R: lo básico

El espacio de trabajo (Workspace)

Directorio de trabajo

El directorio de trabajo o *working directory* es el folder en tu computadora en el que estás trabajando en ese momento. Cuando se le pide a R que abra un archivo o guarde ciertos datos, R lo hará a partir del directorio de trabajo que le hayas fijado.

Para saber en qué directorio te encuentras, se usa el comando `getwd()`.

```
getwd()
```

```
## [1] "/home/animalito/study/aprendeR/lecture_01"
```

Para especificar el directorio de trabajo, se utiliza el comando `setwd()` en la consola. Y volvemos a

```
setwd("/home/animalito/study/")
getwd()
```

Con lo que acabamos de hacer, R buscará archivos o guardará archivos en el folder `/home/animalito/study/`. En R también es posible navegar a partir de el directorio de trabajo. Como siempre,

- "../un_archivo.R" le indica a R que busque un folder arriba del actual directorio de trabajo por el archivo *un_archivo.R*.
- "datos/otro_archivo.R" hace que se busque en el directorio de trabajo, dentro del folder *datos* por el archivo *otro_archivo.R*

Ejemplos básicos

La consola permite hacer operaciones sobre números o caracteres (cuando tiene sentido).

```
# Potencias, sumas, multiplicaciones
2^3 + 67 * 4 - (45 + 5)
```

```
## [1] 226
```

```
# Comparaciones
56 > 78
```

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

```
34 <= 34
```

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

```r
234 < 345
```

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

```r
"hola" == "hola"
```

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

```r
# modulo
10 %% 4
```

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

Estas operaciones también pueden ser realizadas entre vectores

```r
x <- -1:12
x
```

```
##  [1] -1  0  1  2  3  4  5  6  7  8  9 10 11 12
```

```r
x + 1
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13
```

```r
2 * x + 3
```

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

```r
x %% 5 #-- is periodic
```

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

```r
x %/% 5
```

```
##  [1] -1  0  0  0  0  0  1  1  1  1  1  2  2  2
```

Comandos útiles

Para enlistar lso objetos que están en el espacio de trabajo

```r
ls()
```

```
## [1] "x"
```

Para eliminar todos los objetos en un workspace

```r
rm(list = ls()) # se puede borrar solo uno, por ejemplo, nombrándolo
ls()
```

```
## character(0)
```

También se puede utilizar/guardar la historia de comandos utilizados

```r
history()
history(max.show = 5)
history(max.show = Inf) # Muestra toda la historia

# Se puede salvar la historia de comandos a un archivo
savehistory(file = "mihistoria") # Por default, R ya hace esto
# en un archivo ".Rhistory"

# Cargar al current workspace una historia de comandos en particular
loadhistory(file = "mihistoria")
```

Es posible también guardar el workspace -en forma completa- en un archivo con el comando `save.image()` a un archivo con extensión *.RData*. Puedes guardar una lista de objetos específica a un archivo *.RData*. Por ejemplo:

```r
x <- 1:12
y <- 3:45
save(x, y, file = "ejemplo.RData") #la extensión puede ser arbitraria.
```

Después puedo cargar ese archivo. Prueba hacer:

```r
rm(list = ls()) # limpiamos workspace
load(file = "ejemplo.RData") #la extensión puede ser arbitraria.
ls()
```

Nota como los objetos preservan el nombre con el que fueron guardados.

Librerías

R puede hacer muchos análisis estadísticos y de datos. Las diferentes capacidades están organizadas en paquetes o librerías. Con la instalación estándar se instalan también las librerías más comunes. Para obtener una lista de todos los paquetes instalados se puede utilizar el comando `library()` en la consola.

Existen una gran cantidad de paquetes disponibles además de los incluidos por default.

CRAN

CRAN o *Comprehensive R Archive Network* es una colección de sitios que contienen exactamente el mismo material, es decir, las distribuciones de R, las extensiones, la documentación y los binarios. El master de CRAN está en Wirtschaftsuniversität Wien en Austria. Éste se "espeja" (*mirrors*) en forma diaria a muchos sitios alrededor del mundo. En la lista de espejos se puede ver que para México están disponibles el espejo del ITAM, del Colegio de Postgraduados (Texcoco) y Jellyfish Foundation.

