Prácticas de Bioestadística con R

Alfredo Sánchez Alberca
asalber@ceu.es
https://aprendeconalf.es

Tabla de contenidos

Prefacio

¡Bienvenido a Prácticas de Bioestadística con R!

Este libro presenta una recopilación de prácticas de Bioestadística Descriptiva e Inferencial con el lenguaje de programación R, con problemas aplicados a las Ciencias de la Salud.

No es un libro para aprender a programar con R, ya que solo enseña el uso del lenguaje y de algunos de sus paquetes para resolver problemas de Bioestadística. Para quienes estén interesados en aprender a programar en este lenguaje, os recomiendo leer este manual de R.

Capítulos

1. Introducción a R
2. Tipos y estructuras de datos
3. Preprocesamiento de datos
4. Distribuciones de frecuencias y representaciones gráficas

Licencia

- **Compartir bajo la misma licencia**. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Estas condiciones pueden no aplicarse si se obtiene el permiso del titular de los derechos de autor.

Nada en esta licencia menoscaba o restringe los derechos morales del autor.

1 Introducción a R

La gran potencia de cómputo alcanzada por los ordenadores ha convertido a los mismos en poderosas herramientas al servicio de todas aquellas disciplinas que, como la Estadística, requieren manejar un gran volumen de datos. Actualmente, prácticamente nadie se plantea hacer un estudio estadístico serio sin la ayuda de un buen programa de análisis de datos.

R es un potente lenguaje de programación que incluye multitud de funciones para la representación y el análisis de datos. Fue desarrollado por Robert Gentleman y Ross Ihaka en la Universidad de Auckland en Nueva Zelanda, aunque actualmente es mantenido por una enorme comunidad científica en todo el mundo.

Figura 1.1: Logotipo de R

Las ventajas de R frente a otros programas habituales de análisis de datos, como pueden ser SPSS, SAS o Matlab, son múltiples:

- Es software libre y por tanto gratuito. Puede descargarse desde la web http://www.r-project.org/.
- Es multiplataforma. Existen versiones para Windows, Mac, Linux y otras plataformas.
- Está avalado y en constante desarrollo por una amplia comunidad científica distribuida por todo el mundo que lo utiliza como estándar para el análisis de datos.
- Cuenta con multitud de paquetes para todo tipo de análisis estadísticos y representaciones gráficas, desde los más habituales, hasta los más novedosos y sofisticados que no incluyen otros programas. Los paquetes están organizados y documentados en un repositorio CRAN (Comprehensive R Archive Network) desde donde pueden descargarse libremente.

- Es programable, lo que permite que el usuario pueda crear fácilmente sus propias funciones o paquetes para análisis de datos específicos.
- Existen multitud de libros, manuales y tutoriales libres que permiten su aprendizaje e ilustran el análisis estadístico de datos en distintas disciplinas científicas como las Matemáticas, la Física, la Biología, la Psicología, la Medicina, etc.

1.1. Instalación de R

R puede descargarse desde el sitio web oficial de R o desde el repositorio principal de paquetes de R CRAN. Basta con descargar el archivo de instalación correspondiente al sistema operativo de nuestro ordenador y realizar la instalación como cualquier otro programa.

El intérprete de R se arranca desde la terminal, aunque en Windows incorpora su propia aplicación, pero es muy básica. En general, para trabajos serios, conviene utilizar un entorno de desarrollo para R.

1.2. Entornos de desarrollo

Por defecto el entorno de trabajo de R es en línea de comandos, lo que significa que los cálculos y los análisis se realizan mediante comandos o instrucciones que el usuario teclea en una ventana de texto. No obstante, existen distintas interfaces gráficas de usuario que facilitan su uso, sobre todo para usuarios noveles. Algunas de ellas, como las que se enumeran a continuación, son completos entornos de desarrollo que facilitan la gestión de cualquier proyecto:

- RStudio. Probablemente el entorno de desarrollo más extendido para programar con R ya que incorpora multitud de utilidades para facilitar la programación con R.

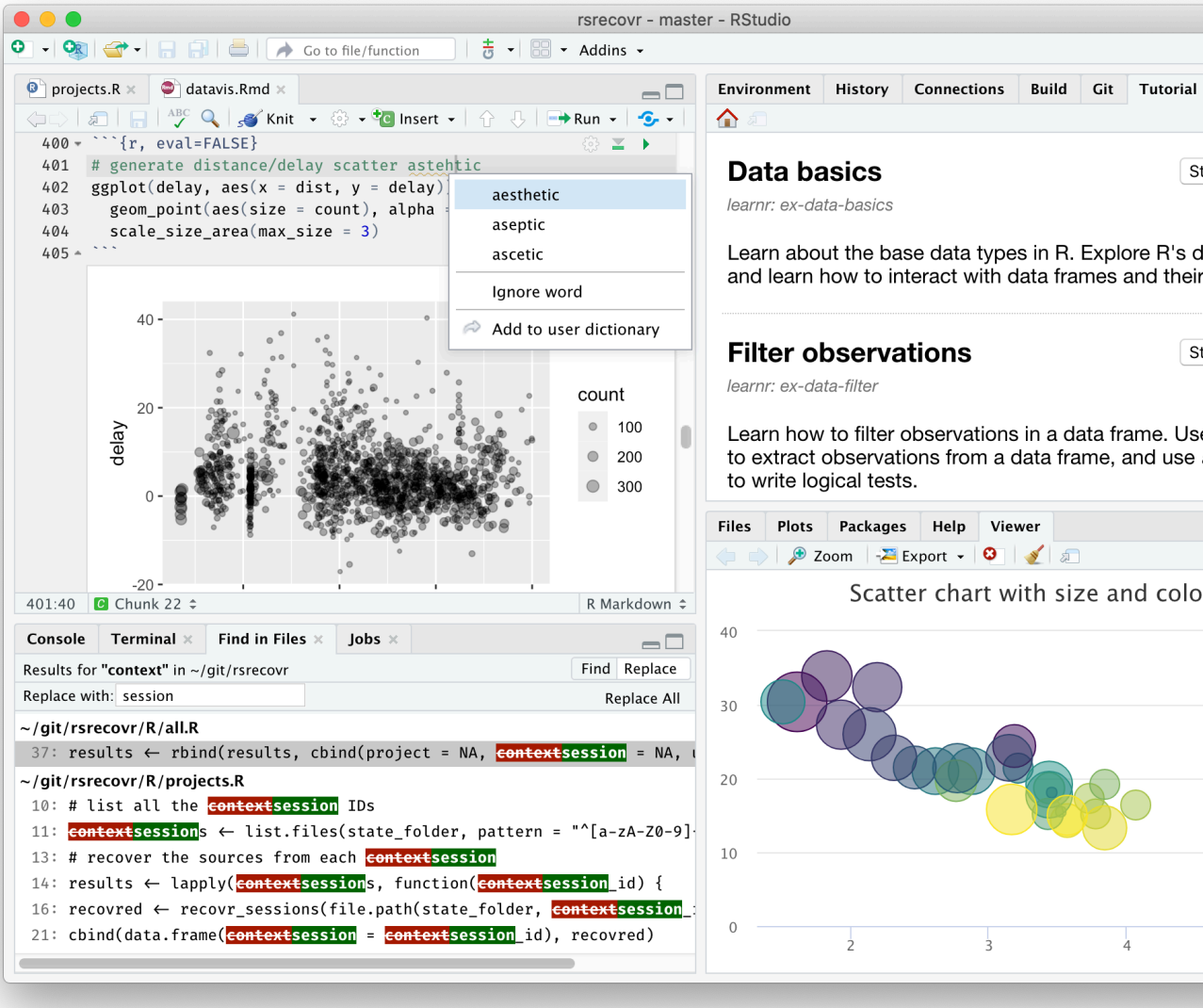


Figura 1.2: Entorno de desarrollo RStudio

- RKWard. Es otra otro de los entornos de desarrollo más completos que además incluye a posibilidad de añadir nuevos menús y cuadros de diálogo personalizados.

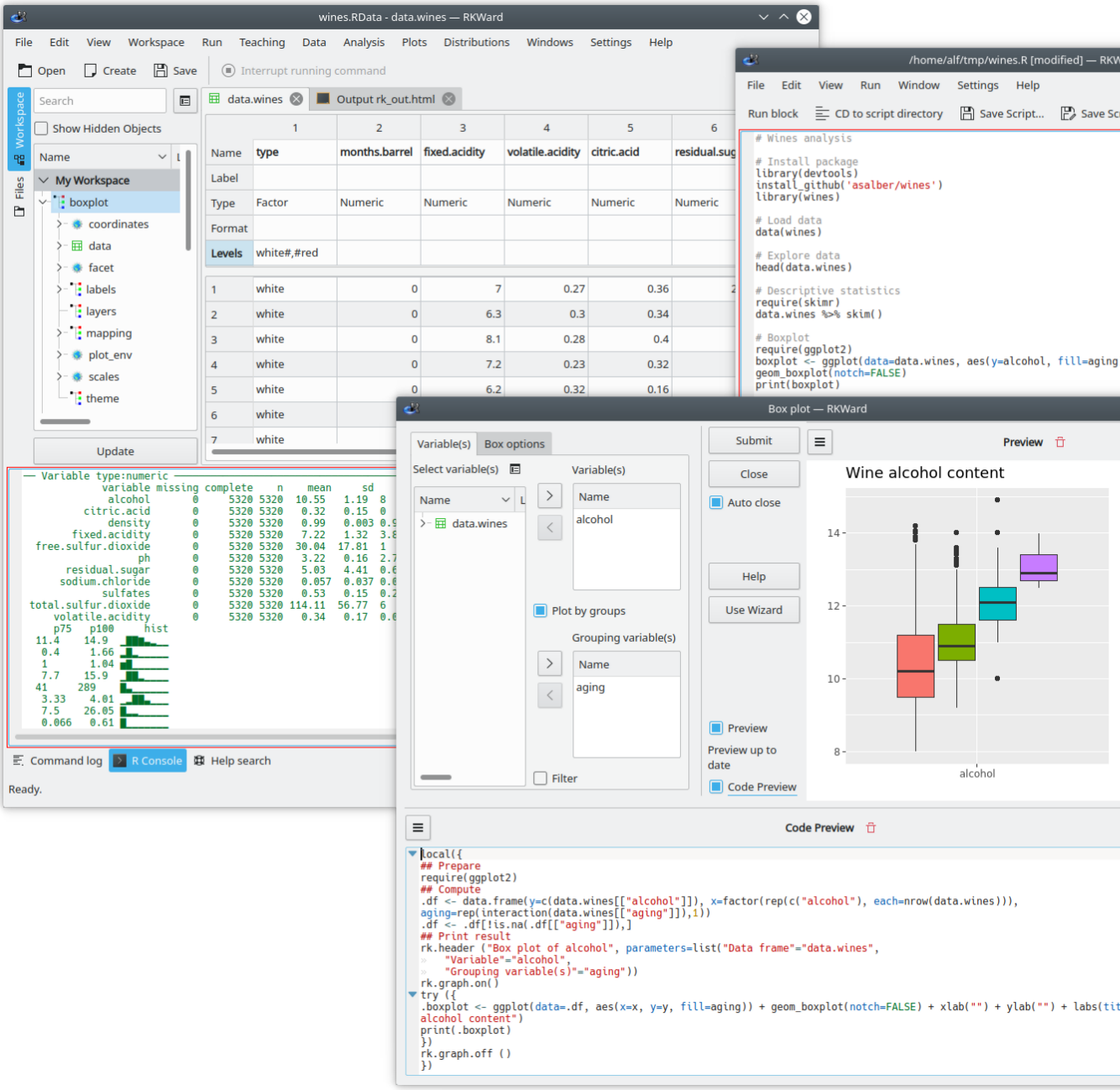


Figura 1.3: Entorno de desarrollo RKWard

- Jupyter Lab. Es un entorno de desarrollo interactivo que permite la creación de documentos que contienen código, texto, gráficos. Aunque no es un entorno de desarrollo específico para R, incluye un kernel para R que permite ejecutar código R en los documentos.

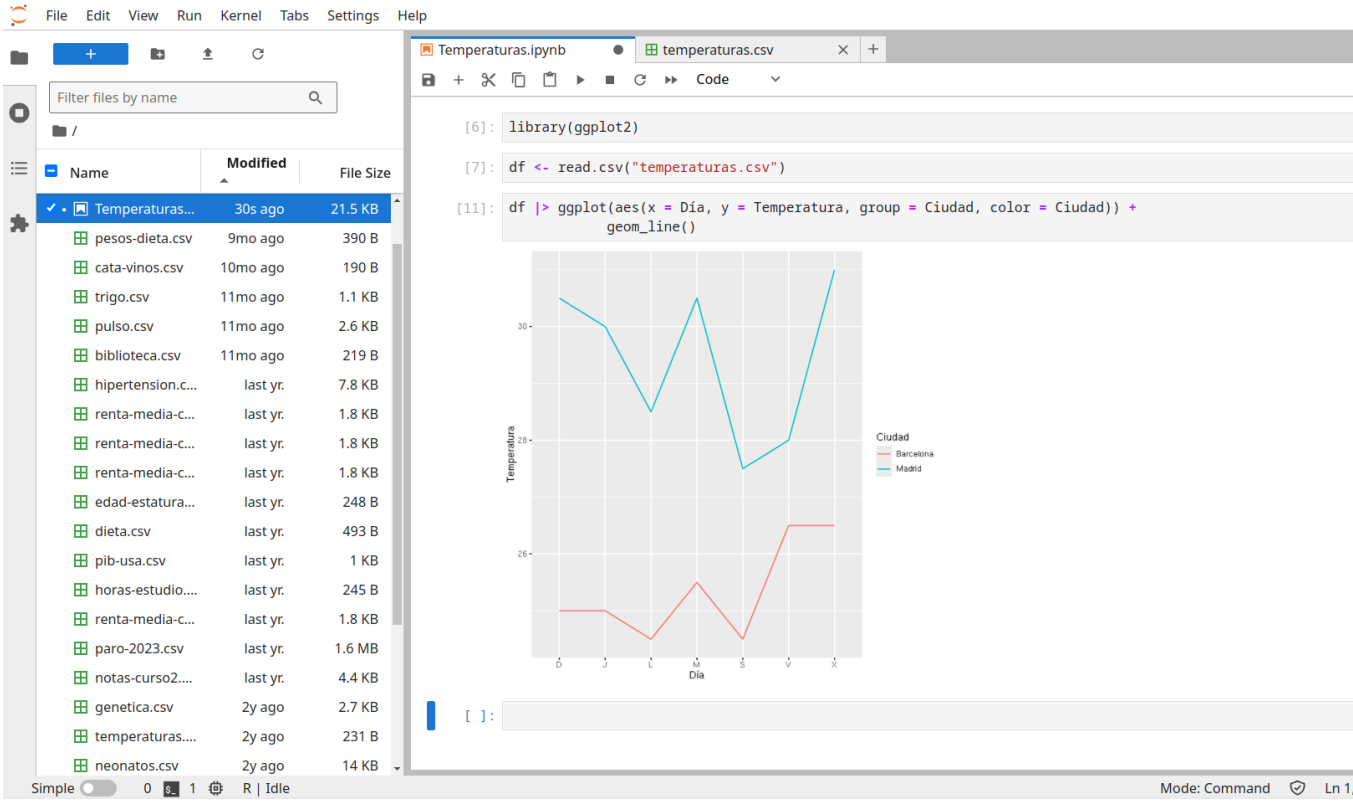


Figura 1.4: Entorno de desarrollo Jupyter Lab

- Visual Studio Code. Es un entorno de desarrollo de propósito general ampliamente extendido. Aunque no es un entorno de desarrollo específico para R, incluye una extensión con utilidades que facilitan mucho el desarrollo con R.

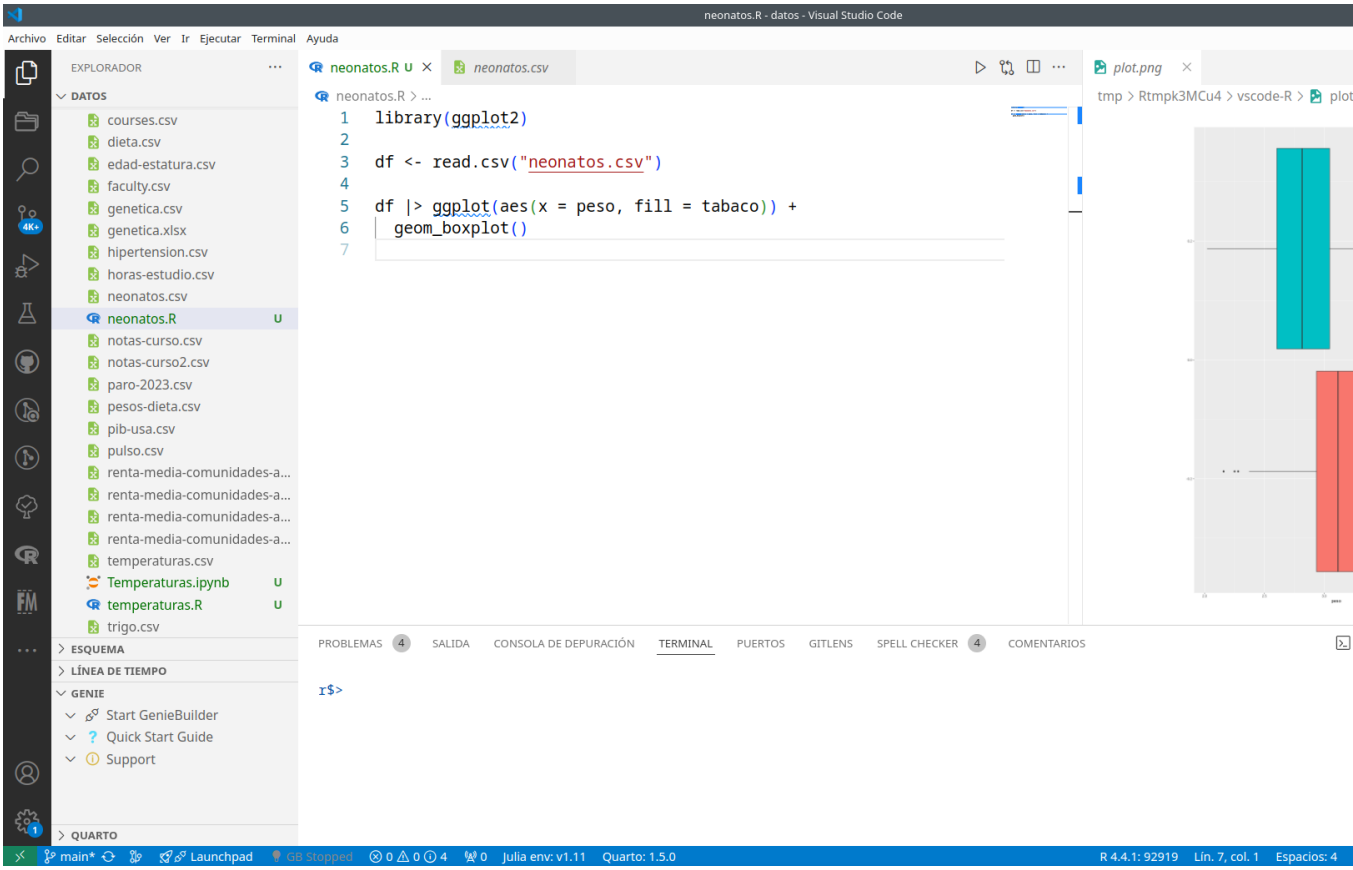


Figura 1.5: Entorno de desarrollo Visual Studio Code

1.3. Instalación de paquetes

R es un lenguaje de programación modular, lo que significa que su funcionalidad se extiende mediante paquetes. Los paquetes son colecciones de funciones, datos y documentación sobre el uso de esas funciones o conjuntos de datos.

