Guía básica de Introducción a R

Este documento fue realizado en *R Markdown*, utilizando funciones creadas por **Edgar Gerardo Alarcón González** que pueden ser consultadas entrando a su GitHub personal: https://github.com/A1arcon/R_Actuarial. De igual manera las ideas y estructura originales fueron realizadas por **Enrique Reyes** en sus notas sobre Modelos no paramétricos y de regresión.

Introducción

R es un programa muy potente para el análisis estadístico de datos, pero no sólo se destaca en esta rama del conocimiento, tiene muchas aplicaciones por ejemplo en: Geografía, Análisis Numérico, Data Science, BigData, entre otras. Esto se debe principalmente a que es un lenguaje sencillo de contribución libre, es decir, pertenece al sistema GNU por lo tanto constantemente hay actualizaciones, nuevas funciones y corrección de errores, sigue la idea de que el conocimiento es de todos, el conocimiento no es estático, por lo tanto se deben compartir los nuevos hallazgos para seguir mejorando.

R se ha popularizado, cada vez son más las empresas, organizaciones e institutos que se acercan a este programa, por ser un **software libre de multiplataforma** (compatible con Windows, Mac, Linux), con muchas paqueterías para el análisis de datos, además de tener una interfaz gráfica amigable *R Studio*.

Instalación

Como se mencionó, R es libre y multiplataforma, por ende no importa el sistema operativo que el usuario tenga, este se puede descargar e instalar de manera gratuita; la versión a utilizar será la más actual disponible, la dirección para descargar el programa se puede visualizar dando clic en los siguientes enlaces:

- Windows
- Mac
- Linux (Debian, Redhat, Suse, Ubuntu)

Pero como el interfaz gráfico no es muy amigable, utilizaremos R por medio de R Studio, ya que este es más visual, más dinámico, se hace más ágil la manipulación de instrucciones, esta interfaz también es multiplataforma y gratuita. La instalación de *R* así como la de *R Studio* se puede consultar en la Biblioteca digital del Github de Edgar Alarcón o bien dando clic aquí.

Cabe mencionar dos cosas: es necesario instalar en este orden los programas, primero *R* y posteriormente *R Studio*, si se instala primero R Studio el programa marcará muchos errores por la ausencia del lenguaje principal; y las versiones base no son necesariamente serán fijas, hay que estar al tanto de las nuevas actualizaciones e instalarlas de manera oportuna, muchas paqueterías dejan de ser estables con versiones anteriores del sistema.

Primeros pasos en R

R Studio abre por default 4 paneles (*panes*/caras):

1. Ventana de variables (Ambiente): en esta sección se mostrarán todos los objetos que se declaren en el código, es de gran utilidad conocer el nombre de las variables, esto evitará la eliminación o sobreescritura de alguna variable, así mismo sabremos que objetos eliminar cuando estos dejen de servir para nuestro análisis. Adicionalmente en esta sección se almacenará el historial del código.

2. Ventana de scripts: en esta sección se crearán y se gardarán los códigos.

3. Ventana de consola: aquí se mostrarán los resultados de los códigos, también se pueden escribir comandos pero no se guardarán.

4. Ventana gráfica: en está sección, se mostrarán las gráficas que se vayan haciendo, no sólo eso, también estarán las paqueterías, la ayuda y los documentos que están dentro del fichero predefinido (Por default en Windows esta carpeta es Documentos).

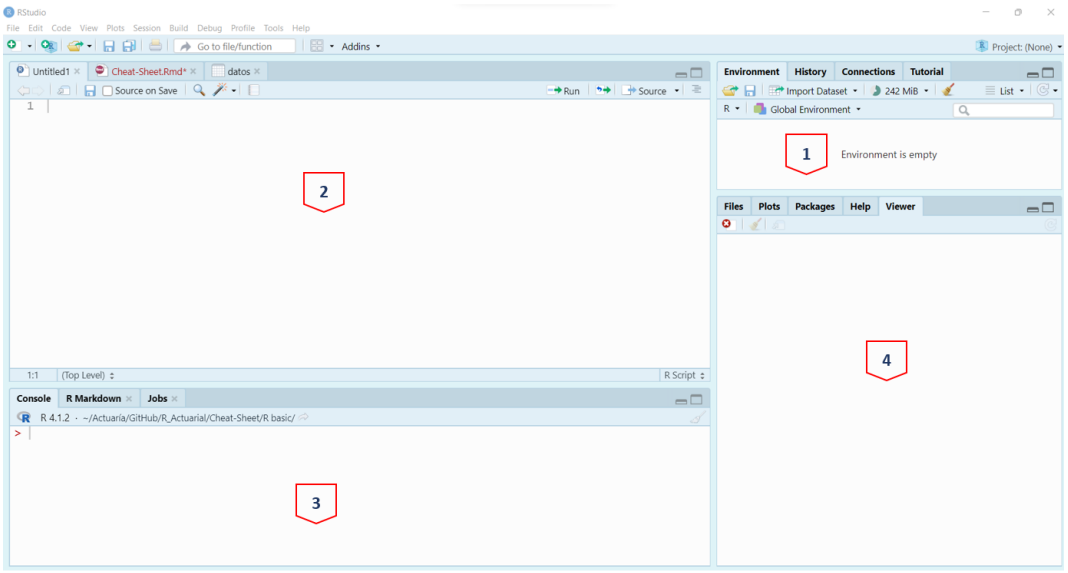


Figure 1: Interfaz principal de *R Studio*.

Directorio de trabajo

Una de las primeras cosas que debemos saber es dónde estamos trabajando, en qué carpeta se guardará todo nuestro contenido. Entonces, para saber cuál es la carpeta de trabajo, debemos poner en la consola el siguiente comando:

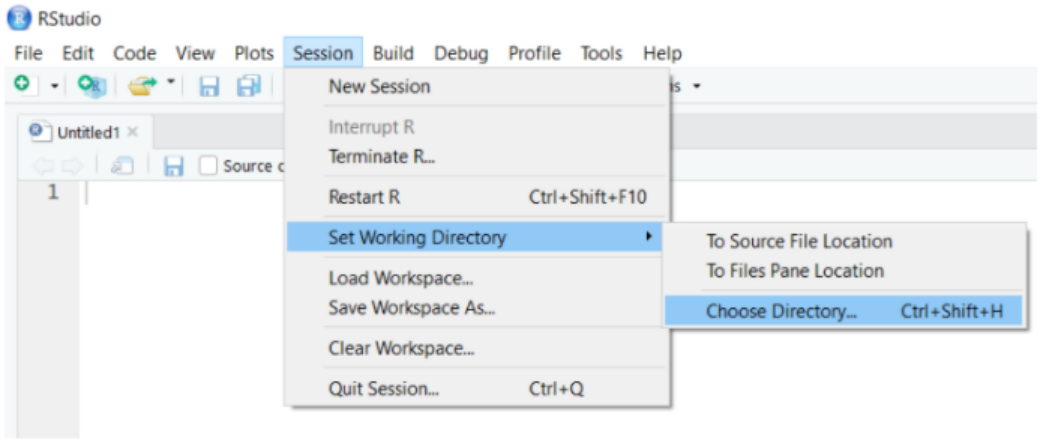
```
getwd()
```

```
## [1] "C:/Users/alarc/Documents/Actuaría/GitHub/R_Actuarial/Cheat-Sheet/R basic"
```

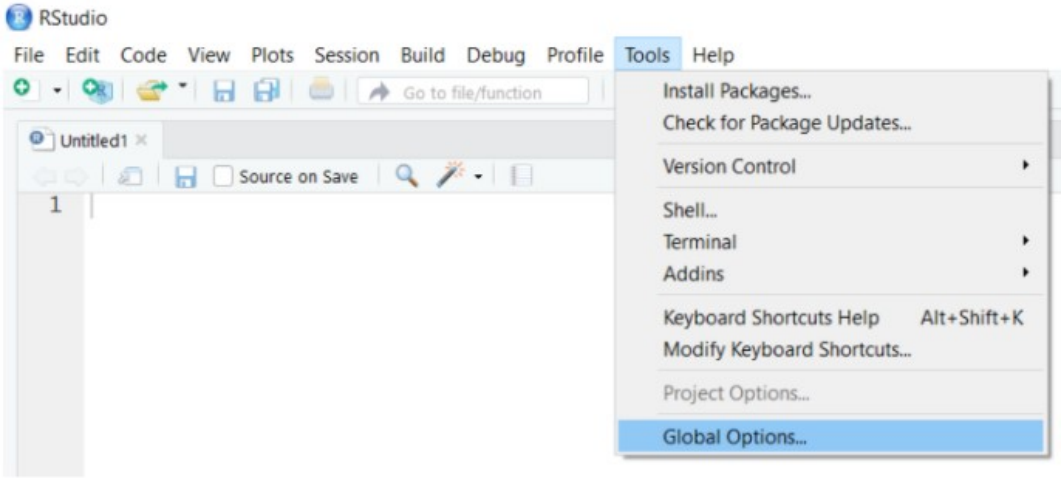
Para cambiar el directorio de trabajo necesitamos obtener la nueva carpeta y la función `setwd()`, entonces si queremos cambiar la dirección a una carpeta llamada Clases, la instrucción será la siguiente:

```
setwd("D:/Clases")
```

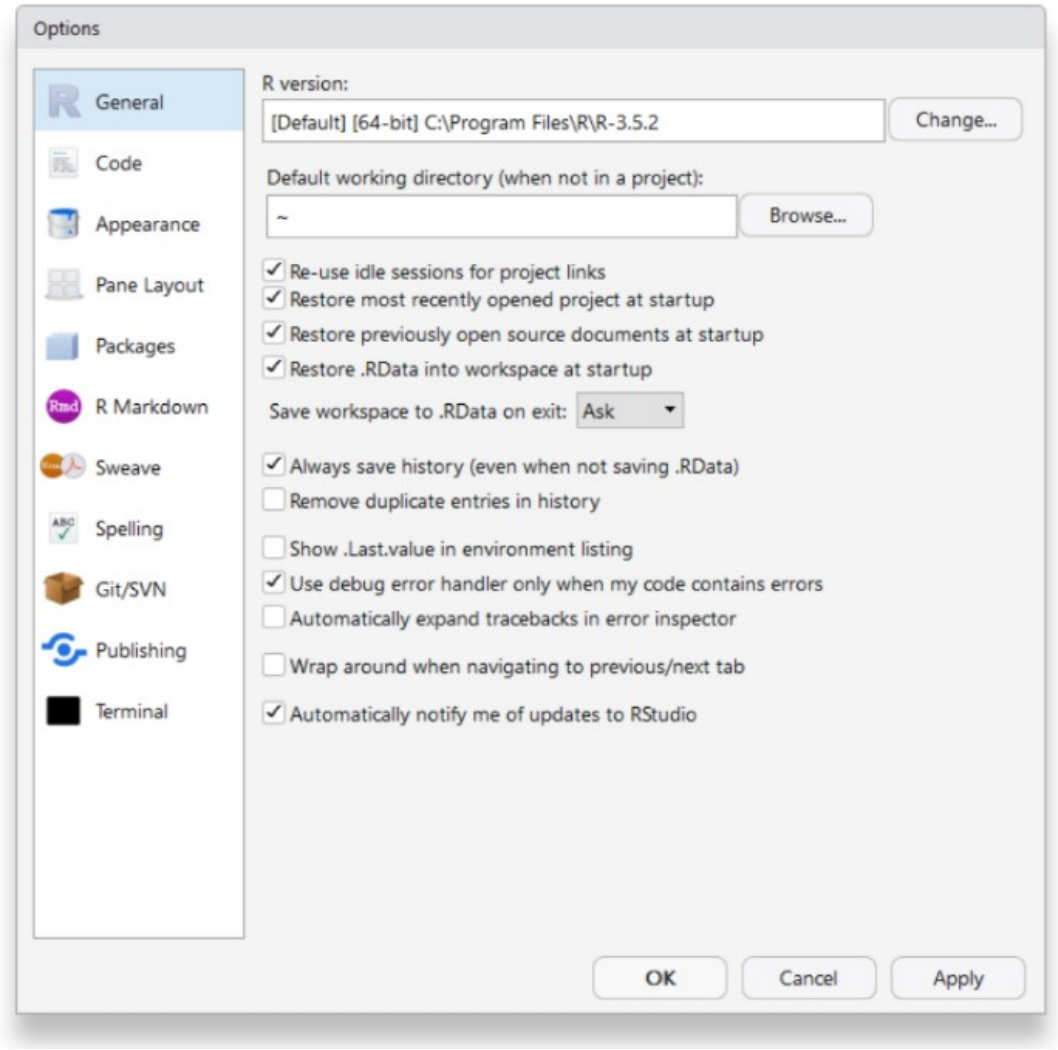
Para hacer este mismo cambio pero de manera visual, tenemos que ir a la pestaña *Session*, ir a *Set working Directory*, y seleccionar *Choose Directory*:



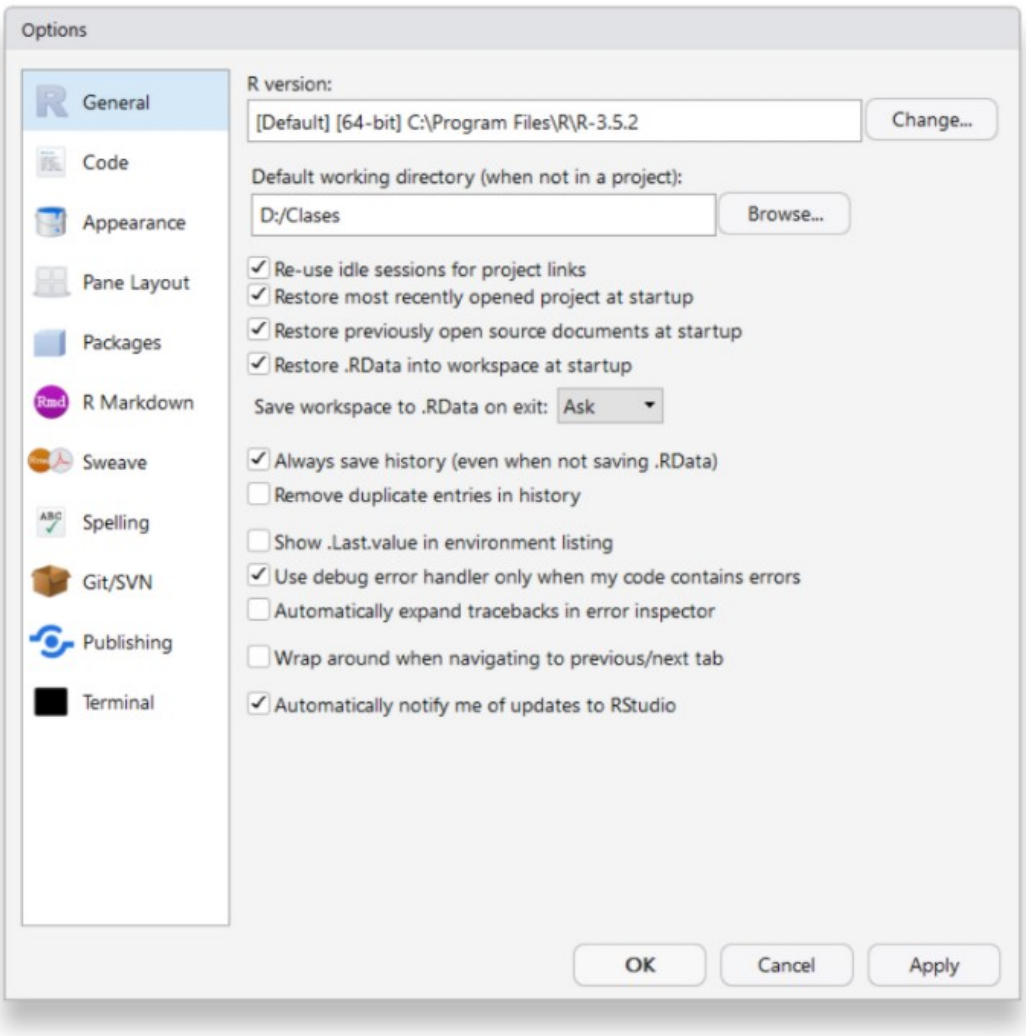
El inconveniente de seleccionar el directorio de esta forma es que en cuanto se cierre el programa se borrará esta configuración y se tendrá que hacer este procedimiento una vez más cuando se vuelva abrir el programa, si se trabajará constantemente en ese directorio se recomienda definir el directorio por *default*, esto se hace seleccionado la pestaña *Tools*, ir a la última opción *Global Options*.



después se abrirá una nueva ventana ahí se debe seleccionar la opción *General* y en la opción *Default working directory*, pulsar el botón *Browse*.

