Universidad Tecnológica de Panamá
Centro de Investigaciones Hidráulicas e Hidrotécnicas

Introducción a R y RStudio

José Ulises Jiménez, MSc.

Jueves 5 de diciembre de 2019

Resumen

Se muestra como instalar R y RStudio, y se presenta una introducción básica de su uso.

Índice

Índice de figuras

1. Introducción

1.1. Acerca de R

R es un conjunto integrado de funciones de software para manipulación de datos, cálculo y visualización gráfica. Este documento trata sobre funciones escritas en R "lenguaje y entorno para la informática estadística"[1], y se ha generado con R version 3.6.1 (2019-07-05).

1.2. Créditos y Licencia

Este documento y los códigos fueron escritos en *Sweave*[2], *R*[3] y LaTeX[4]. Esta obra está bajo una licencia de *Creative Commons* Reconocimiento - Compartir igual 3.0[5].

En resumen, esto significa que Usted es libre de:

1. Compartir: copiar, distribuir y transmitir la obra;
2. Editar: adaptar el trabajo al contexto requerido.

En las siguientes condiciones:

1. Atribución: debe atribuir el trabajo de la manera especificada por el autor o licenciante (pero no de ninguna manera que sugiera que le respalden o su uso de la obra);
2. Compartir por igual: si modifica, transforma o se basa en este trabajo, puede distribuir el trabajo resultante solo bajo una licencia igual, similar o compatible.

Dado que este tipo de licencia es altamente permisiva, este documento se proporciona sin responsabilidades, ni garantías.

1.3. ¿Qué es R?

- R es un programa de última generación para realizar análisis de datos, siendo también un lenguaje de programación, lo cual lo hace muy versátil.
- Como lenguaje de programación, R es un dialecto de un lenguaje de programación denominado S.
- Dentro de los lenguajes de programación se puede clasificar como un lenguaje orientado a objetos de tipo interpretado. Lo que lo hace flexible, potente y posee un tiempo de aprendizaje corto.
- Actualmente se encuentran disponibles más de 15303 paquetes desarrollados en R, que cubren múltitud de campos desde aplicaciones Bayesianas, financieras, graficación de mapas, wavelets, análisis de datos espaciales, etc.

[1] http://www.R-project.org

[2] La función *Sweave* de R proporciona un marco flexible para mezclar texto y código R para la generación automática de documentos.

[3] R es un entorno y lenguaje de programación con un enfoque al análisis estadístico.

[4] LaTeXes un sistema de preparación de documentos para composición de texto de alta calidad

[5] Reconocimiento - Compartir igual (by-sa): se permite el uso comercial de la obra y de las posibles obras derivadas, siempre que su distribución se haga igual que la obra original.

1.3.1. Notas históricas

- 1991: creado en Nueva Zelanda por Ross Ihaka y Robert Gentleman.
- 1993: primer anuncio público de R.
- 1995: Martin Machler convence a Ross y Robert para hacer de R un sofware libre bajo la licencia GNU.
- 1997: se crea el núcleo de desarrollo del código fuente de R, que pasa a controlar todo lo relativo al desarrollo del código fuente.
- 2000: aparece la versión 1.0.0 de R.
- En este momento, la versión más actualizada lleva el número 3.6.1 (03-12-2019).

1.4. Características de R

- R proporciona muchísimas herramientas estadísticas para el análisis de datos: modelos lineales y no lineales para regresión, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, gráficas, etc.
- Como es un lenguaje de programación, permite que los usuarios lo extiendan definiendo sus propias funciones. Gran parte de las funciones de R están escritas en el mismo R, aunque para algoritmos computacionalmente exigentes es posible desarrollar bibliotecas en C, C++ o Fortran que se cargan dinámicamente. La sintaxis es relativamente simple.
- R también puede usarse como herramienta de cálculo numérico, donde puede ser tan eficaz como otras herramientas específicas tales como GNU Octave y su equivalente comercial, MATLAB.
- Es multiplataforma, se puede ejecutar en casi todas los sistemas operativos (Linux, Windows, MacOS).
- Existe una comunidad muy extendida que lo mantiene en permanente actualización.
- Está dividido en paquetes modulares que responden a necesidades específicas.
- Posee exelente capacidades gráficas.
- Posee un excelente paquete, knit, que permite desarrollar presentaciones interactivas que evaluan el código R en el momento.
- Es libre.