El repositorio de paquetes más importante es CRAN (Comprehensive R Archive Network), pero existen otros repositorios como Bioconductor que contiene paquetes específicos para el análisis de datos biológicos.

1.3.1. Instalación de paquetes desde CRAN

Para instalar un paquete en R basta con ejecutar la función `install.packages()` con el nombre del paquete que se desea instalar. Por ejemplo, para instalar el paquete `ggplot2`

que es uno de los paquetes más utilizados para realizar gráficos en R, basta con ejecutar el siguiente comando:

```
install.packages("ggplot2")
```

Los ubicación de los paquete instalados en R depende del sistema operativo, pero puede consultarse en la variable `.libPaths()`.

1.3.2. Instalación de paquetes desde Bioconductor

Para instalar un paquete desde Bioconductor es necesario instalar primero el paquete `BiocManager` y después utilizar la función `BiocManager::install()` con el nombre del paquete que se desea instalar. Por ejemplo, para instalar el paquete `DESeq2` que es uno de los paquetes más utilizados para el análisis de datos de expresión génica, basta con ejecutar el siguiente comando:

```
install.packages("BiocManager")
BiocManager::install("DESeq2")
```

1.4. Actualización de paquetes

Cada cierto tiempo conviene actualizar los paquetes instalados en R para asegurarse de que se dispone de las últimas versiones de los mismos. Para ello se puede utilizar la función `update.packages()`. Por ejemplo, para actualizar todos los paquetes instalados en R sin necesidad de confirmación por parte del usuario, basta con ejecutar el siguiente comando:

```
update.packages(ask = FALSE)
```

2 Tipos y estructuras de datos

Esta práctica contiene ejercicios que muestran cómo trabajar con los tipos y estructuras de datos en R. En concreto, las estructuras de datos que se utilizan son

- Vectores.
- Factores.
- Matrices.
- Listas.
- Dataframes.

2.1. Ejercicios Resueltos

Para la realización de esta práctica se requieren los siguientes paquetes.

```
library(tidyverse)
# Incluye los siguientes paquetes:
# - readr: para la lectura de ficheros csv.
# - dplyr: para el preprocesamiento y manipulación de datos.
library(knitr) # Para el formateo de tablas.
```

Ejercicio 2.1. Realizar las siguientes operaciones con vectores.

a. Crear un vector con los números del 1 al 10.

Solución

2.2. Función c

La función `c()` permite combinar elementos en un vector. Los elementos se introducen separados por comas.

```
numeros <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
numeros
```

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

2.3. Operador :

El operador `inicio:fin` permite crear un vector con la secuencia de números enteros desde el número `inicio` hasta el número `fin`.

```
numeros <- 1:10
numeros
```

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

b. Mostrar el número de elementos del vector anterior.

Solución

```
length(numeros)
```

```
[1] 10
```

c. Crear un vector con los números pares del 1 al 10.

Solución

2.4. Función c

```
pares <- c(2, 4, 6, 8, 10)
pares
```

```
[1]  2  4  6  8 10
```

2.5. Función seq

La función `seq(inicio, fin, salto)` permite crear un vector con la secuencia de números enteros desde el número `inicio` hasta el número `fin` con un salto de `salto`.

```
pares <- seq(2, 10, by = 2)
pares
```

```
[1]  2  4  6  8 10
```

d. Crear un vector con el cuadrado de los elementos del vector anterior.

Solución

El operador ^ permite elevar un número a otro. Cuando se aplica a un vector, eleva cada elemento del vector al número indicado.

```
cuadrados <- pares^2
cuadrados
```

```
[1]   4  16  36  64 100
```

e. Crear un vector con 5 números aleatorios entre 1 y 10.

Solución

La función `sample(vector, n)` permite seleccionar `n` elementos aleatorios de `vector`. El muestreo es sin reemplazamiento.

```
aleatorios <- sample(1:10, 5)
aleatorios
```

```
[1]  3  4  5 10  1
```

f. Crear un vector booleano con los números del vector anterior que son pares.

Solución

El operador `%%` permite calcular el resto de la división entera de dos números. Si el resto es 0, el número es par. Y el operador `==` permite comparar dos vectores elemento a elemento.

```
par <- aleatorios %% 2 == 0
par
```

```
[1] FALSE  TRUE FALSE  TRUE FALSE
```

g. Crear un vector con 100 números aleatorios entre 0 y 1.

Solución

La función `runif(n, min, max)` permite generar `n` números aleatorios entre `min` y `max`.

```
aleatorios2 <- runif(100, 0, 1)
aleatorios2
```

```
  [1] 0.891053847 0.549542916 0.438272027 0.154524597 0.647661075 0.420121032
  [7] 0.453219977 0.947620729 0.100360439 0.612319690 0.404328616 0.632402784
 [13] 0.139879803 0.405395862 0.368278909 0.718734201 0.350693136 0.953217653
 [19] 0.271865823 0.714802996 0.534623191 0.116683595 0.368431539 0.962512388
```

```
 [25] 0.069253915 0.430993993 0.272676231 0.567695538 0.504801705 0.899287539
 [31] 0.733694028 0.307235055 0.630983480 0.809088704 0.137304815 0.857373748
 [37] 0.535310918 0.232112028 0.463858604 0.329170325 0.515030533 0.562275802
 [43] 0.277321492 0.049509382 0.626810341 0.640371124 0.025681507 0.112823939
 [49] 0.248713521 0.937774612 0.296424835 0.717441039 0.529676800 0.553558698
 [55] 0.478533756 0.164942520 0.352589322 0.510333835 0.448018191 0.241118633
 [61] 0.115184974 0.580480553 0.021033541 0.356509073 0.367802356 0.644378388
 [67] 0.814249540 0.913953091 0.592566577 0.223798848 0.127317987 0.028979045
 [73] 0.362386827 0.450190519 0.002673445 0.619867071 0.717661681 0.776532539
 [79] 0.538139816 0.474312478 0.003975592 0.917286967 0.758566601 0.066987475
 [85] 0.253251577 0.809204480 0.555353051 0.399625064 0.617728865 0.789407390
 [91] 0.964750229 0.516433493 0.249310608 0.485464046 0.867486529 0.594526743
 [97] 0.684210896 0.935536078 0.542142871 0.787089645
```

h. Ordenar el vector anterior de menor a mayor.

Solución

La función `sort(vector)` permite ordenar los elementos de un vector de menor a mayor.

```
sort(aleatorios2)
```

```
  [1] 0.002673445 0.003975592 0.021033541 0.025681507 0.028979045 0.049509382
  [7] 0.066987475 0.069253915 0.100360439 0.112823939 0.115184974 0.116683595
 [13] 0.127317987 0.137304815 0.139879803 0.154524597 0.164942520 0.223798848
 [19] 0.232112028 0.241118633 0.248713521 0.249310608 0.253251577 0.271865823
 [25] 0.272676231 0.277321492 0.296424835 0.307235055 0.329170325 0.350693136
 [31] 0.352589322 0.356509073 0.362386827 0.367802356 0.368278909 0.368431539
 [37] 0.399625064 0.404328616 0.405395862 0.420121032 0.430993993 0.438272027
 [43] 0.448018191 0.450190519 0.453219977 0.463858604 0.474312478 0.478533756
 [49] 0.485464046 0.504801705 0.510333835 0.515030533 0.516433493 0.529676800
 [55] 0.534623191 0.535310918 0.538139816 0.542142871 0.549542916 0.553558698
 [61] 0.555353051 0.562275802 0.567695538 0.580480553 0.592566577 0.594526743
 [67] 0.612319690 0.617728865 0.619867071 0.626810341 0.630983480 0.632402784
 [73] 0.640371124 0.644378388 0.647661075 0.684210896 0.714802996 0.717441039
 [79] 0.717661681 0.718734201 0.733694028 0.758566601 0.776532539 0.787089645
 [85] 0.789407390 0.809088704 0.809204480 0.814249540 0.857373748 0.867486529
 [91] 0.891053847 0.899287539 0.913953091 0.917286967 0.935536078 0.937774612
 [97] 0.947620729 0.953217653 0.962512388 0.964750229
```

i. Ordenar el vector anterior de mayor a menor.

Solución

La función `sort(vector, decreasing = TRUE)` permite ordenar los elementos de un vector de mayor a menor.

```
sort(aleatorios2, decreasing = TRUE)
```

```
 [1] 0.964750229 0.962512388 0.953217653 0.947620729 0.937774612 0.935536078
 [7] 0.917286967 0.913953091 0.899287539 0.891053847 0.867486529 0.857373748
[13] 0.814249540 0.809204480 0.809088704 0.789407390 0.787089645 0.776532539
[19] 0.758566601 0.733694028 0.718734201 0.717661681 0.717441039 0.714802996
[25] 0.684210896 0.647661075 0.644378388 0.640371124 0.632402784 0.630983480
[31] 0.626810341 0.619867071 0.617728865 0.612319690 0.594526743 0.592566577
[37] 0.580480553 0.567695538 0.562275802 0.555353051 0.553558698 0.549542916
[43] 0.542142871 0.538139816 0.535310918 0.534623191 0.529676800 0.516433493
[49] 0.515030533 0.510333835 0.504801705 0.485464046 0.478533756 0.474312478
[55] 0.463858604 0.453219977 0.450190519 0.448018191 0.438272027 0.430993993
[61] 0.420121032 0.405395862 0.404328616 0.399625064 0.368431539 0.368278909
[67] 0.367802356 0.362386827 0.356509073 0.352589322 0.350693136 0.329170325
[73] 0.307235055 0.296424835 0.277321492 0.272676231 0.271865823 0.253251577
[79] 0.249310608 0.248713521 0.241118633 0.232112028 0.223798848 0.164942520
[85] 0.154524597 0.139879803 0.137304815 0.127317987 0.116683595 0.115184974
[91] 0.112823939 0.100360439 0.069253915 0.066987475 0.049509382 0.028979045
[97] 0.025681507 0.021033541 0.003975592 0.002673445
```

j. Crear un vector con los días laborables de la semana.

Solución

```
dias_laborables <- c("Lunes", "Martes", "Miércoles", "Jueves", "Viernes")
dias_laborables
```

```
[1] "Lunes"     "Martes"    "Miércoles" "Jueves"    "Viernes"
```

k. Añadir los días del fin de semana al vector anterior y guardar el resultado en una nueva variable.

Solución

```
dias <- c(dias_laborables, "Sábado", "Domingo")
dias
```

```
[1] "Lunes"     "Martes"    "Miércoles" "Jueves"    "Viernes"   "Sábado"
[7] "Domingo"
```

l. Acceder al tercer elemento del vector.

Solución

```
dias_laborables[3]
```

```
[1] "Miércoles"
```

m. Seleccionar los días pares del vector.

Solución

2.6. Índices numéricos

```
dias[c(2, 4, 6)]
```

```
[1] "Martes" "Jueves" "Sábado"
```

2.7. Índices numéricos negativos

```
dias[-c(1, 3, 5, 7)]
```

```
[1] "Martes" "Jueves" "Sábado"
```

2.8. Índices lógicos

```
dias[c(FALSE, TRUE)]
```

```
[1] "Martes" "Jueves" "Sábado"
```

n. Concatenar los elementos del vector en una cadena de texto.

Solución

La función `paste(vector, collapse = " ")` permite concatenar los elementos de un vector en una cadena de texto separados por un espacio.

```
paste(dias, collapse = " ")
```

```
[1] "Lunes Martes Miércoles Jueves Viernes Sábado Domingo"
```

ñ. Concatenar los elementos del vector en una cadena de texto separados por comas.

Solución

```
semana <- paste(dias, collapse = ", ")
semana
```

```
[1] "Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo"
```

o. Dividir la cadena anterior en subcadenas usando como separador la coma.

Solución

La función `strsplit(cadena, separador)` permite dividir una cadena de texto en subcadenas usando como separador el valor de `separador`.

```
strsplit(semana, ", ")
```

```
[[1]]
[1] "Lunes"     "Martes"    "Miércoles" "Jueves"    "Viernes"   "Sábado"
[7] "Domingo"
```

Ejercicio 2.2. Se ha tomado una muestra de alumnos de una clase y se ha recogido la información sobre el sexo de los alumnos obteniendo los siguientes datos:

Mujer, Hombre, Mujer, Hombre, Mujer, Mujer, Hombre, Hombre

a. Crear un vector con los datos de la muestra.

Solución

```
sexo <- c("Mujer", "Hombre", "Mujer", "Hombre", "Mujer", "Mujer", "Hombre", "Hombr
sexo
```

```
[1] "Mujer"  "Hombre" "Mujer"  "Hombre" "Mujer"  "Mujer"  "Hombre" "Hombre"
```

b. Convertir el vector anterior en un factor.

Solución

La función `factor(vector, labels)` permite convertir `vector` en un factor con los niveles o categorías especificados en `labels`. Si no se indica `labels`, los niveles se toman de los elementos del vector y se ordenan alfabéticamente.

```
sexo <- factor(sexo)
sexo
```

```
[1] Mujer  Hombre Mujer  Hombre Mujer  Mujer  Hombre Hombre
```

```
Levels: Hombre Mujer
```

c. Mostrar los niveles del factor.

> Solución
>
> La función `levels(factor)` permite mostrar los niveles del factor `factor`.

```
levels(sexo)
```

```
[1] "Hombre" "Mujer"
```

d. Reordenar los niveles del factor para que la categoría "Mujer" sea la primera.

> Solución

```
sexo <- factor(sexo, levels = c("Mujer", "Hombre"))
sexo
```

```
[1] Mujer  Hombre Mujer  Hombre Mujer  Mujer  Hombre Hombre
Levels: Mujer Hombre
```

Ejercicio 2.3. Realizar las siguientes operaciones con matrices.

a. Crear una matriz de 2 filas y 2 columnas con los números del 1 al 4.

> Solución
>
> La función `matrix(vector, nrow, ncol)` permite crear una matriz con los datos de `vector` el número de filas indicado en `nrow` y el número de columnas indicado en `ncol`.

```
A <- matrix(1:4, nrow = 2, ncol = 2)
A
```

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

b. Añadir a la matriz anterior una nueva columna con los números del 5 y 6.

> Solución
>
> La función `cbind(matriz, vector)` permite añadir una nueva columna a la matriz `matriz` con los datos de `vector`.

```r
A <- cbind(A, 5:6)
A
```

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

c. Crear una matriz de 2 filas y 2 columnas con los números del 1 al 4, rellenando los elementos por filas.

Solución

La función `matrix` rellena los elementos de la matriz por columnas. Para rellenar los elementos por filas, se puede utilizar el parámetro opcional `byrow = TRUE`.

```r
B <- matrix(1:4, nrow = 2, ncol = 2, byrow = TRUE)
B
```

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

d. Crear otra matriz a partir de la anterior añadiendo una fila con los números 5 y 6.

Solución

```r
B <- rbind(B, 5:6)
B
```

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

e. Acceder al elemento de la segunda fila y la primera columna de la matriz anterior.

Solución

```r
B[2, 1]
```

```
[1] 3
```

f. Seleccionar la primera fila de la matriz.

Solución

```
B[1, ]
```

```
[1] 1 2
```

g. Seleccionar la segunda columna de la matriz.

Solución

```
B[, 2]
```

```
[1] 2 4 6
```

h. Multiplicar la matriz A por la matriz B.

Solución

La multiplicación de matrices se realiza con el operador `%*%`.

```
A %*% B
```

```
     [,1] [,2]
[1,]   35   44
[2,]   44   56
```

i. Calcular la transpuesta de la matriz A.

Solución

La función `t(matriz)` permite calcular la transpuesta de `matriz`.

```
t(A)
```

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

Ejercicio 2.4. Realizar las siguientes operaciones con listas.

a. Crear una lista con los siguientes con los datos del siguiente alumno:

- Nombre: Juan.
- Edad: 20 años.

Solución

Para crear una lista se utiliza la función `list(nombre1 = valor1, nombre2 = valor2, ...)`.

```
alumno <- list(Nombre = "Juan", Edad = 20)
alumno
```

```
$Nombre
[1] "Juan"

$Edad
[1] 20
```

b. Obtener la edad del alumno.

Solución

Para acceder a los elementos de una lista se utiliza el operador `$`.

```
alumno$Edad
```

```
[1] 20
```

c. Crear una lista con las siguientes notas del alumno:

- Matemáticas: 7.
- Química: 8.

Solución

```
notas <- list(Matemáticas = 7, Química = 8)
notas
```

```
$Matemáticas
[1] 7

$Química
[1] 8
```

d. Añadir la lista de notas a la lista del alumno.

Solución

```
alumno$Notas <- notas
alumno
```

```
$Nombre
[1] "Juan"

$Edad
[1] 20

$Notas
$Notas$Matemáticas
[1] 7

$Notas$Química
[1] 8
```

e. Añadir a la lista anterior la nota del examen de Física, que ha sido un 6.

Solución

```
alumno$Notas$Física <- 6
alumno
```

```
$Nombre
[1] "Juan"

$Edad
[1] 20

$Notas
$Notas$Matemáticas
[1] 7

$Notas$Química
[1] 8

$Notas$Física
[1] 6
```

Ejercicio 2.5. La siguiente tabla contiene los ingresos y gastos de una empresa durante el primer trimestre del año.

Mes	Ingresos	Gastos	Impuestos
Enero	45000	33400	6450
Febrero	41500	35400	6300
Marzo	51200	35600	7100

a. Crear un data frame con los datos de la tabla.

Solución

Para crear un data frame se utiliza la función `data.frame(columna1 = vector1, columna2 = vector2, ...)`, donde `columna1`, `columna2`, … son los nombres de las columnas y `vector1`, `vector2`, … son los vectores con los datos de cada columna, que deben tener la misma longitud.

```
df <- data.frame(
    Mes = c("Enero", "Febrero", "Marzo"),
    Ingresos = c(45000, 41500, 51200),
    Gastos = c(33400, 35400, 35600)
    )
df
```

```
      Mes Ingresos Gastos
1   Enero    45000  33400
2 Febrero    41500  35400
3   Marzo    51200  35600
```

b. Añadir una nueva columna con los siguientes impuestos pagados.

Mes	Impuestos
Enero	6450
Febrero	6300
Marzo	7100

Solución

2.9. Base

Con las funciones del paquete `base` de R.

```
df$Impuestos <- c(6450, 6300, 7100)
df
```

```
      Mes Ingresos Gastos Impuestos
1   Enero    45000  33400      6450
2 Febrero    41500  35400      6300
3   Marzo    51200  35600      7100
```

2.10. tidyverse

Con las funciones del paquete `dplyr` de `tidyverse`.

```r
df <- df |> mutate(Impuestos = c(6450, 6300, 7100))
df
```

```
      Mes Ingresos Gastos Impuestos
1   Enero    45000  33400      6450
2 Febrero    41500  35400      6300
3   Marzo    51200  35600      7100
```

c. Añadir una nueva fila con los siguientes datos de Abril.

