Capítulo 1: Instalar R

Vamos a empezar con los pasos para la instalación de R en los sistemas operativos Windows, MacOs y Linux.

Descargar R

1. Ve al sitio web oficial de R en https://www.r-project.org/
2. haz clic en donde diga **Download R**.
3. En la lista mirrors, busca https://cran.itam.mx/ y hacer click.

continuamos en la sección correspondiente a tu sistema operativo:

Instalar R en Windows

1. Hacer click donde diga ***Download for Windows*** y guarda el archivo ejecutable.
2. Corre el archivo .exe y seguir las instrucciones de instalación.

Instalar R en macOS

1. Hacer click donde diga ***Download for macOS***.
2. Hacer click en el link del archivo de la versión mas reciente de R.
3. Correr el archivo .pkg y seguir las instrucciones de instalación.

Instalar R en Linux

1. Hacer click donde diga ***Download for Linux***.
2. Selecciona tu distribución de Linux y sigue las instrucciones para instalar desde la terminal.

Capítulo 2: ¿Para qué usar R?

Origenes y usos

R es un software estadístico basado en los lenguajes *S* y *Scheme*. Por un lado, *S* es un lenguaje de programación desarrollado por Rick Bercker, John Chambers y colegas en Bell Labs en los años ochentas. Tiene como principal objetivo alentar y permitir el buen análisis estadístico.

En cuanto a R, todo comenzó en agosto de 1993 como un experimento realizado por Ross Ihaka y Robert Gentelman, ellos reconocieron la necesidad de un mejor ambiente de cálculo del que tenían. Ninguno de los productos comerciales les convencían por lo que decidieron crear una herramienta que les funcionara. Esto los llevo a tomar ideas de *S*, donde les resultó natural usar una sintaxis parecida. Por lo que, en el uso diario, *S* y R son muy similares, pero R no es *S*.

Figura 1: Robert Gentleman (izquierda) y Ross Ihaka (derecha), creadores de R

Fuente: Manual de R

Las diferencias entre ellos son resultado de la herencia de *Scheme*, fundamentalmente el manejo de la memoria y el acceso a las variables dependiendo de dónde fueron definidas. Tambien se distingue el manejo del color, áreas de graficación, rotulación matematica, entre otros.

Algunas fechas importantes en el desarrollo de R: en febrero de 2000 sale finalmente la versión 1.0 de R; en 2001 se publica el primer número de R-News, revista electronica dedicadaa la discusión y anuncios de nuevos procedimientos y paquetes de R; la revista es reemplazada por R-Journal en 2009.

Finalmente, para garantizar que R sea siempre un software libre de codigo abierto, se creó R-foundation que entre sus objetivos están:

1. Avanzar el proyecto de R para cálculo estadístico que proveo de software libre y código abierto para el análisis de datos y gráficas.
2. Guardar y administrar los derechos de copia de R y su documentación. [1]

[1] Barrios, E.(2010). R: Un lenguaje para análisis de datos y graficación. Recuperado de: https://gente.itam.mx/ebarrios/docs/porqueR.pdf

¿Qué se puede hacer con R?

R es una herramienta versátil que, en un principio, ofrece un conjunto robusto de funciones estadísticas para el análisis de datos, evaluación de algoritmos, modelos lineales y no lineales, pruebas estadísticas, entre otros. Además, R puede extenderse hasta campos avanzados como el machine learning, una rama fundamental de la inteligencia artificial. Su alcance abarca diversas áreas, como las siguientes:

- **Análisis y Visualización de Datos:** Procesamiento, análisis estadístico y generación de gráficos avanzados.
- **Tareas Académicas:** Resolución de ejercicios estadísticos, desarrollo de trabajos y tareas académicas.
- **Programación y Desarrollo:** Automatización de procesos y creación de aplicaciones, incluso para desarrollo web.
- **Educación y Aprendizaje:** Una herramienta clave para aprender estadística y programación.
- **Sector Empresarial:** Empresas como Microsoft, American Express, HP y Ford utilizan y promueven el uso de R en sus operaciones.

Capacidades Avanzadas

Además de las funcionalidades básicas, R ofrece herramientas avanzadas que permiten abordar tareas más complejas y especializadas. Estas capacidades lo convierten en una opción poderosa para proyectos innovadores y de gran impacto. Algunos ejemplos destacados incluyen:

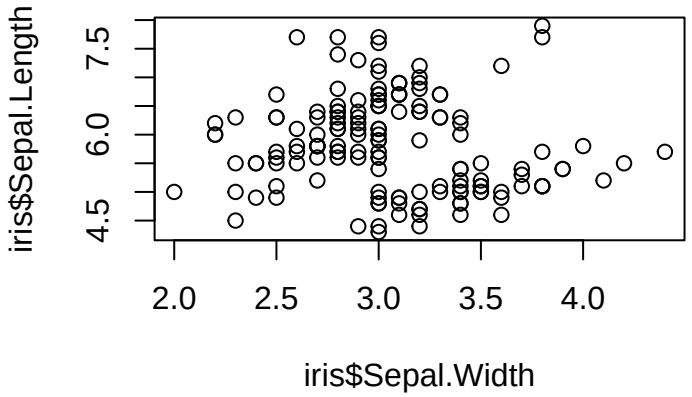
- **Aplicaciones Interactivas:** Permite desarrollar aplicaciones accesibles desde dispositivos móviles para interactuar con datos en tiempo real.
- **Análisis de Redes Sociales:** Con paquetes como ‘rtweet‘, es posible analizar datos de plataformas como Twitter, visualizar interacciones y tendencias, y mucho más.
- **Mapas Interactivos:** Usando herramientas como ‘leaflet‘, se pueden crear mapas dinámicos, trazar datos geoespaciales, añadir capas de información y personalizar visualizaciones.
- **Integración con Sensores de Teléfono:** Mediante aplicaciones como "Sensor Data" (Android) o "SensorLog" (iOS), puedes recopilar datos de los sensores de tu dispositivo móvil y analizarlos directamente en R.

Ejemplos

Gráficos

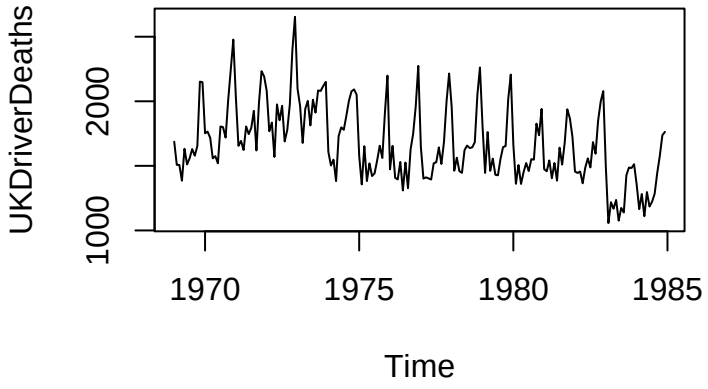
Diagrama de Disperción:

```
plot(iris$Sepal.Width, iris$Sepal.Length)
```



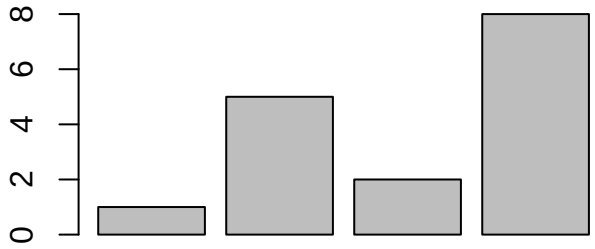
Serie temporal:

```
plot(UKDriverDeaths)
```



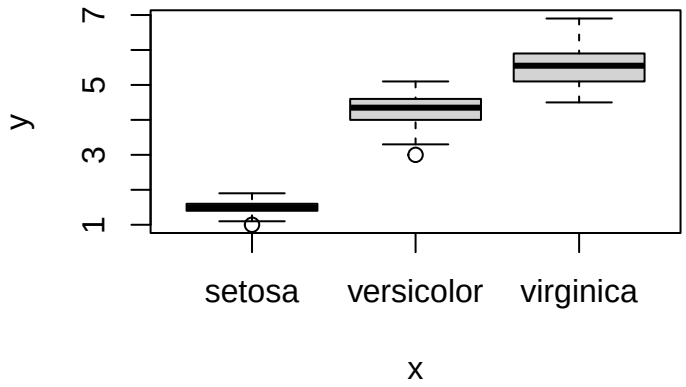
Gráfica de Barras:

```
barplot(c(1, 5, 2, 8))
```