1.5. Diseño del sistema R

El sistema R está dividido en dos partes:

- El sistema "base", que puede ser descargado desde la red de servidores CRAN (Comprehensive R Archive Network) http://cran.r-project.org, contiene las funciones fundamentales y todo lo requerido para que R funcione.
  - Es sistema base contiene los paquetes: utils, stats, datasets, graphics, ...
  - También contiene paquetes recomendados: boot, class, cluster, codetools, ...
- Todo el resto de paquetes pueden ser descargados desde CRAN. Existen en este repositorio más de 15000 paquetes, a eso hay que agregarle la innumerable cantidad de paquetes que existen en sitios personales y que son de libre uso.

1.6. Uso de la consola y las GUI

Hay dos maneras de interaccionar con R:

- por consola
- mediante una interfaz gráfica que hace de puente entre el programa y el usuario.
  - Rstudio
  - Rcommander
  - ...

Es conveniente instalar una interfaz gráfica como Rstudio en http://www.rstudio.com.

1.7. Fuentes para aprender R

Disponibles en CRAN http://cran.r-project.org.

- An Introduction to R https://cran.r-project.org/doc/manuals/R-intro.html.
- Writing R Extensions.
- R Data Import/Export.
- R Installation and Administration (mostly for building R from sources).
- R Internals.
- CRAN Task Views http://cran.r-project.org/web/views/.

1.8. Otras fuentes

Artículo de wikipedia y referencias citadas en http://es.wikipedia.org/wiki/R_(lenguaje_de_programacion).

2. Instalación de R y RStudio

Primero debe instalar R, que es un programa básico con una interfaz simple; y segundo, debe instalar RStudio. En este apartado se muestra como encontrar los archivos ejecutables para la instalación de R y RStudio. Para la instalación de ambos programas, simplemente acepte los valores predeterminados ofrecidos por los programas durante la instalación.

Para instalar el sistema base o entorno R, se deben seguir las instrucciones de la página de CRAN http://cran.r-project.org. Instale la versión correcta de R para el sistema operativo de su computadora.

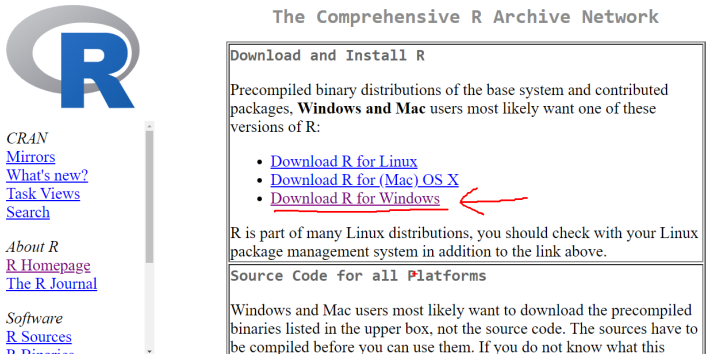

Figura 1. Captura de pantalla de la página principal de CRAN.

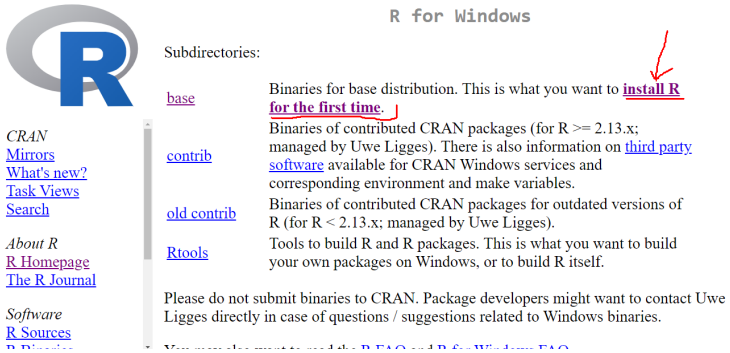

Figura 2. Captura de pantalla para escoger el archivo instalable.

