Universdad Nacional de Colombia Sede De La Paz

Taller 1

Programacion en Lenguajes Estadisticos

- Ronald Mateo Ceballos Lozano

---

1. Resumen/Summary

1. El lenguaje de programación R es un proyecto de código abierto gratuito para computación estadística. Compila para plataformas tipo UNIX y varias versiones de macOS y Windows. Para ejecutar la última versión, necesitarás Windows 7 o posterior o Mac OS X 10.6 y superior. En este tutorial aprenderás cómo instalar R y RStudio.

2. The R programming language is a free open source project for statistical computing. It compiles for UNIX-like platforms and various versions of macOS and Windows. To run the latest version, you will need Windows 7 or later or Mac OS X 10.6 and higher. In this tutorial you will learn how to install R and RStudio.

2. Instalando R

El primer paso para convertirte en un analista o científico de datos es tener R instalado en tu ordenador. R se puede descargar libremente de su página oficial. Ten en cuenta que la instalación requiere hasta 150 MB de espacio de almacenamiento en disco disponible. La página web oficial del proyecto R es la siguiente:

$$https://www.r-project.org/$$

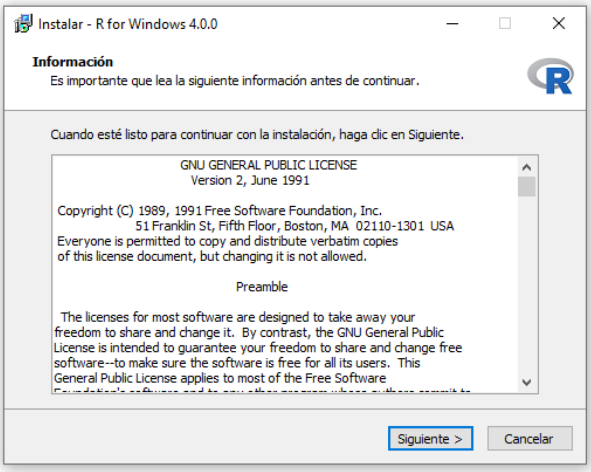
En el enlace anterior puedes encontrar información interesante sobre el proyecto R, la Fundación R, los lanzamientos de nuevas versiones, documentación y otros enlaces interesantes.

3. Instalando R en Windows

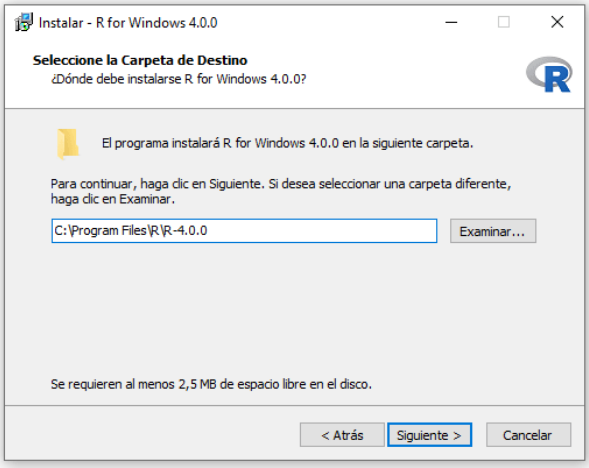
Para instalar R en Windows, puedes acceder al siguiente enlace y hacer clic en “Download R x.x.x for Windows” para comenzar a descargar la última versión de R disponible.

$$https://cran.r-project.org/bin/windows/base/$$

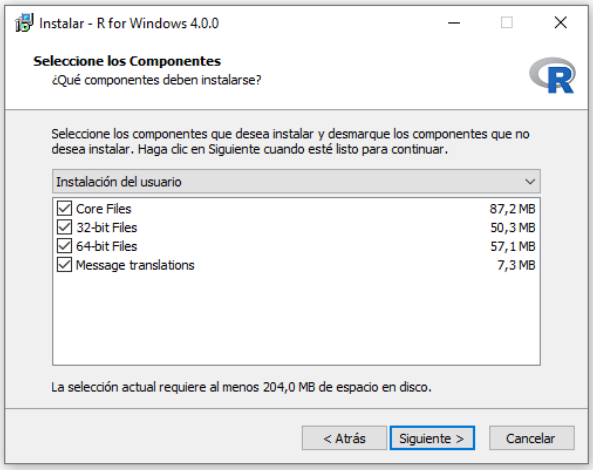
Una vez descargado, ábrelo, selecciona el idioma preferido y haz clic en “Siguiente” a todos los cuadros de diálogo.