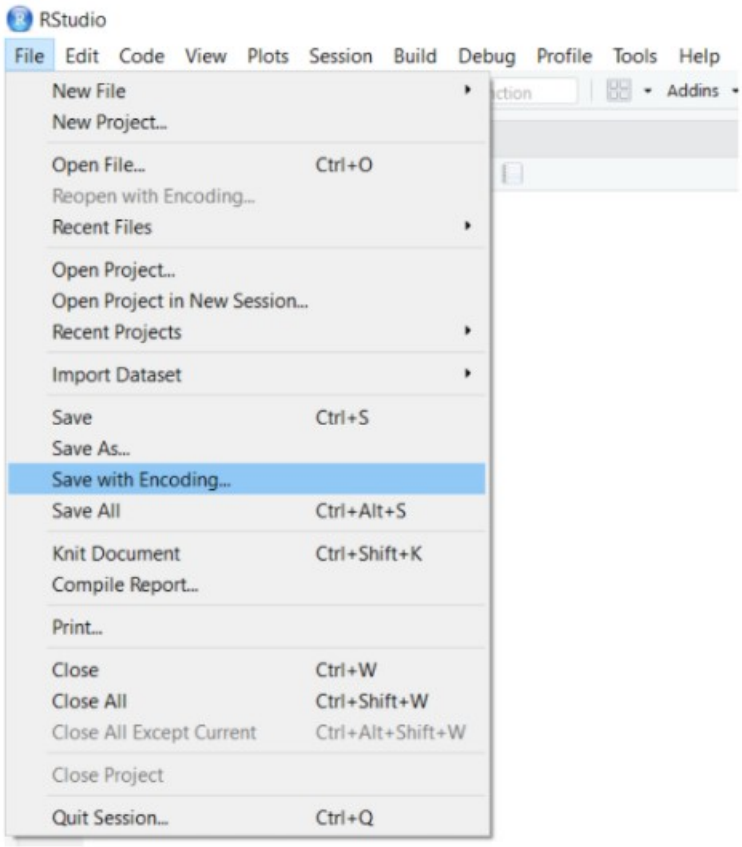


aquí se buscará el directorio deseado, una vez encontrado, se debe presionar el botón `OK` y se guardarán los cambios, de esta forma siempre que se abra *R*, esta será la carpeta de trabajo.

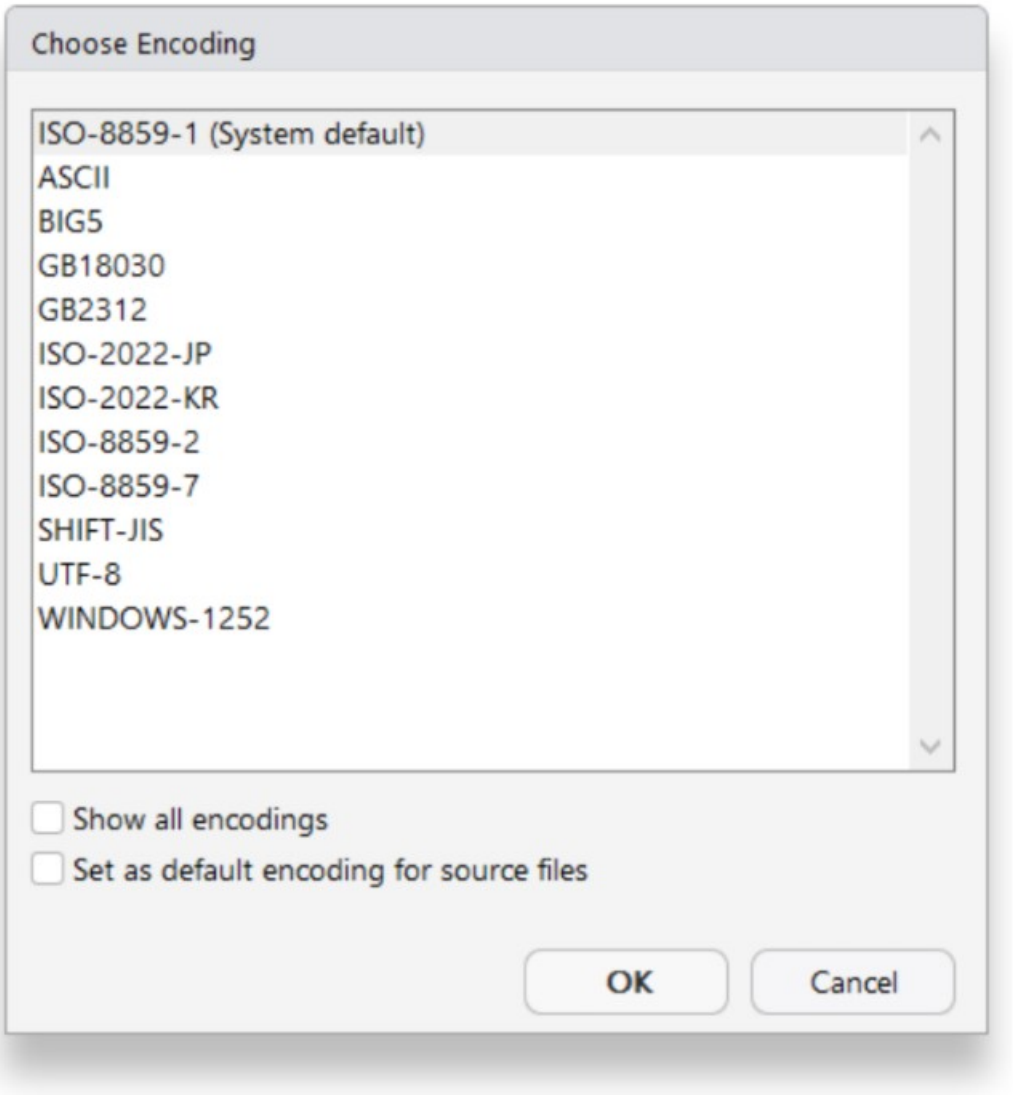


Codificación

Hay varios tipos de codificación, dentro de los más conocidos están **ASCII**, **UTF-8**, **ISO 8859-1**; la codificación universal es **ASCII**, de los más utilizados en la actualidad es **UFT-8** y otra opción para la codificación latina es **ISO 8859-1**, para definir alguna codificación tenemos que ir a la ficha *File* y seleccionar la opción *Save with Encoding*.



una vez seleccionada esta opción aparecerá una ventana con algunas recomendaciones de codificación o se puede ver todas las opciones disponibles, aquí seleccionará la más conveniente, para guardar estos cambios, sólo necesitamos oprimir el botón de `OK`.



Metacomandos / *Shortcuts*

La lista de metacomandos más completa se puede visualizar presionando `CTRL+ALT+K`. Los más importantes se muestran a continuación:

Windows/Linux	Mac	Acción
Ctrl+Shift+H	Cmd+Shift+H	Cambiar directorio de trabajo
Ctrl+Shift+N	Cmd+Shift+N	Nuevo script
Ctrl+Shift+R	Cmd+Shift+R	Inserta una nueva sección (script)
Ctrl+O	Cmd+O	Abrir un documento
Ctrl+S	Cmd+S	Guardar el script actual
Ctrl+Alt+S	Cmd+Alt+S	Guarda todos los scripts abiertos

Windows/Linux	Mac	Acción
Ctrl+L	Cmd+L	Limpia la consola
Ctrl+Enter	Cmd+Enter	Corre una línea del script
Ctrl+Shift+P	Cmd+Shift+P	vuelve a correr lineas anteriores
Ctrl+W	Cmd+W	Cierra el script actual
Ctrl+Shift+W	Cmd+Shift+W	Cierra todos los scripts
ctrl+3	Cmd+3	Abre la ventana de ayuda
Ctrl+4	Cmd+4	Muestra el historial
Ctrl+5	Cmd+5	Abre la ventana del folder de trabajo
Ctrl+6	Cmd+6	Muestra la ventana de gráficas
Ctrl+7	Cmd+7	Muestra los paquetes
Ctrl+8	Cmd+8	Muestra las variables almacenadas
Ctrl+Shift+K	Cmd+Shift+K	Crea un archivo con la paquetería Knit

Primeros Comandos

```
#ayuda/documentación
help()
help.start()
?
#salida
q()
#lista de objetos
ls()
#eliminar objetos especificos
rm()
#guardar variables y el historial
save()
#carga algun historial especifico
load()
```

Comandos de Navegación en el disco duro

```
#dirección actual de trabajo
getwd()
#crea carpetas
dir.create()
#muestra las carpetas en el directorio
```

```
list.files()
#nueva dirección de trabajo
setwd()
#crea un nuevo archivo
file.create()
#elimina archivos
file.remove()
#copia archivos
file.copy()
```

Operaciones Básicas

```
# Sumas
5+5
```

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

```
# Restas
8-3
```

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

```
# Múltiplicaciones
6*5
```

```
## [1] 30
```

```
# Divisiones
7/5
```

```
## [1] 1.4
```

```
# Módulo
8%%2
```

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

```
# Parte entera de la division
6%/%3
```

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

```
# Exponentes
4^5
```

```
## [1] 1024
```

```r
2**3
```

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

```r
# Raíz cudrada
sqrt(144)
```

```
## [1] 12
```

```r
# Signo
sign(1)
```

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

```r
sign(0)
```

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

```r
sign(-1)
```

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

```r
sign(-10)
```

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

```r
sign(10)
```

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

```r
# Valor absoluto
abs(1)
```

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

```r
abs(-1)
```

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

```r
# Función piso
floor(1.4)
```

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

```r
floor(1.9)
```

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

```r
# Función techo
ceiling(1.4)
```

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

```r
ceiling(1.9)
```

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

```r
# Truncar
trunc(1.8564)
```

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

```r
# Redondear
round(1.8564,3)
```

```
## [1] 1.856
```

```r
# Significancia
signif(1.574309463,4)
```

```
## [1] 1.574
```

```r
# Logaritmo natural
log(exp(1))
```

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

```r
# Logaritmo base 10
log10(10)
```

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

```r
# Logaritmo base 2
log2(2)
```

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

```r
# Logaritmo base 1.71828
log1p(1.718282)
```

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

```r
# Logaritmo genérico, cualquier base
log(3,3)
```

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

```r
# Exponencial
exp(2)
```

```
## [1] 7.389056
```

```r
# Factorial
factorial(5)
```

```
## [1] 120
```

```r
# Pi
pi
```

```
## [1] 3.141593
```

```r
# Seno
sin(2*pi)
```

```
## [1] -2.449213e-16
```

```r
sinpi(2)
```

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

```r
# Coseno
cos(2*pi)
```

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

```r
cospi(2)
```

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

```r
# Tangente
tan(2*pi)
```

```
## [1] -2.449294e-16
```

```r
tanpi(2)
```

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

```r
# Arcocoseno
acos(1)
```

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

```r
# Arcoseno
asin(0)
```

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

```r
# Arcotangente
atan(1)
```

```
## [1] 0.7853982
```

```r
# Funciones hiperbólicas
sinh(1)
```

```
## [1] 1.175201
```

```r
cosh(1)
```

```
## [1] 1.543081
```

```r
tanh(1)
```

```
## [1] 0.7615942
```

```r
acosh(2)
```

```
## [1] 1.316958
```

```r
asinh(2)
```

```
## [1] 1.443635
```

```r
atanh(2)
```

```
## [1] NaN
```

```r
atanh(1)
```

```
## [1] Inf
```

```r
atanh(1/2)
```

```
## [1] 0.5493061
```

```r
# Complejos
1i*1i
```

```
## [1] -1+0i
```

```r
1i
```

```
## [1] 0+1i
```

```r
# Parte Real
Re(1+3i)
```

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

```r
Re(2i)
```

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

```r
# Parte Imaginaria
Im(3+1i)
```

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

```r
Im(2)
```

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

```r
# Otras funciones con complejos
Mod(1i)
```

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

```r
Mod(1+1i)
```

```
## [1] 1.414214
```

```r
Mod(7)
```

```
## [1] 7
```

```r
Arg(3i)
```

```
## [1] 1.570796
```

```r
Arg(3)
```

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

```r
Arg(2+7i)
```

```
## [1] 1.292497
```

```r
Conj(1i)
```

```
## [1] 0-1i
```

```r
Conj(4)
```

```
## [1] 4
```

```r
Conj(4+8i)
```

```
## [1] 4-8i
```

Tipos de Variables

Numéricas

```r
#objeto simple numérico
a<-1;a
```

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

```r
b=4;b
```

```
## [1] 4
```

```r
# Todas las operaciones anteriores son objetos numéricos
```

Carácter

```r
#Siempre debe ir entre comillas los archivos de tipo texto
#c1<-Hola #error
z<- "Hola mundo"; z
```

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

```r
a1="Hola"; a1
```

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

```r
b1="a todos"; b1
```

```
## [1] "a todos"
```

```r
# Pega el contenido de dos cadenas de caracter
paste(a1,b1)
```

```
## [1] "Hola a todos"
```

```r
c1="¿Cómo están?"; c1
```

```
## [1] "¿Cómo están?"
```

```r
paste(a1,b1,c1)
```

```
## [1] "Hola a todos ¿Cómo están?"
```

```r
# Cuenta el número de caracteres
nchar(b1)
```

```
## [1] 7
```

```r
# Busca una frase dentro de una cadana de caracteres
grep("todos",b1)
```

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

```r
# Sólo mayusculas
toupper(b1)
```

```
## [1] "A TODOS"
```

```r
# Sólo minusculas
tolower(a1)
```

```
## [1] "hola"
```

```r
# Cambia el contenido de una cadena
sub("todos","nadie",b1)
```

```
## [1] "a nadie"
```

Lógicos

```r
# Objeto lógico
e<-TRUE; e
```

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

```r
f=F; f
```

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

```r
# Validaciónes
## Menor que
6<9
```

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

```r
7<2
```

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

```r
## Mayor
7>5
```

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

```r
## Menor o igual que
5<=7
```

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

```r
5<=5
```

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

```r
5<=1
```

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

```r
## Mayor que
10>1
```

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

```r
## Igual
10==11
```

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

```
## Diferente
2!=3
```

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

Más adelante se ven operaciones con los símbolos `&` (AND) y `|` (OR).

```
# Verificación de tipo de objeto
is.numeric("42")
```

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

```
is.numeric(13)
```

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

```
is.integer(1.78)
```

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

```
is.logical(30<0)
```

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

```
is.logical(F)
```

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

```
is.character(4)
```

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

```
is.character("¿Qué tal?")
```

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

Una observación interesante es que R hace *coherción* a tipo numérico a las variables lógicas, en donde $FALSE \to 0$ y $TRUE \to 1$, por lo que se pueden operar estas variables sin complicaciones.

```
TRUE+15
```

```
## [1] 16
```

```
exp(FALSE)
```

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

```
pi**(-TRUE)
```

```
## [1] 0.3183099
```

Tipos de Objetos

R trabaja con distintos tipos de objetos, estos pueden ser del tipo estructurado o no estructurado. Hablando de los objetos de tipo estructurado trabaja con: vectores, matrices, arreglos y tablas de datos (`data frame`). En el caso de datos no estructurados trabaja principalmente con listas.