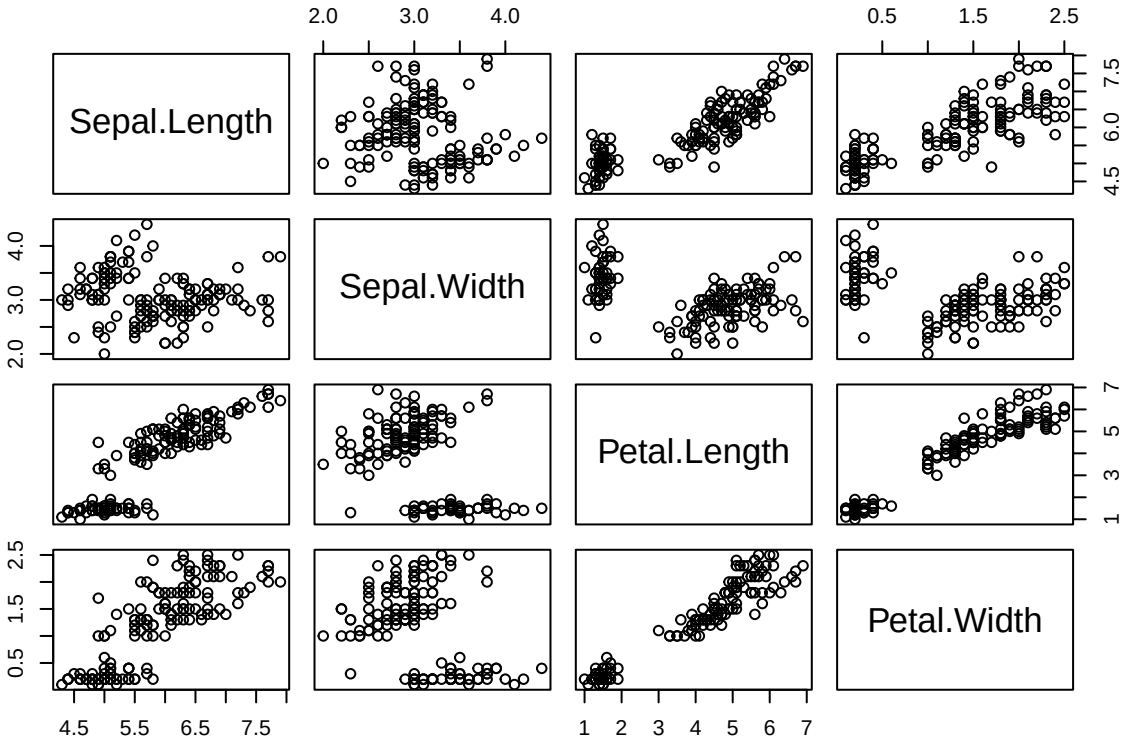
Gráficas de cajas:

```
plot(iris$Species, iris$Petal.Length)
```



Diagramas de dispersión emparejados:

```
plot(iris[-5]) # se elimina de iris el factor con el tipo de flor
```



Gráficos personalizados

```
x <- seq(0, 8*pi, by = 0.1)
plot(x, sin(x), type = "l", xlab = "x", ylab = "Seno",
     main = "Función seno")
```

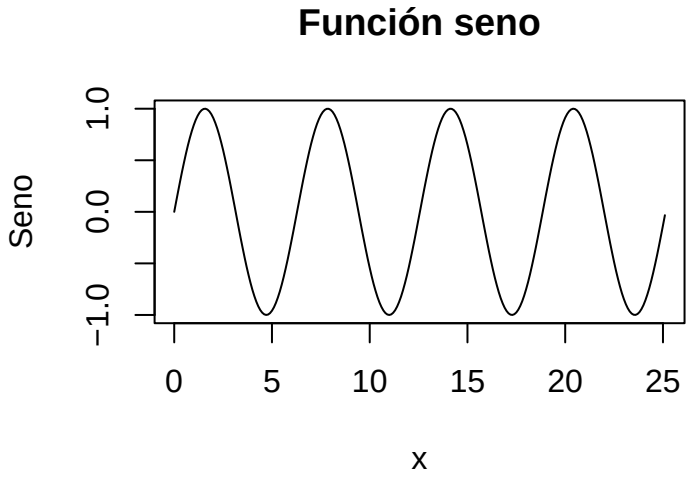


Gráfico de Violín

Para crear este tipo de gráficos, se pueden usar paquetes del ecosistema llamado ***tidyverse*** del cual se hablara más adelante.