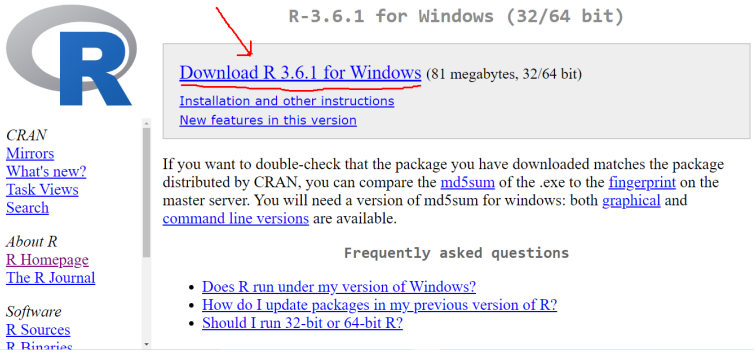

Figura 3. Captura de pantalla de la página CRAN para descargar el archivo de instalación de R para el sistema operativo Windows.

Para instalar RStudio vaya a la página http://rstudio.org/. Siga las instrucciones para instalar la versión correcta de RStudio en su computadora.

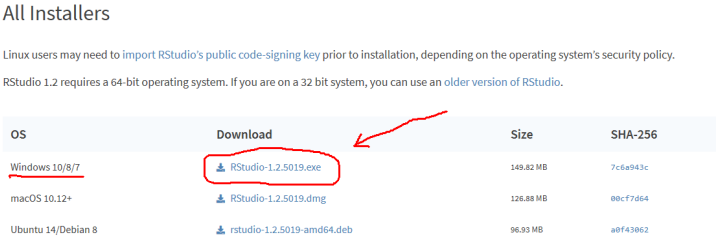

Figura 4. Captura de pantalla de la página de RStudio para descargar el archivo de instalación.

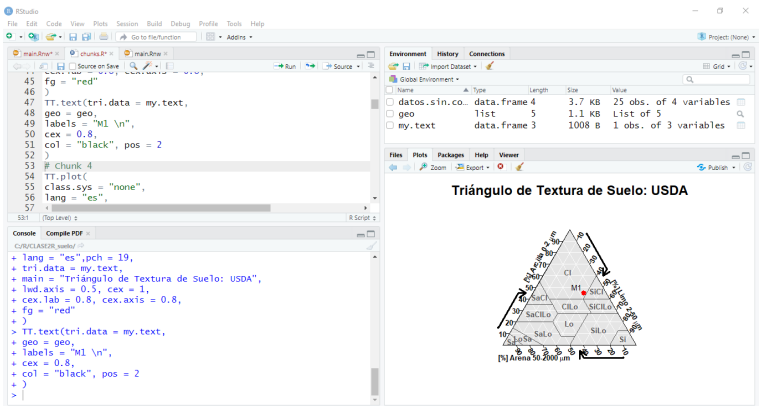

Figura 5. Captura de pantalla de RStudio ejecutándose en el sistema operativo Windows.

RStudio debe mostrar cuatro paneles. El panel superior izquierdo es donde puede ingresar líneas de código en un archivo "script". RStudio permite que se abran muchos archivos de script y utiliza pestañas para ayudarlo a realizar un seguimiento de ellos.

El panel inferior izquierdo es la "consola" donde puede escribir comandos y ver sus respuestas, como una calculadora. Además, si usa un archivo de script y ejecuta sus comandos, la salida de texto aparecerá aquí en la consola.

En el panel superior derecho, verá las variables y sus valores después de declararlas. Finalmente, el panel inferior derecho muestra información de ayuda y diagramas que crea.