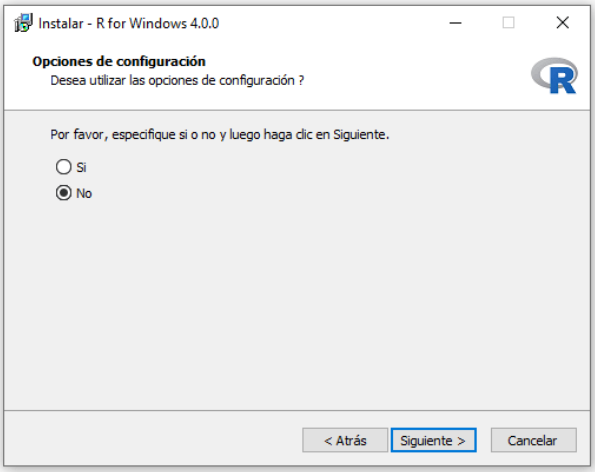
Recuerda seleccionar la ruta donde quieres instalar R:

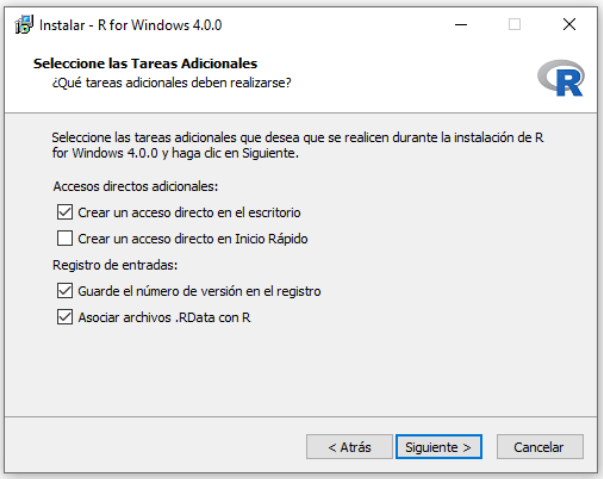


Ten en cuenta que puedes desmarcar los archivos de bits que no corresponden a tu PC:



También puedes especificar personalizar algunas opciones de inicio, aunque podrás modificarlas una vez instalado R si lo prefieres:





Cuando finalice la instalación, podrás comenzar a utilizar R base en Windows.

4. Instalando R en MacOs

El proceso de instalación de R en una Mac es análogo a la instalación en Windows. Puedes descargar el paquete binario (para Mac OS X 10.6 y superior) desde la siguiente URL y luego instalarlo.

$$https://cran.r-project.org/bin/macosx/$$

Si por alguna razón estás utilizando versiones de Mac OS 8.6 a 9.2 (y Mac OS X 10.1) puedes encontrar aquí los paquetes binarios antiguos. Para sistemas Mac OS X y PowerPC Macs anteriores, debes usar este repositorio para descargar R.

5. Instalar RStudio

RStudio es el IDE más popular para R, proporcionando una interfaz gráfica limpia y útil para desarrollar código R. Las principales ventajas de este IDE son el panel para obtener ayuda y mostrar gráficos (incluso interactivos), crear documentos RMarkdown y Sweave, entre otros. Puedes descargar RStudio desde el siguiente enlace, seleccionando el archivo correspondiente a tu sistema operativo:

$$https://rstudio.com/products/rstudio/download/$$

Una vez descargado, abre el asistente de configuración y sige los sencillos pasos de instalación.

6. R en la web(Online)

R está pensado para explotar su potencial que es la “estadística”. Este fantástico lenguaje nos permite una primera toma de contacto con los datos debido a su flexibilidad por la exploración, limpieza y análisis a diferentes fuentes de datos, así como aplicar modelos y algoritmos predictivos puede ser de gran ayuda en el mundo de la análisis de datos.