Vectores

Los vectores sólo pueden almacenar información de un sólo tipo (numérico, carácter o lógico), en caso de combinar un vector de tipo numérico con lógico, obtendremos un vector de tipo numérico como resultado final, en caso de combinar variables de tipo carácter con lógicos o numéricos, el resultado final será un vector de tipo carácter.

```
# Vector numérico
x<-c(1,2,3,4); x; str(x)
```

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

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

```
# Vector de texto
y<-c("R","S","T","W"); y; str(y)
```

```
## [1] "R" "S" "T" "W"
```

```
##  chr [1:4] "R" "S" "T" "W"
```

```
# Vector lógico
z<-c(T,F,F,T); z; str(z)
```

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

```
##  logi [1:4] TRUE FALSE FALSE TRUE
```

```
# Vector número
u=c(T,F,T,5); u; str(u)
```

```
## [1] 1 0 1 5
```

```
##  num [1:4] 1 0 1 5
```

```r
# Vector carácter
v=c("0",1,2,3,4); v; str(v)
```

```
## [1] "0" "1" "2" "3" "4"
```

```
##  chr [1:5] "0" "1" "2" "3" "4"
```

```r
w=c("T",T,F,F,T); w; str(w)
```

```
## [1] "T"     "TRUE"  "FALSE" "FALSE" "TRUE"
```

```
##  chr [1:5] "T" "TRUE" "FALSE" "FALSE" "TRUE"
```

```r
# Se crean dos vectores
x<-c(1,2,3,11,12,20); x
```

```
## [1]  1  2  3 11 12 20
```

```r
y<-c(4,5,6,10,14,21); y
```

```
## [1]  4  5  6 10 14 21
```

```r
# Combinación de dos vectores
z<-c(x,y); z
```

```
##  [1]  1  2  3 11 12 20  4  5  6 10 14 21
```

```r
# Si quisieramos extraer las entradas 1,3,5

## error
## z(1,3,5)
## error
## z[1,3,5]

# Devuelve los elementos del vector seleccionadas
z[c(1,3,5)]
```

```
## [1]  1  3 12
```

```r
# Datos condicionados de un vector
z[z>10]
```

```
## [1] 11 12 20 14 21
```

```r
z[z<15]
```

```
##  [1]  1  2  3 11 12  4  5  6 10 14
```

```r
## Error no acepta intervalos declarados de esta manera
## z[18>z>1]

# Pero si de esta forma
z[z>1&z<18]
```

```
## [1]  2  3 11 12  4  5  6 10 14
```

```r
z[z>1|z<18]
```

```
##  [1]  1  2  3 11 12 20  4  5  6 10 14 21
```

```r
# Actualiza el vector completo
z=z/6; z
```

```
##  [1] 0.1666667 0.3333333 0.5000000 1.8333333 2.0000000 3.3333333 0.6666667
##  [8] 0.8333333 1.0000000 1.6666667 2.3333333 3.5000000
```

```r
# Regresa las posiciones en x/y presentes en z
z[x]
```

```
## [1] 0.1666667 0.3333333 0.5000000 2.3333333 3.5000000        NA
```

```r
z[y]
```

```
## [1] 1.833333 2.000000 3.333333 1.666667       NA       NA
```

Ejemplos de operaciones

```r
# Definamos un vector
g<-c(1,2,3,4); g
```

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

```r
# Total
sum(g)
```

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

```r
# Prodcuto
prod(g)
```

```
## [1] 24
```

```r
# Gasto semanal
gs<-c(12,45,0,9,6,25,30); gs
```

```
## [1] 12 45  0  9  6 25 30
```

```r
# Etiquetas
names(gs)<-c("Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"); gs
```

```
##     Lunes    Martes Miercoles    Jueves   Viernes    Sabado   Domingo
##        12        45         0         9         6        25        30
```

```r
gs[4]
```

```
## Jueves
##      9
```

```r
gs[gs==max(gs)]
```

```
## Martes
##     45
```

```r
names(gs[gs==max(gs)])
```

```
## [1] "Martes"
```

```r
max(gs)
```

```
## [1] 45
```

```r
# Función secuencia
seq(from=7, to=38, by=3)
```

```
##  [1]  7 10 13 16 19 22 25 28 31 34 37
```

```r
c=seq(from=12, to=50); c
```

```
##  [1] 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
## [26] 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

```r
seq(from=100, to=10, by=-10)
```

```
##  [1] 100  90  80  70  60  50  40  30  20  10
```

```r
d=seq(from=1, to=10, by=0.1); d
```

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

```
e=seq(from=1, to=10, length.out = 22); e
```

```
##  [1]  1.000000  1.428571  1.857143  2.285714  2.714286  3.142857  3.571429
##  [8]  4.000000  4.428571  4.857143  5.285714  5.714286  6.142857  6.571429
## [15]  7.000000  7.428571  7.857143  8.285714  8.714286  9.142857  9.571429
## [22] 10.000000
```

```
# Función repetir
rep(1,5)
```

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

```
rep(6,9)
```

```
## [1] 6 6 6 6 6 6 6 6 6
```

```
x<-c(1,2,3);x
```

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

```
rep(x,2)
```

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

```
rep(c(2,6,9),9)
```

```
##  [1] 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9
```

```
rep(1:5,2)
```

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

```
rep(c(1,2,3),c(1,4,8))
```

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

```
rep(1:4,c(1,2,3,4))
```

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

```r
# Remplazar o actualizar uno o varios elementos
x
```

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

```r
x[1]=0; x
```

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

```r
x[c(1,3)]=9;x
```

```
## [1] 9 2 9
```

```r
# Valores faltantes
f=c(1,5,9,NA,5,NA,0); f
```

```
## [1]  1  5  9 NA  5 NA  0
```

```r
# Detecta los elementos vacios o faltantes del vector
sum(is.na(f))
```

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

```r
# Elimina los elementos vacios del vector
f<-f[!is.na(f)]; f
```

```
## [1] 1 5 9 5 0
```

```r
# Función de longitud de un vector
length(f)
```

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

```r
# Minimo
min(f)
```

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

```r
# Maximo
max(f)
```

```
## [1] 9
```

```r
# Media o promedio
mean(f)
```

```
## [1] 4
```

```r
# Crea un vector de tipo caracter
v=c("a","d","g","h","a","x","V","a","d","g","h","a","x","V"); v
```

```
##  [1] "a" "d" "g" "h" "a" "x" "V" "a" "d" "g" "h" "a" "x" "V"
```

```r
# Nos dice los elementos diferentes que tenemos y cuantas veces se repiten
table(v)
```

```
## v
## a d g h V x
## 4 2 2 2 2 2
```

```r
# Calcula el tamaño del vector
length(v)
```

```
## [1] 14
```

```r
# Creamos un vector de tipo categórico para hacer analisis de sus elementos
v1=as.factor(v); v1
```

```
##  [1] a d g h a x V a d g h a x V
## Levels: a d g h V x
```

```r
# Se extraen los elementos del vector sin repetir
levels(v1)
```

```
## [1] "a" "d" "g" "h" "V" "x"
```

```r
# Me dice cuantos elementos hay de cada elemento del vector
summary(v1)
```

```
## a d g h V x
## 4 2 2 2 2 2
```

```r
# Generamos un vector numerico
estaruras<-c(1.7,1.5,1.9,2,1.45,1.7,1.5,1.9,2,1.45,1.68,1.6,1.45,1.72); estaruras
```

```
##  [1] 1.70 1.50 1.90 2.00 1.45 1.70 1.50 1.90 2.00 1.45 1.68 1.60 1.45 1.72
```

```r
length(estaruras)
```

```
## [1] 14
```

```r
# Hace un analisis por clase, en este caso calcula la media
tapply(estaruras,v1,mean)
```

```
##      a      d      g      h      V      x
## 1.6625 1.7500 1.6750 1.8400 1.6100 1.5750
```

```r
# Ordenar vectores numericos
## - Ordena en orden ascendente
estaturas=sort(estaruras); estaturas
```

```
##  [1] 1.45 1.45 1.45 1.50 1.50 1.60 1.68 1.70 1.70 1.72 1.90 1.90 2.00 2.00
```

```r
## - Ordena en orden decreciente
e1=sort(estaruras,decreasing = TRUE); e1
```

```
##  [1] 2.00 2.00 1.90 1.90 1.72 1.70 1.70 1.68 1.60 1.50 1.50 1.45 1.45 1.45
```

```r
## - Ordenar vector tipo caracter
calidad=c("media","baja","media","alta","media","baja","alta","baja"); calidad
```

```
## [1] "media" "baja"  "media" "alta"  "media" "baja"  "alta"  "baja"
```

```r
# Le decimos que baja es peor calidad, media es el intermedio y alta es la mejor calidad
calidad1=ordered(calidad,c("baja","media","alta")); calidad1
```

```
## [1] media baja  media alta  media baja  alta  baja
## Levels: baja < media < alta
```

```r
## - Ordena de manera ascendente
sort(calidad1)
```

```
## [1] baja  baja  baja  media media media alta  alta
## Levels: baja < media < alta
```

```r
## - Ordena de manera descendente
sort(calidad1,decreasing=TRUE)
```

```
## [1] alta  alta  media media media baja  baja  baja
## Levels: baja < media < alta
```

```r
# Creamos dos vectores
y <- c("Aguascalientes", "Baja California", "Baja California", "Chihuahua",
"Zacatecas", "Zacatecas", "Zacatecas", "Baja California", "Chihuahua")
z <- c("H", "M", "M", "M", "H", "M", "M", "M", "M")
cbind(EDO = y, SEXO = z) # forma una base de datos a partir de 2 vectores
```

```
##       EDO               SEXO
##  [1,] "Aguascalientes"  "H"
##  [2,] "Baja California" "M"
##  [3,] "Baja California" "M"
##  [4,] "Chihuahua"       "M"
##  [5,] "Zacatecas"       "H"
##  [6,] "Zacatecas"       "M"
##  [7,] "Zacatecas"       "M"
##  [8,] "Baja California" "M"
##  [9,] "Chihuahua"       "M"
```

```r
table(y) # Tabla de frecuencias
```

```
## y
##  Aguascalientes Baja California       Chihuahua       Zacatecas
##               1               3               2               3
```

```r
table(z)
```

```
## z
## H M
## 2 7
```

```r
table(z, y) # Tabla de contingencia
```

```
##    y
## z   Aguascalientes Baja California Chihuahua Zacatecas
##   H              1               0         0         1
##   M              0               3         2         2
```

```r
# Remplazamos uno o algunos elementos del vector
replace(letters, c(1, 5, 9, 15, 21), c("A", "E", "I", "O", "U"))
```

```
##  [1] "A" "b" "c" "d" "E" "f" "g" "h" "I" "j" "k" "l" "m" "n" "O" "p" "q" "r" "s"
## [20] "t" "U" "v" "w" "x" "y" "z"
```

```r
# Creamos un vector numérico
x <- c(4, 1.5, 6, 4, 10, 20, 1, 15, 0); x
```

```
## [1]  4.0  1.5  6.0  4.0 10.0 20.0  1.0 15.0  0.0
```

```r
# Extraemos la longitud de un vector
n <- length(x); n
```

```
## [1] 9
```

```r
#Creamos un vector tipo lógico
id <- x > 6; id; id*1
```

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

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

```r
sum(id)
```

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

```r
(sum(id)/n)*100
```

```
## [1] 33.33333
```

```r
(1:n)[id]
```

```
## [1] 5 6 8
```

```r
x[id]
```

```
## [1] 10 20 15
```

```r
x[(1:n)[id]]
```

```
## [1] 10 20 15
```

```r
indices<-which(id) #visualiza los indices en los que se cumple id
indices
```

```
## [1] 5 6 8
```

```r
id2 <- y == "Baja California"; id2
```

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

```r
y[id2]
```

```
## [1] "Baja California" "Baja California" "Baja California"
```

```r
z[id2]
```

```
## [1] "M" "M" "M"
```

Matrices

Al igual que los vectores, este tipo de objetos sólo puede ser de un sólo tipo de variable, una matriz no puede almacenar en una columna datos de tipo numérico y en otra de tipo carácter, esta es una limitante que debemos tener presentes.