Los espejos son importantes pues, cada vez que busquen instalar paquetes, se les preguntará qué espejo quieren utilizar para la sesión en cuestión. Del espejo que selecciones, será del cuál R *bajará* el binario y la documentación.

Del CRAN es que se obtiene la última versión oficial de R. Diario se actualizan los espejos. Para más detalles consultar el FAQ.

Para contribuir un paquete en CRAN se deben seguir las instrucciones aquí.

Github

Git es un controlador de versiones muy popular para desarrollar software. Cuando se combina con GitHub se puede compartir el código con el resto de la comunidad. Éste controlador de versiones es el más popular entre los que contribuyen a R. Muchos problemas a los que uno se enfrenta alguien ya los desarrolló y no necesariamente publicó el paquete en CRAN. Para instalar algún paquete desde GitHub, se pueden seguir las instrucciones siguientes

```
install.packages("devtools")
devtools::install_github("username/packagename")
```

Donde `username` es el usuario de Github y `packagename` es el nombre del repositorio que contiene el paquete. Cuidado, no todo repositorio en GitHub es un paquete. Para más información ver el capítulo Git and GitHub en Wickham (2015).

Otras fuentes

Otros lugares en donde es común que se publiquen paquetes es en Bioconductor un projecto de software para la comprensión de datos del genoma humano.

Paquetes recomendados

Hay muchísimas librerías y lo recomendable es, dado un problema y un modelo para resolverlo, revisar si alguien ya implementó el método en algunas de las fuentes de paquetes mencionadas antes. Para una lista de paquetes que son de mucha utilidad ver estas recomendaciones.

Scripting

R es un intérprete. Utiliza un ambiente basado en línea de comandos. Por ende, es necesario escribir la secuencia de comandos que se desea realizar a diferencia de otras herramientas en donde es posible utlizar el mouse o menús.

Aunque los comandos pueden ser ejecutados directamente en consola una única vez, también es posible guardarlos en archivos conocidos como *scripts*. Típicamente, utilizamos la extensión **.R** o **.r**. En RStudio, `CTRL + SHIFT + N` abre inmediatamente un nuevo editor en el panel superior izquierdo.

Se puede *ir editando* el script y corriendo los comandos línea por línea con `CTRL + ENTER`. Esto también aplica para *correr* una selección del texto editable.

Es posible también correr todo el script

```r
source("foo.R")
```

O con el atajo `CTRL + SHIFT + S` en RStudio.

Para enlistar algunos shortcuts comunes en RStudio presiona `ALT + SHIFT + K`. De la misma manera, si utilizas Emacs + ESS, existen múltiples atajos de teclado para realizar todo mucho más eficientemente. Estudiarlos no es tiempo perdido.

Ayuda & documentación

R tiene mucha documentación. Desde la consola se puede accesar a la misma.

Para ayuda general,

```r
help.start()
```

Para la ayuda de una función en especifico, por ejemplo, si se quiere graficar algo y sabemos que existe la funcion `plot` podemos consultar fácilmente la ayuda.

```r
help(plot)
# o tecleando directamente
?plot
```

El segundo ejemplo se puede extender para buscar esa función en todos los paquetes que tengo instalados en mi ambiente al escribir `??plot`.

La documetnación normalmente se acompaña de ejemplos. Para *correr* los ejemplos sin necesidad de copiar y pegar, prueba

```r
example(plot)
```

Para búsquedas más comprensivas, se puede buscar de otras maneras:

```r
apropos("foo") # Enlista todas las funciones que contengan la cadena "foo"
RSiteSearch("foo") # Busca por la cadena "foo" en todos los manuales de ayuda
# y listas de distribución.
```

Estructuras de datos

Vectores

Matrices

Data frames

Listas

Leer y escribir archivos de datos

Lectura

Desde archivo

Datos de muestra

Escritura

Estructuras de programación

IFs

Fors

Whiles

Funciones propias

Referencias

[Wic15] Hadley Wickham. *R packages*. "O'Reilly Media, Inc.", 2015.