3. Instalación de paquetes

- Una vez instalado el sistema base, para instalar paquetes adicionales se utiliza la función:

```
install.packages("nombre del paquete")
```

- Muchos de estos paquetes, además de poseer funciones poseen paquetes de datos.
- Luego de instalar los paquetes, antes de usarlos debe llamarse en cada sesión de R y se utiliza la función:

```
library(nombre del paquete)
```

RStudio facilita la instalación de paquetes con las opciones del menú.

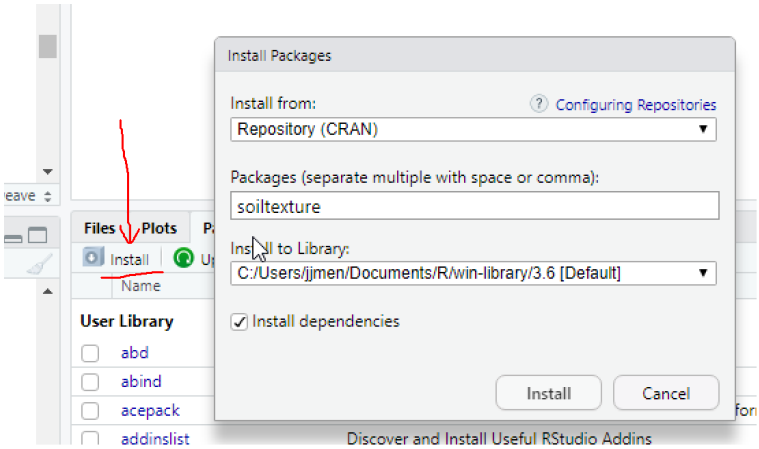

Figura 6. Captura de pantalla de RStudio en la instalación de paquetes.

3.1. Conjunto de datos que acompañan las librerías de R

Además de las funciones asociadas en librerías específicas, R posee paquetes de datos. Si se tipea en consola `data()`, se obtiene una lista de todos los paquetes de datos (principalmente en dataframes) que se encuentran dentro del entorno de trabajo. Para información sobre un dataset en particular se utiliza `data(package="HistData")`.

HistData es un paquete de datos que provee una conjunto de datos importantes en el desarrollo histórico de la Estadística. Si se escribe en consola `sessionInfo()` se obtiene la lista de todos los paquetes (librería de funciones y paquetes de datos) que están incorporados a la sesión de R. Si se desea incorporar otro paquete de datos, se debe escribir `library(nombre del paquete)` en consola, por ejemplo `library(HistData)`.

3.2. Encontrar las librerias que necesitamos

En el repositorio de CRAN, se incluye una página que agrupa las librerías de acuerdo a las tareas que realizan. Los ítems que están disponibles incluyen por ejemplo: Inferencia Bayesiana, Análisis de Clusters, Gráficas y Análisis de Series Temporales. La dirección de la página es http://cran.r-project.org/web/views/.

4. Introduciendo órdenes por consola y códigos

Hay muchas formas de obtener ayuda con R y RStudio. Este sitio, https://rseek.org/, me ha servido muchísimo para conocer que paquetes existen para trabajar en un tema específico. Otra forma es preguntar directamente en la consola con las siguientes órdenes. Prueba estas órdenes:

```
?mean
help(mean)
apropos("mean")
example(mean)
```

