31	807	98,15
2	12	2	2	5,21	4,49	805	75,11
2	12	2	3	5,39	4,31	809	77,33
2	12	3	1	6,85	4,27	806	100,18
2	12	3	2	7,08	5,38	807	102,89
2	12	3	3	6,21	4,02	806	92,33
2	18	1	1	4,92	4,83	806	79,72
2	18	1	2	4,81	5,13	806	78,46
2	18	1	3	5,06	2,15	807	81,48
2	18	2	1	5,72	4,27	807	93,85
2	18	2	2	5,01	3,41	807	85,18
2	18	2	3	5,9	2,99	807	96,04
2	18	3	1	6,56	4,6	807	100,48
2	18	3	2	5,5	4,43	809	87,56
2	18	3	3	6,98	3,45	806	105,56
3	6	1	1	5,28	5,08	807	82,66
3	6	1	2	4,94	4,41	809	78,62
3	6	1	3	4,89	4,54	807	77,97
3	6	2	1	4,45	3,26	806	82,98
3	6	2	2	4,15	2,33	807	79,3

lote	eval	trat	rep	ancho_fol	peso_seco	estadio	largo_fol
3	6	2	3	7,49	2,93	807	120
3	6	3	1	7,13	5,59	806	102,97
3	6	3	2	5,68	5,58	807	85,29
3	6	3	3	7,2	3,86	807	103,75
3	12	1	1	5,22	2,74	809	80,61
3	12	1	2	4,84	3,43	806	76,03
3	12	1	3	4,97	3,18	809	77,62
3	12	2	1	8,33	4,2	807	116,94
3	12	2	2	2,99	3,49	806	51,85
3	12	2	3	2,88	3,72	806	50,49
3	12	3	1	6,41	4,05	809	91,33
3	12	3	2	6,12	4,49	807	87,81
3	12	3	3	5,11	3,88	807	75,46
3	18	1	1	5,43	5	807	86,25
3	18	1	2	5,32	1,89	807	84,91
3	18	1	3	5,15	5,18	806	82,88
3	18	2	1	2,09	3,53	809	55,2
3	18	2	2	6,03	3,56	806	103,29
3	18	2	3	9,32	2,84	806	143,37
3	18	3	1	5,1	4,89	806	81,45
3	18	3	2	6,65	3,16	806	100,27
3	18	3	3	6,33	4,16	807	96,37

B. Instalación de R y RStudio

Instalación de R

Para descargar R debe dirigirse a la página CRAN que se encuentra en el enlace The Comprehensive R Archive Network[7]. Es importante aclarar que la mayoría de las páginas y documentos oficiales se encuentran en inglés.

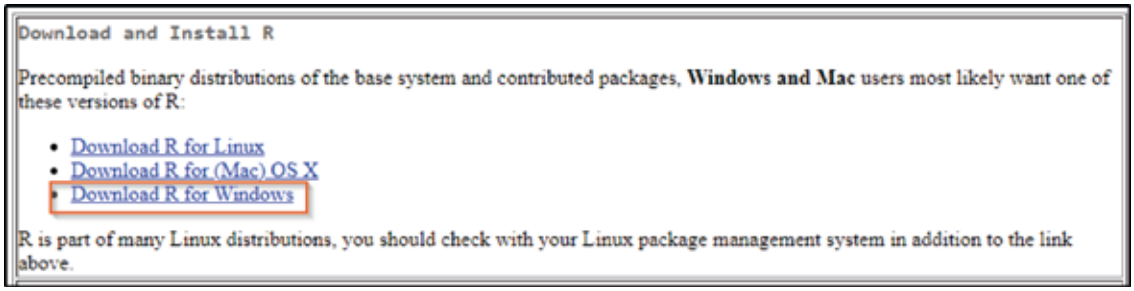

Figura 31. Página CRAN para descargar R

Al ir a la página se observa un cuadro similar al de la Figura 31. Este contiene diferentes enlaces para descargar e instalar R. Aquí se explica el paso a paso para Windows, ya que de forma similar se realiza en los otros dos sistemas operativos. Para bajarlo en su última versión, se hace clic en el hipervínculo Download R por Windows.

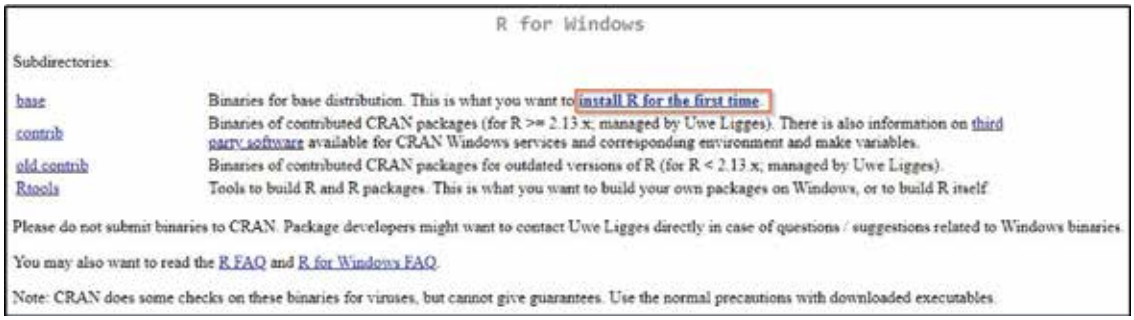

Figura 32. Descargar R para Windows

Allí se debe hacer clic en *install R for the fist time*, y luego sale el enlace para la descarga (Figura 32). Al cliquear el hipervínculo *Download R for Windows*, automáticamente debe descargar un archivo .exe con la última versión de R (a la fecha de elaboración de este manual era la 4.04) (Figura 33).