```r
# Matriz vertical (Mnx1)
M1=matrix(1:6); M1
```

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

```r
# Matriz General (Mnxm)
M2=matrix(1:6,nrow=2); M2
```

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

```r
M3=matrix(1:6,nrow=2,byrow=TRUE); M3
```

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

```r
M=matrix(1:6,nrow=3); M
```

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

```r
# Número de datos en una matriz
length(M)
```

```
## [1] 6
```

```r
# Tipo de matriz
mode(M); str(M)
```

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

```
##  int [1:3, 1:2] 1 2 3 4 5 6
```

```r
# Dimensión
dim(M)
```

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

```r
# Buscar el nombre o etiqueta de nuestras columnas y renglones
M4=M; M4
```

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

```r
rownames(M)<-c("Hugo","Paco","Luis"); M
```

```
##      [,1] [,2]
## Hugo    1    4
## Paco    2    5
## Luis    3    6
```

```
colnames(M)<-c("Edad","Estatura"); M
```

```
##      Edad Estatura
## Hugo    1        4
## Paco    2        5
## Luis    3        6
```

```
dimnames(M4)<-list(c("Hugo","Paco","Luis"),c("Edad","Estatura")); M4
```

```
##      Edad Estatura
## Hugo    1        4
## Paco    2        5
## Luis    3        6
```

```
y1=1:24;y1
```

```
##  [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
```

```
x1=matrix(1:24,nrow=1,byrow=T); x1
```

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

```
y1[1]
```

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

```
x1[1,1]
```

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

```
# Para saber si tenemos una matriz
is.matrix(y1)
```

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

```
is.matrix(x1)
```

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

```
as.matrix(y1,byrow=T)
```

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

```
# Extracción de datos
M2[1,1]
```

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

```
M3[1,3]
```

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

```
## Error es necesario saber la dimensión de nuestra matriz
## M1[1,2]

# Extracción por columna
M2[,2]
```

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

```
# Extracción de renglones
M2[1,]
```

```
## [1] 1 3 5
```

```
M3[]
```

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

```
# Concatenar por columnas
cbind(M3,c(9,8))
```

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

```
# Concatenar por renglón
M5=rbind(M,c(11,10)); M5
```

```
##      Edad Estatura
## Hugo    1        4
## Paco    2        5
## Luis    3        6
##        11       10
```

```
rownames(M5)[4]="Pedro"; M5
```

```
##       Edad Estatura
## Hugo     1        4
## Paco     2        5
## Luis     3        6
## Pedro   11       10
```

Ejemplos de operaciones

```
# Transponer
t(M1)
```

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

```
# Producto matricial
MC<-M1%*%t(M1); MC
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    2    3    4    5    6
## [2,]    2    4    6    8   10   12
## [3,]    3    6    9   12   15   18
## [4,]    4    8   12   16   20   24
## [5,]    5   10   15   20   25   30
## [6,]    6   12   18   24   30   36
```

```r
# Inversa
solve(matrix(1:4,nrow=2))
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

```r
# Solución de un sistema de ecuaciones
solve(matrix(1:4,nrow=2),c(1,-1))
```

```
## [1] -3.5  1.5
```

```r
# Matriz diagonal
diag(1:4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    2    0    0
## [3,]    0    0    3    0
## [4,]    0    0    0    4
```

```r
# Determinante
det(diag(1:4))
```

```
## [1] 24
```

```r
datos<-matrix(c(20,65,174,22,70,180,19,68,170),nrow = 3,byrow=T)
datos
```

```
##      [,1] [,2] [,3]
## [1,]   20   65  174
## [2,]   22   70  180
## [3,]   19   68  170
```

```r
dimnames(datos)<-list(c("hugo","paco","luis"),c("edad","peso","estatura"))
datos
```

```
##      edad peso estatura
## hugo   20   65      174
## paco   22   70      180
## luis   19   68      170
```

```r
# Regresa todos los elementos de el renglón (obs) dado por nombre de sujeto
datos["hugo",]
```

```
##     edad     peso estatura
##       20       65      174
```

```r
# Por número de renglón
datos[1,]
```

```
##     edad     peso estatura
##       20       65      174
```

```r
datos["paco",]
```

```
##     edad     peso estatura
##       22       70      180
```

```r
datos["luis",]
```

```
##     edad     peso estatura
##       19       68      170
```

```r
# Regresa todos los elementos de la columna (var) dada por nombre de columna
datos[,"edad"]
```

```
## hugo paco luis
##   20   22   19
```

```r
# Por numero de columna
datos[,1]
```

```
## hugo paco luis
##   20   22   19
```

```r
datos[,"peso"]
```

```
## hugo paco luis
##   65   70   68
```

```r
datos[,"estatura"]
```

```
## hugo paco luis
##  174  180  170
```

```r
# Extraccion por coordenadas
datos[1,1]
```

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

```r
datos["hugo","edad"]
```

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

```r
datos[2,2]
```

```
## [1] 70
```

```r
datos["paco","peso"]
```

```
## [1] 70
```

```r
datos[3,3]
```

```
## [1] 170
```

```r
datos["luis","estatura"]
```

```
## [1] 170
```

```r
datos["luis",2]
```

```
## [1] 68
```

```r
datos[2,"estatura"]
```

```
## [1] 180
```

```r
# Extracción de más de una variable u observación
datos[,c("edad","estatura")]
```

```
##      edad estatura
## hugo   20      174
## paco   22      180
## luis   19      170
```

```r
datos[c("hugo","luis"),]
```

```
##      edad peso estatura
## hugo   20   65      174
## luis   19   68      170
```

```r
# Extracción de los nombres de reglones y columnas.
dimnames(datos)
```

```
## [[1]]
## [1] "hugo" "paco" "luis"
##
## [[2]]
## [1] "edad"     "peso"     "estatura"
```

```
colnames(datos)
```

```
## [1] "edad"     "peso"     "estatura"
```

```
rownames(datos)
```

```
## [1] "hugo" "paco" "luis"
```

```
# Cálculo de media por variable
apply(datos,2,mean)
```

```
##      edad      peso  estatura
##  20.33333  67.66667 174.66667
```

```
apply(datos,1,mean)
```

```
##     hugo     paco     luis
## 86.33333 90.66667 85.66667
```

Arreglos

```
# Las matrices y los arreglos son muy similares
array(1:6, c(2, 3))
```

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

```
# ¡También se ven como una generalización de matrices!
# array(datos,C(# de renglomes,# de columnas,# de matrices))
array(1:12,c(2,3,2))
```

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

```
array(1:12,c(4,3,1))
```

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

```
array(1:12,c(2,2,3))
```

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

```
# Matrices simultáneas
dat<-array(c(45,46,65,55,170,167,48,49,68,56,169,165),c(2,3,2)); dat
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]   45   65  170
## [2,]   46   55  167
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]   48   68  169
## [2,]   49   56  165
```

```
dimnames(dat)<-list(SEXO=c("hombres","mujeres"),VARIABLES=c("edad","peso","estatura"),
ESTADO=c("CDMX","EdoMex"))
dat
```

```
## , , ESTADO = CDMX
##
##          VARIABLES
## SEXO      edad peso estatura
##   hombres   45   65      170
```

```
##   mujeres   46   55      167
##
## , , ESTADO = EdoMex
##
##          VARIABLES
## SEXO      edad peso estatura
##   hombres   48   68      169
##   mujeres   49   56      165
```

```
# Extracción de etiquetas
dimnames(dat)
```

```
## $SEXO
## [1] "hombres" "mujeres"
##
## $VARIABLES
## [1] "edad"     "peso"     "estatura"
##
## $ESTADO
## [1] "CDMX"   "EdoMex"
```

```
# Extracción de matrices
dat[,,"EdoMex"]
```

```
##          VARIABLES
## SEXO      edad peso estatura
##   hombres   48   68      169
##   mujeres   49   56      165
```

```
dat[,,"CDMX"]
```

```
##          VARIABLES
## SEXO      edad peso estatura
##   hombres   45   65      170
##   mujeres   46   55      167
```

```
# Extracción de columnas y renglones
dat["hombres",,]
```

```
##           ESTADO
## VARIABLES  CDMX EdoMex
##   edad       45     48
##   peso       65     68
##   estatura  170    169
```

```
dat[,"peso",]
```

```
##          ESTADO
## SEXO      CDMX EdoMex
##   hombres   65     68
##   mujeres   55     56
```

```r
# Extracción de más de una variable u observación
dat[,c("edad","estatura"),]
```

```
## , , ESTADO = CDMX
##
##          VARIABLES
## SEXO      edad estatura
##   hombres   45      170
##   mujeres   46      167
##
## , , ESTADO = EdoMex
##
##          VARIABLES
## SEXO      edad estatura
##   hombres   48      169
##   mujeres   49      165
```

Ejemplos de operaciones

```r
# Cálculo de mínimo por sujeto
apply(dat,1,min)
```

```
## hombres mujeres
##      45      46
```

```r
# Cálculo de media por variable
apply(dat,2,mean)
```

```
##     edad     peso estatura
##    47.00    61.00   167.75
```

```r
# Cálculo de la máximo por matriz
apply(dat,3,max)
```

```
##   CDMX EdoMex
##    170    169
```

Listas

```r
# Creación de una lista
familia<-list(padre="Juan",madre="María",edad_padres=c(30,29),num_hijos=3,nom_hijos=c("Axel","Damian","
familia
```

```
## $padre
## [1] "Juan"
##
## $madre
```

```
## [1] "María"
##
## $edad_padres
## [1] 30 29
##
## $num_hijos
## [1] 3
##
## $nom_hijos
## [1] "Axel"   "Damian" "Tania"
##
## $edad_hijos
## [1] 7 5 3
##
## $ciudad
## [1] "Madrid"
```

```
# Regresa el nombre de las "variables"
names(familia)
```

```
## [1] "padre"       "madre"       "edad_padres" "num_hijos"   "nom_hijos"
## [6] "edad_hijos"  "ciudad"
```

```
# Extracción de datos
familia$padre
```

```
## [1] "Juan"
```

```
## Error, pues en listas no acepta extraccion simultanea
## familia[[c(1,3)]]

# Ésta es la forma correcta de haces extracciones de más de un elemento
familia[c(1,3)]
```

```
## $padre
## [1] "Juan"
##
## $edad_padres
## [1] 30 29
```

```
familia[c(4,5,6)]
```

```
## $num_hijos
## [1] 3
##
## $nom_hijos
## [1] "Axel"   "Damian" "Tania"
##
## $edad_hijos
## [1] 7 5 3
```

```
# Otra forma de hacer extraciones en listas es
familia$nom_hijos
```

```
## [1] "Axel"   "Damian" "Tania"
```

```
familia$padre
```

```
## [1] "Juan"
```

```
familia$edad_padres[1]
```

```
## [1] 30
```

Tablas de datos (`data.frame`)

Internas

Este tipo de tablas de datos ya están pre-fabricadas en *R*. Así como el objeto `pi` estas ya existen sin necesidad de definirlas.

```
# data.frame
data("iris")
a<-iris
# Visualizamos los primeros 6
head(a)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# Verificamos que sea un objeto de base de datos
class(a)
```

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

```
# Fijamos la base de datos
attach(a)
Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
##  [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
##  [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
##  [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

```r
# Desfijamos las variables
detach(a)
a$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
##  [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
##  [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
##  [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

```r
# La extracción de informacion es igual que una matriz
a[1,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```r
a[,1]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
##  [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
##  [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
##  [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

```r
a[2,3]
```

```
## [1] 1.4
```

```r
a[4,"Sepal.Width"]
```

```
## [1] 3.1
```

```r
a$Sepal.Length[1]
```

```
## [1] 5.1
```

Creadas

Se pueden crear objetos tipo `data.frame` y manipular como los que ya existen dentro de *R*.

```r
# Creamos dos vectores
y <- c("Aguascalientes", "Baja California", "Baja California", "Chihuahua","Zacatecas", "Zacatecas", "Za
z <- c("H", "M", "M", "M", "H", "M", "M", "M", "M")
x <- c(1, 2, 6, 4, 10, 20, 1, 15, 0)
BD <- data.frame(EDO = y, sexo = z, IDX = x)
names(BD)
```

```
## [1] "EDO"  "sexo" "IDX"
```

```r
# Para manejar las variables de una base de datos i.e. columnas
table(BD$EDO)
```

```
##
##  Aguascalientes Baja California       Chihuahua       Zacatecas
##               1               3               2               3
```

```r
BD$EDO
```

```
## [1] "Aguascalientes"  "Baja California" "Baja California" "Chihuahua"
## [5] "Zacatecas"       "Zacatecas"       "Zacatecas"       "Baja California"
## [9] "Chihuahua"
```

```r
BD$sexo
```

```
## [1] "H" "M" "M" "M" "H" "M" "M" "M" "M"
```

```r
BD$IDX
```

```
## [1]  1  2  6  4 10 20  1 15  0
```

```r
# De esta manera ya podemos manejar las columnas como vectores!
table(BD$EDO)
```

```
##
##  Aguascalientes Baja California       Chihuahua       Zacatecas
##               1               3               2               3
```

```r
table(BD$sexo, BD$EDO)
```

```
##
##    Aguascalientes Baja California Chihuahua Zacatecas
##   H              1               0         0         1
##   M              0               3         2         2
```

```r
mean(BD$IDX)
```

```
## [1] 6.555556
```

```r
sd(BD$IDX)
```

```
## [1] 7.037597
```

```r
summary(BD)
```

```
##      EDO                sexo                IDX        
##  Length:9           Length:9           Min.   : 0.000  
##  Class :character   Class :character   1st Qu.: 1.000  
##  Mode  :character   Mode  :character   Median : 4.000  
##                                        Mean   : 6.556  
##                                        3rd Qu.:10.000  
##                                        Max.   :20.000
```

```r
BD$sexo
```

```
## [1] "H" "M" "M" "M" "H" "M" "M" "M" "M"
```

```r
# Para extraer informacion de una base de datos
BD[1,]
```

```
##              EDO sexo IDX
## 1 Aguascalientes    H   1
```

```r
BD[,1] # equivalente a BD$EDO
```

```
## [1] "Aguascalientes"  "Baja California" "Baja California" "Chihuahua"
## [5] "Zacatecas"       "Zacatecas"       "Zacatecas"       "Baja California"
## [9] "Chihuahua"
```

```r
BD[5,3]
```

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

```r
BD[5:7,]
```

```
##         EDO sexo IDX
## 5 Zacatecas    H  10
## 6 Zacatecas    M  20
## 7 Zacatecas    M   1
```

```r
# Filtrando informacion
id <- BD$EDO == "Zacatecas"
BD_ZAC <- BD[id,]
mean(BD_ZAC$IDX)
```

```
## [1] 10.33333
```

```
tapply(BD$IDX, BD$EDO, median)
```

```
##  Aguascalientes Baja California       Chihuahua       Zacatecas
##               1               6               2              10
```

```
id2 <- BD$sexo == "H"
BD_H <- BD[id2,]
id3 <- BD$IDX <= 5
BD_ID_5 <- BD[id3,] ; BD_ID_5
```

```
##               EDO sexo IDX
## 1  Aguascalientes    H   1
## 2 Baja California    M   2
## 4       Chihuahua    M   4
## 7       Zacatecas    M   1
## 9       Chihuahua    M   0
```

