Curso de R Nivel Intermedio

Luis Manuel Rodarte Solórzano

January 02, 2023

1. Objetos en R

1.1 Vectores

1.1.1. Operación potencia

Para declarar la potencia de un vector en R, se utiliza el símbolo ^ y seguido se escribe el número que respresenta la potencia. Por ejemplo:

```
z <- c(2,3,5,7,11)
z^2  # Retorna un vector que contiene cada elemento del vector z al cuadrado
```

```
## [1]   4   9  25  49 121
```

Ejemplo 2:

```
y <- c(1,3,5,7,9,11)
y^7
```

```
## [1]        1     2187    78125   823543  4782969 19487171
```

Ejemplo 3:

```
w <- c(0,1,1,2,3,5,8,13,21)
y^0.4
```

```
## [1] 1.000000 1.551846 1.903654 2.177906 2.408225 2.609499
```

Ejemplo 4:

```
w <- c(0,1,1,2,3,5,8,13,21)
y^0.4
```

```
## [1] 1.000000 1.551846 1.903654 2.177906 2.408225 2.609499
```

Ejemplo 5:

```
w <- c(0,1,1,2,3,5,8,13,21)
y^(1/3)
```

```
## [1] 1.000000 1.442250 1.709976 1.912931 2.080084 2.223980
```

Ejemplo 6:

```
w <- c(2,3,5,7,11,13,17)
y^(-2/5)
```

```
## [1] 1.0000000 0.6443940 0.5253056 0.4591565 0.4152436 0.3832154
```

1.1.2. Función seq()

Para generar vectores se utiliza la función `seq()`, esta función permite crear secuencias personalizadas, por ejemplo:

```
# Genera un vector que inicia desde el 1 y vaya aumentando de 2 en 2 hasta llegar a 30
seq(1,30, by = 2)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
```

Ejemplo 2:

```
# Genera un vector de 5 números iniciando desde el 1 y terminando en 10
seq(1,10,length.out = 5)
```

```
## [1]  1.00  3.25  5.50  7.75 10.00
```

Ejemplo 3:

```
# Genera un vector de 25 números iniciando desde el 0 y terminando en 10.
seq(0,100,length.out = 25)
```

```
##  [1]   0.000000   4.166667   8.333333  12.500000  16.666667  20.833333
##  [7]  25.000000  29.166667  33.333333  37.500000  41.666667  45.833333
## [13]  50.000000  54.166667  58.333333  62.500000  66.666667  70.833333
## [19]  75.000000  79.166667  83.333333  87.500000  91.666667  95.833333
## [25] 100.000000
```

Ejemplo 4:

```
seq(10,0,by=-0.1)
```

```
##   [1] 10.0  9.9  9.8  9.7  9.6  9.5  9.4  9.3  9.2  9.1  9.0  8.9  8.8  8.7  8.6
##  [16]  8.5  8.4  8.3  8.2  8.1  8.0  7.9  7.8  7.7  7.6  7.5  7.4  7.3  7.2  7.1
##  [31]  7.0  6.9  6.8  6.7  6.6  6.5  6.4  6.3  6.2  6.1  6.0  5.9  5.8  5.7  5.6
##  [46]  5.5  5.4  5.3  5.2  5.1  5.0  4.9  4.8  4.7  4.6  4.5  4.4  4.3  4.2  4.1
##  [61]  4.0  3.9  3.8  3.7  3.6  3.5  3.4  3.3  3.2  3.1  3.0  2.9  2.8  2.7  2.6
##  [76]  2.5  2.4  2.3  2.2  2.1  2.0  1.9  1.8  1.7  1.6  1.5  1.4  1.3  1.2  1.1
##  [91]  1.0  0.9  0.8  0.7  0.6  0.5  0.4  0.3  0.2  0.1  0.0
```

1.1.3. Generación de vectores

Para generar una secuencia donde la diferencia sea uno, se utiliza el siguiente código:

```
# Genera un vector que inicia desde el 1 y va aumentando de 1 en 1 hasta llegar a 30
1:30
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30
```

Ejemplo 2:

```
# Genera un vector que inicia desde del 0.4 y va aumentando de 1 en 1 hasta llegar a 10.4
0.4:10.6
```

```
##  [1]  0.4  1.4  2.4  3.4  4.4  5.4  6.4  7.4  8.4  9.4 10.4
```

Ejemplo 3:

```
# Genera un vector que inicia desde el 0.3 y va aumentando de 1 en 1 hasta llegar a 9.5
0.3:9.5
```

```
##  [1] 0.3 1.3 2.3 3.3 4.3 5.3 6.3 7.3 8.3 9.3
```

1.1.4. Función `rep()`

La función rep() permite repetir elementos, por ejemplo:

```
# Genera un vector que contiene 10 veces el número 2
rep(2,10)
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2
```

Ejemplo 2:

```
rep(1:5,4)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

Ejemplo 3:

```
rep(c(1,2,3),6)
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

Ejemplo 4:

```
# Repite cada elemento del vector dos veces
rep(1:4, each = 2)
```

```
## [1] 1 1 2 2 3 3 4 4
```

1.2. Matrices

1.2.1. Generar matrices

Existen dos formas de declarar matrices en R, una por columnas y la otra por filas. Considere la matriz $A = \begin{pmatrix} 2 & 3 \\ 5 & 8 \\ 13 & 21 \end{pmatrix}$, a continuación, se presentan los datos de las matrices ordenados por columnas y por filas:

```
# Datos ordenados por columnas
d1 <- c(2,5,13,3,8,21)

# Datos ordenados por filas
d2 <- c(2,3,5,8,13,21)
```

El vector `d1` se utiliza para declarar la matriz por columnas, y el vector `d2` para declarar la matriz por filas. Se utiliza la función `matrix()` para declarar una matriz, esta función tiene varias parámetros: el primero es `data`, hace referencia al vector que contiene los datos de la matriz; el segundo es `nrow`, indica el número de filas que contenie la matriz; el tercero, `ncol` representa el número de columnas que contiene la matriz; y el último, `byrow` indica si el vector data está ordenado por filas o columnas.

Por Columnas

Para declarar una matriz por columnas, se ordena el vector `data` por columnas y el valor del parámetro `byrow` debe ser `FALSE`, para hacer referencia de que el vector `data` está ordenado por columnas.

```
A1 <- matrix(data = d1, nrow = 3, ncol = 2, byrow = FALSE) # Por columnas
A1
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    5    8
## [3,]   13   21
```

Por Filas

Para declarar una matriz por filas, se ordena el vector `data` por filas y el valor del parámetro `byrow` debe ser `TRUE`, para hacer referencia de que el vector `data` está ordenado por filas. Por ejemplo:

```
A2 <- matrix(data = d2, nrow = 3, ncol = 2, byrow = TRUE) # Por filas
A2
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    5    8
## [3,]   13   21
```

Algunas cambios y obtención de valores en matrices

```
# Muestra el elemento de la matriz A1 que está en el renglón 1 y la columna 2
A1[1,2]
```

```
## [1] 3
```

```
# Muestra el renglón 2 de la matriz A1
A1[2, ]
```

```
## [1] 5 8
```

```
# Muestra la columna 2 de la matriz A1
A1[,2]
```

```
## [1]  3  8 21
```

```
# Reemplaza el elemento de la matriz A1 en el rengón 2 y la columna 2 por el número 100
A1[2,2]=100

# Muestra la nueva matriz A
A1
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    5  100
## [3,]   13   21
```

```
# Reemplaza el elemento de la matriz A1 en el rengón 2 y la columna 2 por el número 100
A1[2,2]=8

# Muestra la nueva matriz A
A1
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    5    8
## [3,]   13   21
```

1.2.2. Transpuesta

Para obtener la transpuesta de una matriz, se utiliza la función `t()`, por ejemplo:

```
MATRIZ <- matrix(1:12, nrow = 3)
MATRIZ  # Visualizar matriz
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Transpuesta de la matriz

```
t(MATRIZ)   # Se calcula la transpuesta de la matriz
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

Ejemplo 2

```
B <- matrix(c(0,1,1,2,3,5,8,13,21,34,55,89), ncol = 3, byrow = TRUE)
B
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    1
## [2,]    2    3    5
## [3,]    8   13   21
## [4,]   34   55   89
```

```
t(B)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    2    8   34
## [2,]    1    3   13   55
## [3,]    1    5   21   89
```

1.2.3. Diagonal

Para obtener los elementos de la diagonal principal de una matriz, se utiliza la función `diag()`, por ejemplo:

```
MATRIZ <- matrix(seq(1,30,length.out = 9), ncol = 3, byrow = TRUE)
MATRIZ
```

```
##         [,1]   [,2]   [,3]
## [1,]  1.000  4.625  8.250
## [2,] 11.875 15.500 19.125
## [3,] 22.750 26.375 30.000
```

Se imprimen los elementos de la diagonal principal

```
print("Estos son los elementos de la diagonal principal:")
```

```
## [1] "Estos son los elementos de la diagonal principal:"
```

```
diag(MATRIZ)
```

```
## [1]  1.0 15.5 30.0
```

1.2.4. Traza

Para calcular la traza de una matriz, se utiliza la función `sum(diag())`, ya que la suma de los elementos de la diagonal principal es la traza, por ejemplo:

```
Mt <- matrix(seq(1,17,by=2), ncol = 3, byrow = TRUE)
Mt  # Visualizar la matriz
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    7    9   11
## [3,]   13   15   17
```

```
sum(diag(Mt)) # Se calcula la traza
```

```
## [1] 27
```

1.2.5. Determinante

Para calcular el determinante de una matriz cuadrada, se utiliza la función `det()`, por ejemplo:

```
Md <- matrix(c(1,2,3,5,8,13,21,34,55), ncol = 3, byrow = TRUE)
Md
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    5    8   13
## [3,]   21   34   55
```

```
det(Md)  # Se calcula el determinante de la matriz Md
```

```
## [1] 0
```

1.2.6. Inversa

Para calcular la inversa de una matriz cuadrada (cuyo determinante es diferente de cero), se utiliza la función `solve()`, por ejemplo:

```
Minv <- matrix(c(2,3,5,7,11,13,17,19,23), ncol = 3, byrow = TRUE)
Minv
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    5
## [2,]    7   11   13
## [3,]   17   19   23
```

```
# Se calcula la inversa de la matriz Minv
solve(Minv)
```

```
##             [,1]       [,2]        [,3]
## [1,] -0.07692308 -0.3333333  0.20512821
## [2,] -0.76923077  0.5000000 -0.11538462
## [3,]  0.69230769 -0.1666667 -0.01282051
```

```
# Se calcula el producto de la matriz por su inversa
round(Minv %*% solve(Minv), 0)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

1.2.7. Vectores y valores propios

Para calcular los valores y vectores propios de una matriz, se utiliza la función `eigen()`, por ejemplo:

```
MATRIZ <- matrix(c(20, 19, 21, 18, 25, 34, 17, 23, 27), nrow = 3, byrow = TRUE)
MATRIZ
```

```
##      [,1] [,2] [,3]
## [1,]   20   19   21
## [2,]   18   25   34
## [3,]   17   23   27
```

Cálculo de los eigenvalores y eigenvectores

```
# Se calculan los valores y vectores propios
lambda <- eigen(MATRIZ)
lambda
```

```
## eigen() decomposition
## $values
## [1] 68.448394  5.276549 -1.724943
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] -0.5019720 -0.8673498  0.1923837
## [2,] -0.6523395  0.4587200 -0.8184234
## [3,] -0.5678708  0.1930810  0.5414533
```

```
# Primer valor propio
lambda$values[1]
```

```
## [1] 68.44839
```

```
# Primer vector propio
lambda$vectors[,1]
```

```
## [1] -0.5019720 -0.6523395 -0.5678708
```

Ejemplo 2:

```
Mc <- matrix(c(10,-18,6,-11), nrow = 2, ncol = 2, byrow = T)
Mc
```

```
##      [,1] [,2]
## [1,]   10  -18
## [2,]    6  -11
```

Cálculo de los eigenvalores y eigenvectores

```
# Se calculan los valores y vectores propios
lambda <- eigen(Mc)
lambda
```

```
## eigen() decomposition
## $values
## [1] -2  1
##
## $vectors
##           [,1]      [,2]
## [1,] 0.8320503 0.8944272
## [2,] 0.5547002 0.4472136
```

```r
# Primer valor propio
lambda$values[1]
```

```
## [1] -2
```

```r
# Primer vector propio
lambda$vectors[,1]
```

```
## [1] 0.8320503 0.5547002
```

1.2.8. Resolución de sistemas de ecuaciones

Para encontrar la solución de un sistema de ecuaciones, se puede hacer uso de un cálculo matricial. Por ejemplo, considere el siguiente sistema de ecuaciones:

$$2x - 3y - 5z = -19$$
$$3x - 4y + z = -2$$
$$x + y + z = 6$$

El sistema anterior se puede expresar de la siguiente forma:

$$\begin{pmatrix} 2 & -3 & -5 \\ 3 & -4 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -19 \\ -2 \\ 6 \end{pmatrix}$$

A continuación, se presenta el código para solucionar el sistema:

```r
# Matriz de coeficientes
A <- matrix(c(2,-3,-5,3,-4,1,1,1,1), nrow = 3, ncol = 3, byrow = T)
A
```

```
##      [,1] [,2] [,3]
## [1,]    2   -3   -5
## [2,]    3   -4    1
## [3,]    1    1    1
```

