

Conociendo R:

Los primeros pasos

Carlos Iván Espinosa

noviembre 2020

Contents

Por qué usar R	1
Instalando R y RStudio	1
Los primeros pasos	4
Primer Paso: Crear un proyecto	4
El funcionamiento de R	5
Uso de la memoria activa	6
Uso del disco duro	10
Uso de la Web	11
Regresar Introducción R	

Puedes descargar esta lección en pdf aquí

Por qué usar R

Open source (fuente abierta): R es un lenguaje open source, **GRATUITO**

Desarrolladores: Tiene una cantidad enorme de desarrolladores, lo cual permite tener información y paquetes de cualquier tipo de análisis desde SIG hasta análisis de grafos.

Orientado a gráficos y análisis estadísticos: R es un lenguaje que permite desarrollar gráficos y análisis estadísticos. Tiene capacidad para desarrollar pre-procesamiento de datos.

Lenguaje de programación: El hecho de que R sea un lenguaje de programación en lugar de una colección de comandos discretos significa que puede combinar varios comandos, cada uno usando la salida del último, con la combinación resultante siendo bastante poderosa y extremadamente flexible.

Instalando R y RStudio

El primer paso será instalar y . Aunque existen muchas plataformas para trabajar con R, RStudio ofrece algunas ventajas que las analizaremos más adelante, por las que seleccionamos esta plataforma para trabajar.

Para instalar R ingresa en la página web de r-project. Una vez en esta página selecciona descargar (Figura 1). Sigue los pasos para terminar la instalación.

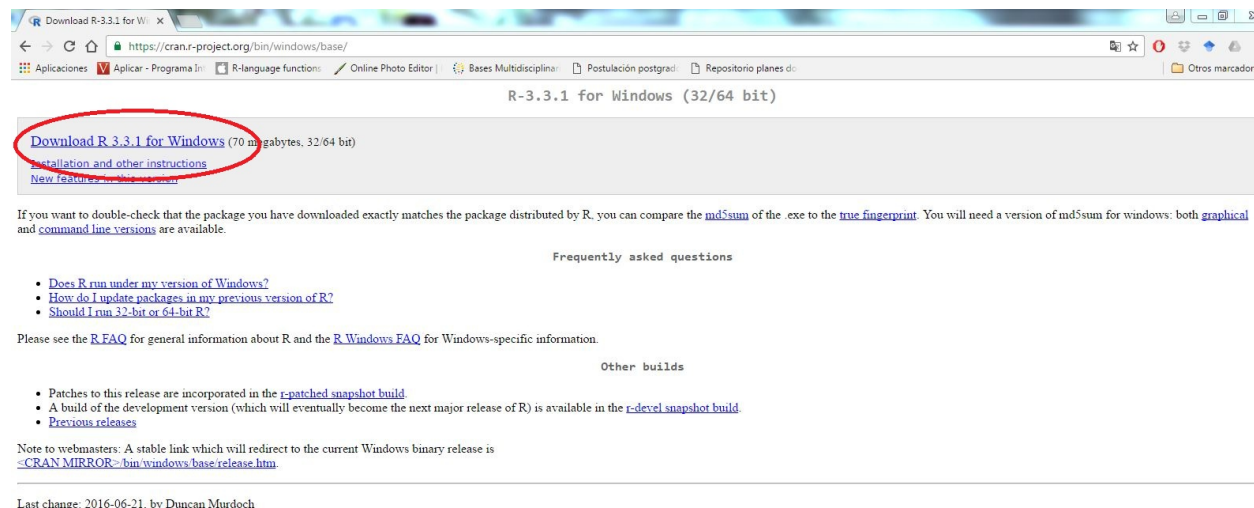


Figure 1: Figura1. Descargando R

La instalación de RStudio es similar a R, debe ingresar en la página de RStudio. Una vez en la página de descarga, seleccione la plataforma acorde a su sistema operativo (Figura 2) y siga los pasos para terminar la instalación.