Al ser un lenguaje muy famoso alrededor del mundo, existe paginas web donde puedes trabajar con el lenguaje R sin necesidad de descargarlo en tu equipo o siempre tener que instalarlo en equipos que no lo tienen.

Esta alternativa se llama ”paiza.io”:

$$https://paiza.io/es/languages/r$$

¿Qué es paiza.IO?

Paiza.IO es un editor y compilador en línea donde puedes escribir y ejecutar código instantáneamente. Ya sea que tengas una nueva idea, estés aprendiendo o enseñando programación, tú y los demás simplemente pueden escribir y ejecutar código

7. Estructuras Basicas de R

Vectores en R

Un vector en R es un conjunto de objetos del mismo tipo concatenados con la función $[c]$. Puedes crear vectores con componentes lógicos, de caracteres, numéricos o complejos, entre otros. Los diferentes términos del vector se denominan componentes. Es importante destacar que puedes verificar la clase de un vector con la función $[class]$ y el tipo de elementos del mismo con la función $[typeof]$.

¿Cómo crear un vector?

La forma más sencilla de crear un vector en R es usando la función $[c]$, que se utiliza para la concatenación de objetos. Puedes guardar en memoria el vector asignándole un nombre con el operador $[<-]$.

```r
# Creando vectores en R con la función 'c'
x <- c(12, 6, 67)
y <- c(2, 13)
y
```

Esto nos genera el siguiente vector:

```
Output

2 13
```

Los vectores también pueden ser no numéricos. Por lo tanto, puedes crear vectores con caracteres, objetos lógicos u otros tipos de objetos de datos, como ya se comentó en la introducción.

```r
caracteres <- c("Madrid", "Barcelona", "Pontevedra", "Murcia")
class(caracteres) # "character"

logico <- c(TRUE, TRUE, FALSE, TRUE)
class(logico) # "logical"
```

Sin embargo, si mezclas distintos tipos de datos dentro de un vector los componentes se transformarán en elementos del mismo tipo. A esto se le llama coerción de tipos de datos.

```r
mezcla <- c(TRUE, "Correcto", 8, 2.2)
mezcla # "TRUE" "Correct"  "8"  "2.2"

class(mezcla)  # "character"
typeof(mezcla) # "character"
```

Ordenando Vectores

Para ordenar un vector, puedes llamar a la función [*sort*] pasando como argumento el vector. Por defecto, la función ordena de forma ascendente.

```r
z <- c(12, 15, 3, 22)
sort(z)
```

Esto nos genera el siguiente vector:

```
Output

3 12 15 22
```

También puedes ordenar los datos en orden decreciente estableciendo el argumento [*decreasing*] como [*TRUE*]:

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

Esto nos genera el siguiente vector:

```
Output

22 15 12 3
```

La función order

Alternativamente, puedes usar corchetes y ordenar los componentes del vector como un índice haciendo uso de la función [*order*] como sigue:

```
# Orden creciente
z[order(z)]  # Equivalente a sort(z)

# Orden decreciente
z[order(-z)] # Equivalente a sort(z, decreasing = TRUE)
```

Concatenar Vectores

Unir dos o más vectores es tan fácil como crear uno. De hecho, solo necesitas llamar a la función [*c*] y pasar los vectores como argumentos para concatenar un vector a otro.

```
x <- c(1, 2, 3)
y <- c(4, 5, 6)
c(x, y)
```

Esto nos genera el siguiente vector:

```
Output

1 2 3 4 5 6
```

Es necesario destacar que el orden de los elementos es relevante. Los vectores se concatenarán según el orden de entrada:

```
c(y, x)
```

Esto nos genera el siguiente vector:

```
Output

4 5 6 1 2 3
```

Crear un vector vacio en R

En ocasiones es necesario inicializar un vector vacío en R y llenarlo dentro de un bucle [*for*]. Con tal objetivo puedes utilizar la función [*c*] sin especificar ningún argumento para crear la estructura vacía. Si lo prefieres también puedes usar la función [*vector*].

```
# Vector vacío
mi_vector <- c()

# Llenando el vector con un bucle
for(i in 1:10) {
  mi_vector[i] <- i
}

mi_vector
```

Esto nos genera el siguiente vector:

```
Output

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

Comparación Dos Vectores

Existen varias formas de comparar vectores en R. Por una parte podemos comparar los elementos uno a uno con algún operador lógico. Ten en cuenta que si un vector es mayor que otro el número de elementos tienen que ser múltiplos entre ellos o surgirá un error.