```r
# Matriz de datos independientes
b <- matrix(c(-19,-2,6), nrow = 3)
b
```

```
##      [,1]
## [1,]  -19
## [2,]   -2
## [3,]    6
```

```r
# Solución del sistema
solve(A, b)
```

```
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
```

La solución es:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

1.2.9. Función `rbind()`

Para agregar una fila a una matriz ya definida, se utiliza la función `rbind()`, por ejemplo:

```
Rb <- matrix(c(2,3,5,-8), nrow = 2)
Rb
```

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    3   -8
```

```
# Se agrega un nuevo renglón o fila a la matriz anterior generándose una nueva matriz
rbind(Rb, c(13,-21))
```

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    3   -8
## [3,]   13  -21
```

1.2.10. Función `cbind()`

Para agregar una columna a una matriz ya definida, se utiliza la función `cbind()`, por ejemplo:

```
Cb <- matrix(c(2,3,5,-8), nrow = 2)
Cb
```

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    3   -8
```

```
# Se agrega una nueva columna a a la matriz anterior generándose una nueva matriz
cbind(Cb, c(13,-21))
```

```
##      [,1] [,2] [,3]
## [1,]    2    5   13
## [2,]    3   -8  -21
```

---

2. Importar datos a R

2.1. Importar archivos .txt

Para importar tablas que se encuentran en archivos .txt a R, cuyas columnas están separadas por un tabulador, se utiliza el siguiente código:

```
datos.txt <- read.delim("1Mints.txt")
head(datos.txt)
```

```
##   Columna1
## 1   324110
## 2  -442472
## 3   626686
## 4  -157678
## 5   508681
## 6   123414
```

Si las columnas están separadas por espacio, se recomienda utilizar la siguiente sintaxis:

```r
datos_txt <- read.csv("1Mints.txt", sep="")
head(datos_txt,10)
```

```
##    Columna1
## 1    324110
## 2   -442472
## 3    626686
## 4   -157678
## 5    508681
## 6    123414
## 7    -77867
## 8    155091
## 9    129801
## 10   287381
```

2.2. Importar archivos .csv

En R se utiliza el paquete **readr** para importar datos desde archivos csv, si el paquete aún no lo has instalado ejecuta el siguiente código en la consola:

El siguiente código se utiliza para importar datos desde un archivo csv:

```r
# install.packages("readr")
library(readr)  # Llamar la paquetería
datos.csv <- read.csv("Indices.csv")  # Código básico
tail(datos.csv, 11)
```

```
##       FECHA INDICES
## 625 2022/01    7.07
## 626 2022/02    7.28
## 627 2022/03    7.45
## 628 2022/04    7.68
## 629 2022/05    7.65
## 630 2022/06    7.99
## 631 2022/07    8.15
## 632 2022/08    8.70
## 633 2022/09    8.70
## 634 2022/10    8.41
## 635 2022/11    7.80
```

Si el archivo que csv que se desea importar no tiene nombre en las columnas, se utiliza la siguiente sintaxis:

```r
# library(readr)
# datos <- read_csv("c:/Documents/Carpeta/name_csv.csv", col_names = FALSE
# datos
```

2.3. Importar archivos excel

En R se utiliza el paquete **readxl** para importar datos desde excel. Si el paquete aún no lo has instalado ejecuta el siguiente código en la consola:

El siguiente código se utiliza para importar datos desde un archivo excel:

```r
# install.packages("readxl")
library(readxl) # Llamar la paquetería
datos.xlsx <- read_excel("PIB_POR_ENTIDAD_FED.xlsx")  # Código básico
head(datos.xlsx, 10)
```

```
## # A tibble: 10 x 20
##    Entid~1 `2003` `2004` `2005` `2006` `2007` `2008` `2009` `2010` `2011` `2012`
##    <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Aguasc~ 1.21e5 1.27e5 1.30e5 1.38e5 1.50e5 1.51e5 1.43e5 1.52e5 1.59e5 1.68e5
##  2 Baja C~ 4.00e5 4.23e5 4.33e5 4.56e5 4.62e5 4.58e5 4.08e5 4.28e5 4.41e5 4.56e5
##  3 Baja C~ 7.60e4 8.15e4 8.74e4 9.37e4 1.06e5 1.09e5 1.08e5 1.11e5 1.15e5 1.17e5
##  4 Campec~ 1.05e6 1.06e6 1.04e6 1.01e6 9.48e5 8.67e5 7.81e5 7.54e5 7.27e5 7.15e5
##  5 Coahui~ 4.37e5 4.49e5 4.59e5 4.80e5 5.00e5 4.98e5 4.21e5 4.90e5 5.23e5 5.50e5
##  6 Colima  6.77e4 6.78e4 6.83e4 7.25e4 7.75e4 7.90e4 7.64e4 8.20e4 8.79e4 9.05e4
##  7 Chiapas 2.48e5 2.38e5 2.40e5 2.48e5 2.53e5 2.58e5 2.57e5 2.71e5 2.79e5 2.85e5
##  8 Chihua~ 3.60e5 3.77e5 3.89e5 4.20e5 4.35e5 4.41e5 4.01e5 4.18e5 4.27e5 4.59e5
##  9 Ciudad~ 2.13e6 2.23e6 2.26e6 2.37e6 2.41e6 2.45e6 2.36e6 2.45e6 2.53e6 2.63e6
## 10 Durango 1.53e5 1.58e5 1.55e5 1.60e5 1.63e5 1.66e5 1.63e5 1.69e5 1.76e5 1.83e5
## # ... with 9 more variables: `2013` <dbl>, `2014` <dbl>, `2015` <dbl>,
## #   `2016` <dbl>, `2017` <dbl>, `2018` <dbl>, `2019` <dbl>, `2020` <dbl>,
## #   `2021` <dbl>, and abbreviated variable name 1: `Entidad Federativa`
```

```
dim(datos.xlsx)
```

```
## [1] 32 20
```

Si deseas definir el tipo de variable que tendrá cada columna, puedes utilizar el siguiente código:

```
#name_excel.xlsx  tiene una tabla con 8 columnas.
library(readxl)
datos_excel <- read_excel("PIB_POR_ENTIDAD_FED.xlsx", #path
                    sheet = "Tabulado", # Nombre de la hoja
                    col_types = c("text", "skip", "skip", "skip",
                                  "skip", "skip", "skip", "skip",
                                  "skip", "skip",
                                  "numeric", "numeric", "numeric", "numeric",
                                  "numeric", "numeric", "numeric", "numeric",
                                  "numeric", "numeric"))

# Tipo de variable. Nota: skip se utiliza para excluir la variable.
head(datos_excel,10)
```

```
## # A tibble: 10 x 11
##    Entid~1 `2012` `2013` `2014` `2015` `2016` `2017` `2018` `2019` `2020` `2021`
##    <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Aguasc~ 1.68e5 1.73e5 1.90e5 1.98e5 2.12e5 2.17e5 2.25e5 2.22e5 2.05e5 2.07e5
##  2 Baja C~ 4.56e5 4.66e5 4.78e5 5.11e5 5.36e5 5.53e5 5.66e5 5.75e5 5.54e5 5.99e5
##  3 Baja C~ 1.17e5 1.15e5 1.16e5 1.31e5 1.34e5 1.48e5 1.73e5 1.59e5 1.22e5 1.41e5
##  4 Campec~ 7.15e5 7.21e5 6.86e5 6.38e5 6.01e5 5.38e5 5.29e5 5.17e5 4.82e5 4.62e5
##  5 Coahui~ 5.50e5 5.38e5 5.59e5 5.64e5 5.72e5 6.02e5 6.08e5 6.04e5 5.36e5 5.67e5
##  6 Colima  9.05e4 9.14e4 9.37e4 9.59e4 9.86e4 1.02e5 1.05e5 1.09e5 1.01e5 1.01e5
##  7 Chiapas 2.85e5 2.81e5 2.94e5 2.88e5 2.87e5 2.78e5 2.68e5 2.62e5 2.54e5 2.68e5
##  8 Chihua~ 4.59e5 4.76e5 4.86e5 5.11e5 5.34e5 5.51e5 5.64e5 5.73e5 5.40e5 5.66e5
##  9 Ciudad~ 2.63e6 2.67e6 2.75e6 2.87e6 2.96e6 3.05e6 3.13e6 3.13e6 2.86e6 2.94e6
## 10 Durango 1.83e5 1.89e5 1.93e5 1.93e5 2.01e5 2.00e5 2.02e5 2.04e5 1.90e5 2.00e5
## # ... with abbreviated variable name 1: `Entidad Federativa`
```

3. Data frame

3.1. Declaración de data frame

Las data frames son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos. Generalmente se utiliza para análisis de datos y su estructura es familiar a otros paquetes estadísticos. Para declarar un data frame, se utiliza la función `data.frame` y entre paréntesis se escriben las columnas que contendrá separadas por comas. Por ejemplo:

```
nombre_data_frame <- data.frame(nombre_columna1 = c(1,2,3,4,5),
                                nombre_columna2 =
                                  c("MARIA","JOSE","LUPITA","HUMBERTO","LUIS"),
                                nombre_columna3 =
                                  c(FALSE,TRUE,FALSE,TRUE,FALSE))
nombre_data_frame
```

```
##   nombre_columna1 nombre_columna2 nombre_columna3
## 1               1           MARIA           FALSE
## 2               2            JOSE            TRUE
## 3               3          LUPITA           FALSE
## 4               4        HUMBERTO            TRUE
## 5               5            LUIS           FALSE
```

Para obtener los valores de las columnas en formato de dataframe, se utiliza la siguiente sintaxis:

```
# Para obtener el valor de nombre_columna3
nombre_data_frame[, 3]
```

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

```
# Para obtener los valores de nombre_columna1 y nombre_columna1
nombre_data_frame[, c(1,3)]
```

```
##   nombre_columna1 nombre_columna3
## 1               1           FALSE
## 2               2            TRUE
## 3               3           FALSE
## 4               4            TRUE
## 5               5           FALSE
```

3.2. Selección de datos usando operadores comparativos

Sea el siguiente data frame:

```
datos.df <- data.frame(edad=c(20,21,27,6,12,26,8),
                       sexo=c("Femenino","Masculino","Femenino",
                              "Femenino","Femenino","Masculino","Femenino"),
                       peso=c(84,66,83,53,54,52,88),
                       estatura=c(1.75,1.59,1.56,1.33,1.77,1.78,1.35))
datos.df
```

```
##   edad      sexo peso estatura
## 1   20  Femenino   84     1.75
## 2   21 Masculino   66     1.59
## 3   27  Femenino   83     1.56
## 4    6  Femenino   53     1.33
## 5   12  Femenino   54     1.77
## 6   26 Masculino   52     1.78
```

```
## 7    8  Femenino   88     1.35
```

Para buscar observaciones específicas en R, se utilizan los operadores comparativos. Por ejemplo, para buscar aquellas observaciones que cuya variable `edad` sea mayor o igual que 25, se utiliza el siguiente código:

Las observaciones cuya edad es mayor o igual a 25 son:

```
# Buscar los datos cuya edad sea mayor o igual que 25
datos.df[datos.df$edad >= 25,]
```

```
##   edad      sexo peso estatura
## 3   27  Femenino   83     1.56
## 6   26 Masculino   52     1.78
```

3.3. Selección de datos usando operadores lógicos & y |

Para buscar aquellas observaciones que cuya edad sea mayor o igual a 25 y cuyo sexo sea Femenino, se utiliza el siguiente código:

```
# Buscar los datos cuya edad sea mayor o igual a 25 y cuyo sexo sea femenino
datos.df[datos.df$edad >= 25 & datos.df$sexo == "Femenino", ]
```

```
##   edad     sexo peso estatura
## 3   27 Femenino   83     1.56
```

3.4. Agregar o quitar columnas

Considere el siguiente Data frame:

```
Mkdo_val <- data.frame(Emisora = c("Cemex CPO","FIHO 12","Apple",
                                   "Wells Fargo","Disney","Cisco"),
                       No._de_Títulos = c(45,100,1,2,1,2),
                       Precio_de_mercado = c(16.71,5.94,2884.79,872.93,3519.56,1068.35))


# Visualizar el data frame
Mkdo_val
```

```
##       Emisora No._de_Títulos Precio_de_mercado
## 1   Cemex CPO             45             16.71
## 2     FIHO 12            100              5.94
## 3       Apple              1           2884.79
## 4 Wells Fargo              2            872.93
## 5      Disney              1           3519.56
## 6       Cisco              2           1068.35
```

```
# Para obtener el valor de Emisora
Mkdo_val$Emisora
```

```
## [1] "Cemex CPO"   "FIHO 12"     "Apple"       "Wells Fargo" "Disney"
## [6] "Cisco"
```

```
# Para obtener los valores de No._de_Títulos y Precio_de_mercado
Mkdo_val[, c(2,3)]
```

```
##   No._de_Títulos Precio_de_mercado
## 1             45             16.71
## 2            100              5.94
## 3              1           2884.79
```

```
## 4              2           872.93
## 5              1          3519.56
## 6              2          1068.35
```

```
# Buscar los datos cuyo No._de_Títulos sean mayores que 10
Mkdo_val[Mkdo_val$No._de_Títulos > 10,]
```

```
##     Emisora No._de_Títulos Precio_de_mercado
## 1 Cemex CPO             45             16.71
## 2   FIHO 12            100              5.94
```

```
# Busca las observaciones cuyo No._de_Títulos sea mayor o igual a 2 y cuyo
# Precio_de_mercado sea mayor que 1000
Mkdo_val[Mkdo_val$No._de_Títulos >= 2 & Mkdo_val$Precio_de_mercado > 1000,]
```

```
##   Emisora No._de_Títulos Precio_de_mercado
## 6   Cisco              2           1068.35
```

```
# Busca las observaciones cuyo No._de_Títulos sea igual a 2 o cuyo
# Precio_de_mercado sea menor que 1000
Mkdo_val[Mkdo_val$No._de_Títulos == 2 | Mkdo_val$Precio_de_mercado < 1000,]
```

```
##       Emisora No._de_Títulos Precio_de_mercado
## 1   Cemex CPO             45             16.71
## 2     FIHO 12            100              5.94
## 4 Wells Fargo              2            872.93
## 6       Cisco              2           1068.35
```

```
# Agrega una nueva columna al Data frame
Mkdo_val$Porcentaje_de_Var_Hist <- c(252,35.30,18.99,85.73,33.17,40.76)
Mkdo_val
```

```
##       Emisora No._de_Títulos Precio_de_mercado Porcentaje_de_Var_Hist
## 1   Cemex CPO             45             16.71                 252.00
## 2     FIHO 12            100              5.94                  35.30
## 3       Apple              1           2884.79                  18.99
## 4 Wells Fargo              2            872.93                  85.73
## 5      Disney              1           3519.56                  33.17
## 6       Cisco              2           1068.35                  40.76
```

```
# Borra la columna Porcentaje_de_Var_Hist
Mkdo_val$Porcentaje_de_Var_Hist <- NULL
Mkdo_val
```

```
##       Emisora No._de_Títulos Precio_de_mercado
## 1   Cemex CPO             45             16.71
## 2     FIHO 12            100              5.94
## 3       Apple              1           2884.79
## 4 Wells Fargo              2            872.93
## 5      Disney              1           3519.56
## 6       Cisco              2           1068.35
```