Trabajando de la forma más básica, podremos ingresar sentencias en la consola, como si usaramos R como una calculadora. El símbolo $<-$ es el operador asignación, también puede utilizarse el signo $=$. En *RStudio* puedes presionar $Alt + -$ para escribir $<-$. Se recomienda usar el *R script* para escribir las órdenes para luego enviarlas a la consola donde se ejecutan presionando Run en el menú o las teclas $crtl + Enter$ en *RStudio* o las teclas $crtl + R$ en *RGUI*.

```
# Puedes usar el símbolo de hashtack (almohadilla o numeral)
# para hacer tus apuntes dentro del script, esas líneas con
# no se ejecutan.
x1 <- 12
print(x1)

## [1] 12
```

```
msg <- "Hola"
msg

## [1] "Hola"
```

```r
# Adición.
2 + 3

## [1] 5
```

```r
# Substracción.
5-1

## [1] 4
```

```r
# Multiplicación.
2*3

## [1] 6
```

```r
# División.
7/3

## [1] 2.333333
```

```r
# Use la función sqrt( ) para obtener la raiz cuadrada de un número.
sqrt(9)

## [1] 3
```

```r
# Potenciación.
9^2

## [1] 81
```

```r
# Logaritmo natural.
log(3)

## [1] 1.098612

# Logaritmo de 3 (base 10), o use log10(3).
log(3,10)

## [1] 0.4771213
```

```r
# Esto es e^3
exp(3)

## [1] 20.08554
```

Más órdenes... prueba reproducirlas.

```r
2+3
2*(3+5)
2+3; 2+4; 2+5
2+2.5
2+2,5  # Produce error, por la coma, debe usar punto decimal.
2*3+5/2
2*(3+5/2)
2*(3+5)/2
2/3+4
2/(3+4)
2^3*5
2^(3*5)
2^-5
2^(-5)
534%/%7
534%%7
534-76*7
pi
2^pi
2(3+5) # Produce error, debe expresar la multiplicación con *.
2*(3+5)
2pi # Produce error, debe expresar la multiplicación con *.
2*pi
2^40
2^(-20)
c(2^40,2^(-20),5)
sqrt(4)
sqrt(8)-8^(1/2)
log10(8)
log(8)/log(10)
7^log(2, 7)
10!  # Produce error, la forma correcta es la siguiente.
factorial(10)
exp(sqrt(8))
choose(5,3)
choose(3,5)
cos(60)
cos(60*pi/180)
```

```
acos(0.5)
acos(0.5)*180/pi
acos(2) # No es un número.
2*exp(1)
exp(pi)-pi^exp(1)
sqrt(2)^2-2
sqrt(2)
print(sqrt(2), 20)
print(sqrt(2), 2)
2^100
print(2^100, 15)
print(2^100, 5)
print(sqrt(2), 22)
print(sqrt(2), 23) # Produce máximo 22 digitos.
print(pi,22)
print(sqrt(2), 3)
print(sqrt(2), 3)^2
1.41^2
round(sqrt(2), 3)
round(sqrt(2), 3)^2
round(2.25, 1)
round(2.35, 1)
round(sqrt(2))
round(sqrt(2), 0)
round(digits=3, sqrt(2))
round(3, sqrt(2))
floor(8.3)
ceiling(8.3)
trunc(8.3)
round(8.3)
floor(-3.7)
ceiling(-3.7)
trunc(-3.7)
round(-3.7)
x=5
x^2
x=x-2
x
x^2
x=sqrt(x)
x
f=function(x){x^2-2^x}
f(30)
f=function(x, y){exp((2*x-y)^2)}
f(0, 1)
```

```
f(1, 0)
rm(list=ls())
f=function(t){t^2-2^t}
a=1
a
ls()
rm(a)
ls()
a # Produce error, se borro a con rm(a), no hay objeto a.
(2+5i)*3
(2+5i)*(3+7i)
(2+5i)/(3+7i)
2+5*i # Produce error, no lleva el *
(3+i)*(2-i) # Produce error, debe llevar un coeficiente i.
(3+1i)*(2-1i)
1+2/3i
1+(2/3)i # Produce error, la forma correcta es la siguiente.
complex(real=1,imaginary=2/3)
z=1+sqrt(2)i # Produce error, la forma correcta es la siguiente.
z=complex(real=1, imaginary=sqrt(2))
z
sqrt(-3) # Produce error las raíces de números negativos.
sqrt(as.complex(-3))
sqrt(2+3i)
exp(2+3i)
sin(2+3i)
acos(as.complex(2))
Re(4-7i)
Im(4-7i)
Mod(4-7i)
Arg(4-7i)
Conj(4-7i)
z=complex(modulus=3, argument=pi/5)
z
Mod(z)
Arg(z)
pi/5
```

```
# Crea un vector de enteros del 1 al 5.
1:5

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