Mes	Ingresos	Gastos	Impuestos
Abril	49700	36300	6850

Solución

2.11. Base

Con las funciones del paquete `base` de R.

```r
df <- rbind(df, list(Mes = "Abril", Ingresos = 49700, Gastos = 36300, Impuestos =
df
```

```
      Mes Ingresos Gastos Impuestos
1   Enero    45000  33400      6450
2 Febrero    41500  35400      6300
3   Marzo    51200  35600      7100
4   Abril    49700  36300      6850
```

2.12. tidyverse

Con las funciones del paquete `dplyr` de `tidyverse`.

```r
df <- df |> add_row(Mes = "Abril", Ingresos = 49700, Gastos = 36300, Impuestos = 6
df
```

```
      Mes Ingresos Gastos Impuestos
1   Enero    45000  33400      6450
2 Febrero    41500  35400      6300
3   Marzo    51200  35600      7100
4   Abril    49700  36300      6850
```

d. Cambiar los ingresos de Marzo por 50400.

Solución

```
df[3, "Ingresos"] <- 50400
df
```

```
      Mes Ingresos Gastos Impuestos
1   Enero    45000  33400      6450
2 Febrero    41500  35400      6300
3   Marzo    50400  35600      7100
4   Abril    49700  36300      6850
```

e. Guardar el data frame en un fichero csv.

Solución

La función `write.csv(dataframe, "fichero.csv")` permite guardar el data frame `dataframe` en el fichero `fichero.csv`.

```
write.csv(df, "datos/ingresos_gastos.csv", row.names = FALSE)
```

Ejercicio 2.6. El fichero colesterol.csv contiene información de una muestra de pacientes donde se han medido la edad, el sexo, el peso, la altura y el nivel de colesterol, además de su nombre.

a. Crear un data frame con los datos de todos los pacientes del estudio a partir del fichero colesterol.csv y mostrar las primeras filas.

Solución

2.13. Base

Con las funciones del paquete `base` de R. La función `read.csv("fichero.csv")` permite leer un fichero csv y cargar los datos en un data frame. Y la función `head(dataframe)` permite mostrar las primeras filas del data frame `dataframe`.

```
df <- read.csv("https://aprendeconalf.es/estadistica-practicas-r/datos/colesterol.
head(df)
```

```
                    nombre edad sexo peso altura colesterol
1 José Luis Martínez Izquierdo   18    H   85   1.79        182
2              Rosa Díaz Díaz   32    M   65   1.73        232
3       Javier García Sánchez   24    H   NA   1.81        191
4         Carmen López Pinzón   35    M   65   1.70        200
5        Marisa López Collado   46    M   51   1.58        148
6           Antonio Ruiz Cruz   68    H   66   1.74        249
```

2.14. tidyverse

Con la función `read_csv` del paquete del paquete `readr` de `tidyverse`.

```
df <- read_csv("https://aprendeconalf.es/estadistica-practicas-r/datos/colesterol.
head(df)
```

```
# A tibble: 6 x 6
  nombre                         edad sexo   peso altura colesterol
  <chr>                         <dbl> <chr> <dbl>  <dbl>      <dbl>
1 José Luis Martínez Izquierdo     18 H        85   1.79        182
2 Rosa Díaz Díaz                   32 M        65   1.73        232
3 Javier García Sánchez            24 H        NA   1.81        191
4 Carmen López Pinzón              35 M        65   1.7         200
5 Marisa López Collado             46 M        51   1.58        148
6 Antonio Ruiz Cruz                68 H        66   1.74        249
```

b. Mostrar las variables del data frame.

Solución

2.15. Base

Con las funciones del paquete `base` de R.

```
colnames(df)
```

```
[1] "nombre"     "edad"       "sexo"       "peso"       "altura"
[6] "colesterol"
```

2.16. tidyverse

Con la función `glimpse` del paquete `dplyr` de `tidyverse`. Esta función muestra las columnas del data frame en filas, de manera que permite ver todas las columnas de un data frame cuando este tiene muchas columnas.

```
glimpse(df)
```

```
Rows: 14
Columns: 6
$ nombre     <chr> "José Luis Martínez Izquierdo", "Rosa Díaz Díaz", "Javier G~
$ edad       <dbl> 18, 32, 24, 35, 46, 68, 51, 22, 35, 46, 53, 58, 27, 20
$ sexo       <chr> "H", "M", "H", "M", "M", "H", "H", "M", "H", "H", "M", "H",~
$ peso       <dbl> 85, 65, NA, 65, 51, 66, 62, 60, 90, 75, 55, 78, 109, 61
$ altura     <dbl> 1.79, 1.73, 1.81, 1.70, 1.58, 1.74, 1.72, 1.66, 1.94, 1.85,~
$ colesterol <dbl> 182, 232, 191, 200, 148, 249, 276, NA, 241, 280, 262, 198, ~
```

c. Mostrar el número de filas del data frame, que corresponde al número de pacientes.

Solución

La función `nrow(dataframe)` permite mostrar el número de filas del data frame `dataframe`.

```
nrow(df)
```

```
[1] 14
```

d. Mostrar 5 filas aleatorias del data frame.

Solución

2.17. Base

La función `sample(vector, n)` permite seleccionar `n` elementos aleatorios de `vector`. El muestreo es sin reemplazamiento.

```
df[sample(nrow(df), 5), ]
```

```
# A tibble: 5 x 6
  nombre                      edad sexo   peso altura colesterol
  <chr>                      <dbl> <chr> <dbl>  <dbl>       <dbl>
1 Carmen López Pinzón           35 M        65   1.7           200
2 Antonio Ruiz Cruz             68 H        66   1.74          249
3 Antonio Fernández Ocaña       51 H        62   1.72          276
4 Marisa López Collado          46 M        51   1.58          148
5 José María de la Guía Sanz    58 H        78   1.87          198
```

2.18. tidyverse

La función `sample_n(dataframe, n)` del paquete `dplyr` de `tidyverse` permite seleccionar `n` filas aleatorias del data frame `dataframe`.

```
df |> sample_n(5)
```

```
# A tibble: 5 x 6
  nombre                          edad sexo   peso altura colesterol
  <chr>                          <dbl> <chr> <dbl>  <dbl>       <dbl>
1 Pilar Martín González             22 M        60   1.66          NA
2 Macarena Álvarez Luna             53 M        55   1.62         262
3 Santiago Reillo Manzano           46 H        75   1.85         280
4 Miguel Angel Cuadrado Gutiérrez   27 H       109   1.98         210
5 Antonio Ruiz Cruz                 68 H        66   1.74         249
```

e. Obtener los datos de colesterol de los pacientes.

Solución

2.19. Base

Con las funciones del paquete `base` de R.

```
df$colesterol
```

```
 [1] 182 232 191 200 148 249 276  NA 241 280 262 198 210 194
```

2.20. tidyverse

Con la función `select` del paquete `dplyr` de `tidyverse`.

```
df |> select(colesterol)
```

```
# A tibble: 14 x 1
   colesterol
        <dbl>
 1        182
 2        232
 3        191
 4        200
 5        148
 6        249
 7        276
 8         NA
 9        241
10        280
11        262
12        198
13        210
14        194
```

f. Obtener los datos del quinto paciente.

Solución

2.21. Base

Con las funciones del paquete `base` de R.

```
df[5, ]
```

```
# A tibble: 1 x 6
```

```
  nombre                 edad sexo   peso altura colesterol
  <chr>                 <dbl> <chr> <dbl>  <dbl>      <dbl>
1 Marisa López Collado    46 M        51   1.58        148
```

2.22. tidyverse

Con la función `slice` del paquete `dplyr` de `tidyverse`.

```
df |> slice(5)
```

```
# A tibble: 1 x 6
  nombre                 edad sexo   peso altura colesterol
  <chr>                 <dbl> <chr> <dbl>  <dbl>      <dbl>
1 Marisa López Collado    46 M        51   1.58        148
```

2.23. Ejercicios Propuestos

Ejercicio 2.7. La siguiente tabla contiene las notas de un grupo de alumnos en dos asignaturas.

Alumno	Grupo	Física	Química
Juan	A	7.0	6.7
María	B	3.5	5.0
Pedro	B	6.0	7.1
Ana	A	5.2	4.5
Luis	A	4.5	NA
Sara	B	9.0	9.2

a. Crear un vector con los nombres de los alumnos.

b. Crear un factor el grupo.

c. Crear un vector con las notas de Física y otro con las notas de Química.

d. Crear un vector con la nota media de las dos asignaturas.

e. Crear un vector booleano con los alumnos que han aprobado el curso. Para aprobar el curso, la nota media de las dos asignaturas debe ser mayor o igual a 5.

f. Crear un vector con los nombres de los alumnos que han aprobado el curso.

g. Crear un data frame con los nombres de los alumnos, sus notas y su media reutilizando los vectores anteriores.

h. Guardar el data frame en un fichero csv.

3 Preprocesamiento de datos

Esta práctica contiene ejercicios que muestran como preprocesar un conjunto de datos en R. El preprocesamiento de datos es una tarea fundamental en el análisis de datos que consiste en la limpieza, transformación y preparación de los datos para su análisis. El preprocesamiento de datos incluye tareas como

- Limpieza de datos.
- Imputación de valores perdidos.
- Recodificación de variables.
- Creación de nuevas variables.
- Transformación de variables.
- Selección de variables.
- Fusión de datos.
- Reestructuración del conjunto de datos.

3.1. Ejercicios Resueltos

Para la realización de esta práctica se requieren los siguientes paquetes.

```
library(tidyverse)
# Incluye los siguientes paquetes:
# - readr: para la lectura de ficheros csv.
# - dplyr: para el preprocesamiento y manipulación de datos.
# - lubridate: para el procesamiento de fechas.
library(knitr) # Para el formateo de tablas.
```

Ejercicio 3.1. La siguiente tabla contiene los ingresos y gastos de una empresa durante el primer trimestre del año.

Mes	Ingresos	Gastos	Impuestos
Enero	45000	33400	6450
Febrero	41500	35400	6300
Marzo	51200	35600	7100
Abril	49700	36300	6850

a. Crear un data frame con los datos de la tabla.

Solución

```
df <- data.frame(
    Mes = c("Enero", "Febrero", "Marzo", "Abril"),
    Ingresos = c(45000, 41500, 51200, 49700),
    Gastos = c(33400, 35400, 35600, 36300),
    Impuestos = c(6450, 6300, 7100, 6850)
    )
df
```

```
      Mes Ingresos Gastos Impuestos
1   Enero    45000  33400      6450
2 Febrero    41500  35400      6300
3   Marzo    51200  35600      7100
4   Abril    49700  36300      6850
```

b. Crear una nueva columna con los beneficios de cada mes (ingresos - gastos - impuestos).

Solución

3.2. Base

Con las funciones del paquete `base` de R.

```
df$Beneficios <- df$Ingresos - df$Gastos - df$Impuestos
df
```

```
      Mes Ingresos Gastos Impuestos Beneficios
1   Enero    45000  33400      6450       5150
2 Febrero    41500  35400      6300       -200
3   Marzo    51200  35600      7100       8500
4   Abril    49700  36300      6850       6550
```

3.3. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`. La función `mutate` permite añadir nuevas columnas a un data frame mediante una fórmula puede hacer referencia a las columnas existentes.

```r
library(tidyverse)
df <- df |>
    mutate(Beneficios = Ingresos - Gastos - Impuestos)
df
```

```
      Mes Ingresos Gastos Impuestos Beneficios
1   Enero    45000  33400      6450       5150
2 Febrero    41500  35400      6300       -200
3   Marzo    51200  35600      7100       8500
4   Abril    49700  36300      6850       6550
```

c. Crear una nueva columna con el factor `Balance` con dos posibles categorías: `positivo` si ha habido beneficios y `negativo` si ha habido pérdidas.

Solución

3.4. Base

Con la función `cut` del paquete `base` de R. La función `cut(vector, breaks, labels)` divide el vector `vector` en intervalos delimitados por los elementos del vector `breaks` y crea un factor asignando a cada intervalo una etiqueta del vector `labels`.

```r
df$Balance <- cut(df$Beneficios, breaks = c(-Inf, 0, Inf), labels = c("negativo",
df
```

```
      Mes Ingresos Gastos Impuestos Beneficios  Balance
1   Enero    45000  33400      6450       5150 positivo
2 Febrero    41500  35400      6300       -200 negativo
3   Marzo    51200  35600      7100       8500 positivo
4   Abril    49700  36300      6850       6550 positivo
```

3.5. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`.

```r
df <- df |>
    mutate(Balance = cut(Beneficios, breaks = c(-Inf, 0, Inf), labels = c("negativ
df
```

```
      Mes Ingresos Gastos Impuestos Beneficios  Balance
1   Enero    45000  33400      6450       5150 positivo
2 Febrero    41500  35400      6300       -200 negativo
3   Marzo    51200  35600      7100       8500 positivo
4   Abril    49700  36300      6850       6550 positivo
```

d. Filtrar el conjunto de datos para quedarse con los nombres de los meses y los beneficios de los meses con balance positivo.

Solución

3.6. Base

Con las funciones del paquete `base` de R.

```
df[df$Balance == "positivo", c("Mes", "Beneficios")]
```

```
    Mes Beneficios
1 Enero       5150
3 Marzo       8500
4 Abril       6550
```

3.7. tidyverse

Con las funciones `filter` y `select` del paquete `dplyr` de `tidyverse`. La función `filter` permite seleccionar las filas de un data frame que cumplen una condición. La función `select` permite seleccionar las columnas de un data frame.

```
df |>
    filter(Balance == "positivo") |>
    select(Mes, Beneficios)
```

```
    Mes Beneficios
1 Enero       5150
2 Marzo       8500
3 Abril       6550
```

Ejercicio 3.2. El fichero `colesterol.csv` contiene información de una muestra de pacientes donde se han medido la edad, el sexo, el peso, la altura y el nivel de colesterol, además de su nombre.

a. Crear un data frame con los datos de todos los pacientes del estudio a partir del fichero `colesterol.csv`.

Solución

3.8. Base

Con las funciones del paquete `base` de R.

```r
df <- read.csv("https://aprendeconalf.es/estadistica-practicas-r/datos/colesterol.
head(df)
```

```
                      nombre edad sexo peso altura colesterol
1 José Luis Martínez Izquierdo   18    H   85   1.79         182
2               Rosa Díaz Díaz   32    M   65   1.73         232
3        Javier García Sánchez   24    H   NA   1.81         191
4          Carmen López Pinzón   35    M   65   1.70         200
5         Marisa López Collado   46    M   51   1.58         148
6            Antonio Ruiz Cruz   68    H   66   1.74         249
```

3.9. tidyverse

Con la función `read_csv` del paquete del paquete `readr` de `tidyverse`.

```r
df <- read_csv("https://aprendeconalf.es/estadistica-practicas-r/datos/colesterol.
head(df)
```

```
# A tibble: 6 x 6
  nombre                        edad sexo   peso altura colesterol
  <chr>                        <dbl> <chr> <dbl>  <dbl>       <dbl>
1 José Luis Martínez Izquierdo    18 H        85   1.79         182
2 Rosa Díaz Díaz                  32 M        65   1.73         232
3 Javier García Sánchez           24 H        NA   1.81         191
4 Carmen López Pinzón             35 M        65   1.7          200
5 Marisa López Collado            46 M        51   1.58         148
6 Antonio Ruiz Cruz               68 H        66   1.74         249
```

b. Crear una nueva columna con el índice de masa corporal, usando la siguiente fórmula

$$\text{IMC} = \frac{\text{Peso (kg)}}{\text{Altura (cm)}^2}$$

Solución

3.10. Base

Con las funciones del paquete `base` de R.

```r
df$imc <- round(df$peso/df$altura^2)
head(df)
```

```
# A tibble: 6 x 7
```

```
  nombre                        edad sexo   peso altura colesterol   imc
  <chr>                        <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl>
1 José Luis Martínez Izquierdo    18 H        85   1.79         182    27
2 Rosa Díaz Díaz                  32 M        65   1.73         232    22
3 Javier García Sánchez           24 H        NA   1.81         191    NA
4 Carmen López Pinzón             35 M        65   1.7          200    22
5 Marisa López Collado            46 M        51   1.58         148    20
6 Antonio Ruiz Cruz               68 H        66   1.74         249    22
```

3.11. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`.

```
df <- df |> mutate(imc = round(peso/altura^2))
head(df)
```

```
# A tibble: 6 x 7
  nombre                        edad sexo   peso altura colesterol   imc
  <chr>                        <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl>
1 José Luis Martínez Izquierdo    18 H        85   1.79         182    27
2 Rosa Díaz Díaz                  32 M        65   1.73         232    22
3 Javier García Sánchez           24 H        NA   1.81         191    NA
4 Carmen López Pinzón             35 M        65   1.7          200    22
5 Marisa López Collado            46 M        51   1.58         148    20
6 Antonio Ruiz Cruz               68 H        66   1.74         249    22
```

c. Crear una nueva columna con la variable `obesidad` recodificando la columna `imc` en las siguientes categorías.

Rango IMC	Categoría
Menor de 18.5	Bajo peso
De 18.5 a 24.5	Saludable
De 24.5 a 30	Sobrepeso
Mayor de 30	Obeso

Solución

3.12. Base

Con la función `cut` del paquete `base` de R.

```
df$Obesidad <- cut(df$imc, breaks = c(0, 18.5, 24.5, 30, Inf), labels = c("Bajo pe
head(df)
```

```
# A tibble: 6 x 8
  nombre                        edad sexo   peso altura colesterol   imc Obesidad
  <chr>                        <dbl> <chr> <dbl>  <dbl>      <dbl> <dbl> <fct>
1 José Luis Martínez Izquier~    18 H        85   1.79        182    27 Sobrepe~
2 Rosa Díaz Díaz                 32 M        65   1.73        232    22 Saludab~
3 Javier García Sánchez          24 H        NA   1.81        191    NA <NA>
4 Carmen López Pinzón            35 M        65   1.7         200    22 Saludab~
5 Marisa López Collado           46 M        51   1.58        148    20 Saludab~
6 Antonio Ruiz Cruz              68 H        66   1.74        249    22 Saludab~
```

3.13. tidyverse

Con las funciones del paquete `dplyr` de `tidyverse`.

```
df <- df |>
    mutate(Obesidad = cut(imc, breaks = c(0, 18.5, 24.5, 30, Inf), labels = c("Baj
head(df)
```

```
# A tibble: 6 x 8
  nombre                        edad sexo   peso altura colesterol   imc Obesidad
  <chr>                        <dbl> <chr> <dbl>  <dbl>      <dbl> <dbl> <fct>
1 José Luis Martínez Izquier~    18 H        85   1.79        182    27 Sobrepe~
2 Rosa Díaz Díaz                 32 M        65   1.73        232    22 Saludab~
3 Javier García Sánchez          24 H        NA   1.81        191    NA <NA>
4 Carmen López Pinzón            35 M        65   1.7         200    22 Saludab~
5 Marisa López Collado           46 M        51   1.58        148    20 Saludab~
6 Antonio Ruiz Cruz              68 H        66   1.74        249    22 Saludab~
```

d. Seleccionar las columnas `nombre`, `sexo` y `edad`.