3.5. Agregar o quitar filas

```
# Agrega una nueva fila al Data frame
Fila_nueva <- c("FEMSA UBD", 5, 165.79)
Mkdo_val <- rbind(Mkdo_val, Fila_nueva)
Mkdo_val
```

```
##       Emisora No._de_Títulos Precio_de_mercado
## 1   Cemex CPO             45             16.71
## 2     FIHO 12            100              5.94
## 3       Apple              1           2884.79
## 4 Wells Fargo              2            872.93
## 5      Disney              1           3519.56
## 6       Cisco              2           1068.35
## 7   FEMSA UBD              5            165.79
```

```
# Borra la fila 1 del Data frame Mkdo_val
Mkdo_val <- Mkdo_val[-1,]
Mkdo_val
```

```
##       Emisora No._de_Títulos Precio_de_mercado
## 2     FIHO 12            100              5.94
## 3       Apple              1           2884.79
## 4 Wells Fargo              2            872.93
## 5      Disney              1           3519.56
## 6       Cisco              2           1068.35
## 7   FEMSA UBD              5            165.79
```

3.6. Función summary()

```
# Retorna un resumen estadístico de la variable Precio_de_mercado del Data
# frame Mkdo_val
summary(as.numeric(Mkdo_val$Precio_de_mercado))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.94  342.57  970.64 1419.56 2430.68 3519.56
```

```
### Datos de Agosto del 2021
```

Considere ahora el siguiente Data frame:

```
Cruceros <- data.frame(Compañías_de_Cruceros = c("RCL","NCLH","CCL"),
                       Precio_por_Acción = c(82.33,27.49,24.26),
                       Capitalización_de_Mercado = c(20.96,10.17,27.9),
                       Ingresos = c(218.06,36.13,141))
Cruceros
```

```
##   Compañías_de_Cruceros Precio_por_Acción Capitalización_de_Mercado Ingresos
## 1                   RCL            82.33                      20.96   218.06
## 2                  NCLH            27.49                      10.17    36.13
## 3                   CCL            24.26                      27.90   141.00
```

```
# Busca las observaciones cuyo Precio por acción sea menor o igual a 30 dlls, cuya
# Capitalización de mercado sea mayor o igual a 20 billones de dlls y cuyos Ingresos
# sean mayores que 100 millones de dlls.

Cruceros[Cruceros$Precio_por_Acción <= 30 &
           Cruceros$Capitalización_de_Mercado >= 20 &
           Cruceros$Ingresos > 100,]
```

```
##   Compañías_de_Cruceros Precio_por_Acción Capitalización_de_Mercado Ingresos
## 3                   CCL            24.26                       27.9      141
```

4. Listas en R

4.1. Declaración de lista

Las listas en R son estructuras de datos que permiten almacenar diferentes tipos de variables. Para declarar una lista en R, se utiliza la función `list()`, y entre paréntesis se escriben las variable separadas por coma, tal como se muestra a continuación:

```
Mi_listilla <- list(a = "I love Chuze", b = exp(1), c = TRUE,
                    d = c(0,1,1,2,3,5,8,13))  # Se declara la lista
Mi_listilla
```

```
## $a
## [1] "I love Chuze"
##
## $b
## [1] 2.718282
##
## $c
## [1] TRUE
##
## $d
## [1]  0  1  1  2  3  5  8 13
```

4.2. Selección de elementos de una lista

Para obtener los valores de la lista, se escribe el nombre de la lista seguido el símbolo de peso `$` y luego el nombre de la variable. Por ejemplo:

```
# Para obtener el valor de a
Mi_listilla$a
```

```
## [1] "I love Chuze"
```

```
# Para obtener el valor de a
Mi_listilla[[1]]
```

```
## [1] "I love Chuze"
```

4.3. Agregar o quitar elementos de una lista

4.3.1. Agregar

Para agregar una nueva variable basta con escribir el nombre de la lista seguido del símbolo de `$` y el nombre la variable, tal como se muestra a continuación:

```
# Agrega la nueva variable ETFS a la lista
Mi_listilla$ETFS <- c("VOO","LIT","QQQ","TAN","ICLN","SPY","DRIV")
Mi_listilla
```

```
## $a
## [1] "I love Chuze"
##
## $b
## [1] 2.718282
##
```

```
## $c
## [1] TRUE
##
## $d
## [1]  0  1  1  2  3  5  8 13
##
## $ETFS
## [1] "VOO"  "LIT"  "QQQ"  "TAN"  "ICLN" "SPY"  "DRIV"
```

4.3.2. Eliminar

Para eleminar una variable de una lista, se escribe `nombre_lista$nombre_variable` y se le asigna el valor de `NULL`.

```
# Para eliminar la variable c de la lista
Mi_listilla$c <- NULL
Mi_listilla
```

```
## $a
## [1] "I love Chuze"
##
## $b
## [1] 2.718282
##
## $d
## [1]  0  1  1  2  3  5  8 13
##
## $ETFS
## [1] "VOO"  "LIT"  "QQQ"  "TAN"  "ICLN" "SPY"  "DRIV"
```

---

5. Programación en R

5.1. Condicional if

La sentencia if se utiliza para comprobar una condición y si la condición es verdadera entonces procesaremos un conjunto de sentencias. La estructura es la siguiente:

```
# if(Condición){
#  Sentencias a realizar si la condición es verdadera
# }
```

Por ejemplo, considere un condicional una que verifique si un número es menor que 10, así que el código correspondiente es:

```
x <- 4
if(x < 10){
  print("Menor que 10")
}
```

```
## [1] "Menor que 10"
```

5.2. Condicional if-else

La sentencia if-else se utiliza para comprobar una condición y si la condición es verdadera entonces procesaremos un conjunto de sentencias y en caso de que la condición sea falsa se procede a realizar otro conjunto de sentencias. La estructura es la siguiente:

```
# if(Condición) {
#   Sentencias a realizar  si la condición es verdadera
# } else {
#   Sentencias a realizar si la condición es falsa
# }
```

Por ejemplo, considere un condicional if-else que verifique si un número es menor que 10, de manera que el código es:

```
w <- 12
if(w < 10){
  print("w es menor que 10")
}else{
  print("w es mayor o igual que 10")
}
```

```
## [1] "w es mayor o igual que 10"
```

Ejemplo 2:

```
z <- 22
if(z != 2){
  print("z es diferente de 2")
}else{
  print("z es igual a 2")
}
```

```
## [1] "z es diferente de 2"
```

5.3. Condicional `ifelse()`

Se recomienda utilizar el condiconal ifelse, cuando la condición es juego tiene más de una expresión. Cuando se realizan comparaciones con vectores se usa expresión condicional que involucra más de una expresión, por ejemplo: considere el vector `nota <- c(50,60,49,87,92,30,90,85)` que representa las calificaciones finales de la asignatura de álgebra de 8 estudiantes. Al realizar la siguiente comparación:

```
nota <- c(50,60,49,87,92,30,90,85)
nota < 70 # Determinar que valores del vector nota son menores que 70
```

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

Retorna un vector de TRUE o FALSE, así que se tiene muchas expreciones a partir de una condición. Por tanto, para condiciones de este tipo se sugiere utilizar el condicional ifelse, ya que dicho condicional trabaja con expresiones de este tipo. Su estructura es la siguiente:

```
# ifelse(Condición, # Sentencia si la condición es verdadera,
# Sentencia si la condición es falsa)
```

Por ejemplo, considere el vector `nota` declarado anteriormente, e idenfique que estudiantes están aprobados o reprobados, teniendo en cuenta que la calificación de aprovación es mayor o igual que 70 puntos. Así que el código es el siguiente:

```
nota <- c(50,60,49,87,92,30,90,85)
ifelse(nota >= 70,"Aprobado","Reprobado")
```

```
## [1] "Reprobado" "Reprobado" "Reprobado" "Aprobado"  "Aprobado"  "Reprobado"
## [7] "Aprobado"  "Aprobado"
```

Ejemplo 2:

```
x <- seq(15, 1, length.out = 20)
ifelse(x > 5,"Mayor que 5","Menor o igual que 5")
```

```
##  [1] "Mayor que 5"         "Mayor que 5"         "Mayor que 5"
##  [4] "Mayor que 5"         "Mayor que 5"         "Mayor que 5"
##  [7] "Mayor que 5"         "Mayor que 5"         "Mayor que 5"
## [10] "Mayor que 5"         "Mayor que 5"         "Mayor que 5"
## [13] "Mayor que 5"         "Mayor que 5"         "Menor o igual que 5"
## [16] "Menor o igual que 5" "Menor o igual que 5" "Menor o igual que 5"
## [19] "Menor o igual que 5" "Menor o igual que 5"
```

Ejemplo 3:

```
Mensualidad <- c(9.99,22.99,32.99,39.99)
ifelse(Mensualidad <= 22.99,"I love Chuze","I don't love Chuze")
```

```
## [1] "I love Chuze"       "I love Chuze"       "I don't love Chuze"
## [4] "I don't love Chuze"
```

5.4. Bucle while

El bucle(ciclo) mientras(while) es la estructura básica que permite repetir varias veces una secuencia de operaciones, mientras se cumpla una determina condición.

```
# while(Condición){
  # Sentencias que se repiten si la condición es verdadera
# }
```

Por ejemplo, imprimir en consola los números de 1 al 10 utilizando el bucle while:

```
i <- 1  # A la variable i se le conoce como contador
while (i <= 10) {
  print(i)
  i <- i+1
}
```

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

Ejemplo 2:

```
contador <- 1
suma <- 0
while (contador <= 10) {
  suma <- suma+contador
  contador <- contador+1
```

```
}
suma # La suma de los 10 primeros números naturales
```

```
## [1] 55
```

Ejemplo 3:

```
contador <- 1
suma <- 0
while (contador <= 10) {
  suma <- suma+contador**2
  contador <- contador+1
}
suma # La suma de los cuadrados de los 10 primeros números naturales
```

```
## [1] 385
```

Ejemplo 4:

```
contador <- 35
while (contador <= 40) {
  print(contador)
  contador <- contador+1
}
```

```
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
```

5.5. Bucle for

El bucle(ciclo) for en R, es una iteración repetitiva (en bucle) de cualquier sentencia, donde en cada iteración se evalúa una misma sentencia a través de los elementos de un vector. La sintaxis del bucle for es:

```
# for(i in vector){
#   Sentencias
# }
```

Por ejemplo, imprimir `i^2 + 3*i` para `i=0,1,2,...,10` utilizando el bucle for, así que el código será el siguiente:

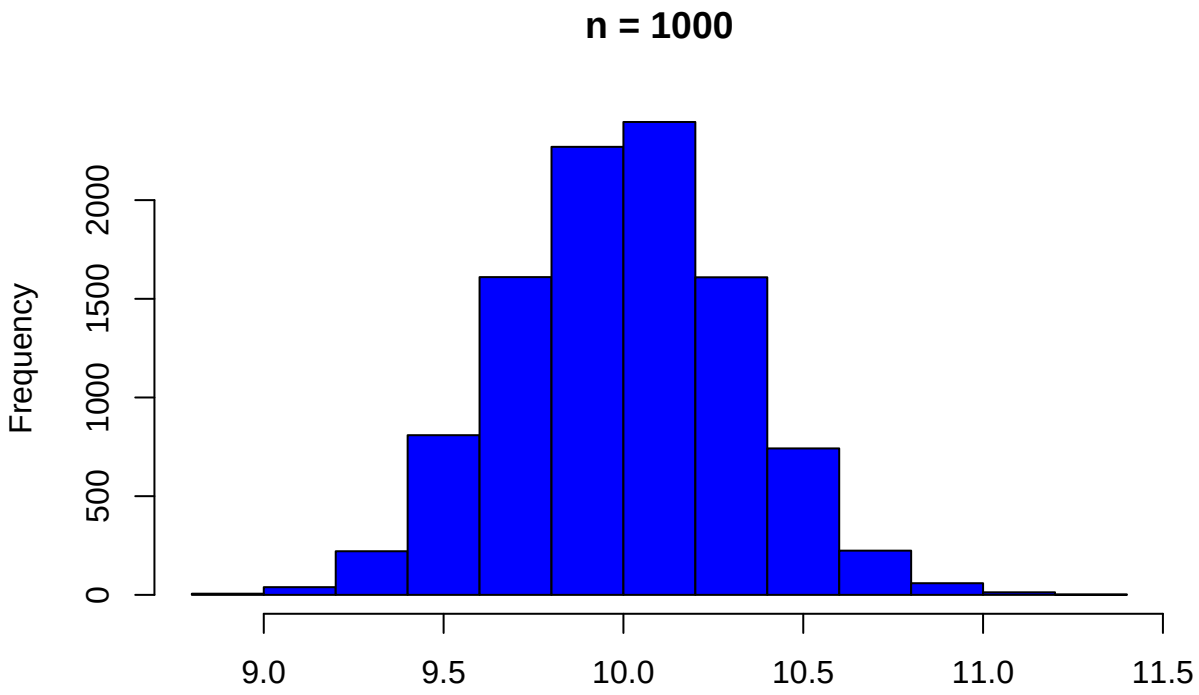
```
for(i in 0:10){
  print(i^2 + 3*i)
}
```

```
## [1] 0
## [1] 4
## [1] 10
## [1] 18
## [1] 28
## [1] 40
## [1] 54
## [1] 70
## [1] 88
## [1] 108
```

```
## [1] 130
```

Ejemplo 2:

```
{sample1000<- c()
n <- 1000
for (i in 1:10000) {
  sample1000[i] <- mean(rexp(n = n, rate = 0.1))
}
hist(sample1000, col = "blue", main = "n = 1000", xlab = "")}
```



6. Funciones

6.1. Mi primera función

En R se declaran las funciones con la palabra `function`, por ejemplo, la siguiente función imprime "Buongiorno principessa" en la consola.:

```
nombre_funcion <- function(){
  print("Buongiorno principessa")
}
nombre_funcion()  # Ejecutar la función declarada
```

```
## [1] "Buongiorno principessa"
```

Ejemplo 2:

```
Mi_función_Ej_2 <- function(){
  print("Pasamos la mayor parte de nuestra vida soñando, sobre todo cuando estamos despiertos")
}
Mi_función_Ej_2()
```

```
## [1] "Pasamos la mayor parte de nuestra vida soñando, sobre todo cuando estamos despiertos"
```

6.2. Funciones con ausencia de parámetros

Las funciones con ausencia de parámetros son aquellas que entre los paréntesis no se indica nada, por ejemplo:

```
# Combinaciones
n <- 6
r <- 2
combinaciones_n_r <- function(){
  return((factorial(n))/(factorial(r)*factorial(n-r))) # Retorna el factorial
}
combinaciones_n_r()  # Ejecuta la función declarada
```

```
## [1] 15
```

Ejemplo 2:

```
# P. Binomial
Binomial <- function(n,y,p){
  # n, y, p son los datos de entrada de la función
  b <- ((factorial(n))/(factorial(y)*factorial(n-y)))*(p^y)*(1-p)^(n-y)
  # Cuerpo de la función
  return(b)  # Retorno
}
prob_de_y <- Binomial(20,14,0.8)
prob_de_y
```

```
## [1] 0.1090997
```

6.3. Funciones con parámetros

Las funciones con parámetros son aquellas que entre los paréntesis se indican los datos de entrada de la función, por ejemplo:

```
suma <- function(x,y){
  # "x" y "y" son los datos de entrada de la función
  q <- x+y   # cuerpo de la función
  return(q) # Retorno
}
w <- suma(1103,4025) # Ejecuta la función declarada
w
```

```
## [1] 5128
```

6.4. Funciones con parámetros por default

Las funciones con parámetros y parametros por defectos son aquellas que entre los paréntesis se indican los datos de entrada de la función, así como aquellos datos que se asume que se conoce desde un principio, pero que se pueden cambiar después, su sintaxis es la siguiente:

‘nombre_funcion <- function(datos de entrada, datos de entrada por defecto){

datos de entrada

datos de entrada por defecto

cuerpo de la función

cuerpo de la función

cuerpo de la función

return(dato de salida) # Retorno }‘

Ejemplo 1:

```
raiz <- function(x,n=2){
  # x dato de entrada
  # n=2 dato de entrada por defecto
  p <- x^(1/n) # cuerpo de la función
  return(p) # retorno
}
raiz(100) # Al no declarar el valor de n, se entiende que toma el valor de 2
```

```
## [1] 10
```

Ejemplo 2:

```
Poiss <- function(y,lambda=2){
  # y dato de entrada
  # lambda=2 datos de entrada por defecto
  p <- ((lambda^y)*exp(-lambda))/factorial(y)  # Cuerpo de la función
  return(p)  # retorno
}
Poiss(4)     # Ejecuta la función declarada con Y=4, P(Y=4)
```

```
## [1] 0.09022352
```

```
Poiss(5)     # Ejecuta la función declarada con Y=5, P(Y=5)
```

```
## [1] 0.03608941
```

```
Poiss(6)     # Ejecuta la función declarada con Y=6, P(Y=6)
```

```
## [1] 0.0120298
```

```
Poiss(6, 3)  # Ejecuta la función declarada con Y=6, P(Y=6) y lambda=3
```

```
## [1] 0.05040941
```

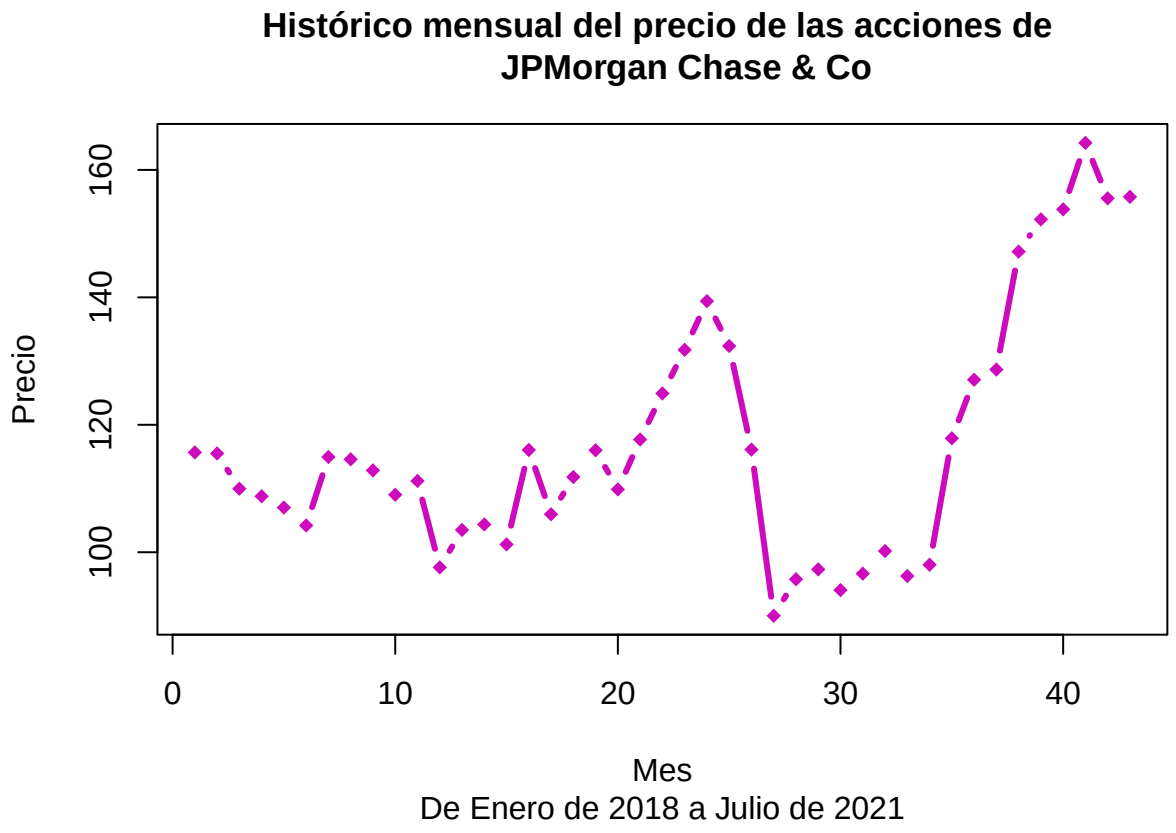
7. Gráficos en R

7.1. Función `plot()`

7.1.1. Declarar ejes x y y.

En R la función `plot()` es usa de manera general para crear gráficos. Para ello, es necesario definir los siguientes argumentos:

```
# x representa al eje x del plano cartesiano
x <- c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,
       25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43)

# y representa al eje y del plano cartesiano
y <- c(115.67,115.5,109.97,108.78,107.01,104.2,114.95,114.58,112.84,
       109.02,111.19,97.62,103.5,104.36,101.23,116.05,105.96,111.8,
       116,109.86,117.69,124.92,131.76,139.4,132.36,116.11,90.03,
       95.76,97.31,94.06,96.64,100.19,96.27,98.04,117.88,127.07,
       128.67,147.17,152.23,153.81,164.24,155.54,155.77
)
plot(x,y,type = "b", pch = 18, lwd = 3, col=30,
     main = "Histórico mensual del precio de las acciones de
     JPMorgan Chase & Co", xlab = "Mes", ylab = "Precio",
     sub = "De Enero de 2018 a Julio de 2021",
     cex.main = 1.1) # pch=18, filled diamond
```