```
tb <- table(BD_ID_5$EDO, BD_ID_5$sexo) ; tb
```

```
##
##                   H M
##   Aguascalientes  1 0
##   Baja California 0 1
##   Chihuahua       0 2
##   Zacatecas       0 1
```

Paqueterías

Las paqueterías son de gran utilidad porque están conformadas por una serie de funciones que nos facilitarán la obtención de resultados de manera rápida y efectiva, además si llegásemos a tener dudas sobre su creación, podemos revisar su documentación desde la sección de ayuda.

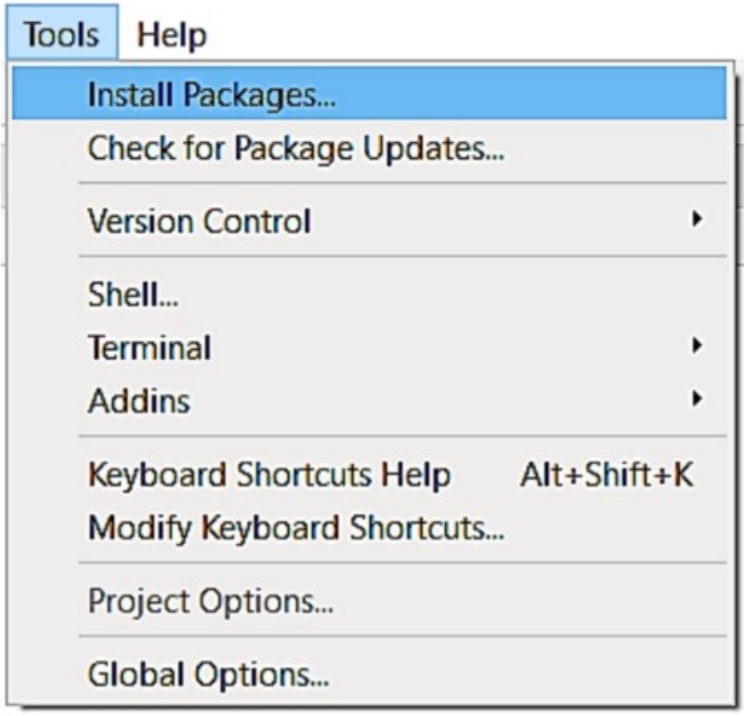
Algunas paqueterías recomendadas para el curso y su forma de instalación en R son:

```
install.packages("sqldf") #Carga base de datos y se realizan consultas
install.packages("sqlite") #Carga base de datos y se realizan consultas
install.packages("RODBC") #Carga base de datos y se realizan consultas
install.packages("RPostgresSQL") #Carga base de datos y se realizan consultas
install.packages("foreing") #Carga archivos de SPSS, SAS, Stata, DBF, Epi info, Minitab
install.packages("plyr") #Extracción de datos y aplicación de funciones a grupos
install.packages("dplyr") #Trabaja con fechas
install.packages("reshape2") #Transformación de datos
install.packages("ggplot2") #Genera graficos
install.packages("rgl") #Gráficos en 3D
install.packages("forecats") #Formateo de datos y creación modelos
install.packages("knitr") #Genera codigos en Latex y Html
install.packages("xtable") #Exporta datos en html y Latex
install.packages("actuar") #Varias distribuciones
install.packages("MASS") #Análisis multivariado
install.packages("alr4") #Conjunto de datos del libro Applied Linear Regression
```

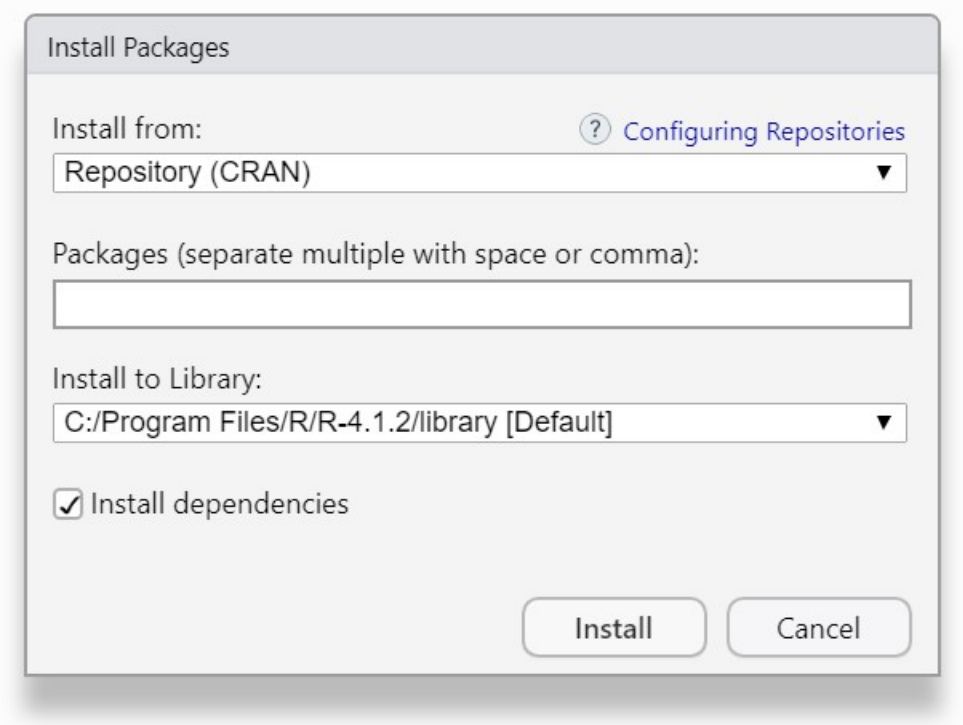
```
install.packages("lmtest") #Pruebas de validación de supuestos

# Esta última paquetería necesita tener Java instalado en el equipo
install.packages("xlsx") #Lectura de archivos de Excel
```

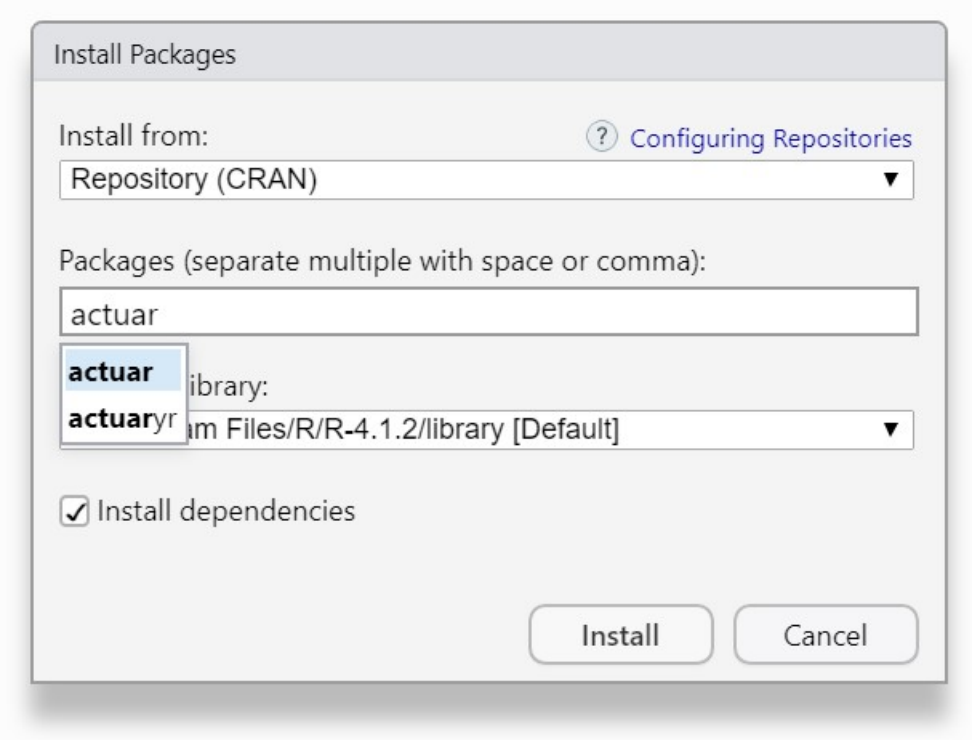
Otro método para instalas una paquetería es ir a la pestaña *Tools*, desplegarla y seleccionar la primera opción: *Install packages*



Eso desplegará la siguiente ventana



Sólo se tiene que colocar el nombre del paquete en la sección *Packages*

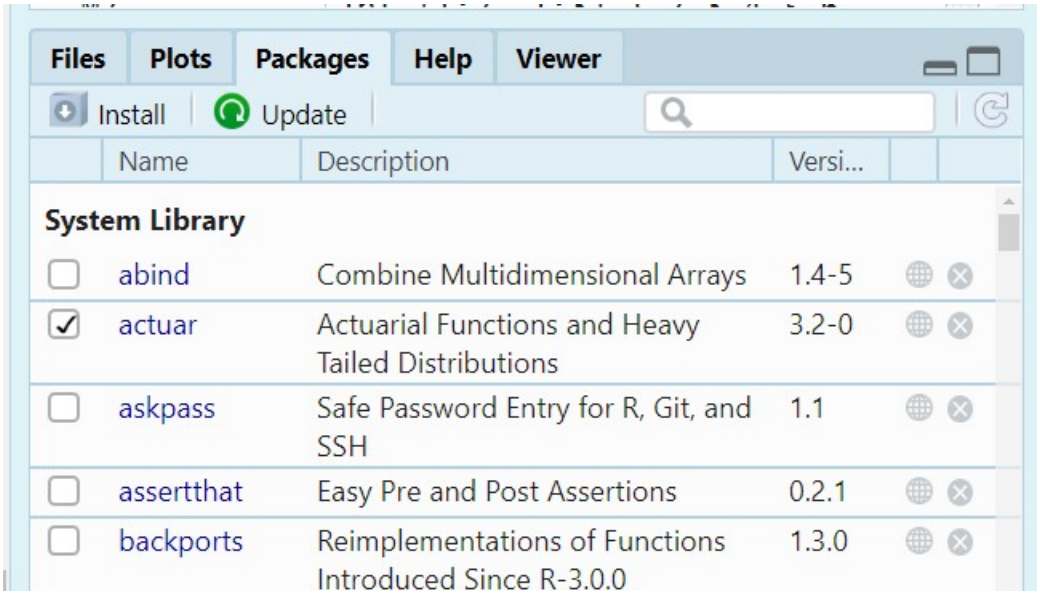


Como se puede observar se autocompleta el nombre de la paquetería, así que una vez seleccionado, sólo se debe presionar el botón *Install* y listo, se instalará el paquete.

La forma de activar las funciones y bases de datos asociadas a cada paquetería es:

```
library(actuar) #Con este comando se activan todas las funciones y bases
# ¡No basta con instalar la paquetería, se tiene que habilitar!
```

Además verificamos que la paquetería se ha activado en el cuarto panel



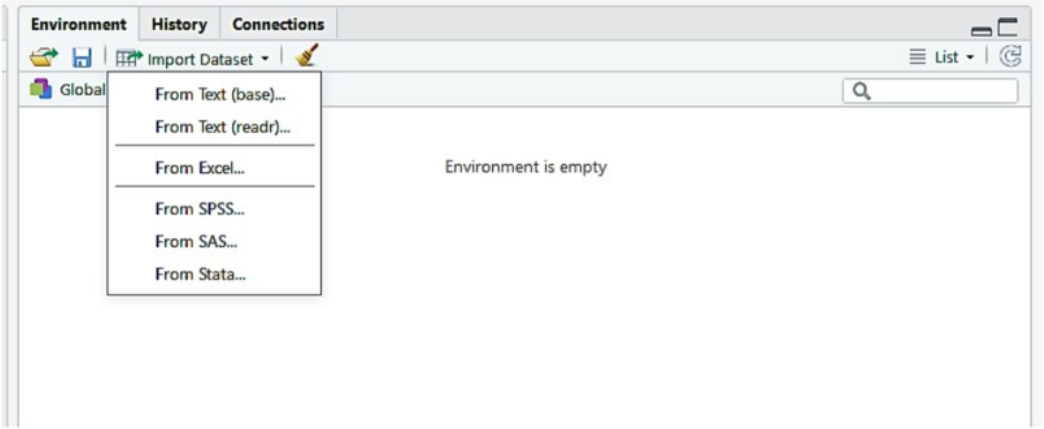
Importando datos

R soporta un gran cantidad de datos, tanto estructurados como no estructurados, puede leer bases guardadas del equipo y datos cargados desde un servidor de internet, a continuación motratemos la forma de llamar las bases de datos, para posteriormente trabajar con ellos.

```
#CSV
library(foreign)
datos1<-read.csv("ejemplo1.csv",header=TRUE)
datos11<-read.table("ejemplo1.csv",sep=",",header=TRUE)
#SPSS
datos2<-read.spss("ejrl.sav",to.data.frame=TRUE)
library(haven)
datos21<- read_sav("ejrl.sav")
#txt
datos3<-read.table("Basetarea.txt",sep="\t",header=TRUE)
#Excel
library(xlsx)
datos4<-read.xlsx("base.xlsx")
library(readxl)
datos41<- read_excel("base.xlsx")
datos42<-read_xlsx("base.xlsx")
#DBF
read.dbf("file")
#Stata
read.dta("file")
read_dta("file")
```

```r
#Minitab
read.mtp("file")
#SAS
read.ssd("file")
#JSON
library("rjson")
fromJSON("data1.json")
```

Otra forma de cargar una base de datos es colocarse en el tercer panel, y seleccionar la opcón *Import Dataset*



Aquí basta con elegir el tipo de archivo que tengamos:

- From Text (base)... (Aquí se subirá un archivo de extensión `.txt`)
- From Text (readr)... (Aquí se cargará un archivo de extensión `.csv`)
- From Excel... (Aquí se importará un archivo de extensión `.xlsx` o `.xls`)
- From SPSS... (Aquí se cargará un archivo de extensión `.sav`)
- From SAS... (Aquí se cargará un archivo de extensión `.sas7bdat` o `.sd7`)
- From Stata... (Aquí se cargará un archivo de extensión `.dta`)

Dos observaciones importantes:

1. Las opciones aquí mostradas dependerán del sistema operativo y la versión de *R*.
2. Son limitadaslas opciones para cargar los datos, por lo cual se recomienda hacer uso de las funciones *read*.

Una vez cargada la base de datos, la forma de oprar será igual al manejo de datos con matrices, pero la gran ventaja de trabajar con esta estructura de datos, es que cada variable puede ser de un tipo diferente, es decir, podemos tener en la base de datos una variable de tipo caracter como Nombre y otra variable de tipo numérica como Calificación, pero cada variable sí tiene que ser de un sólo tipo.

Estadística descriptiva

En la introducción mencionamos el gran potencial de *R* a nivel estadístico, evidentemente la parte descriptiva tiene varias funciones que nos permitirán analizar su comportamiento, de forma numérica y gráfica.

Númericamente

```
#Ejemplo. Consideremos la siguiente muestra que representa las
#calificaciones de alumnos en el primer parcial de Estadística II
x=c(0,1,2,2,3,3,3,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7, 8,8,8,8,9,9,
# Con estos datos haremos un análisis descriptivo
```

Medidas de Tendencia Central

```
# Media, Mediana
mean(x); median(x); quantile(x)[3]
```

```
## [1] 5.48
```

```
## [1] 6
```

```
## 50%
##   6
```

Podemos ver que estas dos medidas son muy similares, la media es un promedio de todos los valores presentes en muestra, al considerar todos los valores esta medida se vuelve muy sensible a datos extremos, además el resultado de operación no necesariamente tomará algún valor en muestra; en cambio la mediana siempre tomará algún valor en muestra y no es sencible a valores extremos, la medida es estable.