```r
# Combinando cinco números en un vector.
c(1, 2.5, 3, 4, 3.5)

## [1] 1.0 2.5 3.0 4.0 3.5
```

```r
# Almacena números de la variable "a".
a = c(1, 2.5, 3, 4, 3.5)
```

```r
# Suma todos los valores de la variable "a".
sum(a)

## [1] 14
```

```r
# Dice cuántos números hay en la variable "a".
length(a)

## [1] 5
```

```r
# Estadísticas descriptivas para la variable "a".
summary(a)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     2.5     3.0     2.8     3.5     4.0
```

```r
# Divide cada valor en "a" por 5.
a/5

## [1] 0.2 0.5 0.6 0.8 0.7
```

```r
# Raíz cuadrada de 2, incluye texto.
cat("La raíz cuadrada de 2 es", sqrt(2),"\n")

## La raíz cuadrada de 2 es 1.414214
```

```r
# Raíz cuadrada de 2, incluye texto, se redondea a dos decimales.
cat("La raíz cuadrada de 2 es", round(sqrt(2),2),"\n")

## La raíz cuadrada de 2 es 1.41
```

Más órdenes... prueba reproducirlas

```
x=c(1,5,6,2,5,7,8,3,5,2,1,0)
x
nombres=c("Pep","Catalina","Joan","Pau")
nombres
nombres=c(Pep,Catalina,Joan,Pau) # Si se nos olvidan las
# comillas... Produce error
c(2,3.5,TRUE,"casa")
x_scan=scan()
1 5 6 2 5 7 8 3 5 2 1 0 # Después de introducir cada número
# presiona enter, al final presiona dos veces enter.
x_scan
notas=scan("http://aprender.uib.es/Rdir/notas.txt")
notas
# Crea un documento notas.txt con los datos anteriores y luego,
#crea un objeto llamando los datos de ese archivo.
notas2=scan("notas.txt")
notas2
x_scan2=scan()
1,5,6,2,5,7,8,3,5,2,1,0 # Introduce toda la secuencia, al final
 # presiona dos veces enter. Produce error, la forma correcta a
 # continuación.
x_scan2=scan(sep=",")
1,5,6,2,5,7,8,3,5,2,1,0 # Introduce toda la secuencia,
# al final presiona dos veces enter.
x_scan2
x_scan3=scan()
4,5 6,2 # después de introducir cada número presiona enter,
# al final presiona dos veces enter. Produce error.
x_scan3=scan(dec=",")
4,5 6,2 # después de introducir cada número presiona enter,
# al final presiona dos veces enter.
x_scan3
x_scan4=scan(sep=",")
Pep, Catalina, Joan, Pau
x_scan4=scan(what="character", sep=",")
x_scan4
x_scan5=scan("http://aprender.uib.es/Rdir/enlatin1.txt",
what="character")
x_scan5
x_scan6=scan("http://aprender.uib.es/Rdir/enlatin1.txt",
what="character", encoding="latin1")
x_scan6
rep(1, 6)
rep("Palma", 5) # Las palabras, siempre entre comillas.
rep(c(1,2,3), times=5)
```