Solución

3.14. Base

Con las funciones del paquete `base` de R.

```
df[, c("nombre", "sexo", "edad")]
```

```
# A tibble: 14 x 3
   nombre                       sexo   edad
   <chr>                        <chr> <dbl>
 1 José Luis Martínez Izquierdo H        18
 2 Rosa Díaz Díaz               M        32
 3 Javier García Sánchez        H        24
 4 Carmen López Pinzón          M        35
```

```
 5 Marisa López Collado             M        46
 6 Antonio Ruiz Cruz                H        68
 7 Antonio Fernández Ocaña          H        51
 8 Pilar Martín González            M        22
 9 Pedro Gálvez Tenorio             H        35
10 Santiago Reillo Manzano          H        46
11 Macarena Álvarez Luna            M        53
12 José María de la Guía Sanz       H        58
13 Miguel Angel Cuadrado Gutiérrez H         27
14 Carolina Rubio Moreno            M        20
```

3.15. tidyverse

Con la función `select` del paquete `dplyr` de `tidyverse`.

```
df |> select(nombre, sexo, edad)
```

```
# A tibble: 14 x 3
   nombre                          sexo   edad
   <chr>                           <chr> <dbl>
 1 José Luis Martínez Izquierdo    H        18
 2 Rosa Díaz Díaz                  M        32
 3 Javier García Sánchez           H        24
 4 Carmen López Pinzón             M        35
 5 Marisa López Collado            M        46
 6 Antonio Ruiz Cruz               H        68
 7 Antonio Fernández Ocaña         H        51
 8 Pilar Martín González           M        22
 9 Pedro Gálvez Tenorio            H        35
10 Santiago Reillo Manzano         H        46
11 Macarena Álvarez Luna           M        53
12 José María de la Guía Sanz      H        58
13 Miguel Angel Cuadrado Gutiérrez H        27
14 Carolina Rubio Moreno           M        20
```

e. Anonimizar los datos eliminando la columna `nombre`.

Solución

3.16. Base

Con las funciones del paquete `base` de R.

```
df[, -1]
```

```
# A tibble: 14 x 7
    edad sexo   peso altura colesterol   imc Obesidad
   <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1    18 H        85   1.79         182    27 Sobrepeso
 2    32 M        65   1.73         232    22 Saludable
 3    24 H        NA   1.81         191    NA <NA>
 4    35 M        65   1.7          200    22 Saludable
 5    46 M        51   1.58         148    20 Saludable
 6    68 H        66   1.74         249    22 Saludable
 7    51 H        62   1.72         276    21 Saludable
 8    22 M        60   1.66          NA    22 Saludable
 9    35 H        90   1.94         241    24 Saludable
10    46 H        75   1.85         280    22 Saludable
11    53 M        55   1.62         262    21 Saludable
12    58 H        78   1.87         198    22 Saludable
13    27 H       109   1.98         210    28 Sobrepeso
14    20 M        61   1.77         194    19 Saludable
```

3.17. tidyverse

Con la función `select` del paquete `dplyr` de `tidyverse`.

```
df |> select(-nombre)
```

```
# A tibble: 14 x 7
    edad sexo   peso altura colesterol   imc Obesidad
   <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1    18 H        85   1.79         182    27 Sobrepeso
 2    32 M        65   1.73         232    22 Saludable
 3    24 H        NA   1.81         191    NA <NA>
 4    35 M        65   1.7          200    22 Saludable
 5    46 M        51   1.58         148    20 Saludable
 6    68 H        66   1.74         249    22 Saludable
 7    51 H        62   1.72         276    21 Saludable
 8    22 M        60   1.66          NA    22 Saludable
 9    35 H        90   1.94         241    24 Saludable
10    46 H        75   1.85         280    22 Saludable
11    53 M        55   1.62         262    21 Saludable
12    58 H        78   1.87         198    22 Saludable
13    27 H       109   1.98         210    28 Sobrepeso
14    20 M        61   1.77         194    19 Saludable
```

f. Reordenar las columnas poniendo la columna `sexo` antes que la columna `edad`.

Solución

3.18. Base

Con las funciones del paquete `base` de R.

```
df[, c(1, 3, 2, 4, 5, 6)]
```

```
# A tibble: 14 x 6
   nombre                          sexo   edad  peso altura colesterol
   <chr>                           <chr> <dbl> <dbl>  <dbl>       <dbl>
 1 José Luis Martínez Izquierdo    H        18    85   1.79         182
 2 Rosa Díaz Díaz                  M        32    65   1.73         232
 3 Javier García Sánchez           H        24    NA   1.81         191
 4 Carmen López Pinzón             M        35    65   1.7          200
 5 Marisa López Collado            M        46    51   1.58         148
 6 Antonio Ruiz Cruz               H        68    66   1.74         249
 7 Antonio Fernández Ocaña         H        51    62   1.72         276
 8 Pilar Martín González           M        22    60   1.66          NA
 9 Pedro Gálvez Tenorio            H        35    90   1.94         241
10 Santiago Reillo Manzano         H        46    75   1.85         280
11 Macarena Álvarez Luna           M        53    55   1.62         262
12 José María de la Guía Sanz      H        58    78   1.87         198
13 Miguel Angel Cuadrado Gutiérrez H        27   109   1.98         210
14 Carolina Rubio Moreno           M        20    61   1.77         194
```

3.19. tidyverse

Con la función `select` del paquete `dplyr` de `tidyverse`.

```
df |> select(nombre, sexo, edad, everything())
```

```
# A tibble: 14 x 8
   nombre                    sexo   edad  peso altura colesterol   imc Obesidad
   <chr>                     <chr> <dbl> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1 José Luis Martínez Izquie~ H        18    85   1.79         182    27 Sobrepe~
 2 Rosa Díaz Díaz             M        32    65   1.73         232    22 Saludab~
 3 Javier García Sánchez      H        24    NA   1.81         191    NA <NA>
 4 Carmen López Pinzón        M        35    65   1.7          200    22 Saludab~
 5 Marisa López Collado       M        46    51   1.58         148    20 Saludab~
 6 Antonio Ruiz Cruz          H        68    66   1.74         249    22 Saludab~
 7 Antonio Fernández Ocaña    H        51    62   1.72         276    21 Saludab~
 8 Pilar Martín González      M        22    60   1.66          NA    22 Saludab~
```

```
 9 Pedro Gálvez Tenorio       H        35    90   1.94         241    24 Saludab~
10 Santiago Reillo Manzano    H        46    75   1.85         280    22 Saludab~
11 Macarena Álvarez Luna      M        53    55   1.62         262    21 Saludab~
12 José María de la Guía Sanz H        58    78   1.87         198    22 Saludab~
13 Miguel Angel Cuadrado Gut~ H        27   109   1.98         210    28 Sobrepe~
14 Carolina Rubio Moreno      M        20    61   1.77         194    19 Saludab~
```

g. Filtrar el data frame para quedarse con las mujeres.

Solución

3.20. Base

Con las funciones del paquete `base` de R.

```
df[df$sexo == "M", ]
```

```
# A tibble: 6 x 8
  nombre                 edad sexo   peso altura colesterol   imc Obesidad
  <chr>                 <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
1 Rosa Díaz Díaz           32 M        65   1.73         232    22 Saludable
2 Carmen López Pinzón      35 M        65   1.7          200    22 Saludable
3 Marisa López Collado     46 M        51   1.58         148    20 Saludable
4 Pilar Martín González    22 M        60   1.66          NA    22 Saludable
5 Macarena Álvarez Luna    53 M        55   1.62         262    21 Saludable
6 Carolina Rubio Moreno    20 M        61   1.77         194    19 Saludable
```

3.21. tidyverse

Con la función filter del paquete `dplyr` de `tidyverse`.

```
df |> filter(sexo == "M")
```

```
# A tibble: 6 x 8
  nombre                 edad sexo   peso altura colesterol   imc Obesidad
  <chr>                 <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
1 Rosa Díaz Díaz           32 M        65   1.73         232    22 Saludable
2 Carmen López Pinzón      35 M        65   1.7          200    22 Saludable
3 Marisa López Collado     46 M        51   1.58         148    20 Saludable
4 Pilar Martín González    22 M        60   1.66          NA    22 Saludable
5 Macarena Álvarez Luna    53 M        55   1.62         262    21 Saludable
6 Carolina Rubio Moreno    20 M        61   1.77         194    19 Saludable
```

h. Filtrar el data frame para quedarse con los hombres mayores de 30 años.

Solución

3.22. Base

Con las funciones del paquete `base` de R.

```
df[df$sexo == "H" & df$edad > 30, ]
```

```
# A tibble: 5 x 8
  nombre                    edad sexo   peso altura colesterol   imc Obesidad
  <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
1 Antonio Ruiz Cruz           68 H        66   1.74         249    22 Saludable
2 Antonio Fernández Ocaña     51 H        62   1.72         276    21 Saludable
3 Pedro Gálvez Tenorio        35 H        90   1.94         241    24 Saludable
4 Santiago Reillo Manzano     46 H        75   1.85         280    22 Saludable
5 José María de la Guía Sanz  58 H        78   1.87         198    22 Saludable
```

3.23. tidyverse

Con la función `filter` paquete `dplyr` de `tidyverse`.

```
df |> filter( sexo == "H" & edad > 30)
```

```
# A tibble: 5 x 8
  nombre                    edad sexo   peso altura colesterol   imc Obesidad
  <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
1 Antonio Ruiz Cruz           68 H        66   1.74         249    22 Saludable
2 Antonio Fernández Ocaña     51 H        62   1.72         276    21 Saludable
3 Pedro Gálvez Tenorio        35 H        90   1.94         241    24 Saludable
4 Santiago Reillo Manzano     46 H        75   1.85         280    22 Saludable
5 José María de la Guía Sanz  58 H        78   1.87         198    22 Saludable
```

i. Filtrar el data frame para quedarse con las filas sin valores perdidos.

Solución

3.24. Base

Con la función `na.omit` del paquete `base` de R. La función `na.omit` elimina las filas con valores perdidos.

```
na.omit(df)
```

```
# A tibble: 12 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
```

```
 1 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 2 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
 3 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
 4 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
 5 Antonio Ruiz Cruz             68 H        66   1.74        249    22 Saludab~
 6 Antonio Fernández Ocaña       51 H        62   1.72        276    21 Saludab~
 7 Pedro Gálvez Tenorio          35 H        90   1.94        241    24 Saludab~
 8 Santiago Reillo Manzano       46 H        75   1.85        280    22 Saludab~
 9 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
10 José María de la Guía Sanz    58 H        78   1.87        198    22 Saludab~
11 Miguel Angel Cuadrado Gut~    27 H       109   1.98        210    28 Sobrepe~
12 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

3.25. tidyverse

Con la función `drop_na` del paquete `tidyr` de `tidyverse`.

```
df |> drop_na()
```

```
# A tibble: 12 x 8
   nombre                     edad sexo   peso altura colesterol   imc Obesidad
   <chr>                     <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 2 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
 3 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
 4 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
 5 Antonio Ruiz Cruz             68 H        66   1.74        249    22 Saludab~
 6 Antonio Fernández Ocaña       51 H        62   1.72        276    21 Saludab~
 7 Pedro Gálvez Tenorio          35 H        90   1.94        241    24 Saludab~
 8 Santiago Reillo Manzano       46 H        75   1.85        280    22 Saludab~
 9 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
10 José María de la Guía Sanz    58 H        78   1.87        198    22 Saludab~
11 Miguel Angel Cuadrado Gut~    27 H       109   1.98        210    28 Sobrepe~
12 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

j. Filtrar el data frame para eliminar las filas con datos perdidos en la columna `colesterol`.

Solución

3.26. Base

Con las funciones del paquete `base` de R. La función `is.na` devuelve `TRUE` cuando se aplica a un valor perdido `NA`. Cuando se aplica a un vector devuelve

un vector lógico con `TRUE` en las posiciones con valores perdidos y `FALSE` en las posiciones con valores no perdidos.

```
df[!is.na(df$colesterol), ]
```

```
# A tibble: 13 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>      <dbl> <dbl> <fct>
 1 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 2 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
 3 Javier García Sánchez         24 H        NA   1.81        191    NA <NA>
 4 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
 5 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
 6 Antonio Ruiz Cruz             68 H        66   1.74        249    22 Saludab~
 7 Antonio Fernández Ocaña       51 H        62   1.72        276    21 Saludab~
 8 Pedro Gálvez Tenorio          35 H        90   1.94        241    24 Saludab~
 9 Santiago Reillo Manzano       46 H        75   1.85        280    22 Saludab~
10 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
11 José María de la Guía Sanz    58 H        78   1.87        198    22 Saludab~
12 Miguel Angel Cuadrado Gut~    27 H       109   1.98        210    28 Sobrepe~
13 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

3.27. tidyverse

Con la función filter del paquete `dplyr` de `tidyverse`.

```
df |> filter(!is.na(colesterol))
```

```
# A tibble: 13 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>      <dbl> <dbl> <fct>
 1 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 2 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
 3 Javier García Sánchez         24 H        NA   1.81        191    NA <NA>
 4 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
 5 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
 6 Antonio Ruiz Cruz             68 H        66   1.74        249    22 Saludab~
 7 Antonio Fernández Ocaña       51 H        62   1.72        276    21 Saludab~
 8 Pedro Gálvez Tenorio          35 H        90   1.94        241    24 Saludab~
 9 Santiago Reillo Manzano       46 H        75   1.85        280    22 Saludab~
10 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
11 José María de la Guía Sanz    58 H        78   1.87        198    22 Saludab~
12 Miguel Angel Cuadrado Gut~    27 H       109   1.98        210    28 Sobrepe~
13 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

k. Imputar los valores perdidos en la columna `colesterol` con la media de los valores no perdidos.

Solución

3.28. Base

Con la función `mean` del paquete `base` de R. La función `mean` calcula la media de un vector. Para que no se tengan en cuenta los valores perdidos se puede usar el argumento `na.rm = TRUE`.

```
media_colesterol <- mean(df$colesterol, na.rm = TRUE)
df$colesterol[is.na(df$colesterol)] <- media_colesterol
df
```

```
# A tibble: 14 x 8
   nombre                      edad sexo   peso altura colesterol   imc Obesidad
   <chr>                      <dbl> <chr> <dbl>  <dbl>      <dbl> <dbl> <fct>
 1 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 2 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
 3 Javier García Sánchez         24 H        NA   1.81        191    NA <NA>
 4 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
 5 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
 6 Antonio Ruiz Cruz             68 H        66   1.74        249    22 Saludab~
 7 Antonio Fernández Ocaña       51 H        62   1.72        276    21 Saludab~
 8 Pilar Martín González         22 M        60   1.66        220.   22 Saludab~
 9 Pedro Gálvez Tenorio          35 H        90   1.94        241    24 Saludab~
10 Santiago Reillo Manzano       46 H        75   1.85        280    22 Saludab~
11 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
12 José María de la Guía Sanz    58 H        78   1.87        198    22 Saludab~
13 Miguel Angel Cuadrado Gut~    27 H       109   1.98        210    28 Sobrepe~
14 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

3.29. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`. La función `ifelse` permite asignar un valor a un vector en función de una condición.

```
df <- df |>
    mutate(colesterol = ifelse(is.na(colesterol), mean(colesterol, na.rm = TRUE),
df
```

```
# A tibble: 14 x 8
   nombre                      edad sexo   peso altura colesterol   imc Obesidad
   <chr>                      <dbl> <chr> <dbl>  <dbl>      <dbl> <dbl> <fct>
```

```
 1 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 2 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
 3 Javier García Sánchez         24 H        NA   1.81        191    NA <NA>
 4 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
 5 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
 6 Antonio Ruiz Cruz             68 H        66   1.74        249    22 Saludab~
 7 Antonio Fernández Ocaña       51 H        62   1.72        276    21 Saludab~
 8 Pilar Martín González         22 M        60   1.66        220.   22 Saludab~
 9 Pedro Gálvez Tenorio          35 H        90   1.94        241    24 Saludab~
10 Santiago Reillo Manzano       46 H        75   1.85        280    22 Saludab~
11 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
12 José María de la Guía Sanz    58 H        78   1.87        198    22 Saludab~
13 Miguel Angel Cuadrado Gut~    27 H       109   1.98        210    28 Sobrepe~
14 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

l. Ordenar el data frame según la columna `nombre`.

Solución

3.30. Base

Con la función `order` del paquete `base` de R. La función `order` devuelve un vector con los índices de las filas ordenadas de menor a mayor.

```
df[order(df$nombre), ]
```

```
# A tibble: 14 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1 Antonio Fernández Ocaña     51 H        62   1.72        276    21 Saludab~
 2 Antonio Ruiz Cruz           68 H        66   1.74        249    22 Saludab~
 3 Carmen López Pinzón         35 M        65   1.7         200    22 Saludab~
 4 Carolina Rubio Moreno       20 M        61   1.77        194    19 Saludab~
 5 Javier García Sánchez       24 H        NA   1.81        191    NA <NA>
 6 José Luis Martínez Izquie~  18 H        85   1.79        182    27 Sobrepe~
 7 José María de la Guía Sanz  58 H        78   1.87        198    22 Saludab~
 8 Macarena Álvarez Luna       53 M        55   1.62        262    21 Saludab~
 9 Marisa López Collado        46 M        51   1.58        148    20 Saludab~
10 Miguel Angel Cuadrado Gut~  27 H       109   1.98        210    28 Sobrepe~
11 Pedro Gálvez Tenorio        35 H        90   1.94        241    24 Saludab~
12 Pilar Martín González       22 M        60   1.66        220.   22 Saludab~
13 Rosa Díaz Díaz              32 M        65   1.73        232    22 Saludab~
14 Santiago Reillo Manzano     46 H        75   1.85        280    22 Saludab~
```

3.31. tidyverse

Con la función `arrange` del paquete `dplyr` de `tidyverse`.

```
df |> arrange(nombre)
```

```
# A tibble: 14 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1 Antonio Fernández Ocaña     51 H        62   1.72         276    21 Saludab~
 2 Antonio Ruiz Cruz           68 H        66   1.74         249    22 Saludab~
 3 Carmen López Pinzón         35 M        65   1.7          200    22 Saludab~
 4 Carolina Rubio Moreno       20 M        61   1.77         194    19 Saludab~
 5 Javier García Sánchez       24 H        NA   1.81         191    NA <NA>
 6 José Luis Martínez Izquie~  18 H        85   1.79         182    27 Sobrepe~
 7 José María de la Guía Sanz  58 H        78   1.87         198    22 Saludab~
 8 Macarena Álvarez Luna       53 M        55   1.62         262    21 Saludab~
 9 Marisa López Collado        46 M        51   1.58         148    20 Saludab~
10 Miguel Angel Cuadrado Gut~  27 H       109   1.98         210    28 Sobrepe~
11 Pedro Gálvez Tenorio        35 H        90   1.94         241    24 Saludab~
12 Pilar Martín González       22 M        60   1.66         220.   22 Saludab~
13 Rosa Díaz Díaz              32 M        65   1.73         232    22 Saludab~
14 Santiago Reillo Manzano     46 H        75   1.85         280    22 Saludab~
```

m. Ordenar el data frame ascendentemente por la columna `sexo` y descendentemente por la columna `edad`.