```
# Moda
# Esta medida se puede obtener de dos maneras, la primera sería manualmente, es decir
# extraer la observación con más repeticiones
moda=c(table(x))
moda[which(moda == max(moda))[1]]
```

```
##  6
## 12
```

Este método es eficiente si y sólo si hay un valor único, R ya tiene una función que calcula la moda para valores discretos, esta se encuentra en la paquetería `modeest`:

```
# Cargamos la paquetería
library(modeest)
# Aplicamos la función
mfv(x)
```

```
## [1] 6
```

Esta función es eficiente y devuelve múltiples modas en caso de que exista más de una.

```
# Percentiles
quantile(x,0.1); quantile(x,0.75); quantile(x,0.99)
```

```
## 10%
##   3

## 75%
##   7

##  99%
## 9.51
```

R genera de forma automática un resumen descriptivo con la función `summary`

```
# Resumen: mínimo, máximo, cuartiles y media
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    4.25    6.00    5.48    7.00   10.00
```

Meidas de Dispersión

```
# Varianza y Desviacion estandar
var(x); s=sd(x)
```

```
## [1] 4.050612
```

```
# Rango
range(x); R=max(x)-min(x); R
```

```
## [1]  0 10
```

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

```
# Rango intercuartílico
RIC<-quantile(x,0.75)-quantile(x,0.25); RIC
```

```
##  75%
## 2.75
```

```
# Coeficiente de variación
cv=s/mean(x)*100; cv
```

```
## [1] 36.72652
```

Medidas de forma

```
# Coeficiente de asimetria
ca=sum((x-mean(x))^3)/(length(x)*sd(x)^3); ca
```

```
## [1] -0.3320303
```

```r
# Con R necesitamos una paquetería extra
library(moments)
skewness(x)
```

```
## [1] -0.3422462
```

Como el coeficiente de asimetria es negativo, se dice el sesgo de la distribución esta a la izquierda de la mediana y hay una "más" acumulación a la derecha de la mediana.

```r
# Coeficiente de curtosis
curtosis=(sum((x-mean(x))^4)/(length(x)*sd(x)^4)); curtosis
```

```
## [1] 3.248305
```

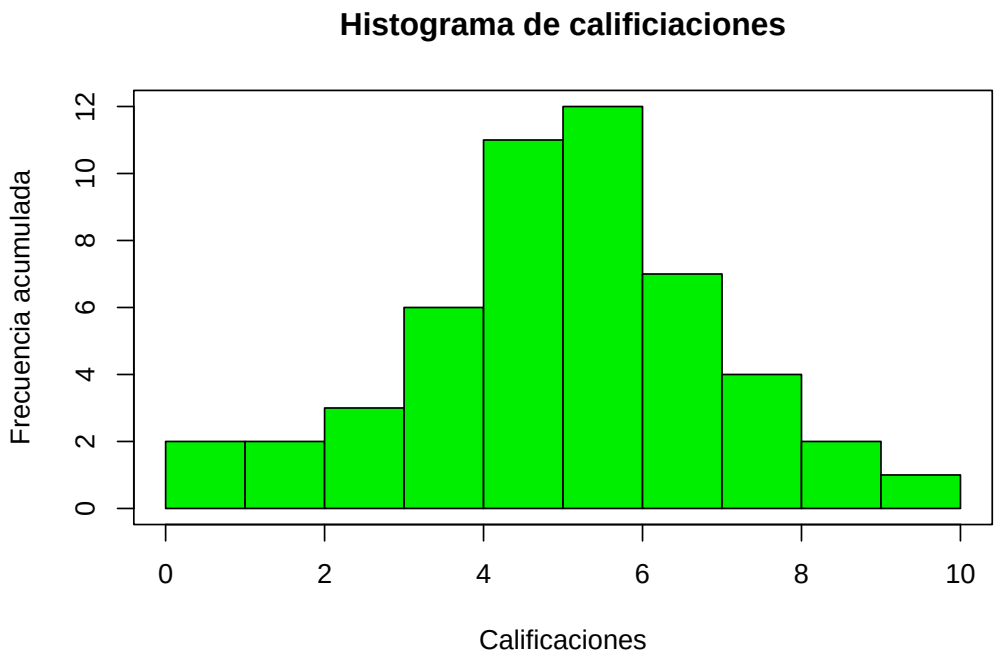
```r
# Con R necesitamos la paquetería moments
kurtosis(x)
```

```
## [1] 3.382241
```

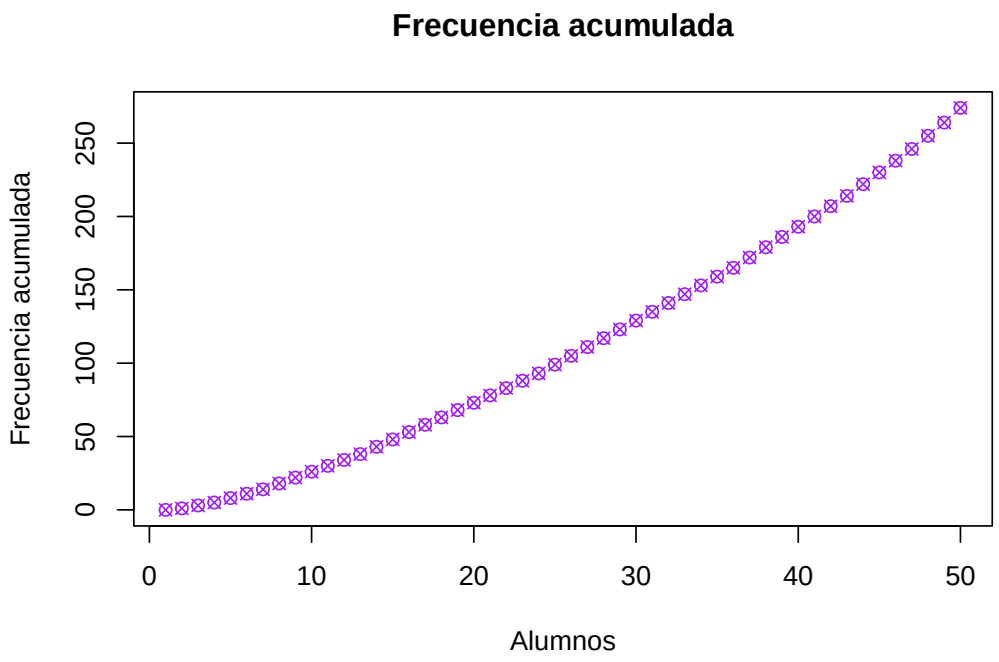
Como el coeficiente de curtosis es mayor a cero, podemos decir que nuestros datos tienen una distribución leptocúrtica, es decir los datos están untanto concentrados en la media, siendo una curva apuntada.

Gráficamente

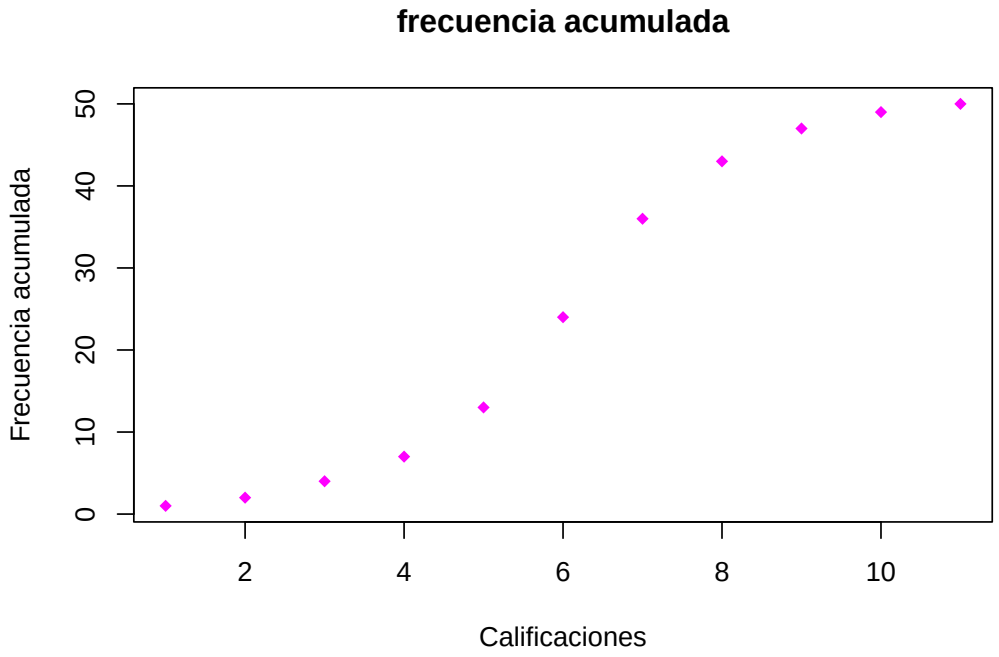
```r
# Histograma de frecuencia acumulada
hist(x,breaks=10,col="green2",main="Histograma de calificiaciones",xlab="Calificaciones",ylab="Frecuenc
box()
```



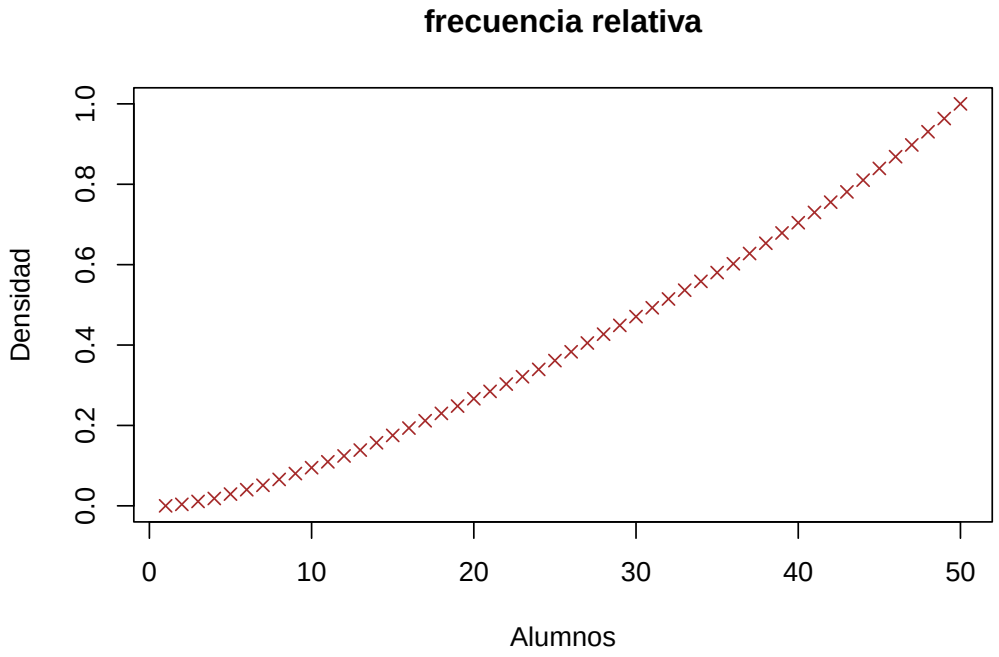
```r
# Frecuencia acumulada
fa=cumsum(x);
plot(fa,main="Frecuencia acumulada",xlab="Alumnos",ylab="Frecuencia acumulada",col="purple",pch=13)
```



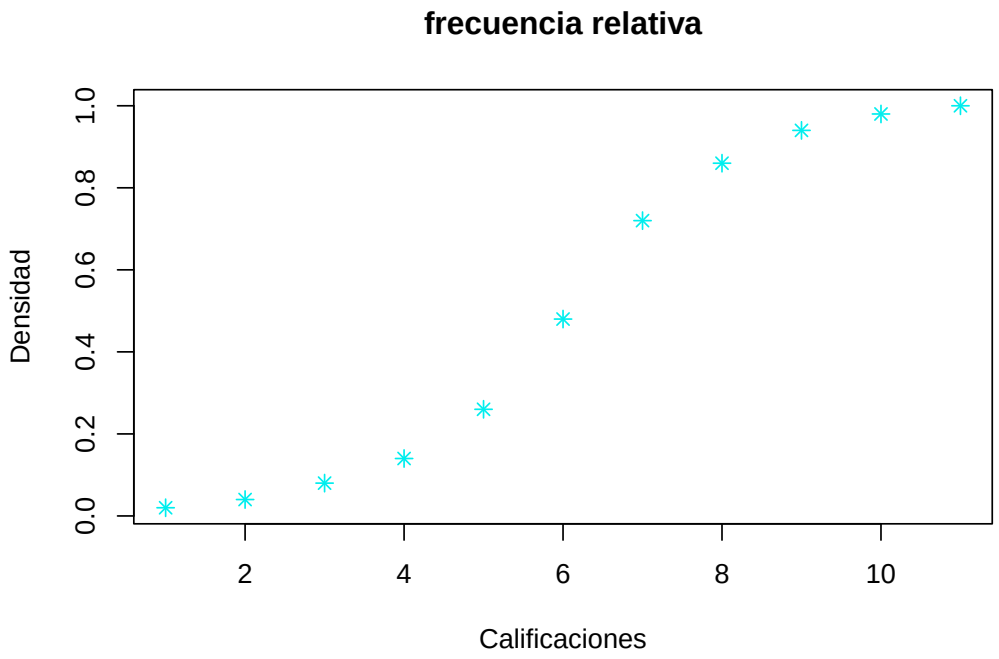

```r
fa1=cumsum(table(x));
plot(fa1, main="frecuencia acumulada",xlab="Calificaciones",ylab="Frecuencia acumulada",
col="magenta",pch=18)
```



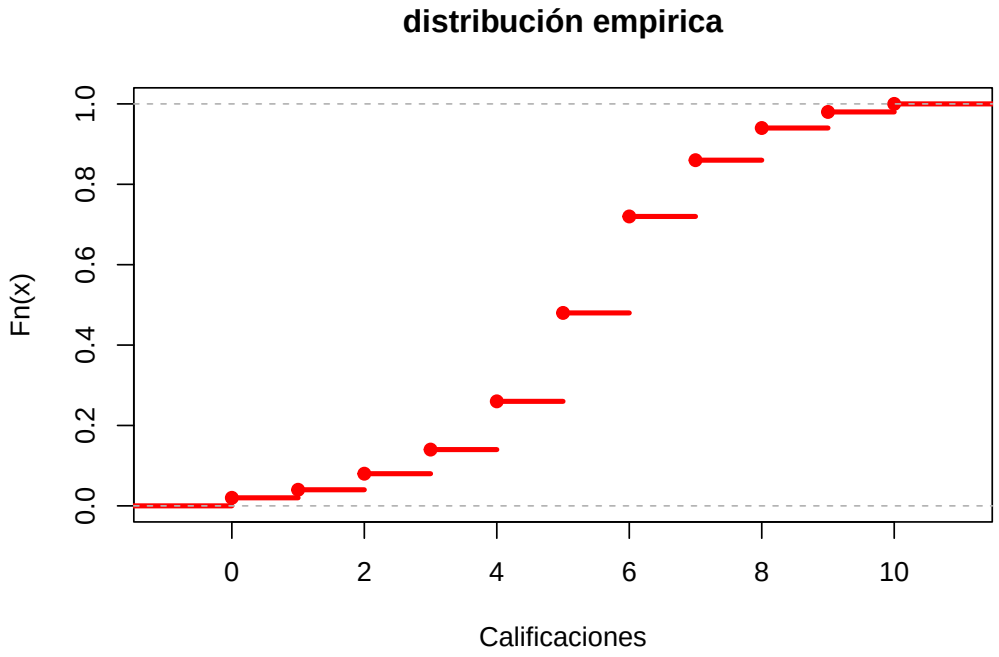