```
x <- c(1, 5)
y <- c(4, 0)
x > y # FALSE TRUE

x <- c(1, 5)
y <- c(4, 0, 1, 3)

# Esto compara 1 > 4, 5 > 0, 1 > 1 y 5 > 3
x > y # FALSE TRUE FALSE TRUE

x <- c(1, 5, 1)
y <- c(4, 0, 1, 3)
x > y # Error
```

También podemos comprobar si los elementos del primer vector están contenidos de la siguiente forma:

```
x %in% y # TRUE FALSE TRUE
```

Otra opción es devolver los elementos comunes entre el primer vector y el segundo:

```
# Devolvemos los elementos comunes
x[x %in% y] # 1 1
```

Por último podríamos comparar si todos los elementos del primer vector están en el segundo con la función [*all*] de la siguiente manera:

```
x <- c(1, 5)
y <- c(4, 5, 1, 3)

all(x %in% y) # TRUE
```

Para aprender un poco mas acerca de los vectores, puedes dirigirte a este link acontinuación y practicar con mas funciones que R nos permite realizar con vectores:

https : //r − coder.com/vectores − r/

Matrices en R

Una matriz en R es una estructura de datos para almacenar objetos del mismo tipo. Si quieres almacenar diferentes objetos dentro de una estructura de datos en R, usa un data frame en su lugar. En este tutorial mostraremos los principales usos y cálculos con matrices en R..

¿Cómo crear una matriz en R?

La función [*matrix*] permite una matriz en RStudio o R base, pasando como input un vector numérico, de caracteres o lógico.

```
data <- 1:6

# Creando la matriz
matrix(data)
```

Esto nos da como resultado:

```
Output

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

Como se puede apreciar en la salida, por defecto se creará una matriz de una columna y tantas filas como la longitud del vector. Sin embargo, es posible establecer el número de columnas o el número de filas con los argumentos [*ncol*] y [*nrow*], respectivamente. También puedes especificar si la matriz está ordenada por filas o por columnas con el argumento byrow. Por defecto, la función ordenará la entrada por columnas [$(byrow = FALSE)$].

```
# Por columnas
matrix(data, ncol = 2, byrow = FALSE) # byrow = FALSE por defecto
matrix(data, ncol = 2, nrow = 3) # Equivalente
matrix(data, nrow = 3) # Equivalente

# Por filas
matrix(data, ncol = 2, byrow = TRUE)
```

Esto nos da como resultado:

Output

```
# Por columnas
     [, 1] [, 2]
[1, ]    1    4
[2, ]    2    5
[3, ]    3    6
```

```
# Por filas
      [, 1] [, 2]
[1, ]    1    2
[2, ]    3    4
[3, ]    5    6
```

Ahora bien, si tienes datos almacenados en vectores o en las columnas de un data frame, puedes usar [*cbind*] para concatenar columnas o [*rbind*] para concatenar filas, siendo el output una matriz. La clase de la salida se puede comprobar con la función [*class*] y la clase de los elementos con la función [*typeof*].

```r
x <- c(2, 7, 3, 6, 1)
y <- c(3, 7, 3, 5, 9)

# Por columnas
cbind(x, y)

# Por filas
rbind(x, y)

# Clase de la salida
class(cbind(x, y))  # "matrix"

# Tipo de dato de los elementos
typeof(cbind(x, y)) # "double"
```

Output

```
# Por columnas
     x  y
[1, ] 2  3
[2, ] 7  7
[3, ] 3  3
[4, ] 6  5
[5, ] 1  9
```

```
# Por filas
  [, 1] [, 2] [, 3] [, 4] [, 5]
x    2    7    3    6    1
y    3    7    3    5    9
```