```
rep(c(1,2,3), each=5)
rep(c(1,2,3,4), c(2,3,4,5))
seq(3, 150, by=4.5)
seq(80, 4, by=-3.5)
seq(80, 4, by=3.5) # Si el signo en el argumento by no es el
# correcto, produce error.
1:15
2.3:12.5
34:-5
-3:5   # Cuidado con los paréntesis.
-(3:5)
seq(2, 10, length.out=10)
seq(2, by=0.5, length.out=10)
x=c(rep(1, 10), 2:10)
x
x=c(0,x,20,30)
x
x=c(rep(1, 10), 2:10)
fix(x)
x=seq(2, 30, by=3)
x
x+2.5
2.5*x
sqrt(x)
2^x
x^2
(1:4)^2
1:4^2
CD=function(x){summary(lm((1:4)~c(1:3, x)))$r.squared}
CD(5)
CD(6)
CD(5:10) # La función no opera con el vector de forma directa,
# produce error.
sapply(5:10, FUN=CD)
1:5+1:5 # Suma entrada a entrada.
(1:5)*(1:5) # Producto entrada a entrada.
(1:5)^(1:5) # Potencia entrada a entrada.
n=1:20 # Secuencia 1,...,20, y la llamamos n por comodidad.
x=3*2^n-20 # Aplicamos la fórmula a n=1,...,20
x
n=0:20
y=n/(n^2+1)
y
(0:20)/((0:20)^2+1)
x=c(1,5,6,2,5,7,8,3,5,2,1,0)
```

```
length(x)
max(x)
min(x)
sum(x)
prod(x)
mean(x)
cumsum(x)
diff(x)
diff(cumsum(x))
sort(x)
sort(x, dec=TRUE)
rev(x)
n=0:200
sum(1/(n^2+1))
i=0:20
x=2^(-i)
y=cumsum(x)
y
x=seq(2, 50, by=1.5)
x
x(3) # Produce error, no se usa paréntesis.
x[3] # La tercera entrada del vector, se usan corchetes.
x[length(x)]  # La última entrada del vector.
x[length(x)-5] # La sexta entrada del vector empezando
# por el final.
n=1:10
x=2*3^n-5*n^3*2^n
x
x[-3] # x sin la tercera entrada.
x[3:7] # Los elementos tercero a séptimo de x.
x[7:3] # Los elementos séptimo a tercero de x.
x[seq(1, length(x), by=2)] # Los elementos de índice impar de x.
x[seq(2, length(x), by=2)] # Los elementos de índice par.
x[-seq(1, length(x), by=2)] # Si borramos los elementos
# de índice impar, quedan los de índice par.
x[(length(x)-5):length(x)] # Los últimos 6 elementos de x.
x[length(x)-5:length(x)] # No olvides los paréntesis.
x=c(1,5,6,2,5,7,8,3,5,2,1,0)
x[x>3] # Elementos mayores que 3.
x[x>2 & x<=5] # Elementos mayores que 2 y menores
# o iguales que 5.
x[x!=2 & x!=5] # Elementos diferentes de 2 y de 5.
x[x>5 | x<=2] # Elementos mayores que 5 o menores
# o iguales que 2.
x[x>=4] # Elementos mayores o iguales que 4.
```

```
x[!x<4] # Esta condición es equivalente a la anterior.
x[x%%4==0] # Elementos múltiplos de 4.
x>3
x[x>3]
x=c(1,5,6,2,5,7,8,3,5,2,1,0)
y=c(2,-3,0,1,2,-1,4,-1,-2,3,5,1)
x[y>0] # Entradas de x correspondientes a entradas
# positivas de y.
x=c(1,5,6,2,5,7,8,3,5,2,1)
x
x[x>3] # Elementos mayores que 3.
which(x>3) # Índices de los elementos mayores que 3.
which(x>2 & x<=5) # Índices de los elementos 2 y <= 5.
which(x!=2 & x!=5) # Índices de los elementos
# diferentes de 2 y 5.
which(x>5 | x<=2) # Índices de los elementos 5 o <= 2.
which(x%%2==0) # Índices de los elementos pares del vector.
which.min(x)
which(x==min(x))
x=2^(0:10)
x
x[20<x & x<30] # Elementos de x estrictamente entre 20 y 30.
length(x[20<x & x<30]) # ¿Cuántas entradas hay entre 20 y 30?
which(x>1500) # Índices de elementos mayores que 1500.
x=c()
x
z=NULL
z
y=c(x, 2, z)
y
exp(pi)>pi^(exp(1)) #¿Es mayor e^pi que pi^e?
1234567%%9==0 # ¿Es 1234567 múltiplo de 9?
x=1:10
x
x[3]=15 # En la posición 3 escribimos 15.
x[11]=25 # Añadimos en la posición 11 un 25.
x
# Sumamos 10 a las entradas en las posiciones 2, 3 y 4.
x[c(2, 3, 4)]=x[c(2, 3, 4)]+10
x
# Igualamos las últimas tres entradas a 0.
x[(length(x)-2):length(x)]=0
x
x[length(x)+3]=2
sum(x)
```