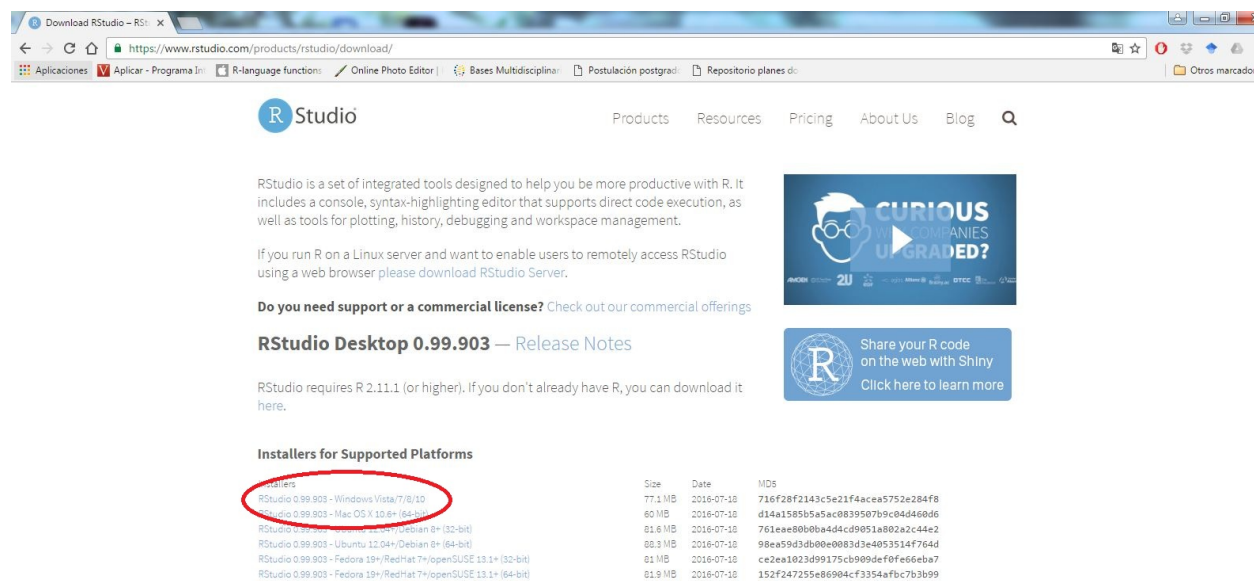


Figure 2: Figura2. Descargando RStudio

Una vez que hemos instalado R y RStudio podremos trabajar con los datos. Una cosa importante es que RStudio no puede funcionar si no hemos descargado R, así que se debe asegurar descargar los dos programas. Trabajaremos en RStudio así que no es necesario que despliegue R.

Nota: “El uso de comandos exige una curva de aprendizaje mayor que el requerido por las interfaces gráficas, pero las ganancias en términos de independencia, creatividad y control no son

comparables. Escribir un código supone una comprensión más profunda de aquello que se desea aplicar” (Elousa 2011).

#Conozcamos RStudio antes de empezar

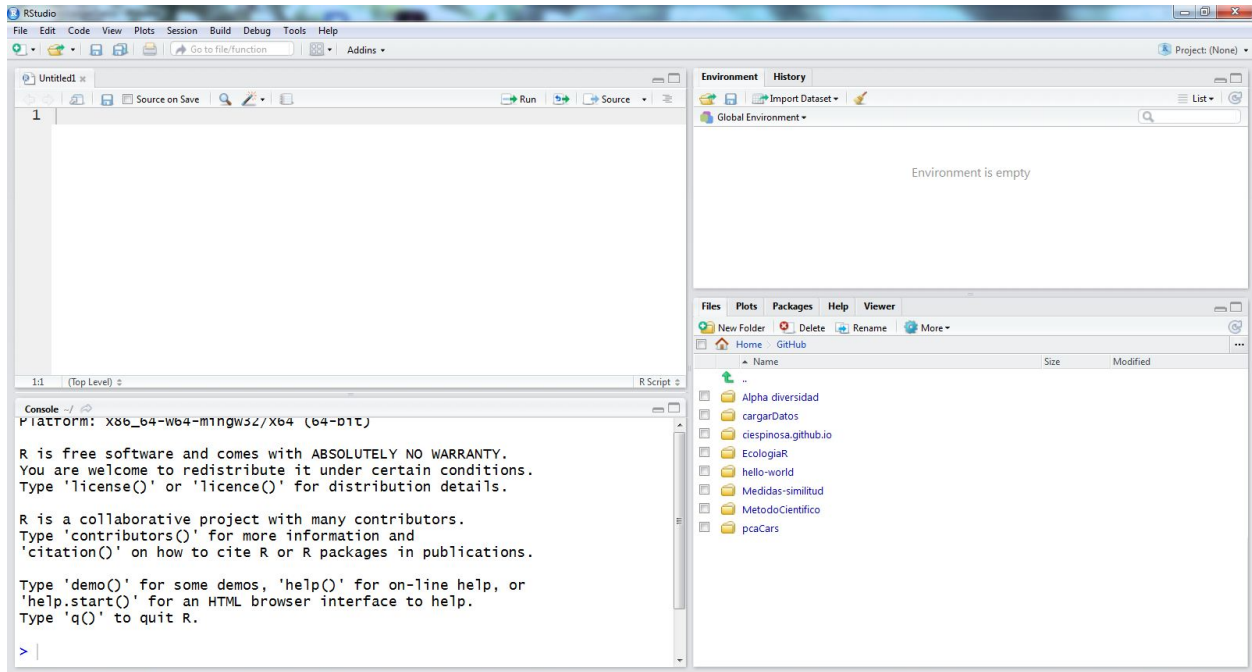



Figure 3: Figura 4. Plataforma RStudio

Como vemos en la figura 4, RStudio está compuesto por cuatro ventanas. Seguramente en su caso, si acaba de abrir RStudio le aparecerán únicamente tres ventanas. A continuación, describiré cada una de las ventanas.