Para conocer los colores que se tiene disponible en R, basta con escribir en consola los siguiente:

```r
colors() # Vector que contiene el nombre de los colores
```

```
##   [1] "white"                "aliceblue"            "antiquewhite"
##   [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
##   [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
##  [10] "aquamarine2"          "aquamarine3"          "aquamarine4"
##  [13] "azure"                "azure1"               "azure2"
##  [16] "azure3"               "azure4"               "beige"
##  [19] "bisque"               "bisque1"              "bisque2"
##  [22] "bisque3"              "bisque4"              "black"
##  [25] "blanchedalmond"       "blue"                 "blue1"
##  [28] "blue2"                "blue3"                "blue4"
##  [31] "blueviolet"           "brown"                "brown1"
##  [34] "brown2"               "brown3"               "brown4"
##  [37] "burlywood"            "burlywood1"           "burlywood2"
##  [40] "burlywood3"           "burlywood4"           "cadetblue"
##  [43] "cadetblue1"           "cadetblue2"           "cadetblue3"
##  [46] "cadetblue4"           "chartreuse"           "chartreuse1"
##  [49] "chartreuse2"          "chartreuse3"          "chartreuse4"
##  [52] "chocolate"            "chocolate1"           "chocolate2"
##  [55] "chocolate3"           "chocolate4"           "coral"
##  [58] "coral1"               "coral2"               "coral3"
##  [61] "coral4"               "cornflowerblue"       "cornsilk"
##  [64] "cornsilk1"            "cornsilk2"            "cornsilk3"
##  [67] "cornsilk4"            "cyan"                 "cyan1"
##  [70] "cyan2"                "cyan3"                "cyan4"
##  [73] "darkblue"             "darkcyan"             "darkgoldenrod"
##  [76] "darkgoldenrod1"       "darkgoldenrod2"       "darkgoldenrod3"
##  [79] "darkgoldenrod4"       "darkgray"             "darkgreen"
##  [82] "darkgrey"             "darkkhaki"            "darkmagenta"
##  [85] "darkolivegreen"       "darkolivegreen1"      "darkolivegreen2"
##  [88] "darkolivegreen3"      "darkolivegreen4"      "darkorange"
##  [91] "darkorange1"          "darkorange2"          "darkorange3"
##  [94] "darkorange4"          "darkorchid"           "darkorchid1"
##  [97] "darkorchid2"          "darkorchid3"          "darkorchid4"
## [100] "darkred"              "darksalmon"           "darkseagreen"
## [103] "darkseagreen1"        "darkseagreen2"        "darkseagreen3"
## [106] "darkseagreen4"        "darkslateblue"        "darkslategray"
## [109] "darkslategray1"       "darkslategray2"       "darkslategray3"
## [112] "darkslategray4"       "darkslategrey"        "darkturquoise"
## [115] "darkviolet"           "deeppink"             "deeppink1"
## [118] "deeppink2"            "deeppink3"            "deeppink4"
## [121] "deepskyblue"          "deepskyblue1"         "deepskyblue2"
## [124] "deepskyblue3"         "deepskyblue4"         "dimgray"
## [127] "dimgrey"              "dodgerblue"           "dodgerblue1"
## [130] "dodgerblue2"          "dodgerblue3"          "dodgerblue4"
## [133] "firebrick"            "firebrick1"           "firebrick2"
## [136] "firebrick3"           "firebrick4"           "floralwhite"
## [139] "forestgreen"          "gainsboro"            "ghostwhite"
## [142] "gold"                 "gold1"                "gold2"
## [145] "gold3"                "gold4"                "goldenrod"
## [148] "goldenrod1"           "goldenrod2"           "goldenrod3"
## [151] "goldenrod4"           "gray"                 "gray0"
## [154] "gray1"                "gray2"                "gray3"
```

```
## [157] "gray4"              "gray5"              "gray6"
## [160] "gray7"              "gray8"              "gray9"
## [163] "gray10"             "gray11"             "gray12"
## [166] "gray13"             "gray14"             "gray15"
## [169] "gray16"             "gray17"             "gray18"
## [172] "gray19"             "gray20"             "gray21"
## [175] "gray22"             "gray23"             "gray24"
## [178] "gray25"             "gray26"             "gray27"
## [181] "gray28"             "gray29"             "gray30"
## [184] "gray31"             "gray32"             "gray33"
## [187] "gray34"             "gray35"             "gray36"
## [190] "gray37"             "gray38"             "gray39"
## [193] "gray40"             "gray41"             "gray42"
## [196] "gray43"             "gray44"             "gray45"
## [199] "gray46"             "gray47"             "gray48"
## [202] "gray49"             "gray50"             "gray51"
## [205] "gray52"             "gray53"             "gray54"
## [208] "gray55"             "gray56"             "gray57"
## [211] "gray58"             "gray59"             "gray60"
## [214] "gray61"             "gray62"             "gray63"
## [217] "gray64"             "gray65"             "gray66"
## [220] "gray67"             "gray68"             "gray69"
## [223] "gray70"             "gray71"             "gray72"
## [226] "gray73"             "gray74"             "gray75"
## [229] "gray76"             "gray77"             "gray78"
## [232] "gray79"             "gray80"             "gray81"
## [235] "gray82"             "gray83"             "gray84"
## [238] "gray85"             "gray86"             "gray87"
## [241] "gray88"             "gray89"             "gray90"
## [244] "gray91"             "gray92"             "gray93"
## [247] "gray94"             "gray95"             "gray96"
## [250] "gray97"             "gray98"             "gray99"
## [253] "gray100"            "green"              "green1"
## [256] "green2"             "green3"             "green4"
## [259] "greenyellow"        "grey"               "grey0"
## [262] "grey1"              "grey2"              "grey3"
## [265] "grey4"              "grey5"              "grey6"
## [268] "grey7"              "grey8"              "grey9"
## [271] "grey10"             "grey11"             "grey12"
## [274] "grey13"             "grey14"             "grey15"
## [277] "grey16"             "grey17"             "grey18"
## [280] "grey19"             "grey20"             "grey21"
## [283] "grey22"             "grey23"             "grey24"
## [286] "grey25"             "grey26"             "grey27"
## [289] "grey28"             "grey29"             "grey30"
## [292] "grey31"             "grey32"             "grey33"
## [295] "grey34"             "grey35"             "grey36"
## [298] "grey37"             "grey38"             "grey39"
## [301] "grey40"             "grey41"             "grey42"
## [304] "grey43"             "grey44"             "grey45"
## [307] "grey46"             "grey47"             "grey48"
## [310] "grey49"             "grey50"             "grey51"
## [313] "grey52"             "grey53"             "grey54"
## [316] "grey55"             "grey56"             "grey57"
```

```
## [319] "grey58"               "grey59"               "grey60"
## [322] "grey61"               "grey62"               "grey63"
## [325] "grey64"               "grey65"               "grey66"
## [328] "grey67"               "grey68"               "grey69"
## [331] "grey70"               "grey71"               "grey72"
## [334] "grey73"               "grey74"               "grey75"
## [337] "grey76"               "grey77"               "grey78"
## [340] "grey79"               "grey80"               "grey81"
## [343] "grey82"               "grey83"               "grey84"
## [346] "grey85"               "grey86"               "grey87"
## [349] "grey88"               "grey89"               "grey90"
## [352] "grey91"               "grey92"               "grey93"
## [355] "grey94"               "grey95"               "grey96"
## [358] "grey97"               "grey98"               "grey99"
## [361] "grey100"              "honeydew"             "honeydew1"
## [364] "honeydew2"            "honeydew3"            "honeydew4"
## [367] "hotpink"              "hotpink1"             "hotpink2"
## [370] "hotpink3"             "hotpink4"             "indianred"
## [373] "indianred1"           "indianred2"           "indianred3"
## [376] "indianred4"           "ivory"                "ivory1"
## [379] "ivory2"               "ivory3"               "ivory4"
## [382] "khaki"                "khaki1"               "khaki2"
## [385] "khaki3"               "khaki4"               "lavender"
## [388] "lavenderblush"        "lavenderblush1"       "lavenderblush2"
## [391] "lavenderblush3"       "lavenderblush4"       "lawngreen"
## [394] "lemonchiffon"         "lemonchiffon1"        "lemonchiffon2"
## [397] "lemonchiffon3"        "lemonchiffon4"        "lightblue"
## [400] "lightblue1"           "lightblue2"           "lightblue3"
## [403] "lightblue4"           "lightcoral"           "lightcyan"
## [406] "lightcyan1"           "lightcyan2"           "lightcyan3"
## [409] "lightcyan4"           "lightgoldenrod"       "lightgoldenrod1"
## [412] "lightgoldenrod2"      "lightgoldenrod3"      "lightgoldenrod4"
## [415] "lightgoldenrodyellow" "lightgray"            "lightgreen"
## [418] "lightgrey"            "lightpink"            "lightpink1"
## [421] "lightpink2"           "lightpink3"           "lightpink4"
## [424] "lightsalmon"          "lightsalmon1"         "lightsalmon2"
## [427] "lightsalmon3"         "lightsalmon4"         "lightseagreen"
## [430] "lightskyblue"         "lightskyblue1"        "lightskyblue2"
## [433] "lightskyblue3"        "lightskyblue4"        "lightslateblue"
## [436] "lightslategray"       "lightslategrey"       "lightsteelblue"
## [439] "lightsteelblue1"      "lightsteelblue2"      "lightsteelblue3"
## [442] "lightsteelblue4"      "lightyellow"          "lightyellow1"
## [445] "lightyellow2"         "lightyellow3"         "lightyellow4"
## [448] "limegreen"            "linen"                "magenta"
## [451] "magenta1"             "magenta2"             "magenta3"
## [454] "magenta4"             "maroon"               "maroon1"
## [457] "maroon2"              "maroon3"              "maroon4"
## [460] "mediumaquamarine"     "mediumblue"           "mediumorchid"
## [463] "mediumorchid1"        "mediumorchid2"        "mediumorchid3"
## [466] "mediumorchid4"        "mediumpurple"         "mediumpurple1"
## [469] "mediumpurple2"        "mediumpurple3"        "mediumpurple4"
## [472] "mediumseagreen"       "mediumslateblue"      "mediumspringgreen"
## [475] "mediumturquoise"      "mediumvioletred"      "midnightblue"
## [478] "mintcream"            "mistyrose"            "mistyrose1"
```

```
## [481] "mistyrose2"          "mistyrose3"          "mistyrose4"
## [484] "moccasin"            "navajowhite"         "navajowhite1"
## [487] "navajowhite2"        "navajowhite3"        "navajowhite4"
## [490] "navy"                "navyblue"            "oldlace"
## [493] "olivedrab"           "olivedrab1"          "olivedrab2"
## [496] "olivedrab3"          "olivedrab4"          "orange"
## [499] "orange1"             "orange2"             "orange3"
## [502] "orange4"             "orangered"           "orangered1"
## [505] "orangered2"          "orangered3"          "orangered4"
## [508] "orchid"              "orchid1"             "orchid2"
## [511] "orchid3"             "orchid4"             "palegoldenrod"
## [514] "palegreen"           "palegreen1"          "palegreen2"
## [517] "palegreen3"          "palegreen4"          "paleturquoise"
## [520] "paleturquoise1"      "paleturquoise2"      "paleturquoise3"
## [523] "paleturquoise4"      "palevioletred"       "palevioletred1"
## [526] "palevioletred2"      "palevioletred3"      "palevioletred4"
## [529] "papayawhip"          "peachpuff"           "peachpuff1"
## [532] "peachpuff2"          "peachpuff3"          "peachpuff4"
## [535] "peru"                "pink"                "pink1"
## [538] "pink2"               "pink3"               "pink4"
## [541] "plum"                "plum1"               "plum2"
## [544] "plum3"               "plum4"               "powderblue"
## [547] "purple"              "purple1"             "purple2"
## [550] "purple3"             "purple4"             "red"
## [553] "red1"                "red2"                "red3"
## [556] "red4"                "rosybrown"           "rosybrown1"
## [559] "rosybrown2"          "rosybrown3"          "rosybrown4"
## [562] "royalblue"           "royalblue1"          "royalblue2"
## [565] "royalblue3"          "royalblue4"          "saddlebrown"
## [568] "salmon"              "salmon1"             "salmon2"
## [571] "salmon3"             "salmon4"             "sandybrown"
## [574] "seagreen"            "seagreen1"           "seagreen2"
## [577] "seagreen3"           "seagreen4"           "seashell"
## [580] "seashell1"           "seashell2"           "seashell3"
## [583] "seashell4"           "sienna"              "sienna1"
## [586] "sienna2"             "sienna3"             "sienna4"
## [589] "skyblue"             "skyblue1"            "skyblue2"
## [592] "skyblue3"            "skyblue4"            "slateblue"
## [595] "slateblue1"          "slateblue2"          "slateblue3"
## [598] "slateblue4"          "slategray"           "slategray1"
## [601] "slategray2"          "slategray3"          "slategray4"
## [604] "slategrey"           "snow"                "snow1"
## [607] "snow2"               "snow3"               "snow4"
## [610] "springgreen"         "springgreen1"        "springgreen2"
## [613] "springgreen3"        "springgreen4"        "steelblue"
## [616] "steelblue1"          "steelblue2"          "steelblue3"
## [619] "steelblue4"          "tan"                 "tan1"
## [622] "tan2"                "tan3"                "tan4"
## [625] "thistle"             "thistle1"            "thistle2"
## [628] "thistle3"            "thistle4"            "tomato"
## [631] "tomato1"             "tomato2"             "tomato3"
## [634] "tomato4"             "turquoise"           "turquoise1"
## [637] "turquoise2"          "turquoise3"          "turquoise4"
## [640] "violet"              "violetred"           "violetred1"
```

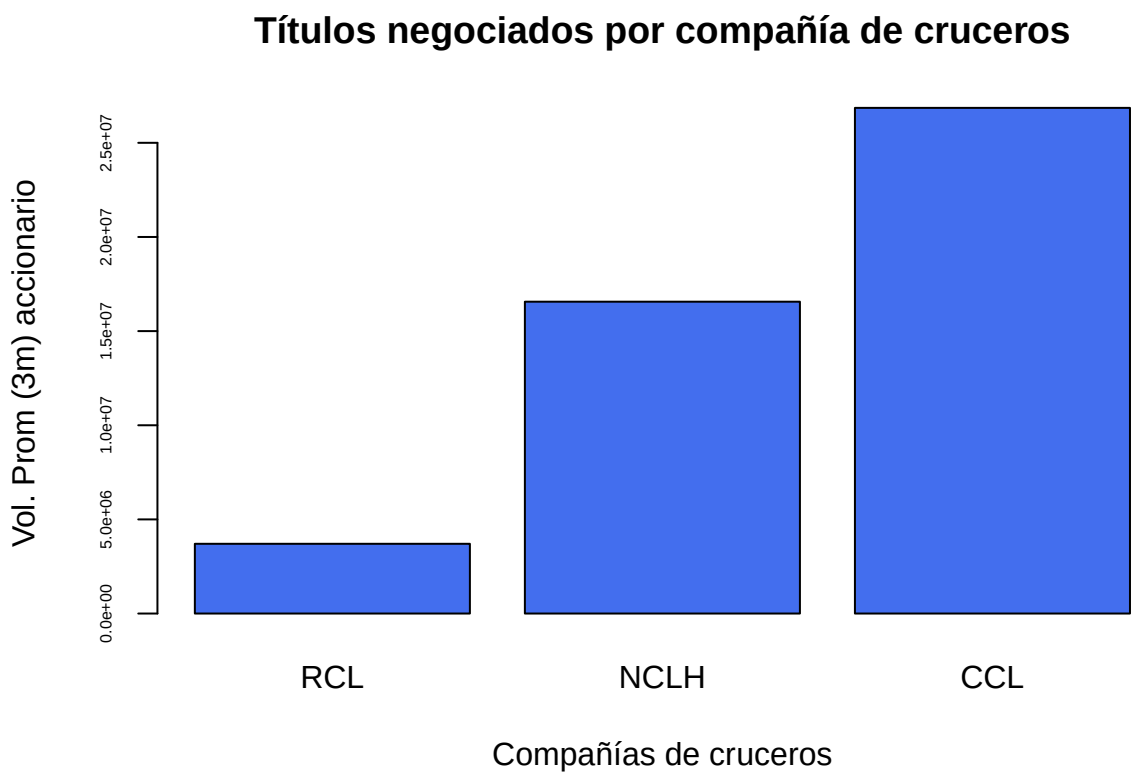
```
## [643] "violetred2"          "violetred3"          "violetred4"
## [646] "wheat"               "wheat1"              "wheat2"
## [649] "wheat3"              "wheat4"              "whitesmoke"
## [652] "yellow"              "yellow1"             "yellow2"
## [655] "yellow3"             "yellow4"             "yellowgreen"
```

7.2. Función `barplot()`

La función `barplot()`, se utiliza para realizar gráficos de barra. Esta gráfica nos muestra la frecuencia con la que se han observado los datos de una variable discreta. Suponga se tiene la siguiente información:

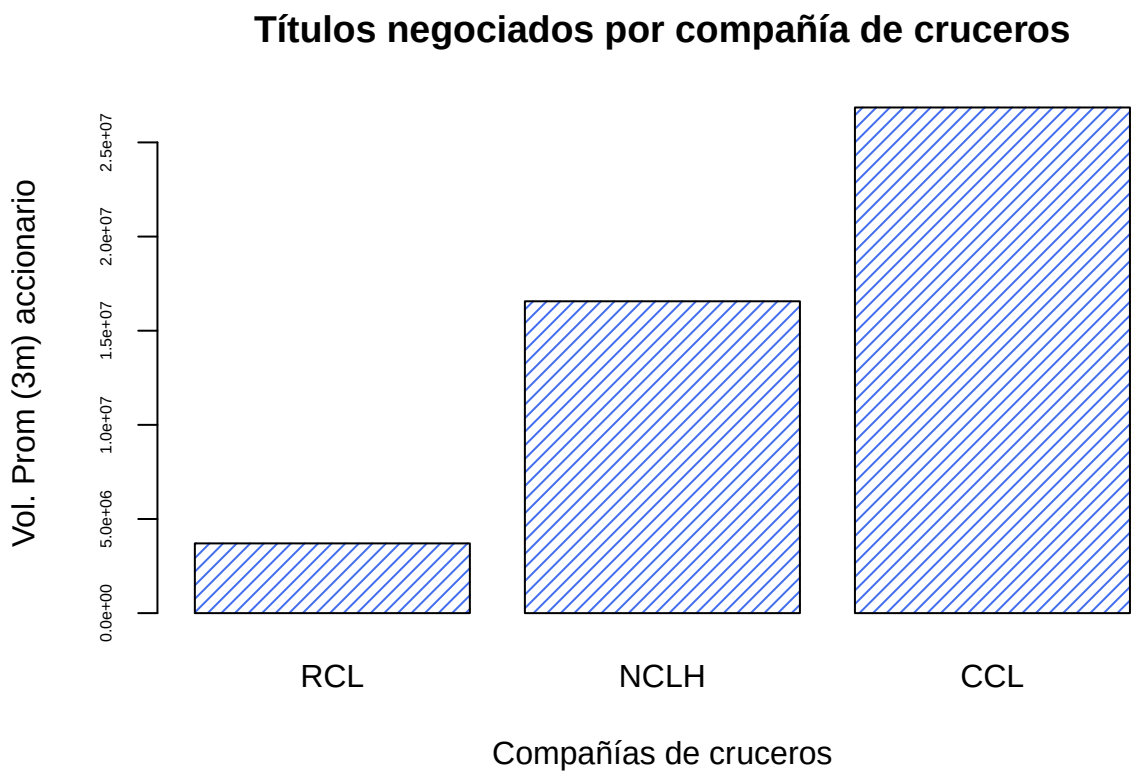
Compañía	Conteo
RCL	3704048
NCLH	16561770
CCL	C26855590

```
barplot(c(3704048,16561770,26855590), names.arg = c("RCL","NCLH","CCL"),
        col = "royalblue2",
        main = "Títulos negociados por compañía de cruceros",
        xlab = "Compañías de cruceros",
        ylab = "Vol. Prom (3m) accionario",
        cex.axis = 0.5, cex.lab = 1)
```



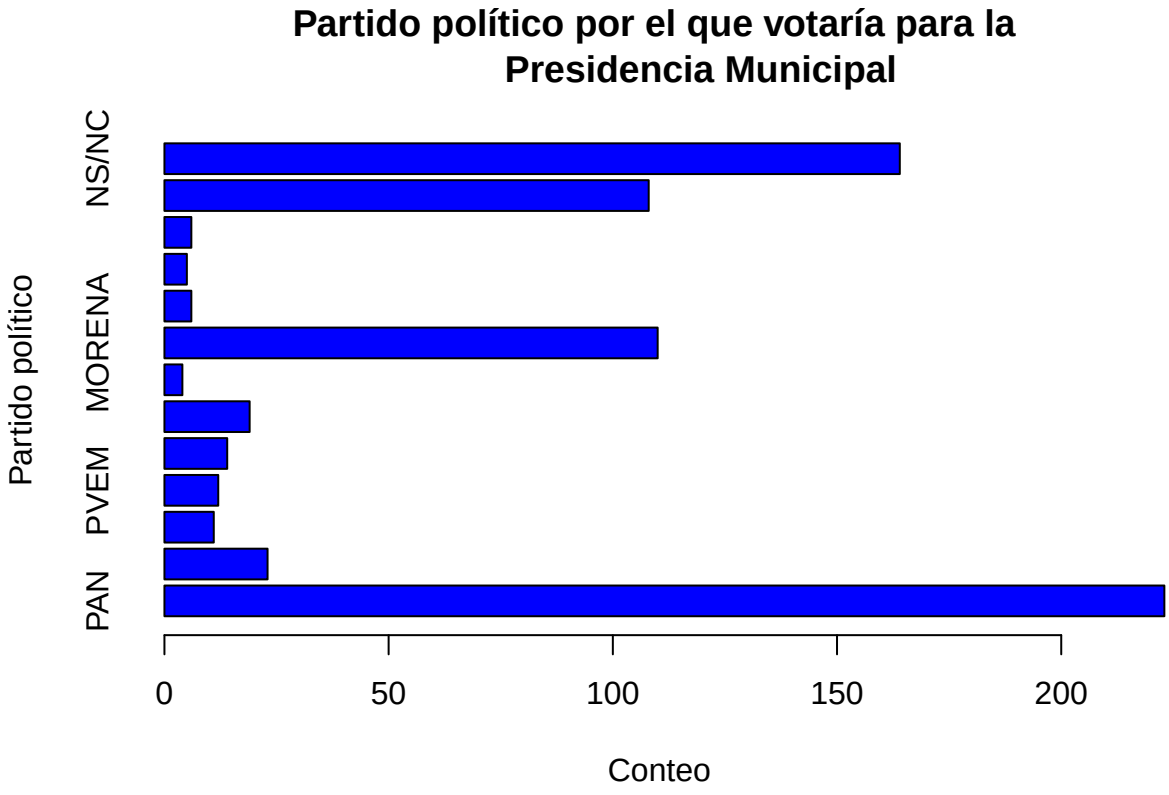
Para cambiar el tipo de relleno de las barras se puede utilizar lo siguiente:

```
barplot(c(3704048,16561770,26855590), names.arg = c("RCL","NCLH","CCL"),
        col = "royalblue2",
        main = "Títulos negociados por compañía de cruceros",
        xlab = "Compañías de cruceros", ylab = "Vol. Prom (3m) accionario",
        cex.axis = 0.5, cex.lab = 1, density = 20)   # density = 20
```



Para cambiar la dirección de las barras se puede utilizar el siguiente código:

```
barplot(c(223,23,11,12,14,19,4,110,6,5,6,108,164),
        names.arg = c("PAN","PRI","PRD","PVEM","PT","MOV.CIUD",
                      "NVA.ALIANZA","MORENA","PARTIDO LIBRE DE AGS",
                      "UNIDOS PODEMOS MÁS","CAND.IND","NINGUNO",
                      "NS/NC"),
        col = "blue",
        main = "Partido político por el que votaría para la
        Presidencia Municipal",
        xlab ="Conteo",
        ylab = "Partido político",
        horiz = TRUE)
```




```
# horiz = TRUE, permite poner las barras de manera horizontal
```

Considere el siguiente ejemplo:

```
nacionalidad <- sample(c("Mexicana","Estadounidense","Canadiense"),
                       size=30,replace = TRUE)
# La función sample() permite tomar una muestra aleatoria de una población,
# dicha muestra puede ser con remplazo o sin remplazo.

nacionalidad
```

```
##  [1] "Canadiense"     "Estadounidense" "Estadounidense" "Canadiense"
##  [5] "Canadiense"     "Estadounidense" "Canadiense"     "Canadiense"
##  [9] "Estadounidense" "Mexicana"       "Canadiense"     "Estadounidense"
## [13] "Estadounidense" "Canadiense"     "Mexicana"       "Estadounidense"
## [17] "Mexicana"       "Estadounidense" "Canadiense"     "Canadiense"
## [21] "Estadounidense" "Mexicana"       "Canadiense"     "Canadiense"
## [25] "Estadounidense" "Mexicana"       "Mexicana"       "Mexicana"
## [29] "Canadiense"     "Canadiense"
```

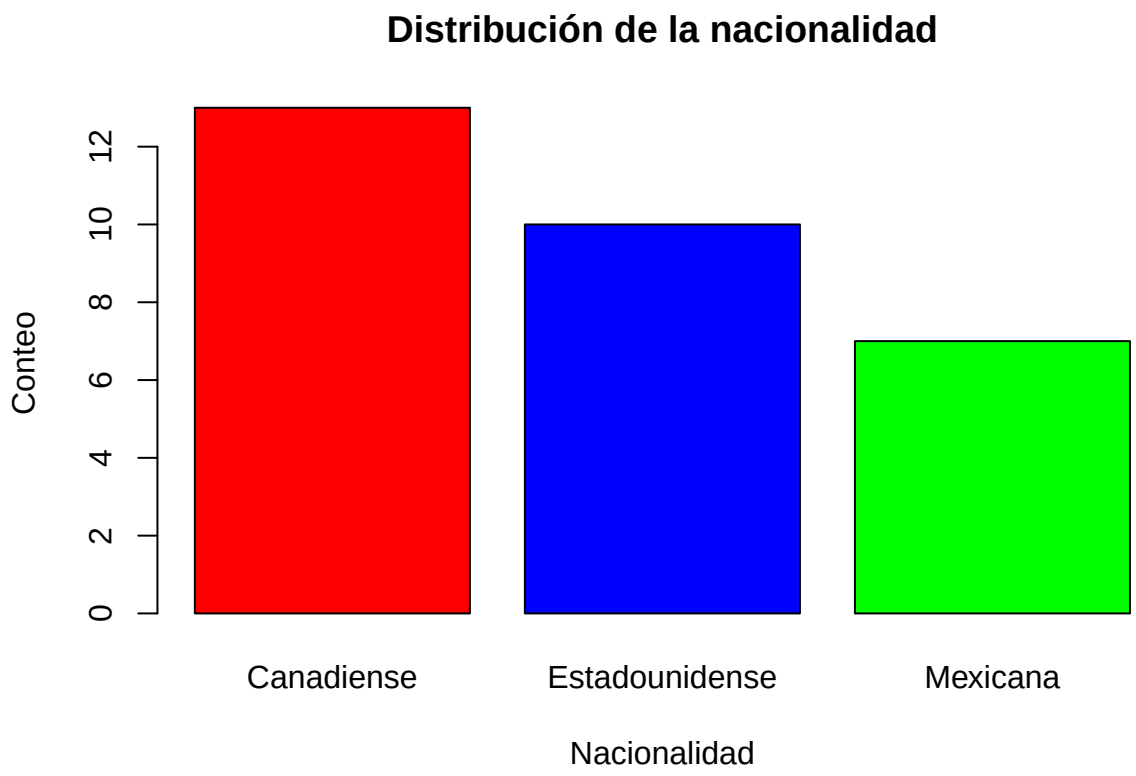
```
tabla <- table(nacionalidad)
# La función table() permite hacer un conteo de cada nivel de la variable

tabla
```

```
## nacionalidad
##     Canadiense Estadounidense       Mexicana
##             13             10              7
```

```
names(tabla) # La función names() retorna el nombre de las columnas
```

```
## [1] "Canadiense"     "Estadounidense" "Mexicana"
```

```
barplot(tabla,names.arg =names(tabla) ,col=c("red","blue","green"),
        main = "Distribución de la nacionalidad",
        xlab = "Nacionalidad", ylab="Conteo" )
```



Considere el siguiente Data frame:

```
Estatus_Finales <- data.frame(Sexo =sample(c("Hombre","Mujer"),
                                           30, replace = TRUE),
                              Estatus = sample(c("Aprobado","Reprobado"),
                                               30, replace = TRUE))
# Tablas cruzadas
Estatus_Finales
```