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

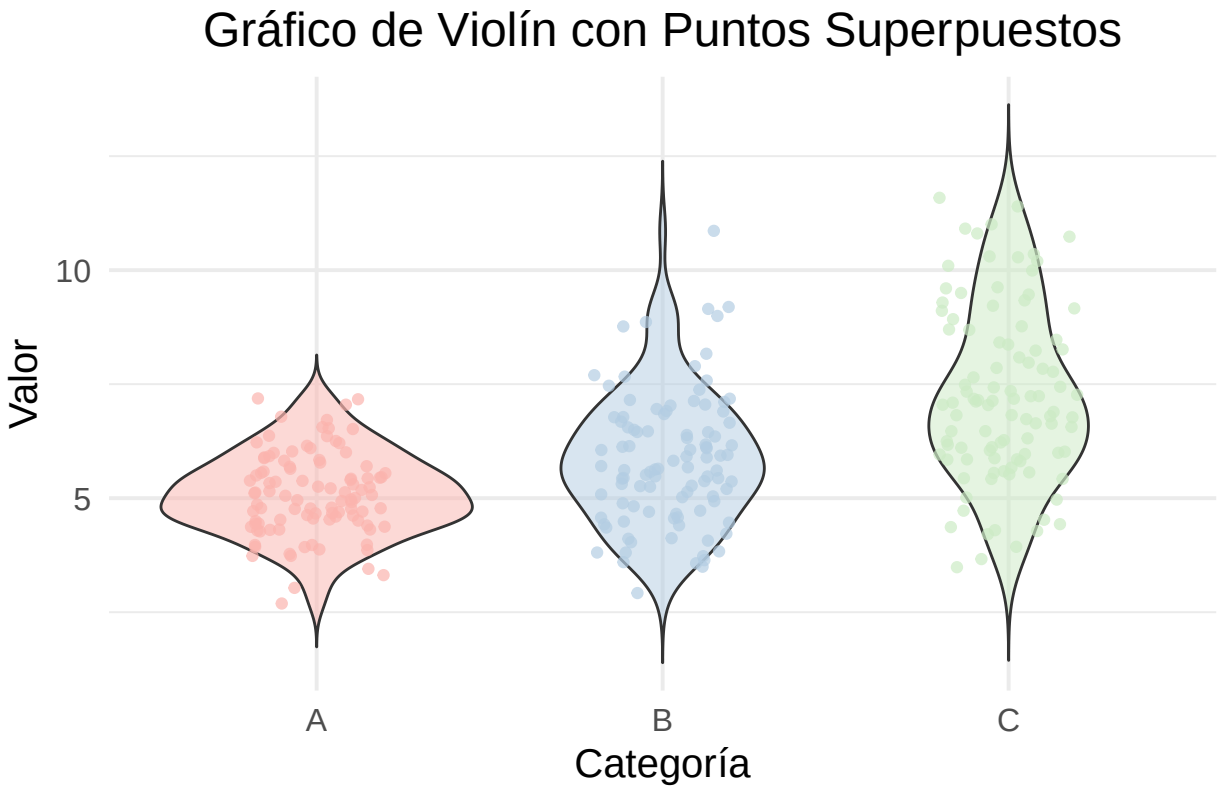
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
set.seed(123)
data <- data.frame(
  category = factor(rep(c("A", "B", "C"), each = 100)),
  value = c(rnorm(100, mean = 5, sd = 1),
            rnorm(100, mean = 6, sd = 1.5),
            rnorm(100, mean = 7, sd = 2))
)


p <- ggplot(data, aes(x = category, y = value, fill = category)) +
  geom_violin(trim = FALSE, alpha = 0.5) +
  geom_jitter(width = 0.2, size = 1.5, alpha = 0.7, aes(color = category)) +
  labs(
    title = "Gráfico de Violín con Puntos Superpuestos",
    x = "Categoría",
    y = "Valor",
    caption = "Fuente: Datos simulados"
  ) +
  theme_minimal(base_size = 15) +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    plot.caption = element_text(hjust = 1),
    legend.position = "none"
  ) +
  scale_fill_brewer(palette = "Pastel1") +
  scale_color_brewer(palette = "Pastel1")

# Mostrar la gráfica
print(p)
```



Fuente: Datos simulados

Gráfico de lineas

```
library(ggplot2)
library(dplyr)


set.seed(123)
date_seq <- seq(as.Date("2023-01-01"), by = "month", length.out = 12)
data <- data.frame(
  date = rep(date_seq, 3),
  value = c(cumsum(rnorm(12, mean = 5, sd = 2)),
            cumsum(rnorm(12, mean = 7, sd = 2)),
            cumsum(rnorm(12, mean = 6, sd = 2))),
  category = factor(rep(c("A", "B", "C"), each = 12))
)