La **primera ventana** (la que en su caso seguramente no tiene, izquierda superior) lo constituye documentos que pueden ser de varios tipos. El tipo básico es un documento con extensión **.R**, el cual sirve para guardar el código desarrollado para el análisis. Vamos a abrir un documento **.R** (script), esto lo podemos hacer de al menos tres maneras, la primera es ir al menú de la consola seleccionar **File > New File > script**, la

segunda forma es seleccionar el icono del documento con una cruz verde  y seleccionar **R script**. La última opción es hacerlo desde el teclado, presione **alt + shift** (mayúsculas) y la tecla **N** obtendrá el mismo resultado.

Existen otros muchos archivos que podemos cargar, pero por ahora veremos solo este.

La **segunda ventana** (derecha superior), esta ventana se verán todos los objetos que iremos cargando o generando durante el trabajo en RStudio, por ahora esta ventana estará vacía. Esta ventana nos muestra la memoria activa de R, lo que en un computador se conoce como la memoria RAM.

Vamos a generar algunos objetos y ver lo que pasa.

```
nombre<- "Carlos Ivan"
apellido<- "Espinosa Iñiguez"
matriz<- matrix(1:20, 5,4)
```

Ahora podemos ver los objetos creados, algunos salen como valores y la matriz sale como datos. Los objetos pueden ser abiertos para ver su estructura. Si hacen clic en el nombre matriz verán que se abre una nueva hoja en la primera ventana que corresponde a estos datos.

La **tercera ventana** (izquierda abajo), corresponde a la consola de R, esta es la consola donde se ejecutarán todos los códigos y se realizarán los análisis. Esta ventana es R. RStudio incorpora a R dentro de su ejecución, por lo que si no hemos instalado R esta ventana no aparecerá.

La **cuarta ventana** (derecha abajo), en esta ventana tenemos varias pestañas. La primera **File** nos muestra todos los archivos que están en la carpeta de mi proyecto, guardados en el disco duro. Pueden ser archivos de datos, gráficos generados o el script de R que estoy generando. La siguiente pestaña **Plots** mostrará los gráficos que va ejecutando en R. La pestaña **Help** puede ser usada para pedir ayuda de algún paquete o función que necesite usar, y no esté claro de cómo hacerlo.

Bueno ya conocemos RStudio ahora si a trabajar.

Nota: El trabajar con códigos requiere que seamos ordenados y sistemáticos, de no cumplir estos requerimientos tendrá un verdadero dolor de cabeza. RStudio es una interesante plataforma que nos permite organizar el trabajo. Siempre que inician un trabajo de análisis siga los pasos propuestos a continuación.

Los primeros pasos

El trabajo de programación requiere ser ordenados, considero que trabajar con RStudio nos ofrece algunas ventajas para organizar ese trabajo.

Primer Paso: Crear un proyecto

Crear un proyecto implica definir un directorio desde el cual R funcionará. Este proceso creará una carpeta a la cual R estará anclado. Además, en esta carpeta encontrará un archivo `.Rproj` desde el cual podrá abrir RStudio.

Para crear un nuevo proyecto en RStudio debemos seguir los siguientes pasos.

1. Abrir RStudio
2. Hacer clic en **file** y seleccionar **new Project**
3. En la nueva ventana que se abrió, podemos seleccionar entre tres opciones, puesto que aún no estamos trabajando con versiones de control, por ahora podemos seleccionar entre nuevo directorio **New Directory** o un directorio existente **Existing Directory**.

Si seleccionamos **New Directory** (nuevo directorio) se generará una nueva carpeta en la ubicación que nosotros definamos, en esta carpeta podremos poner todos los datos y demás información con la cual trabajaremos. Cuando seleccionamos **Existing Directory** (un directorio existente) RStudio generará un proyecto dentro de una carpeta que ya exista en el computador. Por organización, siempre que estoy iniciando un nuevo proyecto prefiero crear un nuevo directorio en el cual colocaré únicamente los archivos que voy a utilizar en los análisis.