Solución

3.32. Base

Con las funciones del paquete `base` de R.

```
df[order(df$sexo, -df$edad), ]
```

```
# A tibble: 14 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1 Antonio Ruiz Cruz           68 H        66   1.74         249    22 Saludab~
 2 José María de la Guía Sanz  58 H        78   1.87         198    22 Saludab~
 3 Antonio Fernández Ocaña     51 H        62   1.72         276    21 Saludab~
 4 Santiago Reillo Manzano     46 H        75   1.85         280    22 Saludab~
 5 Pedro Gálvez Tenorio        35 H        90   1.94         241    24 Saludab~
 6 Miguel Angel Cuadrado Gut~  27 H       109   1.98         210    28 Sobrepe~
 7 Javier García Sánchez       24 H        NA   1.81         191    NA <NA>
```

```
 8 José Luis Martínez Izquie~    18 H        85   1.79        182    27 Sobrepe~
 9 Macarena Álvarez Luna         53 M        55   1.62        262    21 Saludab~
10 Marisa López Collado          46 M        51   1.58        148    20 Saludab~
11 Carmen López Pinzón           35 M        65   1.7         200    22 Saludab~
12 Rosa Díaz Díaz                32 M        65   1.73        232    22 Saludab~
13 Pilar Martín González         22 M        60   1.66        220.   22 Saludab~
14 Carolina Rubio Moreno         20 M        61   1.77        194    19 Saludab~
```

3.33. tidyverse

Con la función `arrange` del paquete `dplyr` de `tidyverse`. Para que la ordenación sea descendente con respecto a una variable se tiene que usar la función `desc` sobre la variable.

```
df |>
    arrange(sexo, desc(edad))
```

```
# A tibble: 14 x 8
   nombre                    edad sexo   peso altura colesterol   imc Obesidad
   <chr>                    <dbl> <chr> <dbl>  <dbl>       <dbl> <dbl> <fct>
 1 Antonio Ruiz Cruz           68 H        66   1.74         249    22 Saludab~
 2 José María de la Guía Sanz  58 H        78   1.87         198    22 Saludab~
 3 Antonio Fernández Ocaña     51 H        62   1.72         276    21 Saludab~
 4 Santiago Reillo Manzano     46 H        75   1.85         280    22 Saludab~
 5 Pedro Gálvez Tenorio        35 H        90   1.94         241    24 Saludab~
 6 Miguel Angel Cuadrado Gut~  27 H       109   1.98         210    28 Sobrepe~
 7 Javier García Sánchez       24 H        NA   1.81         191    NA <NA>
 8 José Luis Martínez Izquie~  18 H        85   1.79         182    27 Sobrepe~
 9 Macarena Álvarez Luna       53 M        55   1.62         262    21 Saludab~
10 Marisa López Collado        46 M        51   1.58         148    20 Saludab~
11 Carmen López Pinzón         35 M        65   1.7          200    22 Saludab~
12 Rosa Díaz Díaz              32 M        65   1.73         232    22 Saludab~
13 Pilar Martín González       22 M        60   1.66         220.   22 Saludab~
14 Carolina Rubio Moreno       20 M        61   1.77         194    19 Saludab~
```

Ejercicio 3.3. El fichero notas-curso2.csv contiene las notas de las asignaturas de un curso en varios grupos de alumnos.

a. Crear un data frame con los datos del curso a partir del fichero notas-curso2.csv.

Solución

```
df <- read_csv("https://aprendeconalf.es/estadistica-practicas-r/datos/notas-curso
df
```

```
# A tibble: 120 x 9
   sexo   turno  grupo trabaja notaA notaB notaC notaD notaE
   <chr>  <chr>  <chr> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Mujer  Tarde  C     N         5.2   6.3   3.4   2.3   2
 2 Hombre Mañana A     N         5.7   5.7   4.2   3.5   2.7
 3 Hombre Mañana B     N         8.3   8.8   8.8   8     5.5
 4 Hombre Mañana B     N         6.1   6.8   4     3.5   2.2
 5 Hombre Mañana A     N         6.2   9     5     4.4   3.7
 6 Hombre Mañana A     S         8.6   8.9   9.5   8.4   3.9
 7 Mujer  Mañana A     N         6.7   7.9   5.6   4.8   4.2
 8 Mujer  Tarde  C     S         4.1   5.2   1.7   0.3   1
 9 Hombre Tarde  C     N         5     5     3.3   2.7   6
10 Hombre Tarde  C     N         5.3   6.3   4.8   3.6   2.3
# i 110 more rows
```

b. Convertir el data frame a formato largo.

Solución

Para convertir un data frame de formato ancho a largo se puede usar la función `pivot_longer` del paquete `tidyr` de `tidyverse`.

```
df_largo <- df |> pivot_longer(notaA:notaE, names_to = "Asignatura", values_to = "
df_largo
```

```
# A tibble: 600 x 6
   sexo   turno  grupo trabaja Asignatura  Nota
   <chr>  <chr>  <chr> <chr>   <chr>      <dbl>
 1 Mujer  Tarde  C     N       notaA        5.2
 2 Mujer  Tarde  C     N       notaB        6.3
 3 Mujer  Tarde  C     N       notaC        3.4
 4 Mujer  Tarde  C     N       notaD        2.3
 5 Mujer  Tarde  C     N       notaE        2
 6 Hombre Mañana A     N       notaA        5.7
 7 Hombre Mañana A     N       notaB        5.7
 8 Hombre Mañana A     N       notaC        4.2
 9 Hombre Mañana A     N       notaD        3.5
10 Hombre Mañana A     N       notaE        2.7
# i 590 more rows
```

c. Crear una nueva columna con la variable `calificación` que contenga las calificaciones de cada asignatura.

Solución

```
df_largo <- df_largo |>
    mutate(Califiación = cut(Nota, breaks = c(0, 4.99, 6.99, 8.99, 10), labels = c
df_largo
```

```
# A tibble: 600 x 7
   sexo   turno  grupo trabaja Asignatura  Nota Califiación
   <chr>  <chr>  <chr> <chr>   <chr>      <dbl> <fct>
 1 Mujer  Tarde  C     N       notaA        5.2 AP
 2 Mujer  Tarde  C     N       notaB        6.3 AP
 3 Mujer  Tarde  C     N       notaC        3.4 SS
 4 Mujer  Tarde  C     N       notaD        2.3 SS
 5 Mujer  Tarde  C     N       notaE        2   SS
 6 Hombre Mañana A     N       notaA        5.7 AP
 7 Hombre Mañana A     N       notaB        5.7 AP
 8 Hombre Mañana A     N       notaC        4.2 SS
 9 Hombre Mañana A     N       notaD        3.5 SS
10 Hombre Mañana A     N       notaE        2.7 SS
# i 590 more rows
```

d. Filtrar el conjunto de datos para obtener las asignaturas y las notas de las mujeres del grupo A, ordenadas de mayor a menor.

Solución

```
df_largo |>
    filter(sexo == "Mujer", grupo == "A") |>
    select(Asignatura, Nota) |>
    arrange(desc(Nota))
```

```
# A tibble: 75 x 2
   Asignatura  Nota
   <chr>      <dbl>
 1 notaB        9.2
 2 notaE        9.2
 3 notaB        8.8
 4 notaB        8.6
 5 notaB        8.6
 6 notaA        8.3
 7 notaB        8.2
```

```
 8 notaB        8.1
 9 notaA        8
10 notaB        8
# i 65 more rows
```

Ejercicio 3.4. Se ha diseñado un ensayo clínico aleatorizado, doble-ciego y controlado con placebo, para estudiar el efecto de dos alternativas terapéuticas en el control de la hipertensión arterial. Se han reclutado 100 pacientes hipertensos y estos han sido distribuidos aleatoriamente en tres grupos de tratamiento. A uno de los grupos (control) se le administró un placebo, a otro grupo se le administró un inhibidor de la enzima conversora de la angiotensina (IECA) y al otro un tratamiento combinado de un diurético y un Antagonista del Calcio. Las variables respuesta final fueron las presiones arteriales sistólica y diastólica.

Los datos con las claves de aleatorización han sido introducidos en una base de datos que reside en la central de aleatorización, mientras que los datos clínicos han sido archivados en dos archivos distintos, uno para cada uno de los dos centros participantes en el estudio.

Las variables almacenadas en estos archivos clínicos son las siguientes:

- CLAVE: Clave de aleatorización
- NOMBRE: Iniciales del paciente
- F_NACIM: Fecha de Nacimiento
- F_INCLUS: Fecha de inclusión
- SEXO: Sexo (0: Hombre 1: Mujer)
- ALTURA: Altura en cm.
- PESO: Peso en Kg.
- PAD_INI: Presión diastólica basal (inicial)
- PAD_FIN: Presión diastólica final
- PAS_INI: Presión sistólica basal (inicial)
- PAS_FIN: Presión sistólica final

El archivo de claves de aleatorización contiene sólo dos variables.

- CLAVE: Clave de aleatorización
- FARMACO: Fármaco administrado (0: Placebo, 1: IECA, 2:Ca Antagonista + diurético)

a. Crear un data frame con los datos de los pacientes del hospital A del fichero de Excel datos-hospital-a.xls.

Solución

```
library(readxl)
dfA <- read_excel("datos/hipertension/datos-hospital-a.xls")
head(dfA)
```

```
# A tibble: 6 x 12
  CLAVE NOMBRE F_NACIM             F_INCLUS             SEXO ALTURA  PESO ESTRES
  <dbl> <chr>  <dttm>              <dttm>              <dbl>  <dbl> <dbl>  <dbl>
1     1 SGL    1941-09-08 00:00:00 1998-07-13 00:00:00     1    165    78     42
2     2 JCZ    1957-07-10 00:00:00 1998-05-09 00:00:00     1    154    74     30
3     3 APZ    1967-08-18 00:00:00 2000-04-01 00:00:00     0    156    81     21
4     4 NDG    1956-05-08 00:00:00 1998-11-13 00:00:00     0    181    82     33
5     5 CLO    1958-11-02 00:00:00 1999-02-24 00:00:00     1    184    78     36
6     6 LFZ    1953-06-13 00:00:00 2000-03-16 00:00:00     0    179    80     22
# i 4 more variables: PAD_INI <dbl>, PAD_FIN <dbl>, PAS_INI <dbl>,
#   PAS_FIN <dbl>
```

b. Crear un data frame con los datos de los pacientes del hospital B del fichero csv datos-hospital-b.csv.

Solución

```
dfB <- read_csv("https://aprendeconalf.es/estadistica-practicas-r/datos/hipertensi
head(dfB)
```

```
# A tibble: 6 x 12
  CLAVE NOMBRE F_NACIM    F_INCLUS    SEXO ALTURA  PESO ESTRES PAD_INI PAD_FIN
  <dbl> <chr>  <date>     <date>     <dbl>  <dbl> <dbl>  <dbl>   <dbl>   <dbl>
1    11 VSH    1965-12-15 1999-12-06     0    170    59   32        90      82
2    12 SZS    1971-03-07 1999-02-13     1    154    61   20.2      92     102
3    13 JSS    1964-01-03 1998-10-31     1    162    49   30        86      94
4    14 BMH    1941-08-16 1999-09-16     0    162    77   26        93      77
5    15 DGM    1969-01-24 1999-08-19     1    173    95   18        81      77
6    16 POJ    1966-10-22 2000-10-29     1    177    63   19        72      96
# i 2 more variables: PAS_INI <dbl>, PAS_FIN <dbl>
```

c. Fusionar los datos de los dos hospitales en un nuevo data frame.

Solución

3.34. Base

Con la función `rbind` del paquete `base` de R.

```r
df <- rbind(dfA, dfB)
head(df)
```

```
# A tibble: 6 x 12
  CLAVE NOMBRE F_NACIM             F_INCLUS             SEXO ALTURA  PESO ESTRES
  <dbl> <chr>  <dttm>              <dttm>              <dbl>  <dbl> <dbl>  <dbl>
1     1 SGL    1941-09-08 00:00:00 1998-07-13 00:00:00     1    165    78     42
2     2 JCZ    1957-07-10 00:00:00 1998-05-09 00:00:00     1    154    74     30
3     3 APZ    1967-08-18 00:00:00 2000-04-01 00:00:00     0    156    81     21
4     4 NDG    1956-05-08 00:00:00 1998-11-13 00:00:00     0    181    82     33
5     5 CLO    1958-11-02 00:00:00 1999-02-24 00:00:00     1    184    78     36
6     6 LFZ    1953-06-13 00:00:00 2000-03-16 00:00:00     0    179    80     22
# i 4 more variables: PAD_INI <dbl>, PAD_FIN <dbl>, PAS_INI <dbl>,
#   PAS_FIN <dbl>
```

3.35. tidyverse

Con la función `bind_rows` del paquete `dplyr` de `tidyverse`.

```r
df <- dfA |> bind_rows(dfB)
head(df)
```

```
# A tibble: 6 x 12
  CLAVE NOMBRE F_NACIM             F_INCLUS             SEXO ALTURA  PESO ESTRES
  <dbl> <chr>  <dttm>              <dttm>              <dbl>  <dbl> <dbl>  <dbl>
1     1 SGL    1941-09-08 00:00:00 1998-07-13 00:00:00     1    165    78     42
2     2 JCZ    1957-07-10 00:00:00 1998-05-09 00:00:00     1    154    74     30
3     3 APZ    1967-08-18 00:00:00 2000-04-01 00:00:00     0    156    81     21
4     4 NDG    1956-05-08 00:00:00 1998-11-13 00:00:00     0    181    82     33
5     5 CLO    1958-11-02 00:00:00 1999-02-24 00:00:00     1    184    78     36
6     6 LFZ    1953-06-13 00:00:00 2000-03-16 00:00:00     0    179    80     22
# i 4 more variables: PAD_INI <dbl>, PAD_FIN <dbl>, PAS_INI <dbl>,
#   PAS_FIN <dbl>
```

d. Crear un data frame con los datos de las claves de aleatorización del fichero csv claves-aleatorizacion.csv.

Solución

```r
claves <- read_csv("https://aprendeconalf.es/estadistica-practicas-r/datos/hiperte
head(claves)
```

```
# A tibble: 6 x 2
  CLAVE FARMACO
```

```
  <dbl> <chr>
1     1 Ca Antagonista + Diurético
2     2 Ca Antagonista + Diurético
3     3 Placebo
4     4 Ca Antagonista + Diurético
5     5 Ca Antagonista + Diurético
6     6 Placebo
```

e. Fusionar el data frame con los datos clínicos y el data frame con claves de aleatorización en un nuevo data frame.

Solución

Para fusionar las columnas de dos data frames usando una misma columna como clave en ambos data frames se puede la función `left_join` del paquete `dplyr` de `tidyverse`.

```
df <- df |> left_join(claves, by = "CLAVE")
head(df)
```

```
# A tibble: 6 x 13
  CLAVE NOMBRE F_NACIM             F_INCLUS             SEXO ALTURA  PESO ESTRES
  <dbl> <chr>  <dttm>              <dttm>              <dbl>  <dbl> <dbl>  <dbl>
1     1 SGL    1941-09-08 00:00:00 1998-07-13 00:00:00     1    165    78     42
2     2 JCZ    1957-07-10 00:00:00 1998-05-09 00:00:00     1    154    74     30
3     3 APZ    1967-08-18 00:00:00 2000-04-01 00:00:00     0    156    81     21
4     4 NDG    1956-05-08 00:00:00 1998-11-13 00:00:00     0    181    82     33
5     5 CLO    1958-11-02 00:00:00 1999-02-24 00:00:00     1    184    78     36
6     6 LFZ    1953-06-13 00:00:00 2000-03-16 00:00:00     0    179    80     22
# i 5 more variables: PAD_INI <dbl>, PAD_FIN <dbl>, PAS_INI <dbl>,
#   PAS_FIN <dbl>, FARMACO <chr>
```

f. Convertir la columna del sexo en un factor con dos niveles: `Hombre` y `Mujer`.

Solución

3.36. Base

Con la función del paquete `base` de R.

```
df$SEXO <- factor(df$SEXO, levels = c(0, 1), labels = c("Hombre", "Mujer"))
head(df)
```

```
# A tibble: 6 x 13
  CLAVE NOMBRE F_NACIM             F_INCLUS             SEXO   ALTURA  PESO ESTRES
  <dbl> <chr>  <dttm>              <dttm>              <fct>   <dbl> <dbl>  <dbl>
```

```
1     1 SGL    1941-09-08 00:00:00 1998-07-13 00:00:00 Mujer    165    78     42
2     2 JCZ    1957-07-10 00:00:00 1998-05-09 00:00:00 Mujer    154    74     30
3     3 APZ    1967-08-18 00:00:00 2000-04-01 00:00:00 Homb~    156    81     21
4     4 NDG    1956-05-08 00:00:00 1998-11-13 00:00:00 Homb~    181    82     33
5     5 CLO    1958-11-02 00:00:00 1999-02-24 00:00:00 Mujer    184    78     36
6     6 LFZ    1953-06-13 00:00:00 2000-03-16 00:00:00 Homb~    179    80     22
# i 5 more variables: PAD_INI <dbl>, PAD_FIN <dbl>, PAS_INI <dbl>,
#   PAS_FIN <dbl>, FARMACO <chr>
```

3.37. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`.

```
df <- df |> mutate(SEXO = factor(SEXO, levels = c(0, 1), labels = c("Hombre", "Muj
head(df)
```

```
# A tibble: 6 x 13
  CLAVE NOMBRE F_NACIM             F_INCLUS            SEXO  ALTURA  PESO ESTRES
  <dbl> <chr>  <dttm>              <dttm>              <fct>  <dbl> <dbl>  <dbl>
1     1 SGL    1941-09-08 00:00:00 1998-07-13 00:00:00 Mujer    165    78     42
2     2 JCZ    1957-07-10 00:00:00 1998-05-09 00:00:00 Mujer    154    74     30
3     3 APZ    1967-08-18 00:00:00 2000-04-01 00:00:00 Homb~    156    81     21
4     4 NDG    1956-05-08 00:00:00 1998-11-13 00:00:00 Homb~    181    82     33
5     5 CLO    1958-11-02 00:00:00 1999-02-24 00:00:00 Mujer    184    78     36
6     6 LFZ    1953-06-13 00:00:00 2000-03-16 00:00:00 Homb~    179    80     22
# i 5 more variables: PAD_INI <dbl>, PAD_FIN <dbl>, PAS_INI <dbl>,
#   PAS_FIN <dbl>, FARMACO <chr>
```

g. Crear una nueva columna con la edad de los pacientes en el momento de inclusión en el estudio.