Ten en cuenta que puedes usar cualquier tipo de datos dentro de una matriz, siempre que sean homogéneos.

```r
matrix(c(TRUE, TRUE, FALSE, TRUE), ncol = 2)
matrix(c("rojo", "verde", "naranja", "negro"), ncol = 2)
```

```
Output

     [, 1]  [, 2]              [, 1]    [, 2]
[1, ] TRUE FALSE         [1, ] "rojo"  "naranja"
[2, ] TRUE  TRUE         [2, ] "verde" "negro"
```

Además, puedes conocer las dimensiones de la matriz con la función [*dim*].

```r
matriz <- matrix(1:12, ncol = 2, byrow = FALSE)

# Dimensiones de la matriz
dim(matriz) # 6 2
```

El primer número de la salida de la función dim indica el número de filas (6) y el segundo el número de columnas (2). Ten en cuenta que la función dim también se puede utilizar para crear una matriz en R.

```r
A <- c(3, 1, 6, 1, 2, 9)
dim(A) <- c(3, 2)
```

```
Output

     [, 1] [, 2]
[1, ]    3     1
[2, ]    1     2
[3, ]    6     9
```

A modo esquemático, la siguiente tabla muestra las funciones más comunes relacionadas con matrices.

Función	Descripción
dim(), nrow(), ncol()	Número de filas/columnas
diag()	Diagonal de una matriz
*	Multiplicación elemento a elemento
%*%	Producto matricial
%o%	Producto exterior
%x%	Producto de Kronecker
cbind(), rbind()	Agregar filas/columnas
t()	Matriz traspuesta
solve(A)	Inversa de la matriz A
solve(A, b)	Solución a $Ax = b$
eigen()	Autovalores y autovectores
qr()	Descomposición QR
chol()	Descomposicion de Cholesky
svd()	Descomposición singular

Dataframes en R

¿Que es un data frame en R?

Los data frames (marcos de datos) son el objeto más habitual para almacenar datos en R. En este tipo de objeto, cada individuo o fecha corresponde a una fila y cada columna corresponde a una variable. Dentro de este tipo de estructura puedes almacenar diferentes tipos de datos.

¿Cómo crear un data frame en R?

En R es muy sencillo crear un nuevo data frame. Puedes unir tus variables haciendo uso de la función data.frame para convertir tus datos a la estructura de datos de tipo [$dataframe$]. Primero, necesitas tener algunas variables almacenadas para crear el marco de datos en R. En este ejemplo vamos a definir algunas variables de datos meteorológicos. Ten en cuenta que todos los vectores tienen la misma longitud.

```
temp <- c(20.37, 18.56, 18.4, 21.96, 29.53, 28.16,
          36.38, 36.62, 40.03, 27.59, 22.15, 19.85)
humedad <- c(88, 86, 81, 79, 80, 78,
             71, 69, 78, 82, 85, 83)
precipitaciones <- c(72, 33.9, 37.5, 36.6, 31.0, 16.6,
                     1.2, 6.8, 36.8, 30.8, 38.5, 22.7)
mes <- c("enero", "febrero", "marzo", "abril", "mayo", "junio",
         "julio", "agosto", "septiembre", "octubre", "noviembre", "diciembre")
```

Para unir los datos puedes usar la función [$data.frame$]. En este ejemplo, vamos a almacenar el data frame en una variable llamada [$datos$]:

```r
datos <- data.frame(mes = mes, temperatura = temp, humedad = humedad,
                    precipitaciones = precipitaciones)
names(datos) # Nombres de las variables (columnas)
```

```
"mes"  "temperatura"  "humedad"  "precipitaciones"
```

Primero, es habitual mostrar los primeros valores para hacer algunas comprobaciones. Para ello, puedes hacer uso de la función [*head*] en R, que por defecto mostrará las 6 primeras filas del data frame.

```r
# Primeras filas de nuestro conjunto de datos
head(datos)
```

Output

	mes	temperatura	humedad	precipitaciones
1	enero	20.37	88	72.0
2	febrero	18.56	86	33.9
3	marzo	18.40	81	37.5
4	abril	21.96	79	36.6
5	mayo	29.53	80	31.0
6	junio	28.16	78	16.6