4. Si hemos elegido **New Directory** tendremos dos casilleros, el primero indica el nombre que le vamos a poner a la carpeta **Directory Name** y el segundo nos indica donde alojaremos esa carpeta **browse**.
5. Si hemos elegido **Existing Directory**, la siguiente ventana nos permite decir cuál es la carpeta que quiero enlazar. Hacer clic en **browse** buscar la carpeta donde colocaré el proyecto y aceptar.

##Segundo Paso: Los códigos usados

Si bien podemos trabajar directamente en la consola de R para ejecutar los códigos, lo mejor es que desde el principio nos acostumbremos a generar scripts, donde tengamos la información limpia y podamos saber lo que estamos haciendo. Un Script es un archivo donde tendremos los códigos de R, referenciados y organizados.

Algunos consejos iniciales

- Todos los códigos usados para realizar los diferentes análisis, deben siempre ir acompañados de una nota que explique lo que están haciendo. La nota debe estar precedida por `#`, con lo cual R no lee esta parte como código (Figura 4).
- Recuerden siempre colocar sus archivos de datos en la carpeta del proyecto que generaron en el primer paso.
- Si está probando cambios en el código, una vez que tiene un código que funciona, borre el código con error o que no le funcionó, tenga siempre su código limpio.
- Una vez que escribe el código, este puede ser ejecutado desde la consola haciendo clic en la consola en



el ícono run que se encuentra en la parte superior derecha de la ventana del script. Sin embargo, una mejor forma es tecleando **ctrl** y **enter**.

- Cada vez que en el código iniciamos un nuevo tema, podemos poner un título seguido por cuatro guiones medios (- - - -), esto genera en RStudio una estructura de índice que puede ser navegada (Figura 5 círculo rojo abajo a la izquierda).

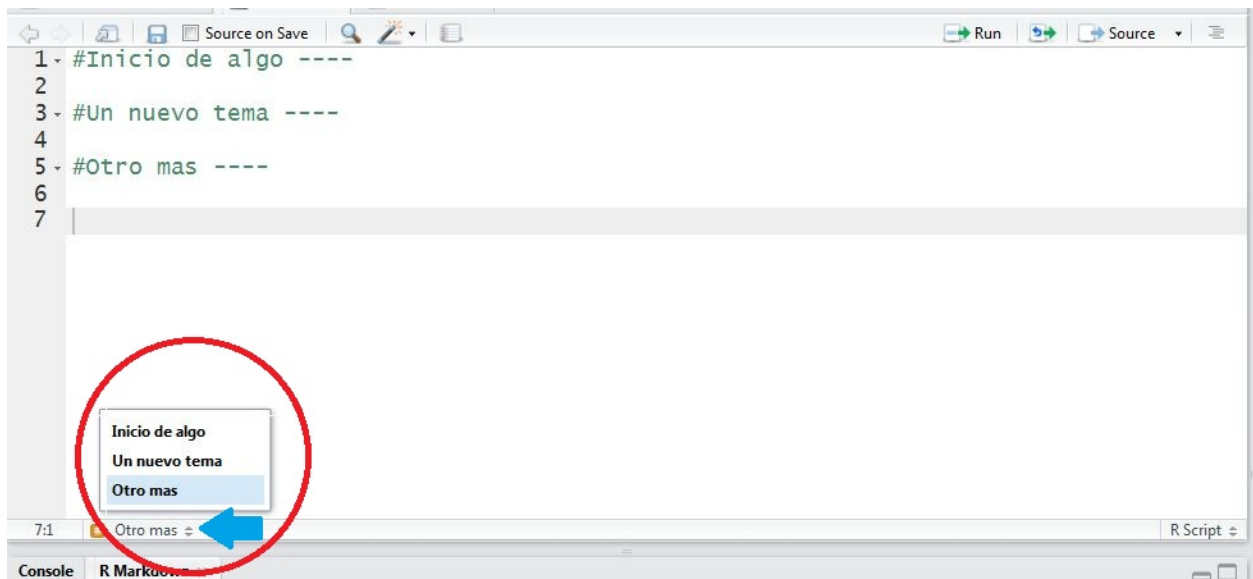


Figure 4: Figura 5. Estructura del script en RStudio

Nota: Es importante que sea organizado en la ejecución de los códigos, siga los consejos y podrá mantener un código limpio y facil de acceder y revisar.

El funcionamiento de R

Hemos hablado mucho de R pero hasta ahora no hemos dicho que es, ¿es un programa?, no, realmente no es un programa, es un entorno de programación. R es considerado un dialecto del lenguaje *S* el cual fue desarrollado por los Laboratorios AT&T Bell.