Solución

3.38. Base

Con la función del paquete `base` de R.

```
df$EDAD <- as.numeric(difftime(df$F_INCLUS, df$F_NACIM, units = "days")/365)
head(df[, c("F_NACIM", "F_INCLUS", "EDAD")])
```

```
# A tibble: 6 x 3
  F_NACIM             F_INCLUS             EDAD
  <dttm>              <dttm>              <dbl>
1 1941-09-08 00:00:00 1998-07-13 00:00:00  56.9
2 1957-07-10 00:00:00 1998-05-09 00:00:00  40.9
```

```
3 1967-08-18 00:00:00 2000-04-01 00:00:00  32.6
4 1956-05-08 00:00:00 1998-11-13 00:00:00  42.5
5 1958-11-02 00:00:00 1999-02-24 00:00:00  40.3
6 1953-06-13 00:00:00 2000-03-16 00:00:00  46.8
```

3.39. tidyverse

Con las funciones `interval` y `time_length` del paquete `lubridate` de `tidyverse`. La función `interval` permite crear un intervalo de tiempo entre dos fechas y la función `time_length` permite calcular la longitud de un intervalo en una determinada unidad de tiempo.

```
df <- df |> mutate(AGE = time_length(interval(F_NACIM, F_INCLUS), "years"))
head(df |>  select(F_NACIM, F_INCLUS, AGE))
```

```
# A tibble: 6 x 3
  F_NACIM             F_INCLUS              AGE
  <dttm>              <dttm>              <dbl>
1 1941-09-08 00:00:00 1998-07-13 00:00:00  56.8
2 1957-07-10 00:00:00 1998-05-09 00:00:00  40.8
3 1967-08-18 00:00:00 2000-04-01 00:00:00  32.6
4 1956-05-08 00:00:00 1998-11-13 00:00:00  42.5
5 1958-11-02 00:00:00 1999-02-24 00:00:00  40.3
6 1953-06-13 00:00:00 2000-03-16 00:00:00  46.8
```

h. Crear una nueva columna con el índice de masa corporal (IMC) de los pacientes.

Solución

3.40. Base

Con las funciones del paquete `base` de R.

```
df$IMC <- df$PESO/(df$ALTURA/100)^2
head(df[, c("PESO", "ALTURA", "IMC")])
```

```
# A tibble: 6 x 3
   PESO ALTURA   IMC
  <dbl>  <dbl> <dbl>
1    78    165  28.7
2    74    154  31.2
3    81    156  33.3
4    82    181  25.0
5    78    184  23.0
6    80    179  25.0
```

3.41. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`.

```
df <- df |> mutate(IMC = PESO/(ALTURA/100)^2)
head(df |> select(PESO, ALTURA, IMC))
```

```
# A tibble: 6 x 3
   PESO ALTURA   IMC
  <dbl>  <dbl> <dbl>
1    78    165  28.7
2    74    154  31.2
3    81    156  33.3
4    82    181  25.0
5    78    184  23.0
6    80    179  25.0
```

i. Crear una nueva columna para la evolución de la presión arterial diastólica y otra con la evolución de la presión arterial sistólica.

Solución

3.42. Base

Con las funciones del paquete `base` de R.

```
df$EVOL_PAD <- df$PAD_FIN - df$PAD_INI
df$EVOL_PAS <- df$PAS_FIN - df$PAS_INI
head(df[, c("PAD_INI", "PAD_FIN", "EVOL_PAD", "PAS_INI", "PAS_FIN", "EVOL_PAS")])
```

```
# A tibble: 6 x 6
  PAD_INI PAD_FIN EVOL_PAD PAS_INI PAS_FIN EVOL_PAS
    <dbl>   <dbl>    <dbl>   <dbl>   <dbl>    <dbl>
1      78     104       26     176     175       -1
2      95     114       19     162     160       -2
3      93     102        9     141     150        9
4      86      91        5     162     161       -1
5      89      94        5     165     162       -3
6      74      99       25     141     148        7
```

3.43. tidyverse

Con la función `mutate` del paquete `dplyr` de `tidyverse`.

```r
df <- df |> mutate(EVOL_PAD = PAD_FIN - PAD_INI, EVOL_PAS = PAS_FIN - PAS_INI)
head(df |> select(PAD_INI, PAD_FIN, EVOL_PAD, PAS_INI, PAS_FIN, EVOL_PAS))
```

```
# A tibble: 6 x 6
  PAD_INI PAD_FIN EVOL_PAD PAS_INI PAS_FIN EVOL_PAS
    <dbl>   <dbl>    <dbl>   <dbl>   <dbl>    <dbl>
1      78     104       26     176     175       -1
2      95     114       19     162     160       -2
3      93     102        9     141     150        9
4      86      91        5     162     161       -1
5      89      94        5     165     162       -3
6      74      99       25     141     148        7
```

j. Guardar el data frame en un fichero csv.

Solución

3.44. Base

Con la función `write.csv` del paquete `base` de R.

```r
write.csv(df, "datos/hipertension/datos-ensayo-clinico.csv")
```

3.45. tidyverse

Con la función `write_csv` del paquete `readr` de `tidyverse`.

```r
df  |> write_csv("datos/hipertension/datos-ensayo-clinico.csv")
```

3.46. Ejercicios Propuestos

Ejercicio 3.5. Los ficheros `vinos-blancos.xls` y `vinos-tintos.csv` contienen información sobre las características de vinos blancos y tintos portugueses de la denominación "Vinho Verde". Las variables almacenadas en estos archivos son las siguientes:

Variable	Descripción	Tipo (unidades)
tipo	Tipo de vino	Factor (blanco, tinto)
meses.barrica	Mesesde envejecimiento en barrica	Numérica(meses)
acided.fija	Cantidadde ácidotartárico	Numérica(g/dm3)
acided.volatil	Cantidad de ácido acético	Numérica(g/dm3)

Variable	Descripción	Tipo (unidades)
acido.citrico	Cantidad de ácidocítrico	Numérica(g/dm3)
azucar.residual	Cantidad de azúcarremanente después de la fermentación	Numérica(g/dm3)
cloruro.sodico	Cantidad de clorurosódico	Numérica(g/dm3)
dioxido.azufre.libre	Cantidad de dióxido de azufreen formalibre	Numérica(mg/dm3)
dioxido.azufre.total	Cantidadde dióxido de azufretotal en forma libre o ligada	Numérica(mg/dm3)
densidad	Densidad	Numérica(g/cm3)
ph	pH	Numérica(0-14)
sulfatos	Cantidadde sulfato de potasio	Numérica(g/dm3)
alcohol	Porcentajede contenidode alcohol	Numérica(0-100)
calidad	Calificación otorgada porun panel de expertos	Numérica(0-10)

a. Crear un data frame con los datos de los vinos blancos partir del fichero de Excel `vinos-blancos.xlsx`.

b. Crear un data frame con los datos de los vinos tintos partir del fichero csv `vinos-tintos.csv`.

c. Fusionar los datos de los vinos blancos y tintos en un nuevo data frame.

d. Convertir el tipo de vino en un factor.

e. Imputar los valores perdidos del alcohol con la media de los valores no perdidos para cada tipo de vino.

f. Crear un factor `Envejecimiento` recodificando la variable `meses.barrica` en las siguientes categorías.

Rango en meses	Categoría
Menos de 3	Joven
Entre 3 y 12	Crianza
Entre 12 y 18	Reserva
Más de 18	Gran reserva

g. Crear un factor `Dulzor` recodificando la variable `azucar.residual` en las siguientes categorías.

Rango azúcar	Categoría
Menos de 4	Seco
Más de 4 y menos de 12	Semiseco
Más de 12 y menos de 45	Semidulce
Más de 45	Dulce

h. Filtrar el conjunto de datos para quedarse con los vinos Reserva o Gran Reserva con una calidad superior a 7 y ordenar el data frame por calidad de forma descendente.

i. ¿Cuántos vinos blancos con un contenido en alcohol superior al 12 % y una calidad superior a 8 hay en el conjunto de datos?

4 Distribuciones de frecuencias y representaciones gráficas

En esta práctica contiene ejercicios que muestran como hacer un resumen descriptivos de un conjunto de datos mediante la construcción de tablas de frecuencias y la representación gráfica de las mismas. En particular, se muestra cómo construir los siguientes tipos de gráficos:

- Diagramas de barras.
- Diagramas de sectores.
- Diagramas de cajas.
- Histogramas.
- Polígonos de frecuencias.

4.1. Ejercicios Resueltos

Para la realización de esta práctica se requieren los siguientes paquetes:

```
library(tidyverse)
# Incluye los siguientes paquetes:
# - readr: para la lectura de ficheros csv.
# - dplyr: para el preprocesamiento y manipulación de datos.
# - ggplot2: para la representación gráfica.
library(knitr) # para el formateo de tablas.
```

Ejercicio 4.1. En una encuesta a 25 matrimonios sobre el número de hijos que tenían se obtuvieron los siguientes datos:

$$1, 2, 4, 2, 2, 2, 3, 2, 1, 1, 0, 2, 2, 0, 2, 2, 1, 2, 2, 3, 1, 2, 2, 1, 2$$

a. Crear un conjunto de datos con la variable `hijos`.

Solución

4.2. Base

```
df <- data.frame(hijos = c(1, 2, 4, 2, 2, 2, 3, 2, 1, 1, 0, 2, 2, 0, 2, 2, 1, 2, 2
```

4.3. tidyverse

```
library(tidyverse)
df <- tibble(hijos = c(1, 2, 4, 2, 2, 2, 3, 2, 1, 1, 0, 2, 2, 0, 2, 2, 1, 2, 2, 3,
```

b. Construir la tabla de frecuencias.

Solución

4.4. Base

Para obtener las frecuencias absolutas se puede usar la función `table`, y para las frecuencias relativas la función `prop.table` ambas del paquete `base` de R.
Posteriormente, para obtener las frecuencias acumuladas se puede usar la función `cumsum` aplicada a las frecuencias absolutas y relativas.

```
library(knitr)
# Frecuencias absolutas.
ni <- table(df$hijos)
# Frecuencias relativas
fi <- prop.table(ni)
# Frecuencias acumuladas.
Ni <- cumsum(ni)
# Frecuencias relativas acumuladas.
Fi <- cumsum(fi)
# Creación de un data frame con las frecuencias.
tabla_frec <- cbind(ni, fi, Ni, Fi)
kable(tabla_frec)
```

	ni	fi	Ni	Fi
0	2	0.08	2	0.08
1	6	0.24	8	0.32
2	14	0.56	22	0.88
3	2	0.08	24	0.96
4	1	0.04	25	1.00

4.5. tidyverse

Para obtener la tabla de frecuencias podemos usar la función `count` del paquete `dplyr` de `tidyverse`.

```
library(knitr)
count(df, hijos) |>
    mutate(fi = n/sum(n), Ni = cumsum(n), Fi = cumsum(n)/sum(n)) |>
    kable()
```

hijos	n	fi	Ni	Fi
0	2	0.08	2	0.08
1	6	0.24	8	0.32
2	14	0.56	22	0.88
3	2	0.08	24	0.96
4	1	0.04	25	1.00

c. Dibujar el diagrama de barras de las frecuencias absolutas, relativas, absolutas acumuladas y relativas acumuladas.

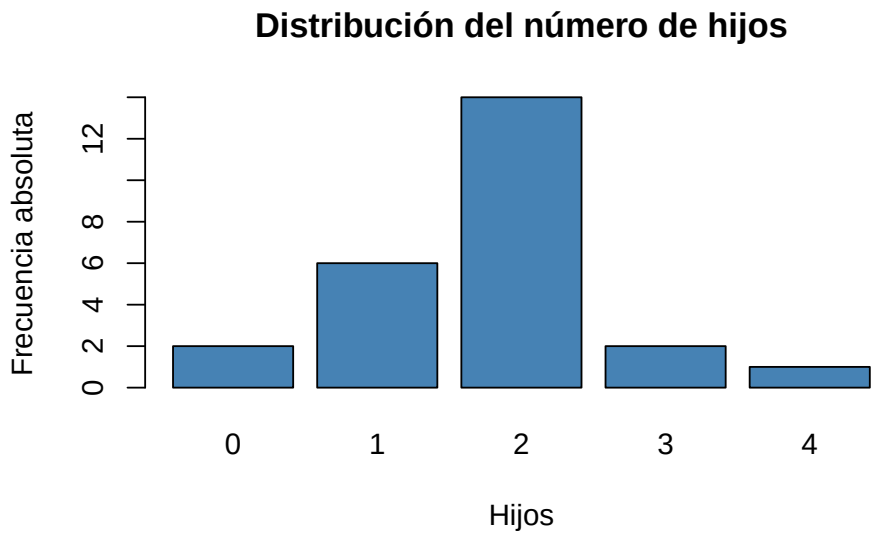
Solución

4.6. Base

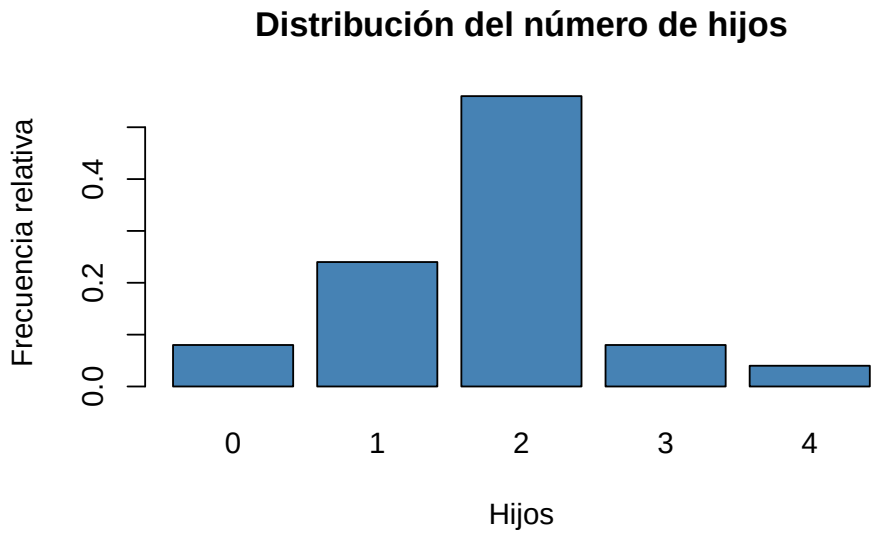
Para dibujar un diagrama de barras podemos usar la función `barplot` del paquete `graphics`.
Parámetros:

- `height`: vector con las alturas de las barras.
- `col`: color de las barras.
- `main`: título del gráfico.
- `xlab`: etiqueta del eje x.
- `ylab`: etiqueta del eje y.

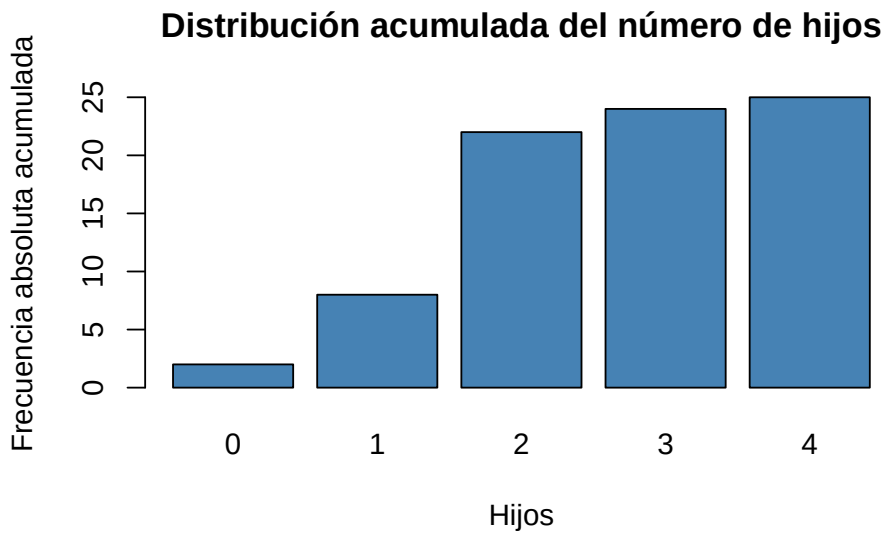
```
# Diagrama de barras de frecuencias absolutas.
barplot(ni, col = "steelblue", main = "Distribución del número de hijos", xlab = "
```



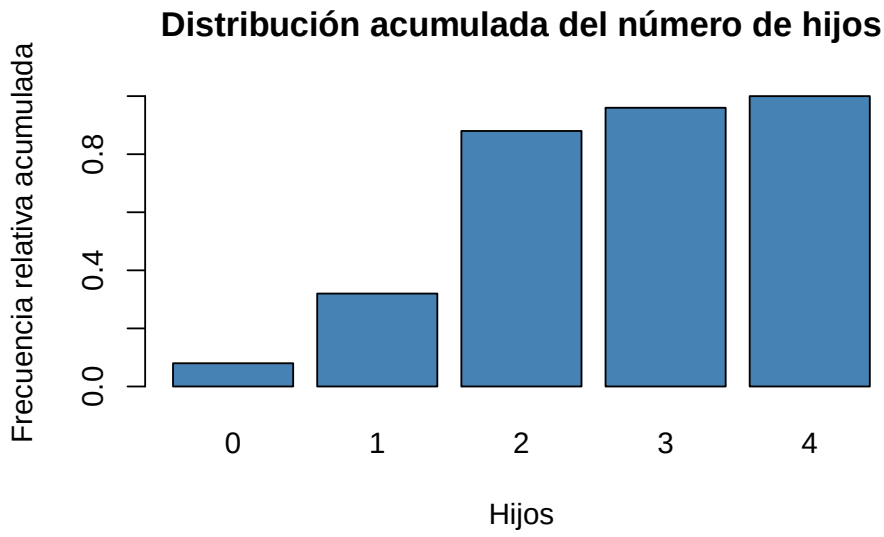

```
# Diagrama de barras de frecuencias relativas.
barplot(fi, col = "steelblue", main = "Distribución del número de hijos", xlab = "
```




```
# Diagrama de barras de frecuencias absolutas acumuladas.
barplot(Ni, col = "steelblue", main = "Distribución acumulada del número de hijos"
```




```
# Diagrama de barras de frecuencias relativas acumuladas.
barplot(Fi, col = "steelblue", main = "Distribución acumulada del número de hijos"
```



4.7. tidyverse

Para dibujar un diagrama de barras podemos usar la función `geom_bar` del paquete `ggplot2` de `tidyverse`. Esta función calcula automaticamente las

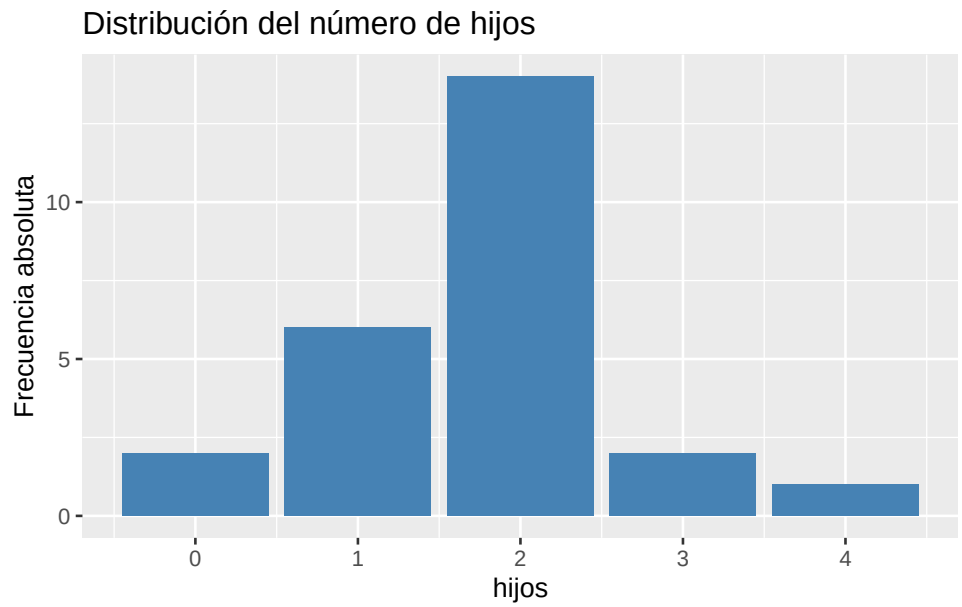
frecuencias absolutas de la columna indicada en la dimensión `x` para barras horizontles o `y` para barras verticales.
Parámetros:

- color: color de las barras del borde de las barras.
- fill: color de relleno de las barras.
- width: anchura de las barras (valor entre 0 y 1).