```
sum(x, na.rm=TRUE)
mean(x, na.rm=TRUE)
is.na(x)
which(is.na(x))   # Índices de entradas NA.
y=x  # Creamos una copia de x y la llamamos y.
# Cambiamos los NA de y por la media del resto de entradas.
y[is.na(y)]=mean(y, na.rm=TRUE)
y
x[!is.na(x)]
sum(x[!is.na(x)])
cumsum(x)
cumsum(x, na.rm=TRUE)  # Produce error,
# cumsum no admite na.rm.
cumsum(x[!is.na(x)])
na.omit(x)
sum(na.omit(x))
cumsum(na.omit(x))
x_sinNA=na.omit(x)
x_sinNA
attr(x_sinNA, "na.action")=NULL
attr(x_sinNA, "class")=NULL
x_sinNA
Ciudades=c("Madrid","Palma","Madrid","Madrid","Barcelona","Palma","Madrid")
Ciudades
Ciudades.factor=factor(Ciudades)
Ciudades.factor
S=c("M","M","F","M","F","F","F","M","M","F")
Sex=as.factor(S)
Sex
#Esto definirá el mismo factor
Sex2=factor(S)
Sex2
# Si queremos añadir un nuevo nivel.
Sex3=factor(S, levels=c("M","F","B"))
Sex3
Sex4=factor(S, levels=c("M","F","B"),
labels=c("Masc","Fem","Bisex"))
# Si queremos cambiar los nombres
Sex4
levels(Sex)
levels(Sex4)
Notas=factor(c(1,2,2,3,1,3,2,4,2,3,4,2))
Notas
levels(Notas)=c("Muy.mal","Mal","Bien","Muy.bien")
Notas
```

```
Notas_2niv=Notas
levels(Notas_2niv)=c("Mal","Mal","Bien","Bien")
Notas_2niv
Notas=ordered(Notas,
levels=c("Muy.mal","Mal","Bien","Muy.bien"))
Notas
x=c(1,2,-3,-4,5,6)
L=list(nombre="x",vector=x,media=mean(x),sumas=cumsum(x))
L
# Produce error, debe usar nombre del objeto seguido del símbolo
# de dólar $ y el nombre de la variable, como sigue a continuación.
sumas
L$nombre
L$vector
L$media
L$sumas
L[[1]]
# Esto es un vector.
L[[4]]
# Y podemos operar con él.
3*L[[4]]
# Esto es una list, no un vector.
L[4]
# Produce error porque NO podemos operar con él.
3*L[4]
str(L)
names(L)
```

```
# Conocer la dirección de la carpeta de trabajo.
getwd()

## [1] "C:/R/CLASE1R"

# Ajustar la dirección de la carpeta de trabajo.
setwd("C:/R/CLASE1R")
```

```
# lista los archivos con extensión csv
# en la carpeta de trabajo.
list.files(pattern = 'csv')

## [1] "coord_tocumen.csv" "microplastic.csv"

# lista los archivos con extensión xlsx
# en la carpeta de trabajo.
list.files(pattern = 'xlsx')
```

```
## [1] "Datos1.xlsx"

# lista los archivos con extensión txt
# en la carpeta de trabajo.
list.files(pattern = 'txt')

## [1] "latorre_Holdridge.txt" "notas.txt"
```

Prueba estas funciones también:

```
install.packages("soiltexture")
library(soiltexture)
citation()
citation("soiltexture")
sessioninfo::session_info(include_base = TRUE)
sessionInfo()
q() # Lo escribes en la consola para cerrar del programa.
```