Aunque mucha gente se asusta cuando hablamos de R, como un lenguaje de programación, la realidad es que R es un lenguaje bastante simple, está orientado a *Objetos*. Por otro lado, a diferencia de otros lenguajes de programación los comandos escritos en el teclado, son ejecutados directamente sin necesidad de construir ejecutables.

En R tenemos al menos tres grandes categorías de objetos; funciones, datos y resultados. Cada una de estas tiene unas características propias. Las *funciones* normalmente se encuentran dentro de paquetes, estos objetos traen comandos que permiten manipular los datos. Accedemos a las funciones a través de comandos. Los *datos* son matrices o vectores de información los cuales son manipulados por las funciones. Los *resultados* son objetos resultantes de la manipulación de los datos (Figura 3).

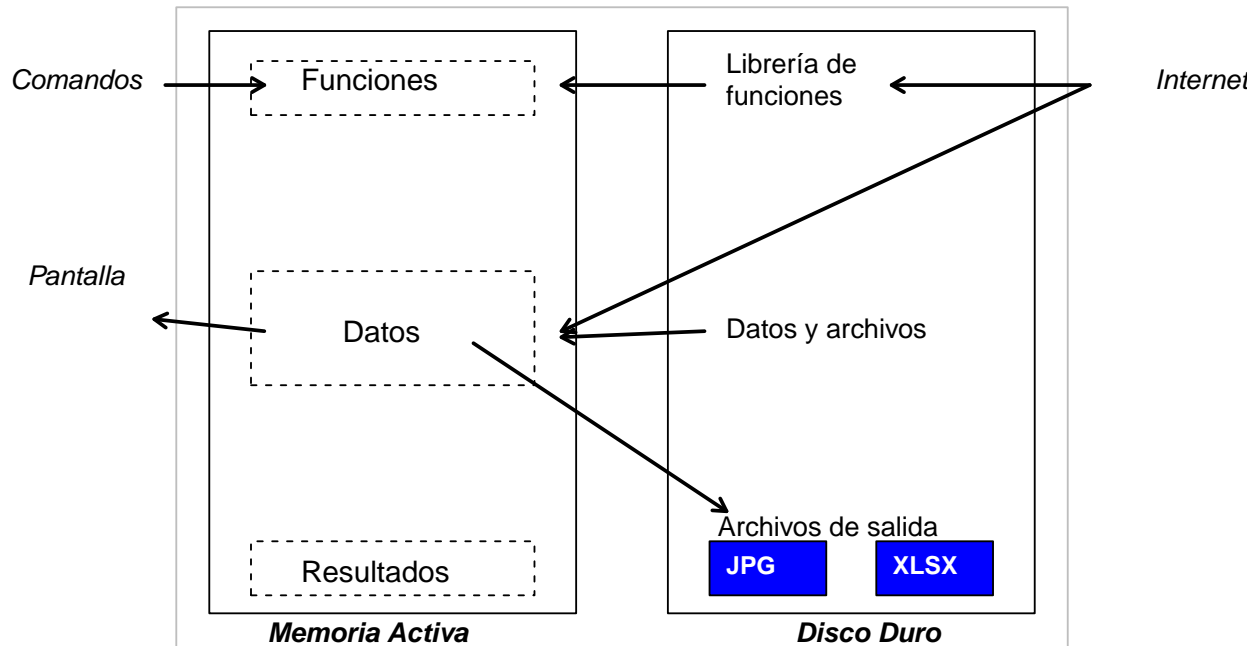


Figure 5: Figura 3. Esquema del funcionamiento de r. Basado en Paradis 2003

El funcionamiento de R se da en la memoria activa, así que cada vez que iniciamos a trabajar con R es necesario llamar los datos desde el disco duro a la memoria activa. Para llamar paquetes (donde tenemos las funciones) normalmente utilizamos la función `library`, mientras que para llamar los datos utilizamos funciones como `read.table`. Algunas veces nos interesa grabar un resultado desde la memoria activa al disco duro para esto podemos utilizar funciones como `write.table`. Finalmente, no todos los paquetes disponibles se bajan cuando instalamos R, de hecho, solo se baja un paquete conocido como paquete base. Cuando se necesita un nuevo paquete, este debe llamarse desde internet, para ello utilizamos la función `install.packages`. Existe un repositorio de todos los paquetes disponibles (Comprehensive R Archive Network, **CRAN**) y varios países tienen espejos de estos repositorios (espejos CRAN) a partir de los cuales podemos descargar los paquetes.

Nota: Es muy importante que antes de empezar a trabajar con R seamos conscientes de cómo funciona, esto facilitará entender lo que está haciendo cuando introduce un código.