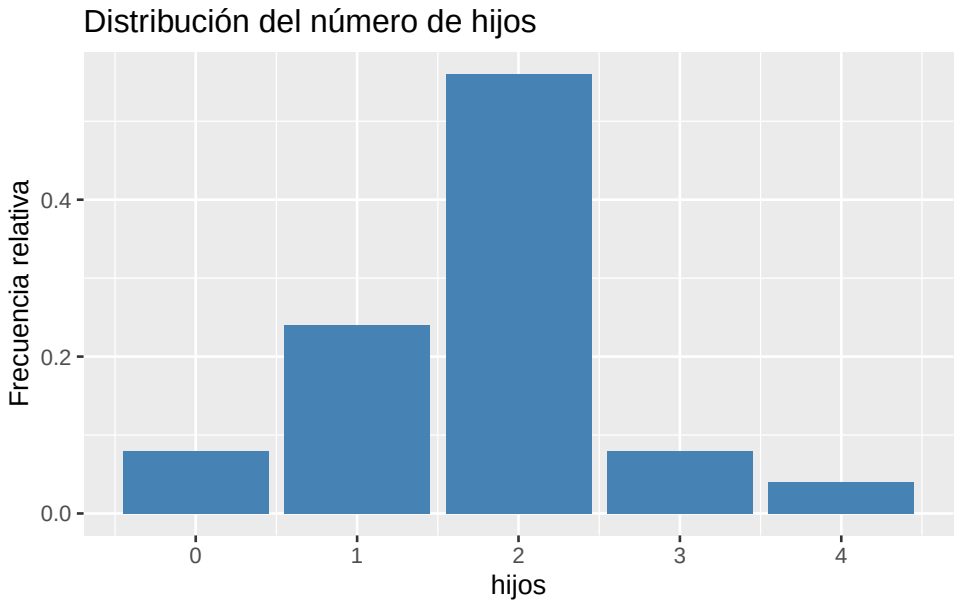
Para dibujar el diagrama de barras de frecuencias relativas o acumuladas, se le pude pasar como parámetro la función `after_stat`:

- `after_stat(count/sum(count))`: para las frecuencias relativas.
- `after_stat(cumsum(count))`: para las frecuencias absolutas acumuladas.
- `after_stat(cumsum(count)/sum(count))`: para las frecuencias relativas acumuladas.

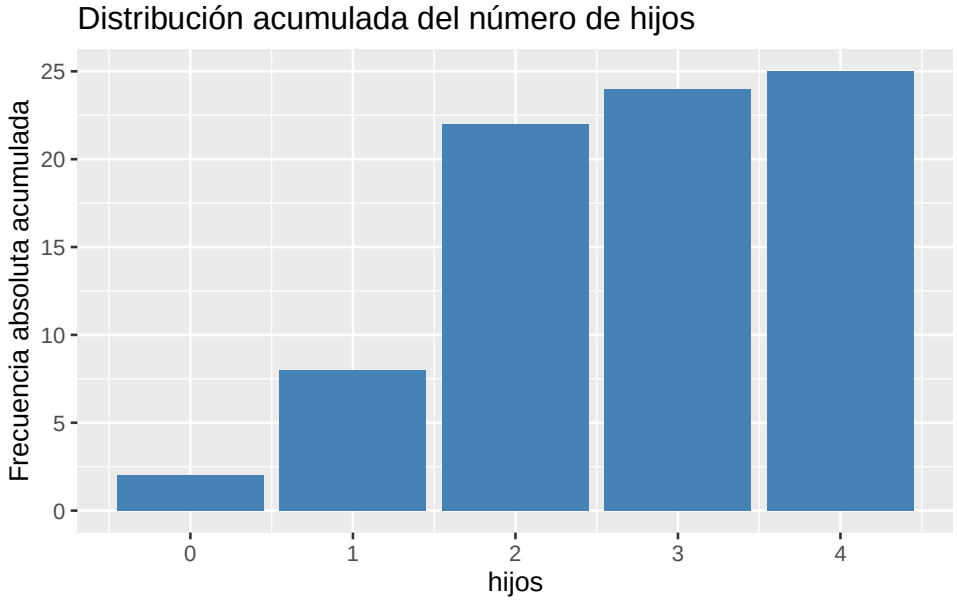
```
# Diagrama de barras de frecuencias absolutas
ggplot(df, aes(x = hijos)) +
    geom_bar(fill = "steelblue") +
    labs(title = "Distribución del número de hijos", y = "Frecuencia absoluta")
```



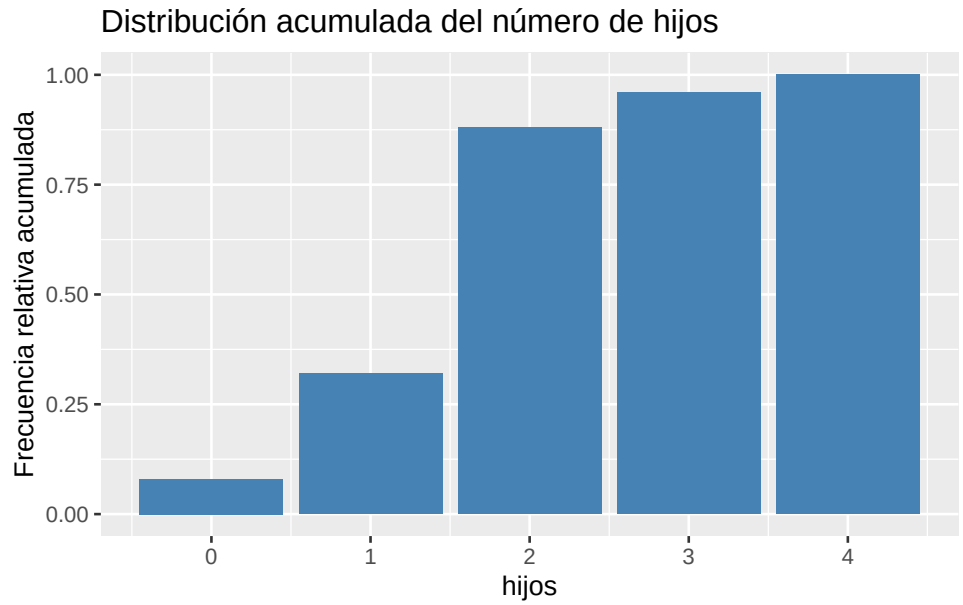

```
# Diagrama de barras de frecuencias relativas
ggplot(df, aes(x = hijos)) +
    geom_bar(aes(y = after_stat(count/sum(count))), fill = "steelblue") +
    labs(title = "Distribución del número de hijos", y = "Frecuencia relativa")
```




```
# Diagrama de barras de frecuencias acumuladas
ggplot(df, aes(x = hijos)) +
    geom_bar(aes(y = after_stat(cumsum(count))), fill = "steelblue") +
    labs(title = "Distribución acumulada del número de hijos", y = "Frecuencia abs
```



```r
# Diagrama de barras de frecuencias acumuladas
ggplot(df, aes(x = hijos)) +
    geom_bar(aes(y = after_stat(cumsum(count)/sum(count))), fill = "steelblue") +
    labs(title = "Distribución acumulada del número de hijos", y = "Frecuencia rel
```



d. Dibujar el polígono de frecuencias relativas.

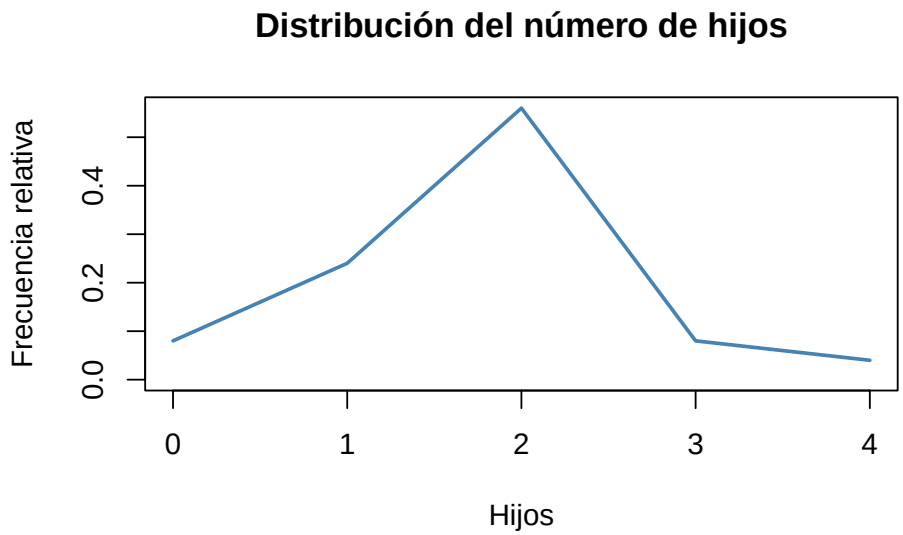
Solución

4.8. Base

Para dibujar el polígono de frecuencias podemos usar la función `plot` del paquete `graphics`.
Parámetros:

- `x`: tabla con las frecuencias.
- `type`: tipo de gráfico. Para un polígono de frecuencias se usa `"l"` (línea).
- `col`: color de la línea.
- `main`: título del gráfico.
- `xlab`: etiqueta del eje x.
- `ylab`: etiqueta del eje y.

```r
# Frecuencias relativas.
plot(fi, type = "l", col = "steelblue", main="Distribución del número de hijos", x
```

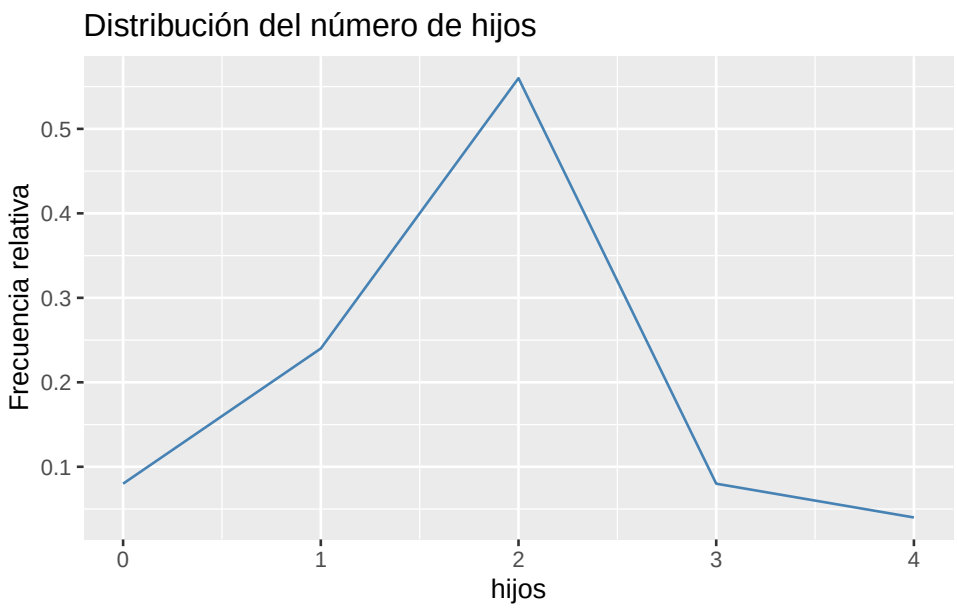


4.9. tidyverse

Par dibujar el polígono de frecuencias podemos usar la función `geom_line` del paquete `ggplot2` de `tidyverse`, que conecta con segmentos los puntos con coordenadas pasadas en las dimensiones `x` e `y`.
Parámetros:

- `col`: color de la línea.
- `size`: grosor de la línea.
- `linetype`: tipo de línea (por ejemplo, “solid”, “dashed”, “dotted”).

```
library(ggplot2)
count(df, hijos) |>
    mutate(fi = n/sum(n)) |>
    ggplot(aes(x = hijos, y = fi)) +
    geom_line(col = "steelblue") +
    labs(title = "Distribución del número de hijos", y = "Frecuencia relativa")
```



Ejercicio 4.2. En un servicio de atención al cliente se han registrado el número de llamadas de clientes cada día del mes de noviembre, obteniendo los siguientes datos:

15, 23, 12, 10, 28, 50, 12, 17, 20, 21, 18, 13, 11, 12, 26
30, 6, 16, 19, 22, 14, 17, 21, 28, 9, 16, 13, 11, 16, 20

a. Crear un conjunto de datos con la variable `llamadas`.

Solución

```
df <- data.frame(llamadas = c(15, 23, 12, 10, 28, 50, 12, 17, 20, 21, 18, 13, 11,
```

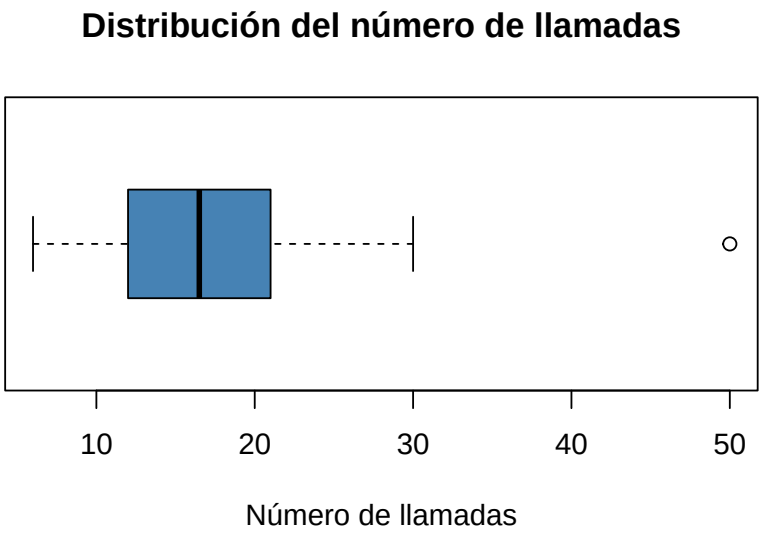
b. Dibujar el diagrama de cajas. ¿Existe algún dato atípico? En el caso de que exista, eliminarlo y proceder con los siguientes apartados.

Solución

4.10. Base

Con la función `boxplot` del paquete `graphics`.

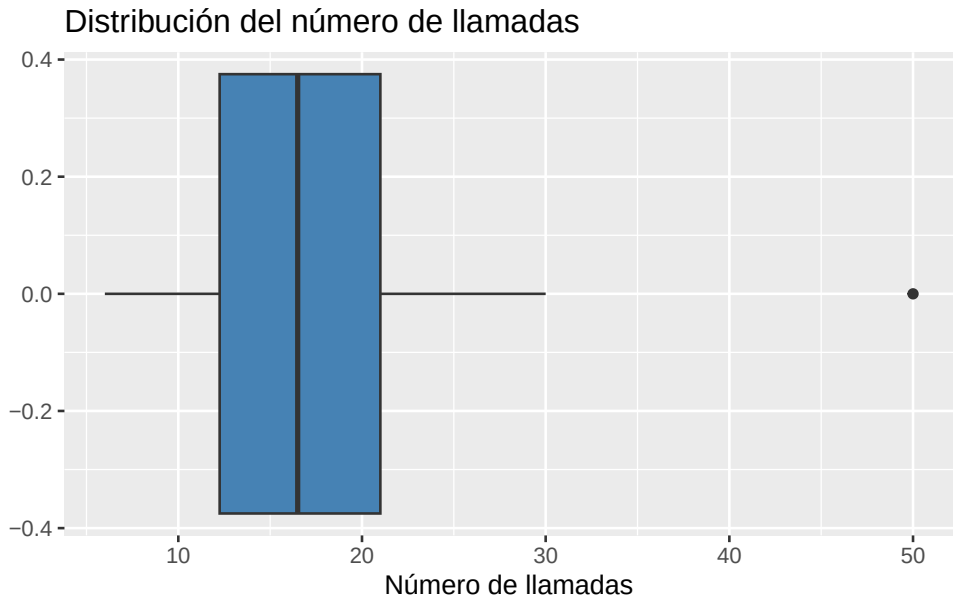
```
# Frecuencias relativas.
boxplot(df$llamadas, col = "steelblue", main="Distribución del número de llamadas"
```



4.11. tidyverse

Con la función la función `geom_boxplot` del paquete `ggplot2` de `tidyverse`.

```
library(tidyverse)
ggplot(df, aes(x = llamadas)) +
    geom_boxplot(fill = "steelblue") +
    labs(title = "Distribución del número de llamadas", x = "Número de llamadas")
```



Hay un día con 50 llamadas, que es un valor atípico en comparación con el resto de días.

4.12. Base

Con las funciones del paquete `base` de R.

```
# Eliminación del dato atípico.
df <- df[df$llamadas != 50, , drop = F]
```

4.13. tidyverse

Con la función `filter` del paquete `dplyr` de `tidyverse`.

```
df <- filter(df, llamadas != 50)
```

c. Construir la tabla de frecuencias agrupando en 5 clases.