```
# Frecuencia relativa
fr=cumsum(x)/sum(x);
plot(fr,main="frecuencia relativa",xlab="Alumnos",ylab="Densidad",col="brown",pch=4)
```



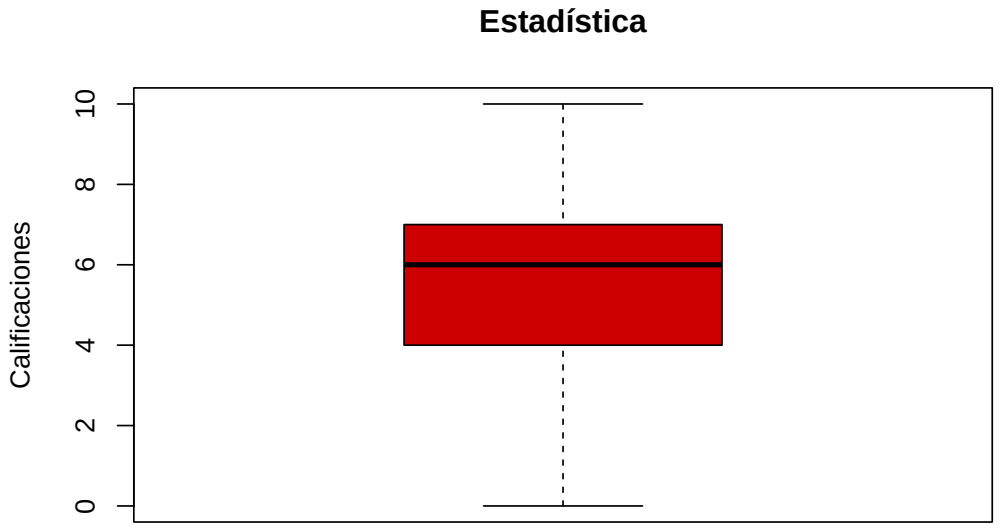
```
# Frecuencia relativa
fr1=cumsum(moda)/sum(moda);
plot(fr1, main="frecuencia relativa",xlab="Calificaciones",ylab="Densidad",col="cyan2",
pch=8)
```




```
# Gráfica de distribución acumulada empírica
ECDF1=ecdf(x);
plot(ECDF1, col="red",lwd=3,xldo="datos",yldo=" ",
main="distribución empirica", xlab="Calificaciones")
```



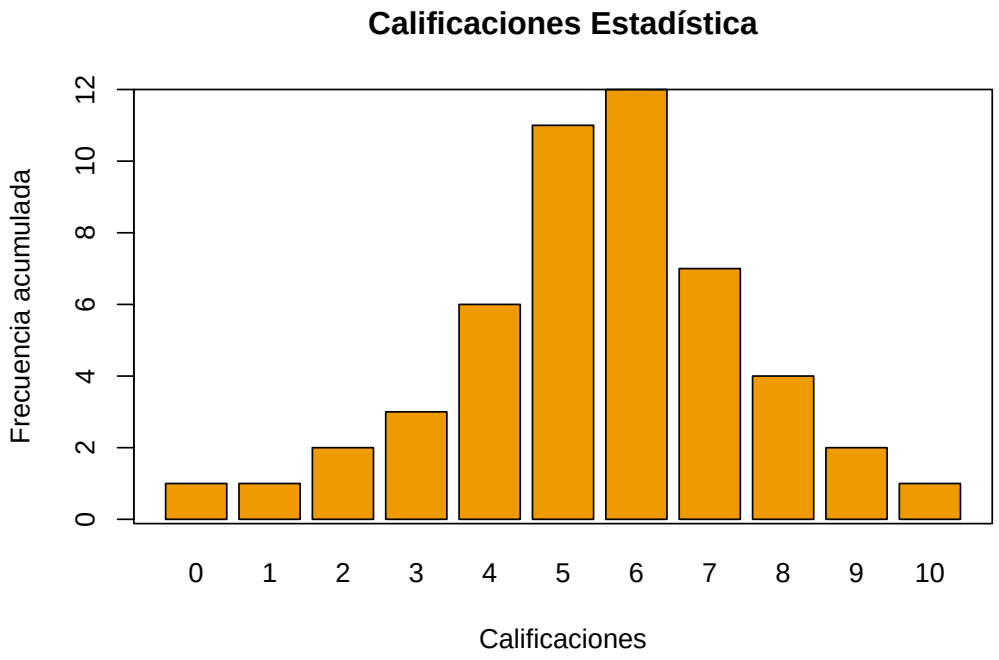

```r
# Diagrama de caja
boxplot(x,col="red3",main="Estadística",ylab="Calificaciones")
```



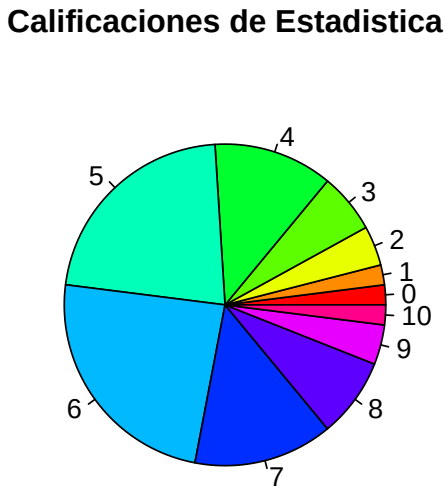

```r
# Tallo de hoja
stem(x)
```

```
##
##   The decimal point is at the |
##
##    0 | 00
##    2 | 00000
##    4 | 00000000000000000
##    6 | 000000000000000000
##    8 | 000000
##   10 | 0
```

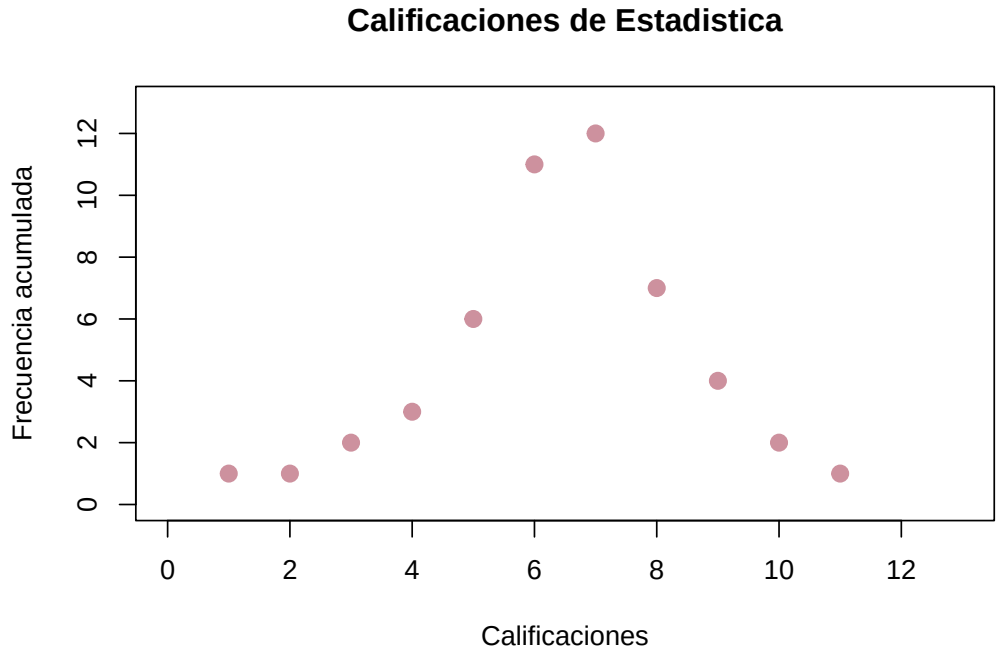
```r
# Gráfico de barras
barplot(moda,col="orange2",main="Calificaciones Estadística",
ylab="Frecuencia acumulada",xlab="Calificaciones");
box()
```




```r
# Diagrama de pie
pie(moda,main="Calificaciones de Estadistica",col=rainbow(11))
```




```
# Gráfico de dispersión
plot(moda,col="pink3",pch=20, cex=2,main="Calificaciones de Estadistica",
ylab="Frecuencia acumulada",xlab ="Calificaciones",ylim=c(0,13),
xlim=c(0,13),bcolor="red")
```



R posee una gran diversidad de gráficos descriptivos que se pueden utilizar. Es recomendable explorar un

poco más por ejemplo la librería `ggplot2`.

Probabilidad

R de forma predeterminada (en el paquete `STATS`), ya tiene cargadas 16 distribuciones, 5 discretas (binomial, poisson, geométrica, hipergeométrica, binomial negativa) y 11 continuas (uniforme, beta, cauchy, ji-cuadrada, exponencial, gamma, F, log-normal, normal, T-student, weibull), estas distribuciones son las más conocidas. Con el paquete `actuar` se añaden otras 21 distribuciones más, con las que se pueden hace mezclas y transformaciones conforme sea el caso de estudio. De hecho existe el paquete `dist` con el cual se pueden crear distribuciones como mezclas a voluntad y está especializado en este ramo de la probabilidad.

Con las distribuciones de R se pueden generar:

- `r`: Muestras pseudo-aleatorias (`r` debe anteceder a la identificación de la distribución)
- `q`: Cuantiles (`q` debe anteceder a la identificación de la distribución)
- `d`: Función de densidad/masa de probabilidad (`d` debe anteceder a la identificación de la distribución)
- `p`: Función de distribución acumulada (`p` debe anteceder a la identificación de la distribución)

Muestra Aleatoria (`r`)

Ejemplo con distribución Normal Estándar.

```
# Necesitamos una muestra aleatoria de tamaño 1000
# Semilla
set.seed(1978)
# Creamos la muestra
sim<-rnorm(1000)
# Observamos los primeros 20 elementos de la muestra
sim[1:20]
```

```
##  [1] -1.02331764  0.39436819 -0.56102909  2.45981477  0.86510097 -1.05075361
##  [7] -1.42896137  0.21684883 -0.34349509 -1.69987825  0.59720539 -0.01361331
## [13]  0.46775929 -1.70486307 -1.55470442 -0.60600234  0.54179878  1.83888064
## [19]  0.39619325  0.61401606
```

```
# Calculamos media, varianza y desviación estándar de la muestra
mean(sim); var(sim); sd(sim)
```

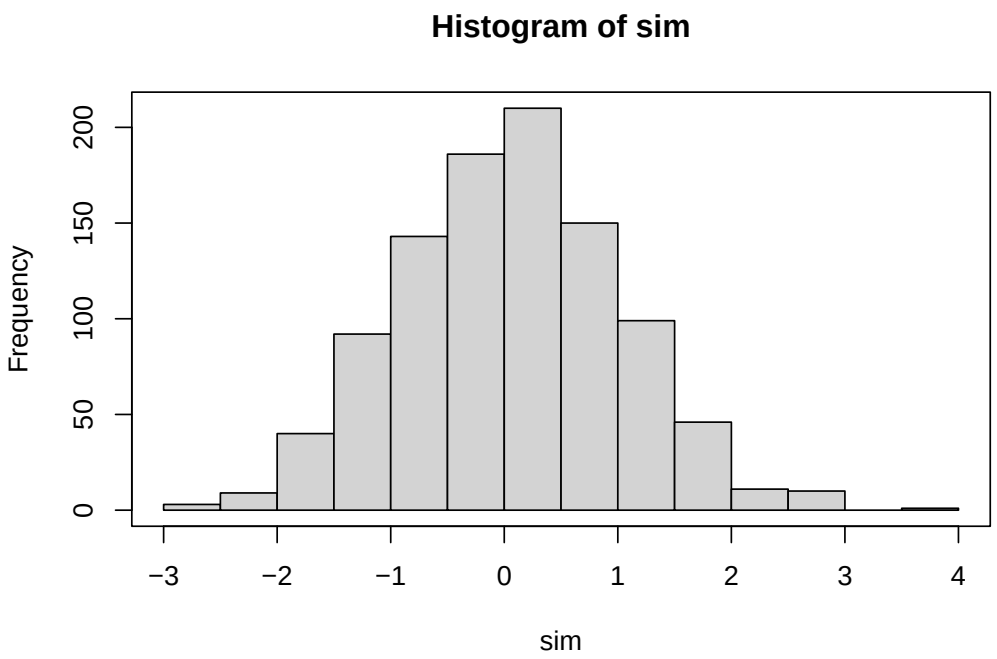
```
## [1] 0.05557488
```