Uso de la memoria activa

Vamos a trabajar en R

1. Abra RStudio y cree un proyecto
2. Genere un Script para escribir el código.
3. Ahora a trabajar

Ejecución de comandos e impresión

Cuando estamos trabajando con R, nosotros podemos desarrollar una serie de acciones u operaciones que se ejecutan en la consola, luego de lo cual el resultado puede ser impreso en la consola. Veamos unos ejemplos.

```
#Ejecución de comandos e impresión
```

```
pi*3
```

```
## [1] 9.424778
```

```
log(234)
```

```
## [1] 5.455321
```

Cuando nosotros introducimos el comando en R y lo ejecutamos, podemos ver la salida de esa expresión, R imprime el resultado directamente en la consola. Podemos usar la función `print()` para pedir a R que imprima el resultado.

```
##La función de impresión
```

```
print(pi*3); print(pi*3, digits = 4) #El ; nos permite separar dos líneas de código
```

```
## [1] 9.424778
```

```
## [1] 9.425
```

```
print(log(234), digits = 10)
```

```
## [1] 5.455321115
```

Como vemos con el uso de la función `print()` tenemos la misma salida, imprime el resultado en la consola. Sin embargo, nos permite cambiar ciertas propiedades de lo que se imprime, en este caso controlar el número de decimales. Cuando trabajamos con tablas, la función `print` nos permite controlar la salida de esas tablas.

```
##Controlar la salida de tablas
```

```
print(as.table(matrix(c(5:0,1,0,0),3,3)))
```

```
##   A B C
```

```
## A 5 2 1
```

```
## B 4 1 0
```

```
## C 3 0 0
```

```
print(as.table(matrix(c(5:0,1,0,0),3,3)), zero.print = ".")
```

```
##   A B C
```

```
## A 5 2 1
```

```
## B 4 1 .
```

```
## C 3 . .
```

En este caso el argumento `zero.print` nos permite decirle que queremos substituir los seros por un punto podemos cambiar el punto por un “-”. Inténtalo y mira lo que obtienes.

Aunque la función `print` nos permite tener cierto control sobre lo que imprimimos en la consola, esta función solo puede ser aplicada a una expresión a la vez. Cuando queremos que se imprima una expresión compuesta por varias expresiones, podemos usar la función `cat()`

```
##La función cat para imprimir vectores
```

```
cat("Juan tiene", 17*pi , "años", "...\\n")
```

```
## Juan tiene 53.40708 años ...
```

La función **cat** permite imprimir varias cosas juntas, intente usar la expresión `print` para ejecutar el mismo vector. ¿Qué sucedió?. El argumento final “...-n” nos dice como debería terminar el vector. Remplace los puntos por asteriscos.

Hasta aquí, todos los códigos que hemos ejecutado se han impreso en la consola pero no constan en ningún lado, todos los resultados de esos códigos no pueden ser usados para realizar ninguna otra acción. Hemos usado R como una calculadora, la cual nos da un resultado, pero no podemos volver a acceder a ese resultado de ninguna forma.

Para acceder a los resultados necesitamos asignar el resultado a un símbolo, un nombre.

Asignación y generación de objetos

El trabajo que hicimos en la sección anterior no generó objetos, como había comentado los objetos son las estructuras sobre las cuales R opera. Podemos generar objetos usando la función `<-`. Existen otras formas de asignar, como “=”, o “->”, sin embargo, es mejor no usarlas puesto que por ejemplo el “=” puede ser confundida con la expresión *igual que*. En el caso de la función “->” si bien cuando generamos un objeto sencillo puede usarse, cuando estamos generando una expresión más compleja es muy difícil entender si al final está la asignación. Por estas razones es mejor que siempre use “<-”.

Trabajemos nuevamente en R

```
##Asignando resultados a símbolos
set.seed(23)
edad <- round(rnorm(5, 25,5),2)
```

Un momento, algo paso no veo el resultado.

Puede chequear el “Environment” en RStudio, ¿qué puede ver ahí?. Efectivamente, aparece el nombre `edad`. Hemos generado un objeto que está representado por el símbolo `edad`

Usar la función asignar “<-”, y hemos asignado el resultado de nuestro código a un símbolo (nombre). Realmente es un símbolo, R reconoce los símbolos, es por eso que **Edad** no es lo mismo que **edad**

Ahora, ¿cómo imprimimos el resultado de `edad`?. Podemos usar las mismas funciones que usamos antes para imprimir (`print` y `cat`) o teclear directamente el nombre.

```
#Imprimir los objetos
print(edad)
```

```
## [1] 25.97 22.83 29.57 33.97 29.98
```

```
edad
```

```
## [1] 25.97 22.83 29.57 33.97 29.98
```