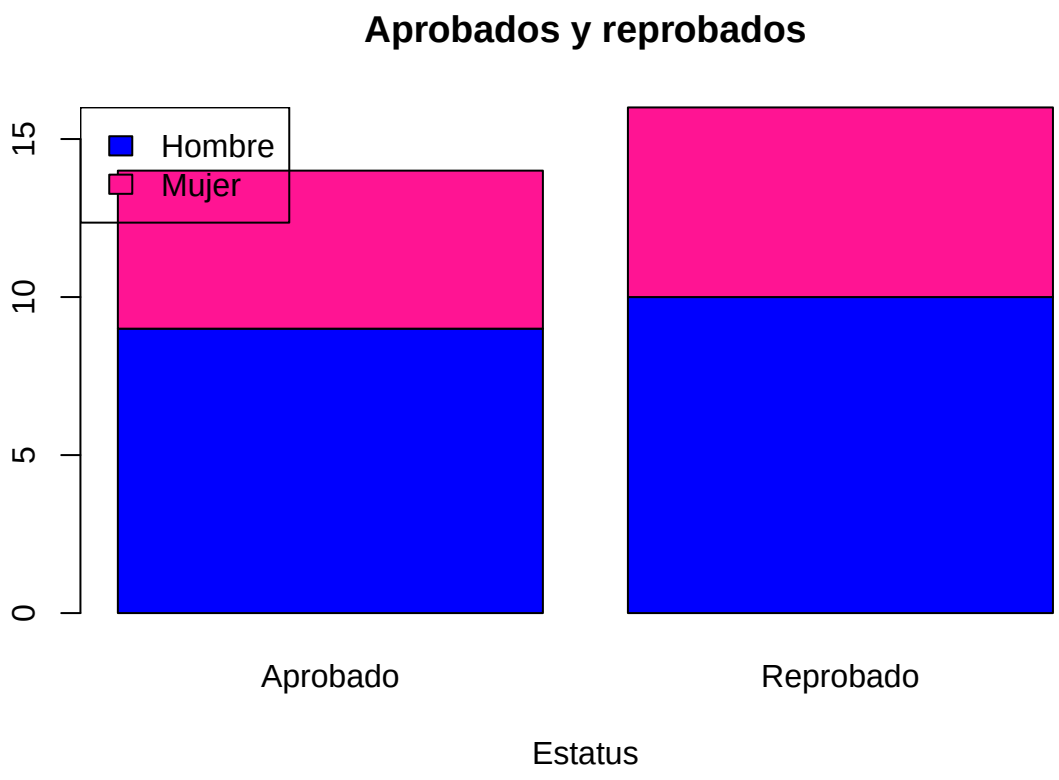
```
##      Sexo   Estatus
## 1   Mujer  Aprobado
## 2  Hombre Reprobado
## 3   Mujer Reprobado
## 4  Hombre  Aprobado
## 5   Mujer  Aprobado
## 6   Mujer Reprobado
## 7  Hombre Reprobado
## 8  Hombre Reprobado
## 9  Hombre Reprobado
```

```
## 10 Hombre Reprobado
## 11  Mujer Reprobado
## 12 Hombre Reprobado
## 13 Hombre  Aprobado
## 14 Hombre  Aprobado
## 15 Hombre  Aprobado
## 16  Mujer Reprobado
## 17 Hombre  Aprobado
## 18 Hombre  Aprobado
## 19  Mujer Reprobado
## 20 Hombre  Aprobado
## 21 Hombre Reprobado
## 22  Mujer  Aprobado
## 23 Hombre Reprobado
## 24 Hombre Reprobado
## 25  Mujer  Aprobado
## 26 Hombre Reprobado
## 27  Mujer  Aprobado
## 28 Hombre  Aprobado
## 29  Mujer Reprobado
## 30 Hombre  Aprobado
```

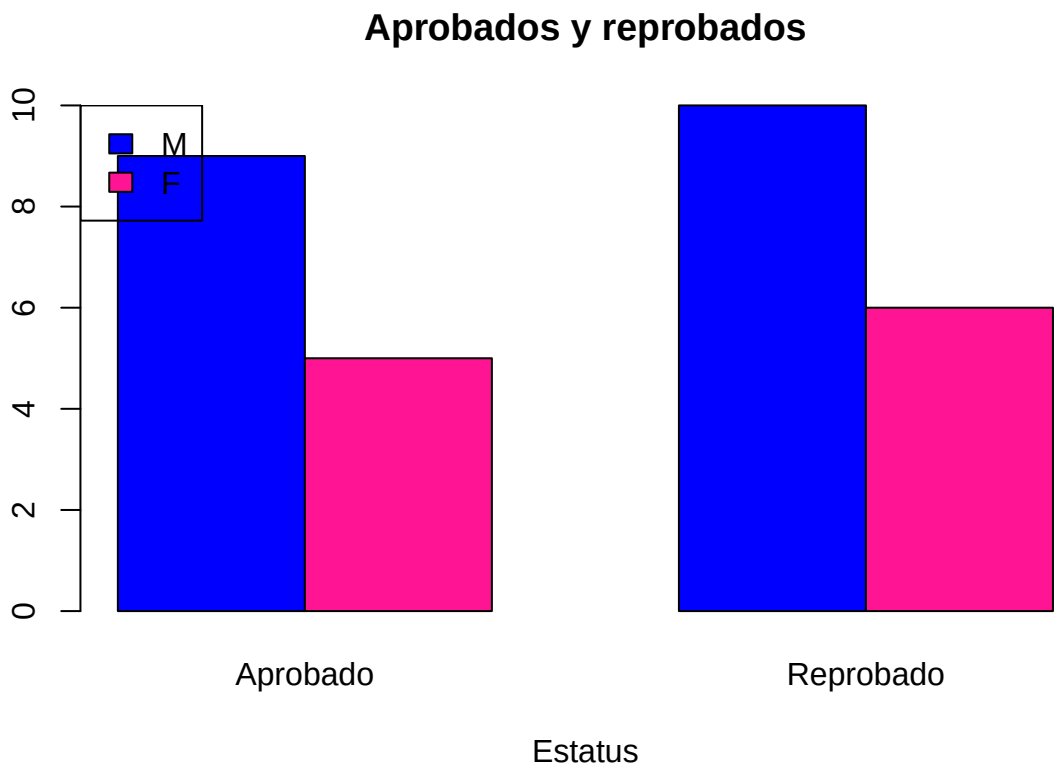
```
reporte <- table(Estatus_Finales$Sexo, Estatus_Finales$Estatus)
reporte
```

```
##
##          Aprobado Reprobado
##   Hombre        9        10
##   Mujer         5         6
```

```
# Gráfica de barras
barplot(reporte, main = "Aprobados y reprobados", xlab = "Estatus",
        col = c("blue","deeppink"))
legend("topleft", c("Hombre","Mujer"), fill = c("blue","deeppink"))
```



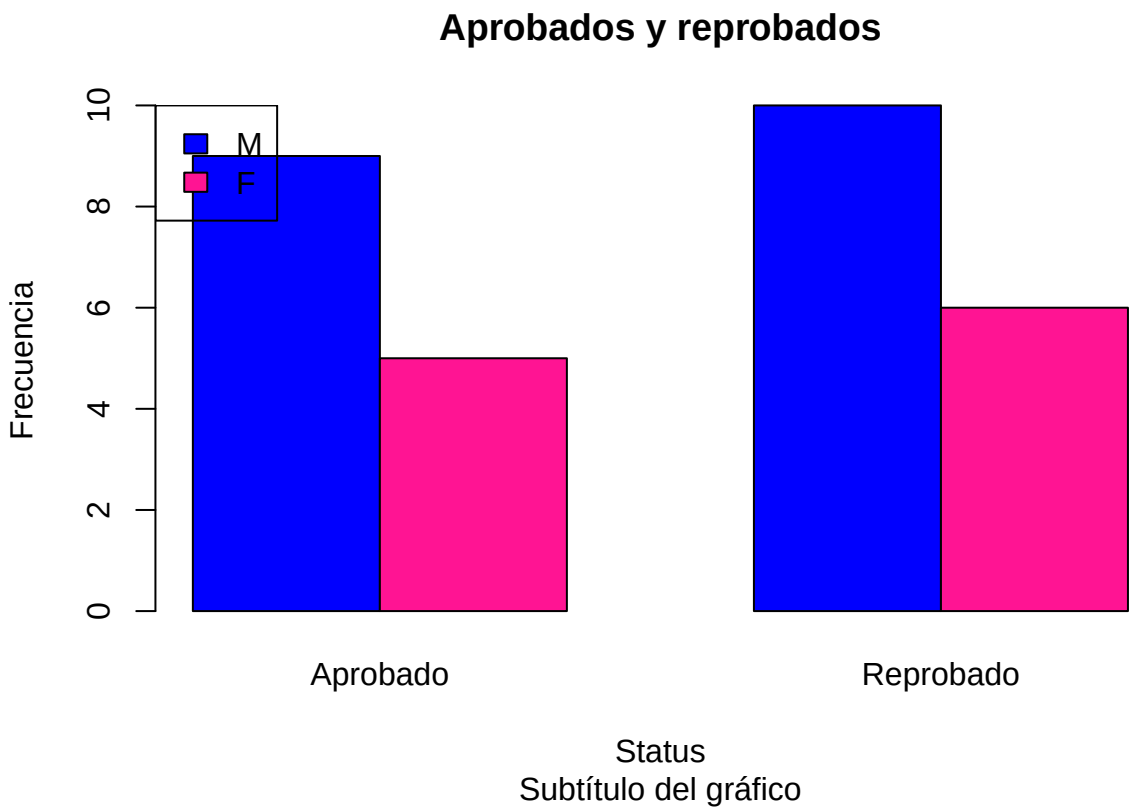

```
barplot(reporte, main = "Aprobados y reprobados", xlab = "Estatus",
        col = c("blue","deeppink"), beside = TRUE)
#  beside = TRUE, permite poner una barra junto a la otra

legend("topleft", c("M","F"), fill = c("blue","deeppink"))
```



7.2.1. Agregar texto a la gráfica

Para agregar texto a la gráfica, se utiliza los parámetros "main" para título del gráfico, "sub" para subtítulo del gráfico, "xlab" nombre del eje x y "ylab" nombre del eje y. En el siguiente código se visualiza un gráfico con texto.

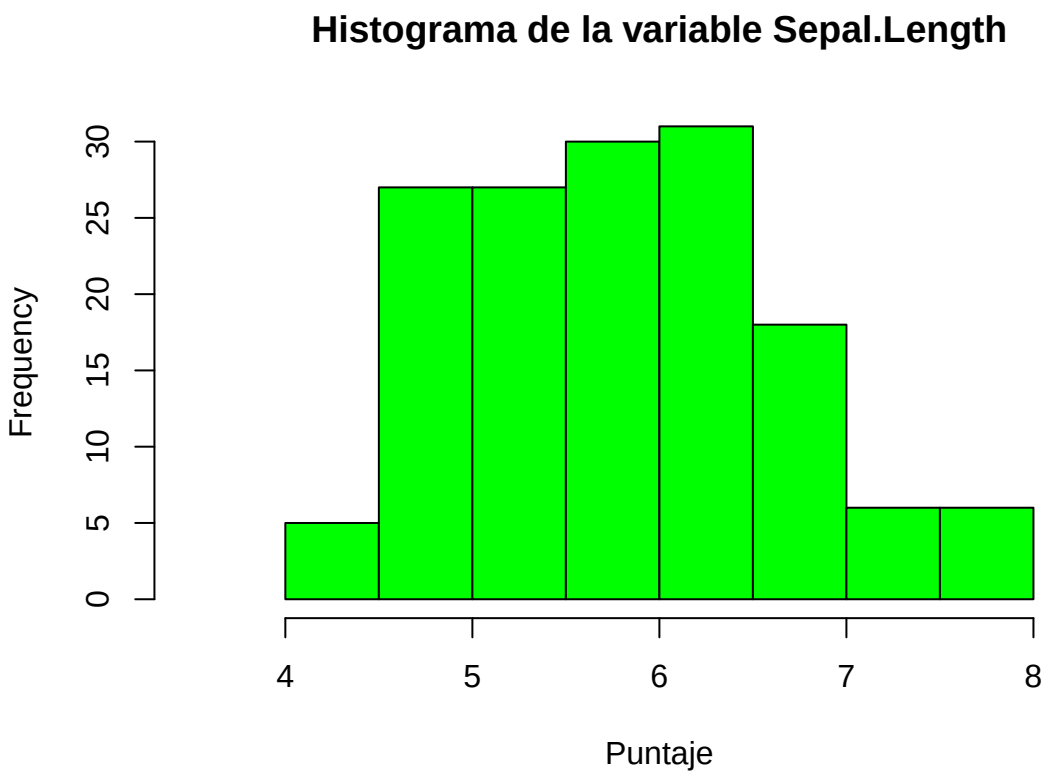
```
barplot(reporte, main = "Aprobados y reprobados", xlab = "Status",
        col = c("blue","deeppink"), beside = TRUE,
        sub = "Subtítulo del gráfico",ylab = "Frecuencia")
legend("topleft", c("M","F"), fill = c("blue","deeppink"))
```



7.3. Función `hist()`

La función `hist()`, se utiliza para realizar histograma. Esta gráfica nos muestra la distribución de los datos usando barras. Suponga que se tiene la información de la edad de 32 niños y se desea conocer su distrución, para ello se utiliza el siguiente código:

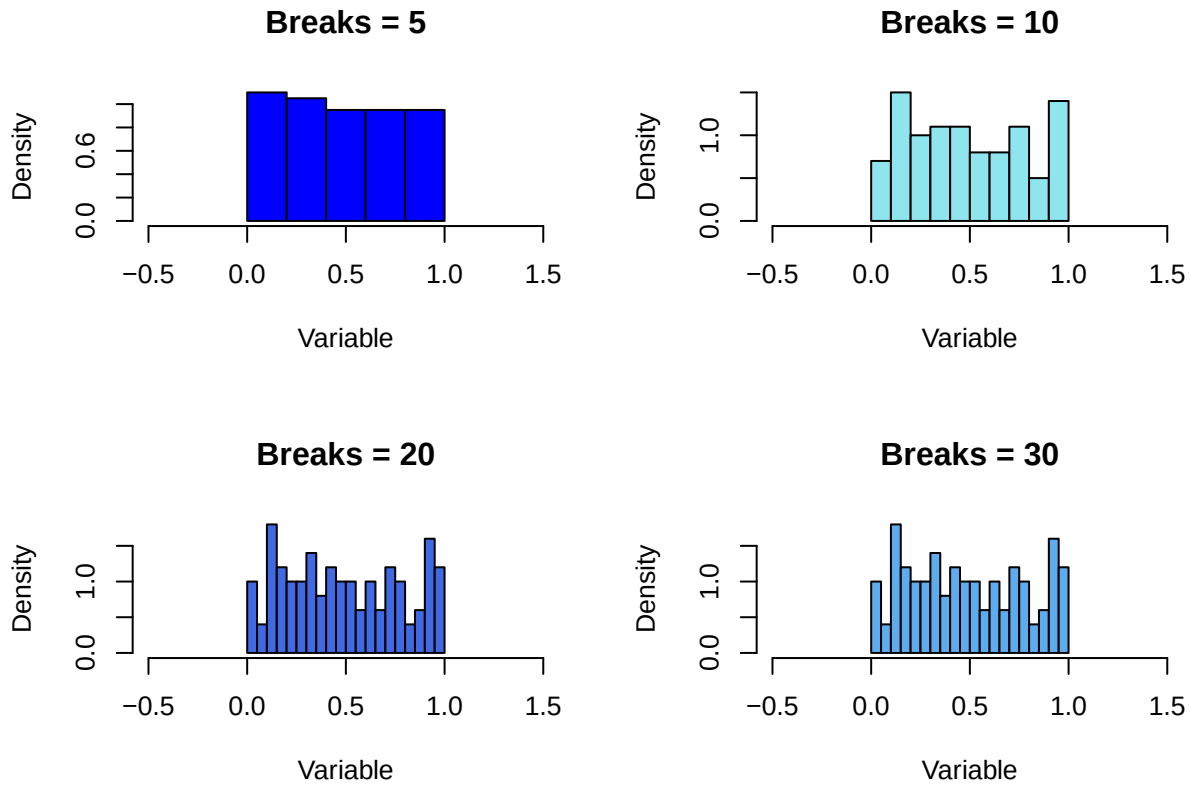
```
hist(iris$Sepal.Length, main = "Histograma de la variable Sepal.Length",
     xlab = "Puntaje", xlim = c(3.5, 8.5), col = colors()[254])
```