```
## [1] 0.9358718
```

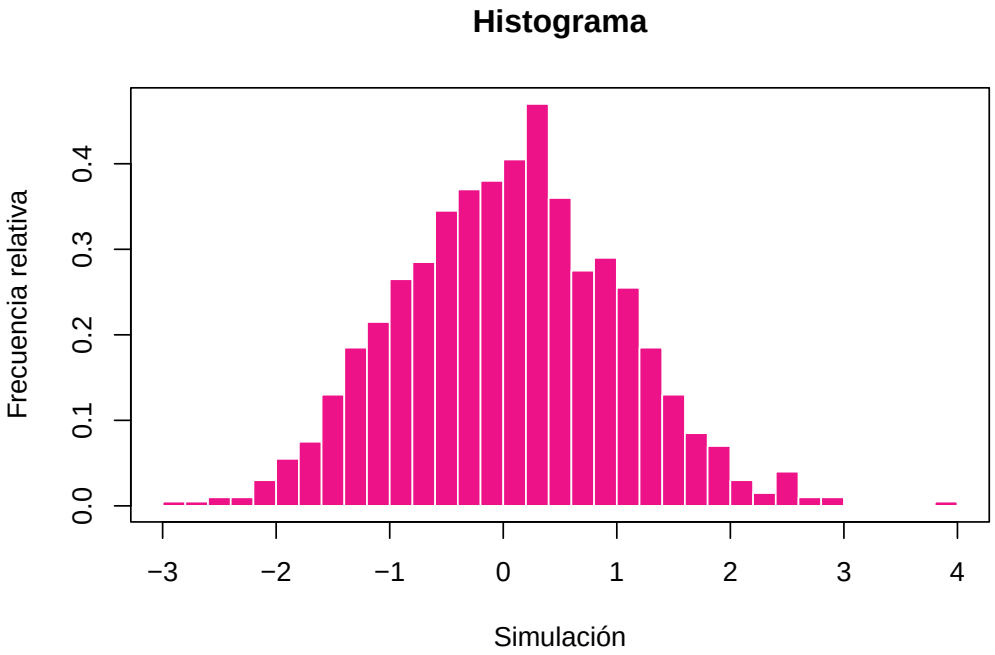
```
## [1] 0.9674047
```

Se observa que la esperanza/media muestral $\overline{x} = 0.0556$ es cercana a la esperanza teórica $\mu = 0$. Lo mismo pasa con la desviación estándar muestral $s = 0.9674$ es similar a la desviación teórica $\sigma = 1$.

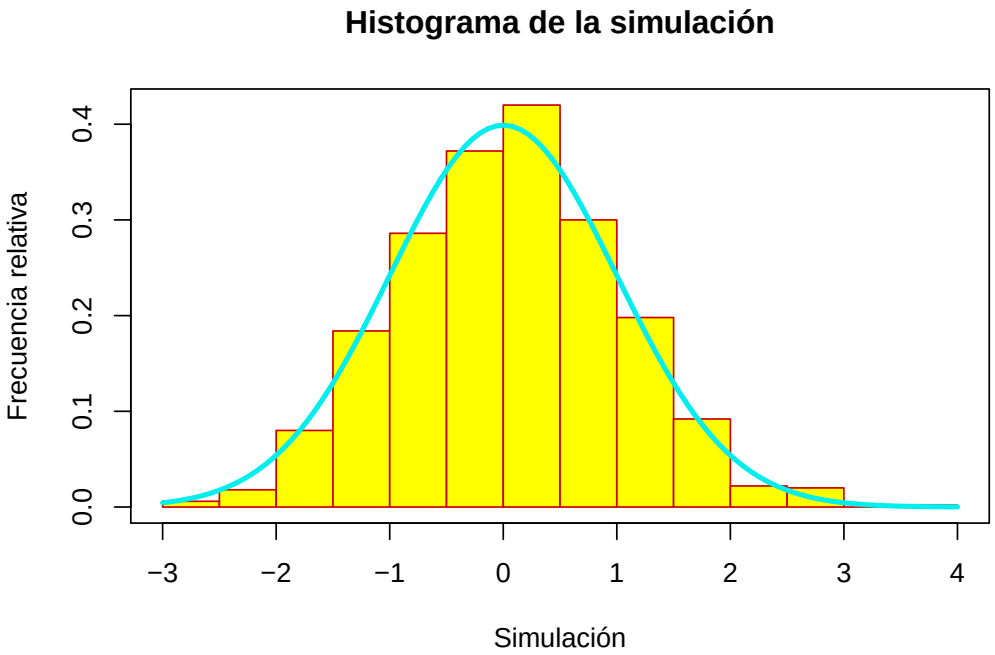
```r
# Graficamos el histograma que R crea por default
hist(sim)
box()
```



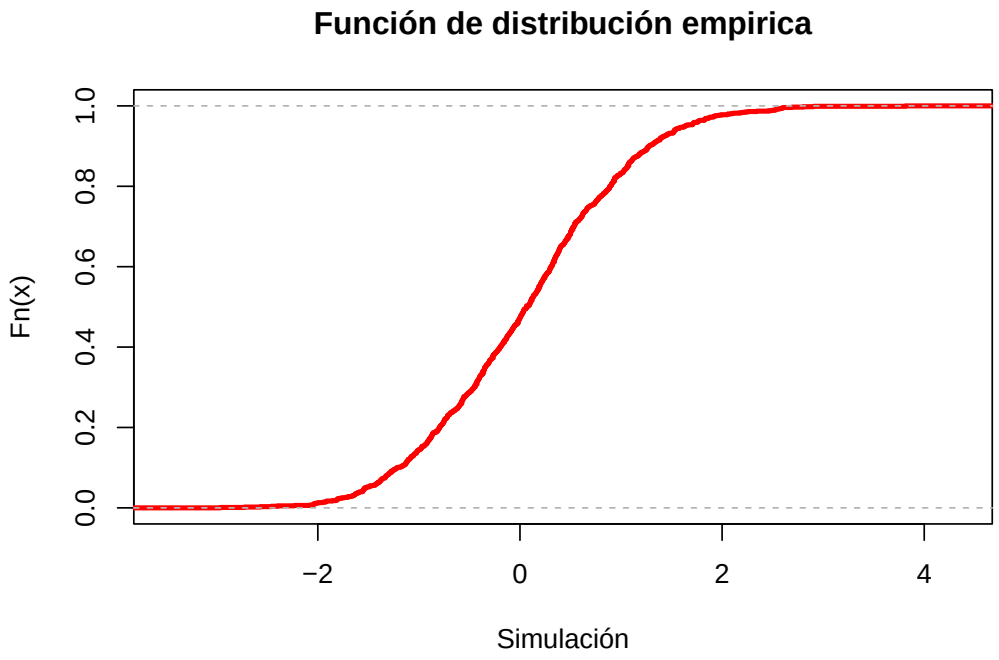

```r
# Agregamos algunos paramétros a la función, para obtener un mejor resultado
hist(sim,col="deeppink2",breaks=30,border="white",freq=0,xlab="Simulación", ylab = "Frecuencia relativa
box()
```




```
# Ponemos en un mismo plano la densidad empírica y el histograma
hist(sim,col="yellow",border="red3",freq=0, xlab="Simulación", main="Histograma de la simulación",ylab =
curve(dnorm(x),-3,4,add=T,col="cyan2",lwd=3)
box()
```



```r
# Graficamos la función de distribución empírica
fde=ecdf(sim)
plot(fde, col="red",lwd=3,xlab="Simulación", main="Función de distribución empirica")
```

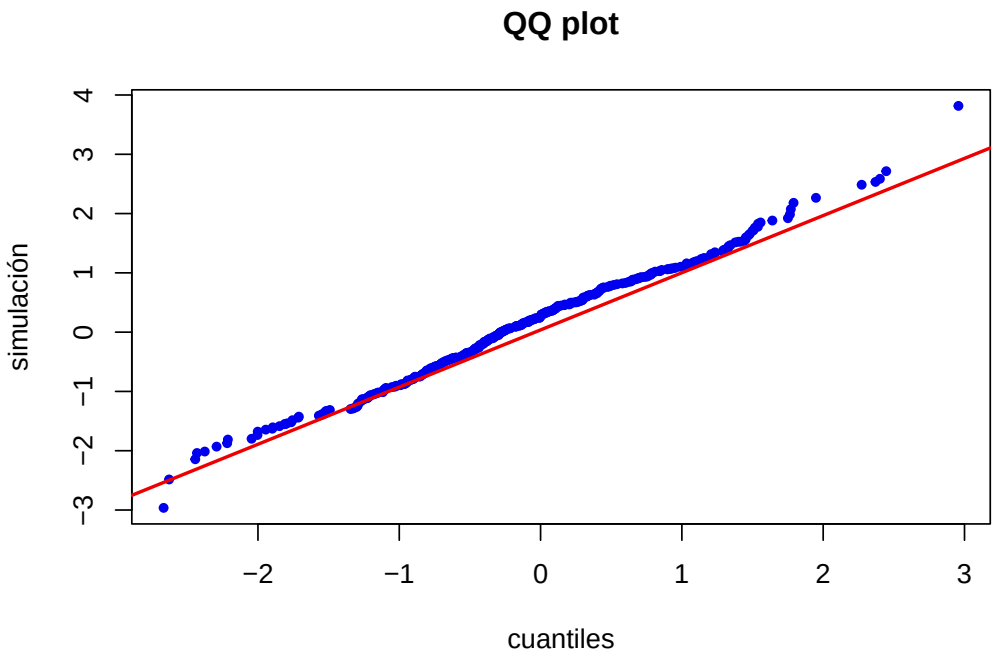


Cuantiles (q)

```r
# Cuantiles
probabilidades=c(0.8,0.9,0.95,0.975,0.99)
qnorm(probabilidades)
```

```
## [1] 0.8416212 1.2815516 1.6448536 1.9599640 2.3263479
```

```r
# Gráfica cuantil cuantil
qqplot(qnorm(sim),sim,col="blue2",pch=20,xlab="cuantiles", ylab="simulación",main="QQ plot")
qqline(sim,col="red2",lwd=2)
```

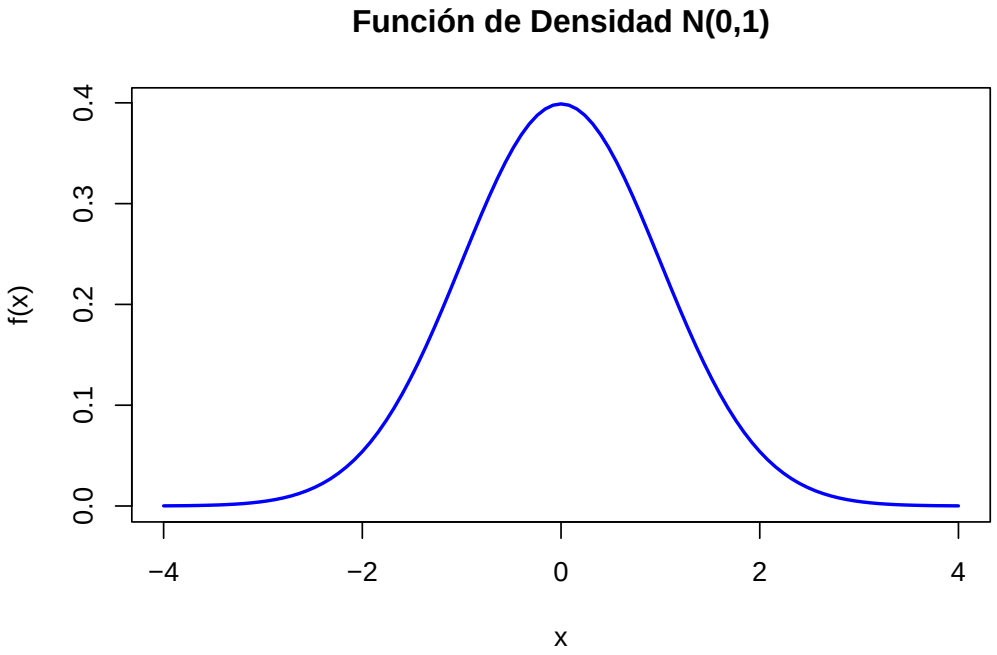


Densidad (d)

```
# Función de densidad
valores=c(-2.3,-2,-1.6,-1.3,-1.0,1,1.3,1.6,2,2.3)
dnorm(valores)
```

```
##  [1] 0.02832704 0.05399097 0.11092083 0.17136859 0.24197072 0.24197072
##  [7] 0.17136859 0.11092083 0.05399097 0.02832704
```

```
# Densidad teórica
curve(dnorm(x),xlim=c(-4,4),col="blue",lwd=2, xlab="x",ylab="f(x)", main="Función de Densidad N(0,1)")
```

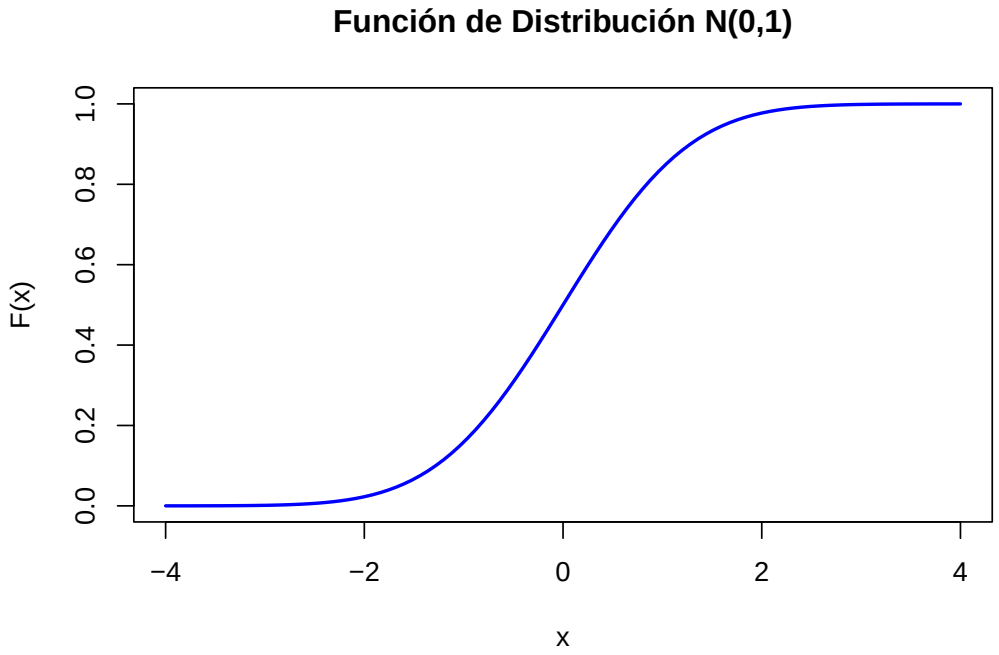


Distribución (p)

```
# Función de distribución acumulada
valores=c(-2,-1.6,-1.0,1,1.6,2)
pnorm(valores)
```

```
## [1] 0.02275013 0.05479929 0.15865525 0.84134475 0.94520071 0.97724987
```

```
# Distribución acumulada teórica
curve(pnorm(x),xlim=c(-4,4),col="blue",lwd=2, xlab="x",ylab="F(x)",main="Función de Distribución N(0,1)
```



Programación

Estructuras de control y ciclos

if

```
#Se ejecuta el comando si la condición es TRUE
if(3 > 2) print(' :D ')
```

```
## [1] " :D "
```

```
x=1; Y=NULL
if(x > 3){
y <- 10
}else{
y<-0};
y
```

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

```
ifelse(3 < 2,print(':D'),print(':O'))
```

```
## [1] ":O"
```

```
## [1] ":O"
```

for

```
x<-c("a","b","c","d")
for(i in 1:6) {
print(x[i]) # Imprime cada uno de los elementos de x.
} # Los dos últimos, al no existir en x aparecen como "NA"
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
## [1] NA
## [1] NA
```

while

```
f <- 5 # Valor inicial
n <- 0
while(f > 0.001) {
n <- n + 1
f <- f / n
print(f)
}
```

```
## [1] 5
## [1] 2.5
## [1] 0.8333333
## [1] 0.2083333
## [1] 0.04166667
## [1] 0.006944444
## [1] 0.0009920635
```

repeat

```
v <- c("Hola","mundo")
cnt <- 2
repeat {
  print(v)
  cnt <- cnt+1
  if(cnt > 5) {
    break
    }
}
```

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

Crear una función

```
#Sucesión de fibonacci
fibo=function(n){
  Res=numeric(n)
  if(n==1){
    Res[1]=1}
  if(n==2){
    Res[1:2]=c(1,1)}
  if(n>2){
    Res[1:2]=c(1,1)
    for(i in 3:n){
      Res[i]=Res[i-1]+Res[i-2]
    }
  }
Res
}
####Ejemplo
fibo(10)
```

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