Vamos a jugar un poco. Usaremos la función `cat()` para imprimir un objeto que tiene varios elementos, imprimiremos cada uno de los elementos que están asociados. En este caso tenemos un nombre de una persona y la edad de esas personas. Para esto usamos una función que genera un loop, en este caso la función `for()`, esta función permite definir un símbolo en este caso “i” y darle un valor secuencial que yo defino. Así donde está “i” R generará un ciclo en el cual i toma el valor de 1, en la siguiente vuelta 2 y así sucesivamente hasta 5.

```
#Vector de nombres

nom <- c("Juan", "Pedro", "Ana", "Sol", "Juliana")

#Usamos la función cat para imprimir
```



```
for(i in 1:length(nom)) cat("La edad de", nom[i], "es", edad[i], "...\\n")
```

```
## La edad de Juan es 25.97 ...
## La edad de Pedro es 22.83 ...
## La edad de Ana es 29.57 ...
## La edad de Sol es 33.97 ...
## La edad de Juliana es 29.98 ...
```

En esta segunda parte hemos trabajado en la memoria activa de la computadora la memoria RAM. Cuando define una variable en el símbolo del sistema como en este caso “edad”, la variable se mantiene en su espacio de trabajo (Workspace). El espacio de trabajo se mantiene en la memoria activa de la computadora, pero se puede guardar en el disco cuando sale de R. La definición de la variable permanece en el espacio de trabajo hasta que la elimina. Podemos ver los objetos generados en el “Environment” de RStudio.

Aunque, RStudio me permite ver el workspace, podemos pedirle a R que nos muestre lo que tiene en su “workspace” usando la función `ls()`.

```
#Listado del workspace
ls()
```

```
## [1] "apellido" "edad"      "i"          "matriz"    "nom"       "nombre"    "x"
## [8] "y"
```

Podemos pedir que nos diga las características de esos objetos

```
#listado con características
ls.str()
```

```
## apellido : chr "Espinosa Iñiguez"
## edad : num [1:5] 26 22.8 29.6 34 30
## i : int 5
## matriz : int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
## nom : chr [1:5] "Juan" "Pedro" "Ana" "Sol" "Juliana"
## nombre : chr "Carlos Ivan"
## x : int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
## y : int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
```

Eliminando objetos

Puede ser que necesite borrar algunos objetos que ya no los ocuparé, para ello usaremos la función `rm()`. Veamos como lo podemos hacer.

```
ls(); rm(i); ls()
```

```
## [1] "apellido" "edad"      "i"          "matriz"    "nom"       "nombre"    "x"
## [8] "y"

## [1] "apellido" "edad"      "matriz"    "nom"       "nombre"    "x"         "y"
```

Como ven en la primera salida de nuestro workspace consta “i”, pero la segunda vez que ejecutamos `ls` el objeto “i” ya no aparece. Podemos usar la expresión `rm(list=ls())` para borrar todos los objetos en el workspace o usar la expresión `rm(list=setdiff(ls(), c(“edad”, “nom”)))`, para eliminar todo aquello que no sea “edad” ni “nom”

Uso del disco duro

Hasta ahora hemos trabajado en la memoria activa, eso quiere decir, que cuando cerremos R todos los objetos que hemos desarrollado se pierden. Es posible que eso sea lo que quiero, pero seguramente hay resultados que me gustaría grabar en mi disco duro para poder acceder a ellos.

A diferencia de programas tradicionales, para grabar los datos necesito usar una función y no un menu de ventanas.

Guardando datos

```
sink("names_output.txt")
for(i in 1:length(nom)) cat("La edad de", nom[i], "es", edad[i], "...\\n")
sink()
```

La función **sink** lo que hace es redireccionar la salida a un archivo .txt Debemos volver a correr la función **sink** para que se cierre la conexión con el archivo.

Podríamos generar un reporte con los datos y los resultados que hemos obtenido utilizando la función **sink**.

```
dta <- data.frame(edad, nom, gen=c(rep("Hombres",2), rep("Mujeres",3)))
resul <- tapply(dta$edad, dta$gen, mean)
```

```
sink("reporte1.txt")

print(resul)

summary(dta)
cat("Conclusión: La mayor media de edad se presenta en",
    names(resul[resul==max(resul)]), "(", max(resul), ")",
    ", mientras que la menor edad en",
    names(resul[resul==min(resul)]), "(", min(resul), ")")
sink()
```

Para guardar las tablas podemos usar la función **write.csv**. Veamos.

```
write.csv(dta, file="datosEdad.csv", row.names=FALSE)
```

Ahora podemos ver los archivos que se han generado usando la función **dir**

```
dir()

## [1] "ConociendoR.Rmd"      "ConociendoR.Rproj"  "datosEdad.csv"
## [4] "imagen"              "index.html"         "index.pdf"
## [7] "index.Rmd"           "index_files"        "LICENSE"
## [10] "names_output.txt"    "reporte1.txt"       "Rlogo.png"
## [13] "RStudio-Logo.png"
```

Leyendo los datos

Como hemos visto hasta ahora, R trabaja con unos datos los puede procesar y modificar, todo esto usando la memoria activa (workspace), sin embargo, esa memoria es temporal, por lo que si queremos que algo de lo que hemos procesado se mantenga como un archivo, necesito decirle a R que lo guarde en mi disco duro.