7.3.1. Número de cortes

```
variable <- runif(100)
# La función runif genera una muestra aleatoria de la distribución uniforme
# continua entre 0 y 1.

par(mfrow=c(2,2)) # Permite crear un gráfico con 4 figuras

hist( variable,
      main="Breaks = 5",
      xlab="Variable",
      xlim = c(-0.5,1.5), #Rango del eje x
      col= colors()[27],
      freq = FALSE,  # Distribución de probabilidad
      breaks= 5 # Número de cortes
      )
hist( variable,
      main="Breaks = 10",
      xlab="Variable",
      xlim = c(-0.5,1.5), #Rango del eje x
      col= colors()[44],
      freq = FALSE,  # Distribución de probabilidad
      breaks= 10 # Número de cortes
      )
hist( variable,
```

```
      main="Breaks = 20",
      xlab="Variable",
      xlim = c(-0.5,1.5), #Rango del eje x
      col= colors()[562],
      freq = FALSE,  # Distribución de probabilidad
      breaks= 20 # Número de cortes
      )
hist( variable,
      main="Breaks = 30",
      xlab="Variable",
      xlim = c(-0.5,1.5), #Rango del eje x
      col= colors()[617],
      freq = FALSE,  # Distribución de probabilidad
      breaks= 30 # Número de cortes
      )
```



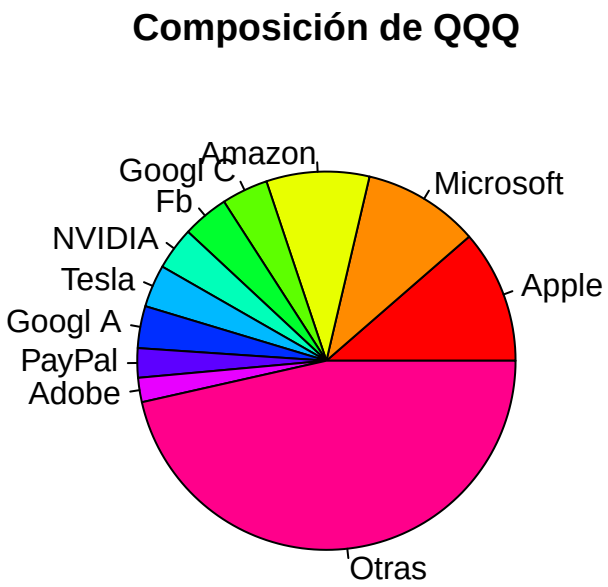

```
par(mfrow=c(1,1)) # Reestablecer el entorno, un gráfico con 1 figura
```

7.4. Función `pie()`

La función `pie()`, se utiliza para realizar diagramas de pastel. El primer argumento de esta función, son lo valores que demarcarán las divisiones del círculo que representa el 100% del área. Luego se indican los valores que determinarán la construcción de etiquetas. En el siguiente ejemplo se muestra un diagrama de pastel:

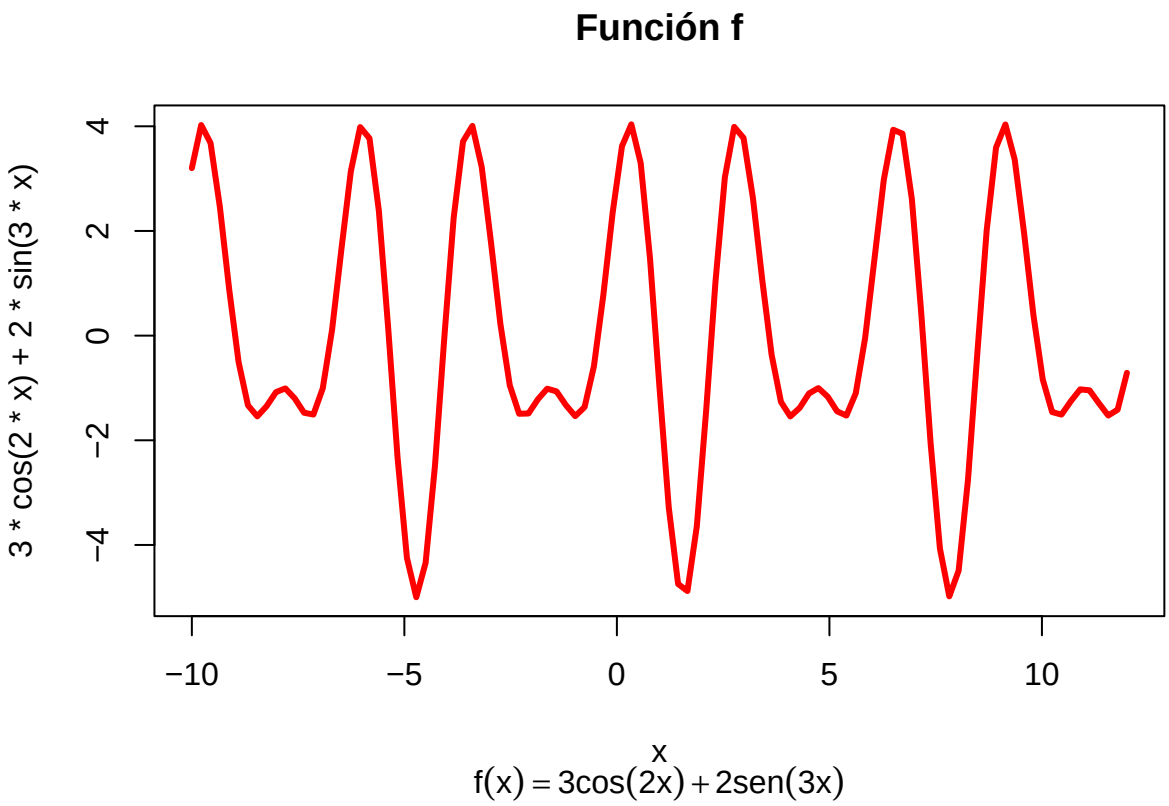
```
porcentajes <- c(0.1139,0.0995,0.0880,0.0397,0.0396,0.0365,0.0363,0.0359,
                 0.0251,0.0208,0.4648)
```

```
etiqueta_porcentajes <- c("Apple","Microsoft","Amazon",
                          "Googl C", "Fb","NVIDIA",
                          "Tesla","Googl A","PayPal","Adobe","Otras")

pie(porcentajes, labels=etiqueta_porcentajes,
    main = "Composición de QQQ", col=rainbow(length(porcentajes)))
```



7.5. Función curve()

```
curve(3*cos(2*x)+2*sin(3*x), #Ley de asignación, f(x)=3cos(2x)+2cos(3x)
      from = -10, # Inicio
      to =12, # Final
      main= "Función f", # Título
      col="red",  #color
      lwd = 3, # Grosor del línea
      sub = expression(f(x)==3*cos(2*x)+2*sen(3*x))
      )
```



8. R Markdown

8.1. Crear un archivo de R Markdown

Es muy sencillo generar documentos Rmarkdown desde Rstudio, que combinan texto, imágenes e instrucciones de R, más los resultados que dichas instrucciones produzcan (estos resultados pueden ser simples valores numéricos, tablas o gráficos). Dichos documentos pueden exportarse en tres formatos:

-**html:** para ser visualizados a través de navegadores web.

-**doc:** para ser directamente editados con Microsoft Word o LibreOffice Writer.

-**pdf:** Para esta última opción es preciso que el ordenador del usuario disponga de una instalación válida de LaTeX. LaTeX es un completo (y complejo) sistema de edición de textos de código abierto que puede descargarse libremente para Windows (Miktex), para Mac (Mactex) y para Linux (TexLive, si bien lo habitual es que en los sistemas linux texlive venga ya instalado por defecto).

Para crear un documento Rmarkdown en Rstudio basta acceder a File -> New File -> R Markdown:

8.2. Sintaxis Markdown

8.2.1. Cabeceras (Títulos y subtítulos)

```
# Esto es un H1
## Esto es un H2
```

```
### Esto es un H3
#### Esto es un H4
##### Esto es un H5
###### Esto es un H6
```

8.2.2. Viñetas y listas numeradas

```
# - Elemento
# - Elemento
# - Elemento
# 1. Elemento
# 2. Elemento
# 3. Elemento
```

- Elemento
- Elemento
- Elemento

1. Elemento
2. Elemento
3. Elemento

8.2.3. En lugar de - podemos utilizar * :

```
# * Elemento
# * Elemento
#     + Subelemento
#         - Desagregación
# * Elemento
```

- Elemento
- Elemento
  - Subelemento
    - Desagregación
- Elemento

8.2.4. Formatos

Para poner el texto en cursiva utilizamos * o _ antes y después del texto.

`*cursiva*` o `_cursiva_` *cursiva* o *cursiva*

Para poner el texto en negrita utilizamos ** o ___ antes y después del texto.

`**negrita**` o `__negrita__` **negrita** o **negrita**

8.2.5. Color

```
<span style="color:blue">Texto *azul cursiva* normal</span>
```

Texto *azul cursiva* normal

8.2.6. Tabla

Para crear tablas es debemos indicar cuales son los elementos de la cabecera y separar los campos con el símbolo |.

```
Cabecera A | Cabecera B
– | –
Campo A1 | Campo B1
Campo A2 | Campo B2
```

Cabecera A	Cabecera B
Campo A1	Campo B1
Campo A2	Campo B2

8.2.7. Ecuaciones

```
La *función Gamma* satisface $\Gamma(n) = (n-1)!\quad\forall n\in\mathbb N$ a través de
la integral de Euler

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt\,.$$
```

La *función Gamma* satisface $\Gamma(n) = (n-1)! \quad \forall n \in \mathbb{N}$ a través de la integral de Euler

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt\,.$$

8.2.8. Lista de algunos símbolos matemáticos

Significado	Código	Resultado
Letra alfa	`$\alpha$`	α
Letra beta	`$\beta$`	β
Letra gamma	`$\gamma$`,`$\Gamma$`	γ, Γ
Letra mi	`$\mu$`	μ
Letra sigma	`$\sigma$`, `$\Sigma$`	σ, Σ
Equis barra	`$\bar{x}$`	$\bar{x}$
Potencia/Superíndice	`$a^{2x}$`	a^{2x}
Subíndice	`$x_{ij}$`	x_{ij}
Raíz Cuadrada	`$\sqrt{x}$`	$\sqrt{x}$
Aproximado	`$\approx$`	$\approx$
Desigualdad	`$\neq$`	$\neq$
Más menos	`$\pm$`	$\pm$
Sumatorio	`$\sum_{i=0}^{n}$`	$\sum_{i=0}^{n}$
Integral	`$\int_{a}^{b}$`	$\int_a^b$

Significado	Código	Resultado
Negrita	`$\mathbf{x}$`	$\mathbf{x}$
Color azul	`$\color{blue}{\text{x}}$`	$\color{blue}{\text{x}}$
Espacio entre símbolos	`$x\ y$`	$x\ y$
Texto	`$\text{Texto}$`	Texto
Fracción pequeña	`$\frac{x}{y}$`	$\frac{x}{y}$
Fracción grande	`$\dfrac{x}{y}$`	$\dfrac{x}{y}$

Podemos especificar que los delimitadores en general, se adapten a la altura de la expresión que rodean, usando `\left` y `\right` (Si se pone un punto, posterior a la palabra left o right, el lado donde va el punto, no operará).

Ejemplo 1 al no usarlos: `$(\dfrac{a}{b})$` $(\dfrac{a}{b})$

Ejemplo 1 al usarlos: `$\left(\dfrac{a}{b}\right)$` $\left(\dfrac{a}{b}\right)$

Ejemplo 2 al no usarlos: `$$\int_{0}^1 x\ dx = \frac{x^2}{2}]_{0}^{1} = \frac{1}{2}$$`

$$\int_{0}^1 x\ dx = \frac{x^2}{2}]_{0}^{1} = \frac{1}{2}$$

Ejemplo 2 al usarlos: `$$\int_{0}^1 x\ dx = \left.\frac{x^2}{2}\right]_{0}^{1} = \frac{1}{2}$$`

$$\int_{0}^1 x\ dx = \left.\frac{x^2}{2}\right]_{0}^{1} = \frac{1}{2}$$