p <- ggplot(data, aes(x = date, y = value, color = category, group = category)) +
  geom_line(size = 1.2) +
  geom_point(size = 3) +
  labs(
    title = "Tendencia Temporal de Múltiples Series",
    subtitle = "Ejemplo de visualización de datos temporales con ggplot2",
    x = "Fecha",
    y = "Valor Acumulado",
    color = "Categoría",
    caption = "Fuente: Datos simulados"
  ) +
  theme_minimal(base_size = 15) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 20, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 16),
    plot.caption = element_text(hjust = 1, size = 12),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12)
  ) +
  scale_color_manual(values = c("A" = "#FF5733", "B" = "#33FF57", "C" = "#3357FF"))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
print(p)
```

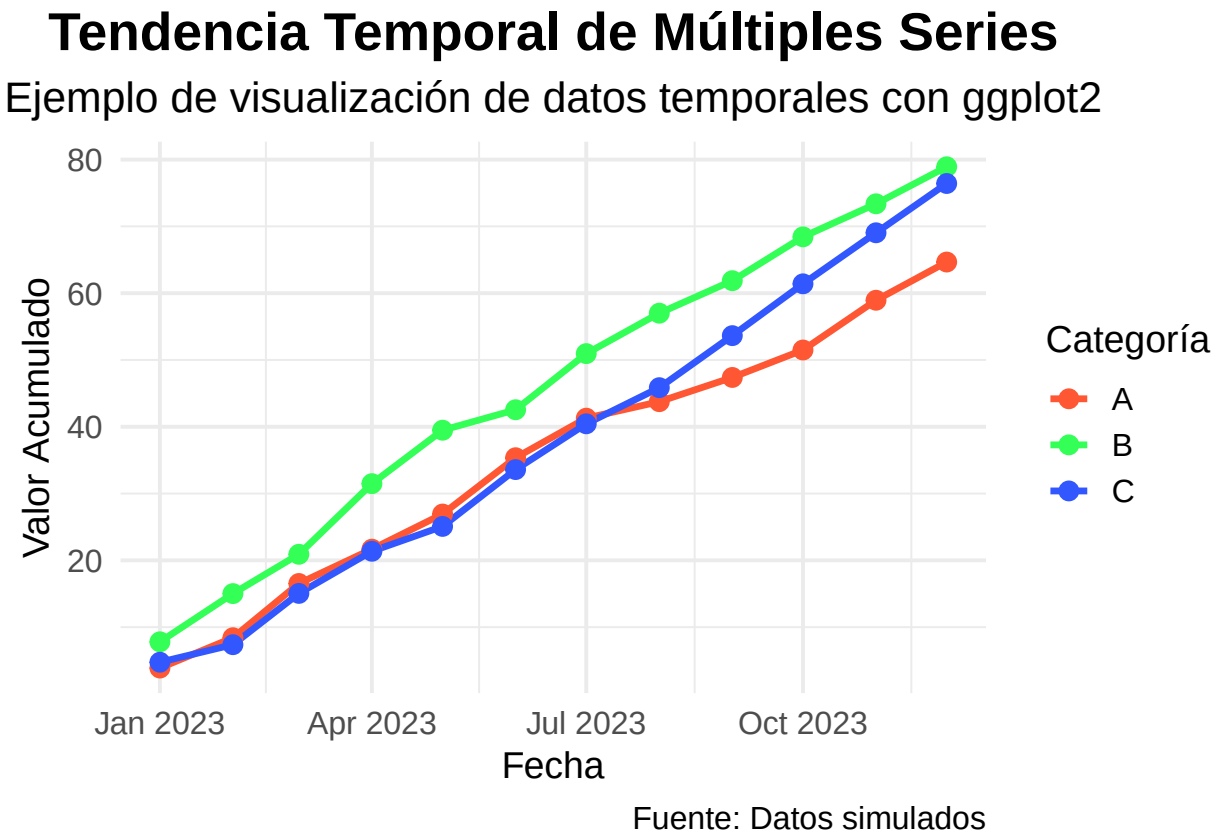
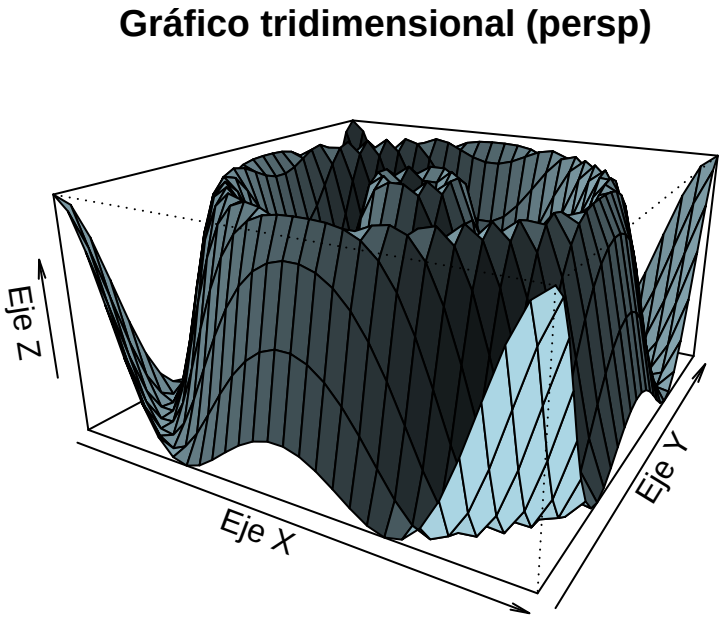

Tendencia Temporal de Múltiples Series
Ejemplo de visualización de datos temporales con ggplot2
80
60
40
20
Valor Acumulado
Jan 2023
Apr 2023
Jul 2023
Oct 2023
Fecha
Categoría
A
B
C
Fuente: Datos simulados

Gráfico tridimensional

```r
# Crear datos para la superficie
x <- seq(-10, 10, length.out = 30)
y <- seq(-10, 10, length.out = 30)
z <- outer(x, y, function(x, y) sin(sqrt(x^2 + y^2)))