Solución

4.14. Base

Para agrupar los datos en intervalos se puede utilizar la función `cut` del paquete base de R, y para contar las frecuencias absolutas y relativas las funciones `table`, y `prop.table` respectivamente.

```r
# Frecuencias absolutas. Creación automática de 5 clases con intervalos cerrados a
```

```r
library(knitr)
ni <- table(cut(df$llamadas, breaks = 5, right = F))
# Creación manual de 5 clases.
ni <- table(cut(df$llamadas, breaks = seq(5, 30, 5)))
# Frecuencias relativas
fi <- prop.table(ni)
# Frecuencias acumuladas.
Ni <- cumsum(ni)
# Frecuencias relativas acumuladas.
Fi <- cumsum(fi)
# Creación de un data frame con las frecuencias.
tabla_frec <- cbind(ni, fi, Ni, Fi)
kable(tabla_frec)
```

	ni	fi	Ni	Fi
(5,10]	3	0.1034483	3	0.1034483
(10,15]	9	0.3103448	12	0.4137931
(15,20]	9	0.3103448	21	0.7241379
(20,25]	4	0.1379310	25	0.8620690
(25,30]	4	0.1379310	29	1.0000000

4.15. tidyverse

Con la fución `count` del paquete `dplyr` de `tidyverse`.

```r
library(knitr)
mutate(df, llamadas_int = cut(llamadas, breaks = seq(5, 30, 5))) |>
    count(llamadas_int) |>
    mutate(fi = n/sum(n), Ni = cumsum(n), Fi = cumsum(n)/sum(n)) |>
    kable()
```

	llamadas_int	n	fi	Ni	Fi
(5,10]	3	0.1034483	3	0.1034483	
(10,15]	9	0.3103448	12	0.4137931	
(15,20]	9	0.3103448	21	0.7241379	
(20,25]	4	0.1379310	25	0.8620690	
(25,30]	4	0.1379310	29	1.0000000	

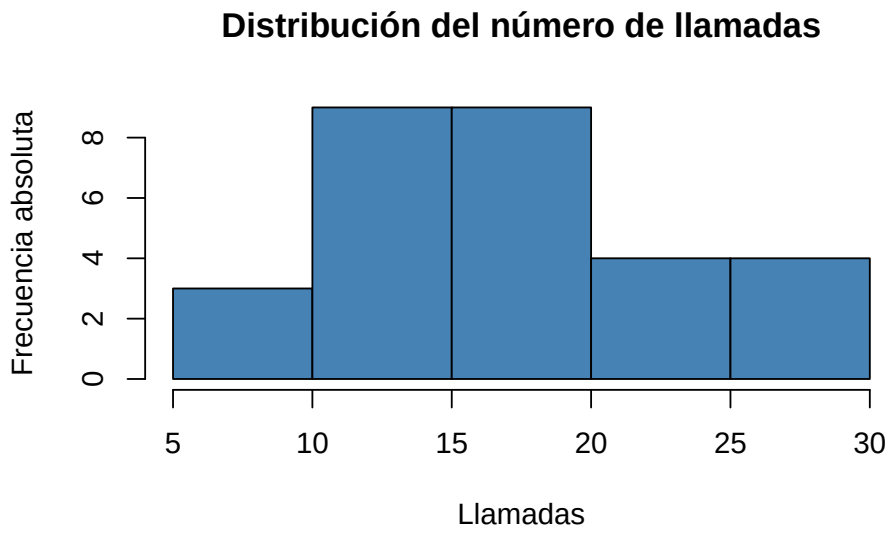
d. Dibujar el histograma de frecuencias absolutas, relativas, absolutas acumuladas y relativas acumuladas correspondiente a la tabla anterior.

Solución

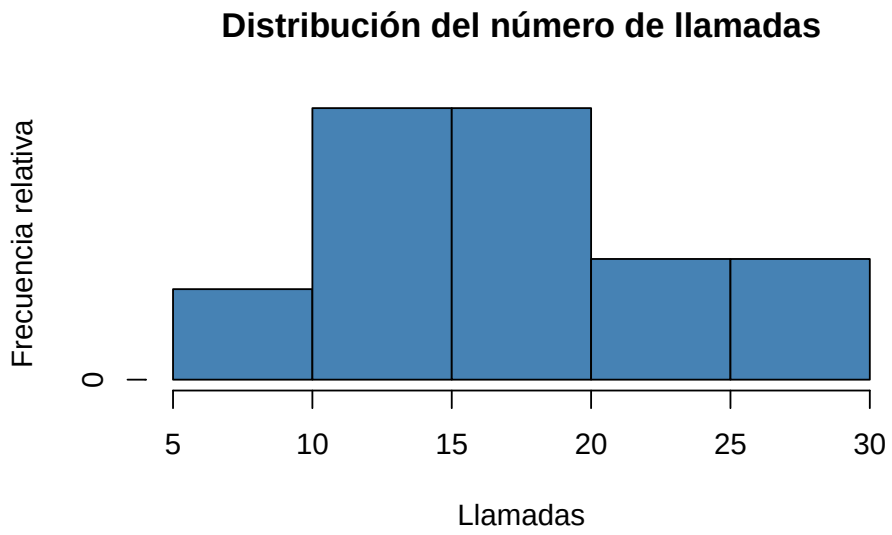
4.16. Base

Con la función `hist` del paquete `graphics`.

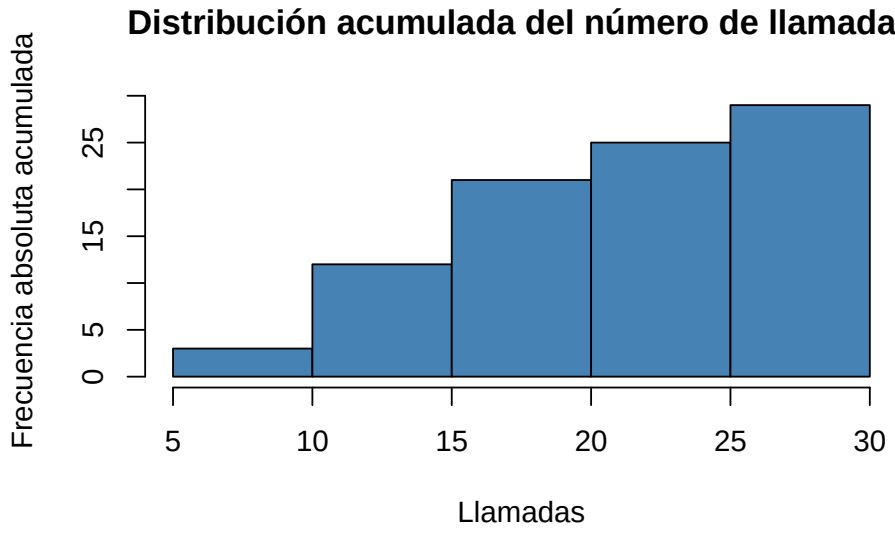
```
# Histograma de frecuencias absolutas.
histo <- hist(df$llamadas, breaks = seq(5, 30, 5), col = "steelblue", main="Distri
```



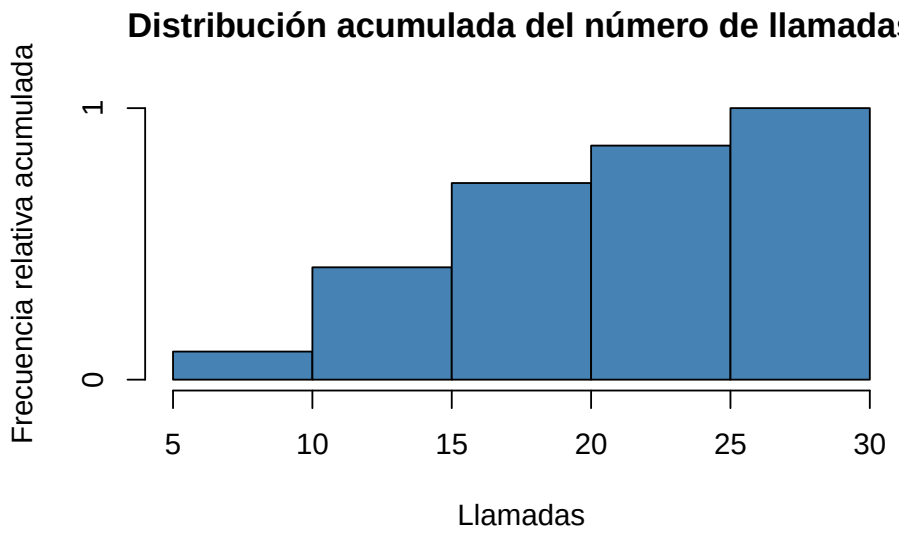

```
ni <- histo$counts
# Histograma de frecuencias relativas.
histo$counts <- ni/sum(ni)
plot(histo, col = "steelblue", main="Distribución del número de llamadas", xlab="L
```




```
# Histograma de frecuencias absolutas acumuladas.
histo$counts <- cumsum(ni)
plot(histo, col = "steelblue", main="Distribución acumulada del número de llamadas
```



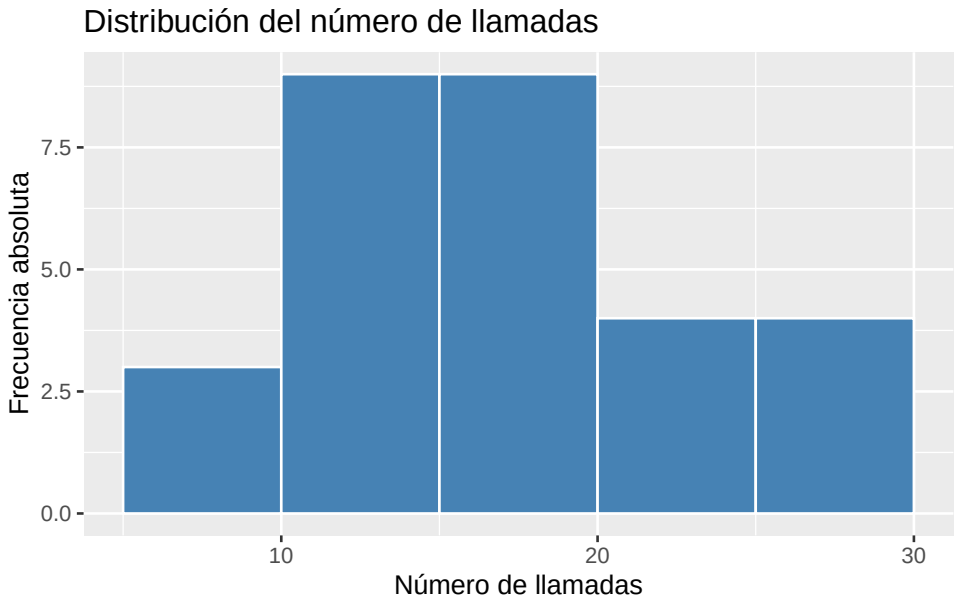

```
# Histograma de frecuencias relativas acumuladas.
histo$counts <- cumsum(ni)/sum(ni)
plot(histo, col = "steelblue", main="Distribución acumulada del número de llamadas
```



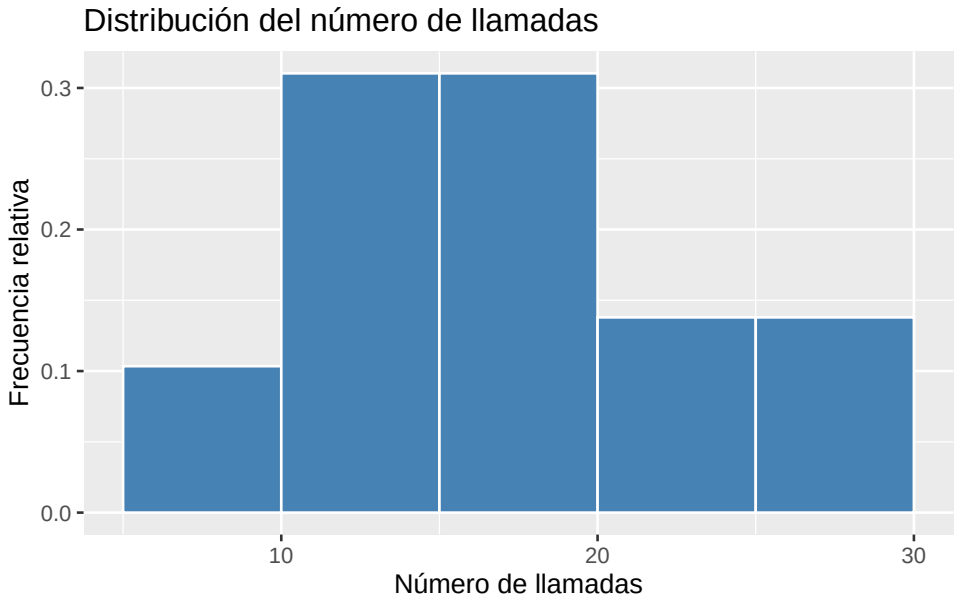
4.17. tidyverse

Con la función la función `geom_histogram` del paquete `ggplot2` de `tidyverse`.

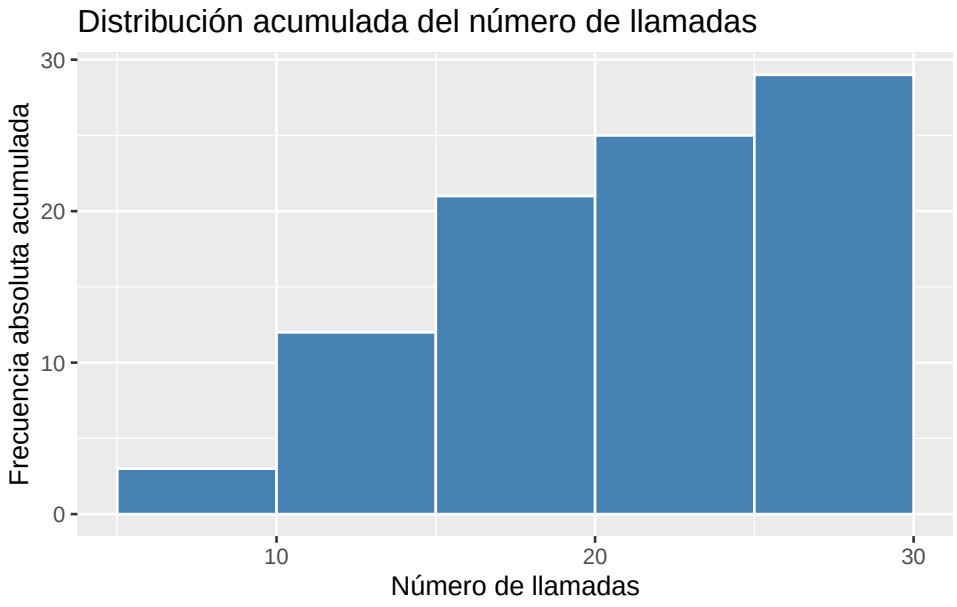
```
# Histograma de frecuencias absolutas
ggplot(df, aes(x = llamadas)) +
    geom_histogram(breaks = seq(5, 30, 5), fill = "steelblue", col = "white") +
    labs(title = "Distribución del número de llamadas", x = "Número de llamadas",
```



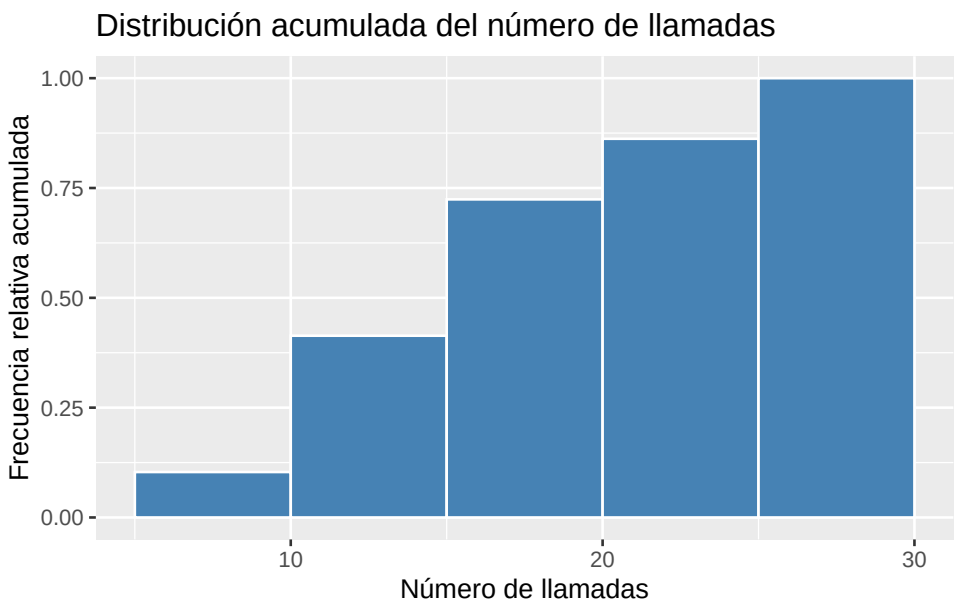

```
# Histograma de frecuencias relativas
ggplot(df, aes(x = llamadas)) +
    geom_histogram(aes(y = after_stat(count/sum(count))), breaks = seq(5, 30, 5),
    labs(title = "Distribución del número de llamadas", x = "Número de llamadas",
```



```r
# Histograma de frecuencias acumuladas
ggplot(df, aes(x = llamadas)) +
    geom_histogram(aes(y = after_stat(cumsum(count))), breaks = seq(5, 30, 5), fil
    labs(title = "Distribución acumulada del número de llamadas", x = "Número de l
```




```r
# Histograma de frecuencias relativas acumuladas
ggplot(df, aes(x = llamadas)) +
    geom_histogram(aes(y = after_stat(cumsum(count)/sum(count))),  breaks = seq(5,
    labs(title = "Distribución acumulada del número de llamadas", x = "Número de l
```



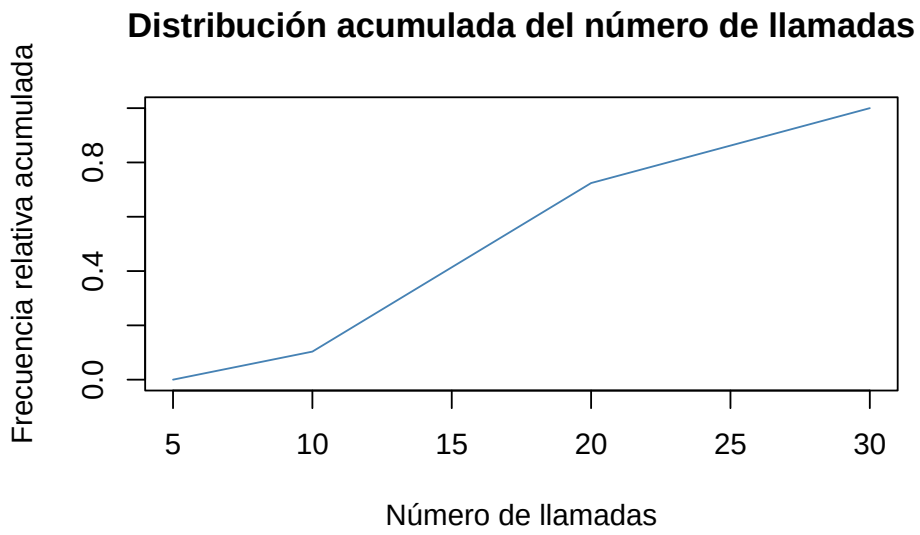
e. Dibujar el polígono de frecuencias relativas acumuladas (ojiva).

Solución

4.18. Base

Con la función `plot` del paquete `graphics`.

```
# Ojiva
cortes = seq(5, 30, 5)
ni <- table(cut(df$llamadas, breaks = cortes))
Fi <- c(0, cumsum(ni)/sum(ni))
plot(cortes, Fi, type = "l", col = "steelblue", main = "Distribución acumulada del
```



4.19. tidyverse

Con la función geom_line del paquete ggplot2 de tidyverse.

```
library(ggplot2)
# Ojiva
cortes <- seq(5, 30, 5)
tabla_frec <- mutate(df, llamadas_int = cut(df$llamadas, breaks = cortes)) |>
    count(llamadas_int) |>
    mutate(cortes = cortes[-1], Fi = cumsum(n)/sum(n)) |>
    select(cortes, Fi)
tabla_frec <- rbind(data.frame(cortes = cortes[1], Fi = 0), tabla_frec)
ggplot(tabla_frec, aes(x = cortes , y = Fi)) +
    geom_line(col = "steelblue") +
    labs(title = "Distribución del número de llamadas", x = "Número de llamadas",
```