En segundo lugar, puedes utilizar la función [*summary*] que devolverá un resumen estadístico de las variables (columnas) del conjunto de datos.

```r
summary(datos)
```

Resumen descriptivo de los datos

mes	temperatura	humedad	precipitaciones
april :1	Min. :18.40	Min. :69.0	Min. : 1.20
august :1	1st Qu.:20.24	1st Qu.:78.0	1st Qu.:21.18
december:1	Median :24.87	Median :80.5	Median :32.45
february:1	Mean :26.63	Mean :80.0	Mean :30.37
january :1	3rd Qu.:31.24	3rd Qu.:83.5	3rd Qu.:36.98

También puedes hacer uso de los data frames de ejemplo que R proporciona. Para buscarlos, puedes llamar a la función [*data*]:

```r
data()
```

Una vez ejecutado, se abrirá una ventana con una lista de conjuntos de datos disponibles:

```
Data sets in package "datasets":
 AirPassengers Monthly Airline Passenger Numbers 1949-1960
 BJsales Sales Data with Leading Indicator
 …
```

Ahora puedes cargar el que quieras con:

```
data(nombre_del_conjunto_de_datos)
```

Como ejemplo, si quieres cargar el conjunto de datos AirPassengers en el espacio de trabajo puedes escribir lo siguiente:

```
data(AirPassengers)
```

Accediendo a los datos del data frame

Hay varias formas de acceder a las columnas almacenadas en los data frames:

[1.] Usando el signo del dollar y el nombre de la columna.

[2.] Usando corchetes con el índice de la columna después de una coma.

Como ejemplo, si quieres seleccionar la columna mes del data frame que creamos antes, ejecuta lo siguiente:

```
datos$mes
datos[, 1] # Equivalente
```

También puedes seleccionar varias variables a la vez. Para ello puedes:

[1.] Crear una secuencia de índices.

[2.] Crear un vector con la función $[c]$ con los nombres de las variables o índices que quieras seleccionar.

```
# Seleccionando las columnas 1 a 3 con una secuencia
datos[, 1:3]

# Seleccionando columnas con la función 'c'
datos[, c("temperatura", "precipitaciones")]
datos[, c(2, 4)] # Equivalente
```

Del mismo modo, puedes acceder a filas del data frame con $[datos[1,]]$ o $[datos[1:2,]]$ para seleccionar la primera fila o la primera y la segunda, o seleccionar solo algunos puntos de datos seleccionando filas y columnas a la vez:

```r
# Observación de la primera
# fila y segunda columna
data[1, 2]

# Primera y segunda fila
# de la segunda columna
data[1:2, 2]
```

Añadir columnas y filas a un data frame

A veces necesitas modificar los datos para agregar nuevas filas o columnas, o eliminarlas. Para los siguientes ejemplos, utilizaremos el conjunto de datos [*cars*], registrado en la década de 1920, que forma parte de los conjuntos de datos de ejemplo de R. Puedes cargarlo ejecutando [*data*(*cars*)]. La base de datos contiene 50 filas y 2 variables:

[1.] speed: velocidad (mph).

[2.] dist: distancia de frenado (ft).

Si ejecutas [*head*(*cars*)] obtendrás el siguiente resultado:

Output of head(cars)

	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10

Supón que quieres crear una nueva variable para transformar la velocidad en kilómetros por hora (km/h) y la distancia en metros. Recuerda que:

$$kilómetro = milla/0.62137 y metro = pie/3.2808$$

Entonces, ahora podrías agregar dos nuevas columnas llamadas [*kph*] (kilómetros por hora) y [*meters*] (metros) con el siguiente código:

```r
cars$kph <- cars$speed / 0.62137
cars$meters <- cars$dist / 3.2808
```

También puedes hacer uso de la función [*cbind*]. En caso de que quisieses agregar una nueva fila, podrías usar la función [*rbind*].

```
kph <- cars$speed / 0.62137
meters <- cars$dist / 3.2808
cars <- cbind(cars[, c(1, 2)], kph, meters)
```

Resultando en:

Output de head(cars) con las nuevas variables

	speed	dist	kph	meters
1	4	2	6.437388	0.6096074
2	4	10	6.437388	3.0480371
3	7	4	11.265430	1.2192148
4	7	22	11.265430	6.7056815
5	8	16	12.874777	4.8768593
6	9	10	14.484124	3.0480371

Agrega nuevas filas con la función rbind y nuevas columnas con la función cbind.