# Graficar la superficie 3D
persp(
  x, y, z,
  main = "Gráfico tridimensional (persp)",
  xlab = "Eje X",
  ylab = "Eje Y",
  zlab = "Eje Z",
  col = "lightblue",
  theta = 30,    # Ángulo de rotación horizontal
  phi = 20,      # Ángulo de elevación
  expand = 0.5, # Escala vertical
  shade = 0.5    # Sombras
)
```



Este gráfico muestra una superficie tridimensional basada en la función $z = sen(\sqrt{x^2 + y^2})$.

Distribuciones empíricas

Supongamos que se obtener una distribución empírica para la variable altura de los hombres de la base de datos **medidas del cuerpo**.

Primero vamos a cargar los datos que vamos a usar:

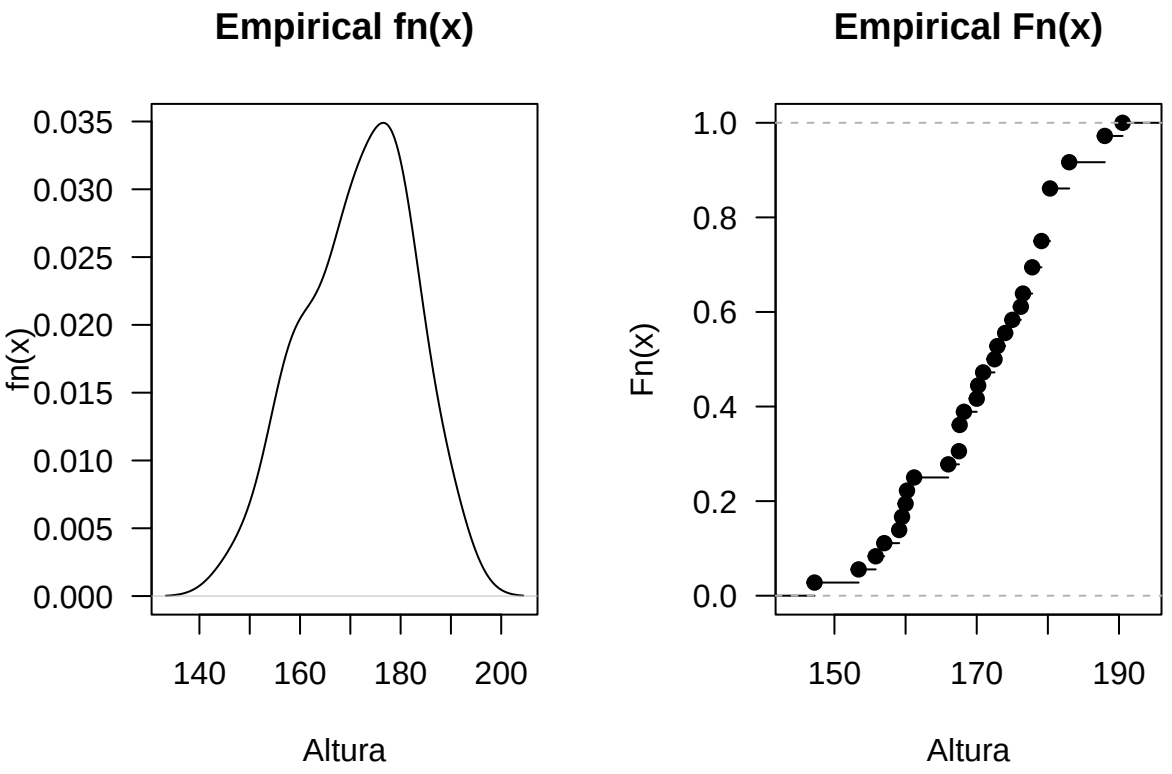
```
url <- 'https://raw.githubusercontent.com/fhernanb/datos/master/medidas_cuerpo'
datos <- read.table(file=url, header=T)
```

Solución

```
emp_f <- density(datos$altura)
emp_F <- ecdf(datos$altura)
```

Ahora vamos a dibujar $f_n(x)$ y $F_n(x)$

```
par(mfrow=c(1, 2))
plot(emp_f, las=1, main="Empirical fn(x)", xlab="Altura", ylab="fn(x)")
plot(emp_F, las=1, main="Empirical Fn(x)", xlab="Altura")
```



Intervalos de Confianza

Suponga que se quiere obtener un intervalo de confianza bilateral del 90% para la altura promedio de los hombres de la base de datos medidas del cuerpo.

Solución

Para calcular el intervalo de confianza, primero se carga la base de datos usando la url apropiada, luego se crea un subconjunto de datos y se aloja en el objeto hombres como sigue a continuación:

```
url <- 'https://raw.githubusercontent.com/fhernanb/datos/master/medidas_cuerpo'
datos <- read.table(file=url, header=TRUE)
hombres <- datos[datos$sexo=="Hombre", ]
```

Una vez leídos los datos, se analiza la normalidad de la variable altura de los hombres, a partir de un qqPlot y un histograma

```
par(mfrow=c(1, 2))
require(car)  # Debe instalar antes el paquete car
```

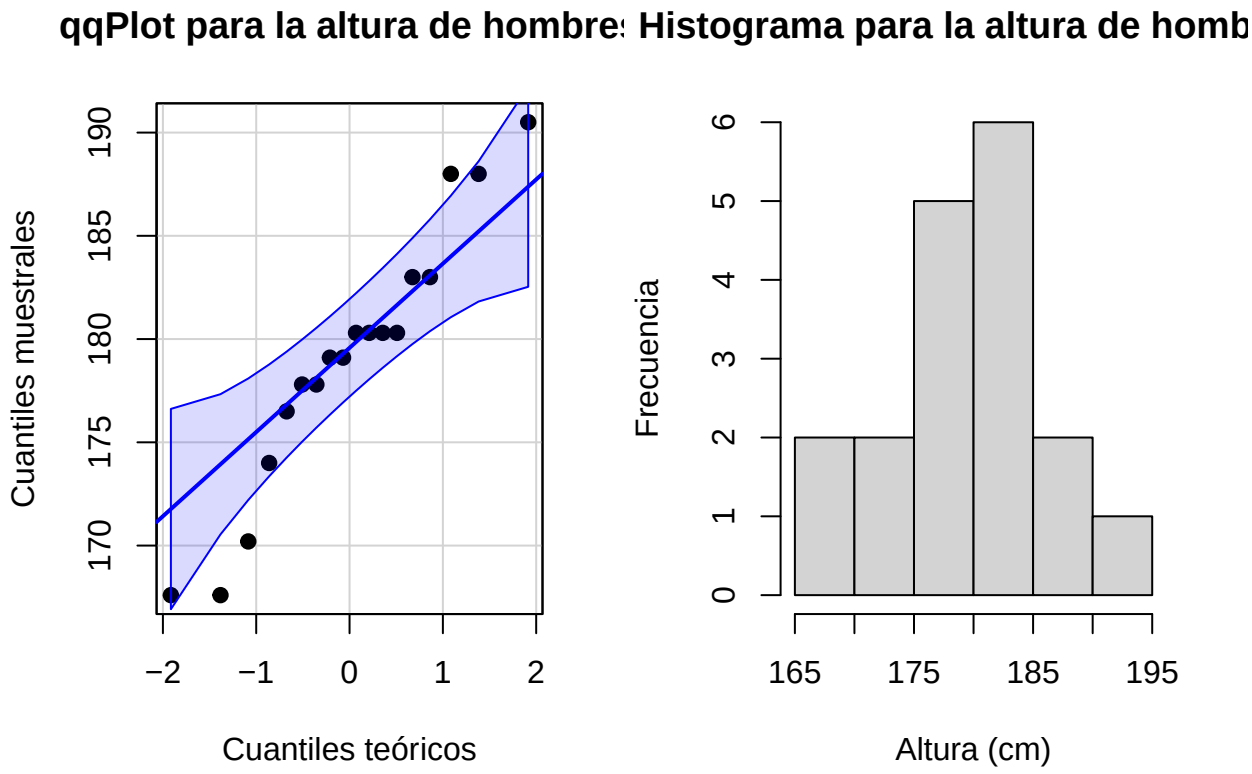
```
## Loading required package: car
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
qqPlot(hombres$altura, pch=19, id=FALSE,
       main='qqPlot para la altura de hombres',
       xlab='Cuantiles teóricos',
       ylab='Cuantiles muestrales')
hist(hombres$altura, freq=TRUE,
     main='Histograma para la altura de hombres',
     xlab='Altura (cm)',
     ylab='Frecuencia')
```



Entonces, en la respuesta se muestra el qqPlot e histograma para la variable altura, de estas figuras no se observa un claro patrón normal, sin embargo, al aplicar la prueba Shapiro-Wilk a la muestra de alturas de los hombres se obtuvo un valor-P de 0.3599, por lo tanto, se asume que la muestra de alturas provienen de una población normal.

Una vez chequeado el supuesto de normalidad se puede usar la función t.test sobre la variable de interés para construir el intervalo de confianza. El resultado de usar t.test es una lista, uno de los elementos de esa lista es justamente el intevalo de confianza y para extraerlo es que se usa $conf.int al final de la instrucción. A continuación se muestra el código utilizado.