La información en el disco duro puede ser llamada al workspace de R, pero nuevamente, a diferencia de cualquier otro programa, este paso lo debemos hacer usando una función.

Vamos a volver a leer los datos que habíamos generado en R y guardado en el disco duro. Tenemos dos tipos de archivos un *.txt* y un *.csv*, para cada uno de ellos puedo usar una función. Veamos

```
dta2 <- read.csv("datosEdad.csv", header=T)
edad2 <- read.table("names_output.txt")
```

```
dta2; edad2
```

```
##      edad      nom      gen
## 1 25.97      Juan Hombres
## 2 22.83      Pedro Hombres
## 3 29.57       Ana Mujeres
## 4 33.97       Sol Mujeres
## 5 29.98 Juliana Mujeres

##   V1   V2 V3      V4 V5   V6  V7
## 1 La edad de      Juan es 25.57 ...
## 2 La edad de      Pedro es 22.06 ...
## 3 La edad de      Ana es 26.82 ...
## 4 La edad de      Sol es 25.93 ...
## 5 La edad de Juliana es 25.55 ...
```

Leyendo desde una url

Una ventaja de R es que podemos cargar los datos desde cualquier fuente, por ejemplo desde una página web. Vamos usar unos datos subidos en la página: datos

```
url <- "https://raw.githubusercontent.com/Ciespinosa/datos_practicas/master/AMEBIASIS_LOJA.csv"
```

```
dtaAm <- read.csv2(url, header = TRUE, sep = ",")
```

```
head(dtaAm, 5)
```

```
##   Cantón Distrito  Sexo Edad.en.años Consultas      Parroquia
## 1   LOJA    11D01 Hombre          1          1    CHUQUIRIBAMBA
## 2   LOJA    11D01 Hombre         13          2    CHUQUIRIBAMBA
## 3   LOJA    11D01 Hombre         14          1    CHUQUIRIBAMBA
## 4   LOJA    11D01 Hombre          2          1    CHUQUIRIBAMBA
## 5   LOJA    11D01 Hombre          2          1 TAQUIL (MIGUEL RIOFRÍO)
```

¿En donde estan los datos?

Se encuentra en la memoria activa. Si apago R los datos se perderán, debo guardarlos en el disco duro. Prueben grabar los datos en el disco duro.

Hasta ahora hemos trabajado en la **consola**, cuando usabamos R como una calculadora, usando la **memoria activa**, asignando a un símbolo (nombre) unos datos, y finalmente, el disco duro cuando leemos o grabamos unos datos en el disco duro.

Aunque si lo pensamos bien, hemos usado tambien el **internet** cuando leímos datos de la red.

Uso de la Web

R puede usar el **internet** para; i) descargar datos, ii) subir datos y iii) descargar paquetes. Hay muchas otras aplicaciones para descargar datos de la web, además, podemos usar R para subir datos a bases de datos o sencillamente hacer una pagina web. Por ahora al menos quiero que sepan que es posible hacerlo, aunque esto sería otro curso.

Los paquetes y su instalación

Los paquetes es como R organiza las funciones que nos permiten desarrollar diferentes acciones. Así, muchas funcione de estadística básica como; la media la desviación estándar, entre otras se encuentran dentro de un paquete que se denomina paquete base.

Cuando instalamos R, algunos paquetes son instalados por defecto, pero R tiene miles de paquetes así que si nosotros necesitamos trabajar con un paquete más específico lo debemos instalar.

Instalando paquetes Nuevamente recuerde R no funciona con pestañas por lo que instalar un paquete se lo debe hacer usando código. En este caso se usa la función `install.packages()`

```
#install.packages("xlsx")
```

Esta función nos permite descargar del internet este paquete y grabarlo en nuestro disco duro. Este paquete tiene varias funciones para leer, editar y escribir archivos .xlsx.

Para llevar del disco duro a nuestra consola, que es donde usaremos el paquete, es necesario usar una nueva función en este caso la función `library(xlsx)`.

La instalación de los paquetes se debe hacer unicamente una vez, cuando hemos descargado el paquete este esta disponible en nuestro disco duro así que podemos llamarlo desde ahí cada vez que lo necesitemos.