Eliminar columnas y filas a un data frame

Ahora, si quieres borrar variables o filas de un data frame, tienes varias opciones:

[1.] Usar el signo menos (-) e indicar las columnas o filas que quieras borrar.

[2.] Crear un subconjunto de los datos que quieras conservar.

Como ejemplo, borraremos las variables [*speed*] y [*disty*] para evitar sobrescribir el conjunto de datos original guardaremos nuestros resultados en un nuevo data frame llamado [*cars*2].

```
# Borrando la primera y segunda columna con el signo -
cars2 <- cars[, -c(1, 2)]

# Seleccionar solo las columnas que queremos conservar
cars2 <- cars[, c("kph", "meters")]
```

Si vuelves a utilizar la función head, podrás ver el nuevo data frame.

```
head(cars2)
```

Output of head(cars2)

	kph	meters
1	6.437388	0.6096074
2	6.437388	3.0480371
3	11.265430	1.2192148
4	11.265430	6.7056815
5	12.874777	4.8768593
6	14.484124	3.0480371

8. Ejemplo con Visualización de Datos

Veamos ahora un ejemplo más avanzado. Vamos a simular una variable aleatoria normal de tamaño 200 con media de 105 y desviación típica de 2. Luego, haremos un resumen, un histograma y el diagrama de caja correspondiente.

```r
# Semilla para reproductibilidad
set.seed(1)

# Generando los datos
x <- rnorm(n = 200, mean = 105, sd = 2)

# Primeros elementos de los datos
head(x)

# Resumen estadístico de los datos
summary(x)
```

Al ejecutar esto nos da como resultado:

```
Output

107.1996 105.7260 105.0169 104.4441 105.6405 104.2416

 Min.   1st Qu.   Median  Mean   3rd Qu.    Max.
100.5   103.9     105.3   105.4   106.9    111.1
```

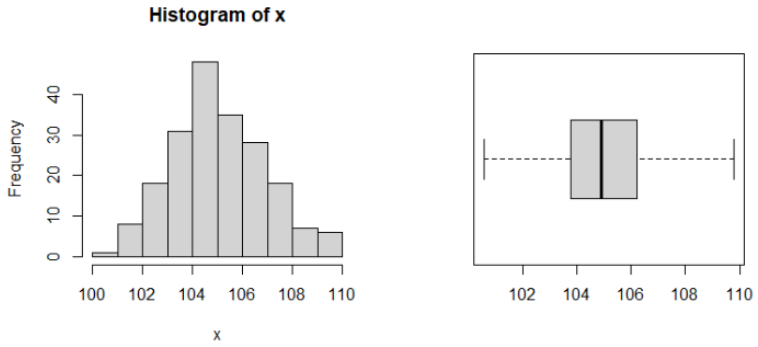
Para graficar los resultados tenemos el siguiente codigo:

```r
# Dividiendo la pantalla gráfica en dos columnas
par(mfcol = c(1, 2))

# Dibujando el histograma
hist(x)

# Dibujando el gráfico de cajas
boxplot(x, horizontal = TRUE)
```

Esto nos genera las siguientes graficas:



9. Bibliografía

Referencias

[1] ¿Cómo instalar R?, R CODER Online, Recuperado el 12 de mayo de 2022: https://r-coder.com/instalar-r/

[2] Vectores en R, R CODER Online, Recuperado el 12 de mayo de 2022: https://r-coder.com/vectores-r/

[3] Matrices en R, R CODER Online, Recuperado el 12 de mayo de 2022: https://r-coder.com/matrices-r/

[4] Data frame en R, R CODER Online, Recuperado el 12 de mayo de 2022: https://r-coder.com/data-frame-en-r/

[5] R Online, Paiza,Io, Recuperado el 12 de mayo de 2022: https://paiza.io/es/languages/r