```
res <- t.test(x=hombres$altura, conf.level=0.90)
res$conf.int
```

```
## [1] 176.4384 181.7172
## attr(,"conf.level")
## [1] 0.9
```

A partir del resultado obtenido se puede concluir, con un nivel de confianza del 90%, que la altura promedio de los estudiantes hombres se encuentra entre 176.4 cm y 181.7 cm.

Capítulo 3: Abrir R por primera vez.

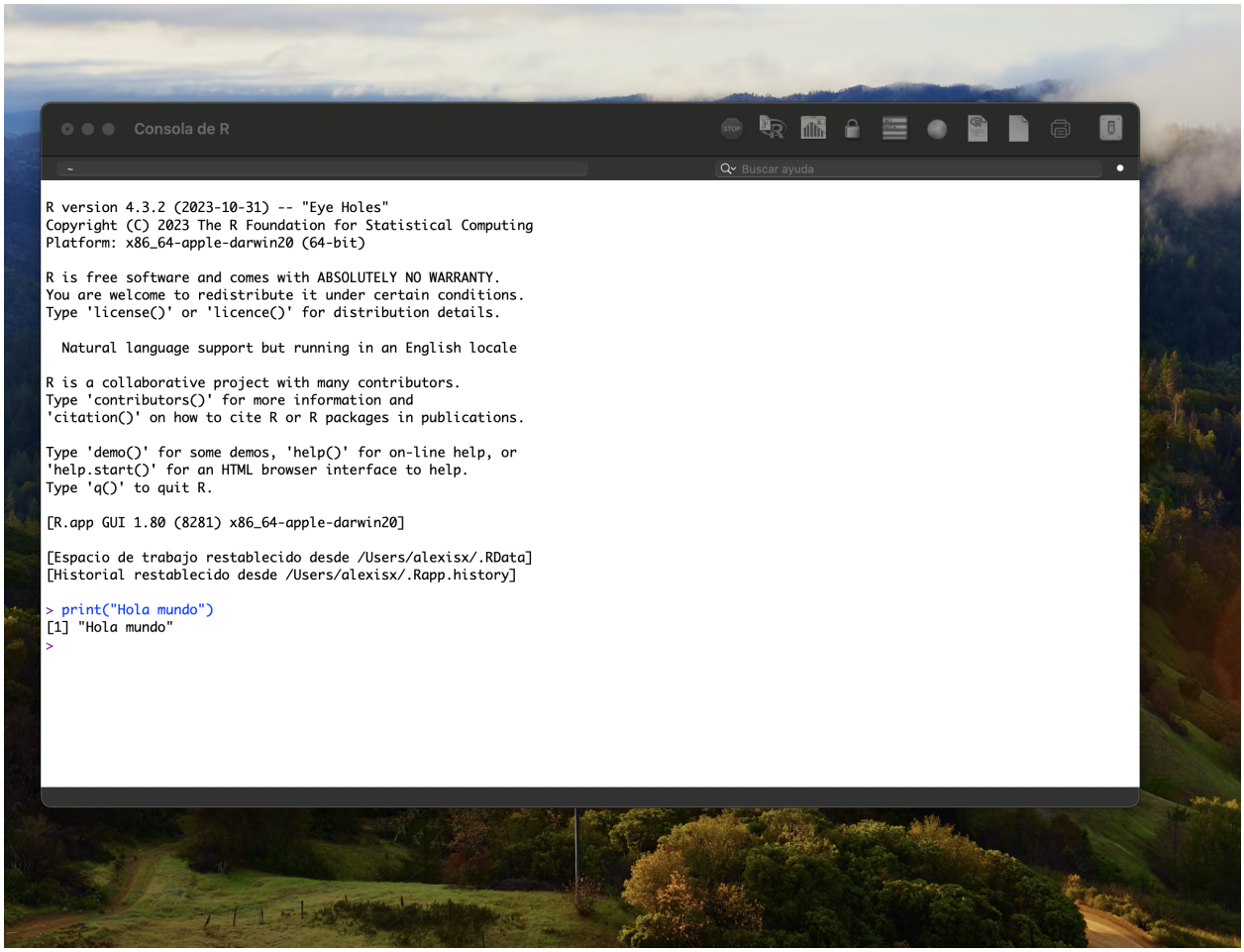
Una vez que hayamos instalado R en nuestro sistema operativo, abramos la consola y ejectuemos nuestro primer comando paso a paso.

1. Localiza donde está guardado el programa de R y abrelo. Notese que una vez abierto nos abrira una consola donde se puede empezar a escribir comandos y generar respuestas.
2. Una vez abierta la consola ejecute el siguiente comando:

```
print("Hola mundo")
```

```
## [1] "Hola mundo"
```

Note como en este ejemplo, al ejecutar la función print() y escribir el texto entre comillas *"Hola mundo"* la consola unicamente nos regresa el texto dentro de la función.



Con esto empezamos nuestro camino para usar R como una herramienta :)