Al ejecutar el archivo descargado, el computador pide permisos para instalar el *software*, y al aceptarlo se genera una ventana que permite elegir el idioma de preferencia. Una vez realizado, se hace clic en la opción "siguiente" a todas las ventanas que aparezcan. Seguidamente, se puede ingresar a R desde la lista de programas, o por medio del acceso directo del escritorio.

7 https://cran.r-project.org/index.html

Figura 33. Última versión de R

Instalación de RStudio

Para descargar RStudio, debe dirigirse a la página oficial que se encuentra en el enlace *Download the RStudio IDE*. Aquí se puede acceder a la versión gratis o comercial del *software*, según las necesidades del usuario (Figura 34).

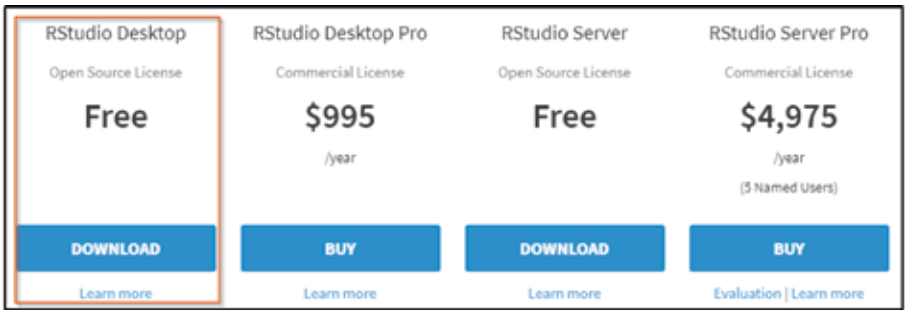

Figura 34. Opciones de descarga de RStudio

Para descargarlo se hace clic en el hipervínculo que se encuentra en el cuadro en *DOWNLOAD*. Luego aparece un aviso como el de la Figura 35. Observe que es necesario tener instalado R para que RStudio funcione. En esta página se debe oprimir el enlace *DOWNLOAD RSTUDIO FOR WINDOWS*. Automáticamente se descarga un archivo .exe. Al ejecutarlo, el computador pide permisos para instalar el *software*.

Después, se muestra el directorio en el que quedará ubicado el programa, y el nombre de la capeta en el menú de inicio, y empezará la instalación. Una vez hecho esto, se puede acceder a RStudio desde la lista de programas, o por medio de acceso directo en el escritorio.

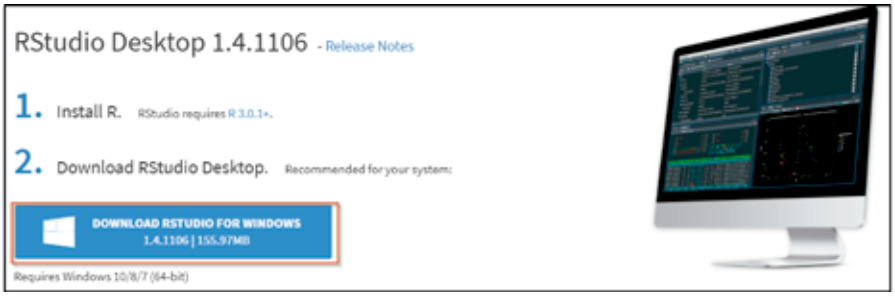

Figura 35. Descargar RStudio para Windows

Bibliografía

Gough, B. (2009). *GNU Scientific Library Reference Manual*. Network Theory Ltd.

Usuga, O., & Hernández, F (2021). Manual de R. https://fhernanb.github.io/Manual-de-R/

Vinasco, L. E (2012). *Estadística Descriptiva con Minitab versión 15.0*. Cali: Pontificia Universidad Javeriana.

R Core Team. (2020). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.r-project.org/

Clifford Blair, R., & Taylor, R. A. (2008). *Bioestadística*. México: Pearson Educación.

RStudio Team. (2020). *RStudio: Integrated Development Environment for r.* Boston, MA: RStudio, PBC.

Santana, J. S., & Farfán, E. M. (2014). *El arte de programar en R: un lenguaje para la estadística*. Instituto Mexicano de Tecnología del Agua, *1*.

Peng, R. D. (2016). *R programming for data science*. Leanpub. http://lib.21h.io/library/VHTNFN7I/download/KMHYHXWZ/R_Programming_for_Data_Science_184p_.pdf

Triola, M. F. (2018). *Estadística (12 Ed.)*. México: Pearson Education.

Wickham, H. (2016). *Ggplot2: Elegant Graphics for Data Analysis*. New York: Springer-Verlag. https://ggplot2.tidyverse.org

Wickham, H., & Bryan, J. (2019). *Readxl: Read Excel Files*. https://CRAN.R-project.org/ package=readxl

Wickham, H., François, R., Henry, L., & Müller, K. (2020). *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr