

RLadies Cuernavaca, Meetup R y RStudio

Leticia Vega Alvarado

12 Jan 2022

Índice

	Objetivo	3
1	Introducción	3
1.1	¿Qué es R?	3
1.2	Historia	3
1.3	Distribuciones de R	4
1.4	Instalando R	4
1.5	¿Qué es RStudio?	4
1.5.1	Distribuciones	5
1.5.2	Instalación	5
2	Primeras nociones, un tour por RStudio	5
2.1	Ambiente de trabajo	5
2.2	Consola	6
2.3	Moviéndose entre los directorios	6
2.3.1	¿Cuál es mi directorio de trabajo?	7
2.3.2	¿Cómo cambio de directorio?	7
2.3.3	Consultando el contenido de un directorio	7
2.4	Paquetes de R	8
2.4.1	Visualizando los paquetes instalados	8
2.4.2	Cargando un paquete	8
2.4.3	Visualizando las funciones de un paquete determinado	9
2.4.4	Instalando paquetes	9
2.5	Consultando la ayuda	9
2.6	Sección de visualización de gráficas	10
2.7	<i>Environment</i>	11
2.8	<i>History</i> , comandos creados en una sesión	12
2.9	Saliendo de R	12
3	Apéndice 1	13
3.1	Iniciando R desde línea de comandos	13
3.2	Manejo de paquetes desde línea de comandos	13

3.3	Consultando la ayuda desde línea de comandos.	14
3.3.1	Ayuda en general	15
3.3.2	Ayuda relacionada a un término	15
3.3.3	Ayuda sobre ejemplos de uso de una función	16
3.3.4	Ayuda sobre argumentos de una función	17
3.3.5	Búsquedas aproximadas	17
3.4	Guardando y Cargando el Espacio de Trabajo desde línea de comandos	17
3.4.1	Almacenando, en un archivo, objetos específicos creados en una sesión	17
3.5	Guardando y Cargando el Historial de Comandos desde línea de comandos	18

Objetivo

Tener una visión general de qué es **R**, para qué sirve y cómo instalarlo, así como conocer el ambiente integrado **RStudio**, cómo instalarlo, sus características principales y las diferentes ventanas que integran el ambiente de trabajo.

1 Introducción

1.1 ¿Qué es R?

R es un entorno integrado para el manejo de datos, cálculo y procedimientos gráficos y estadísticos. Los principales aspectos que ofrece son:

1. Facilidad para el manejo y el almacenamiento de datos.
2. Un conjunto de operadores para cálculo con vectores y matrices.
3. Una colección extensa e integrada de herramientas intermedias para el análisis estadístico de datos.
4. Multitud de facilidades gráficas.
5. Un lenguaje de programación simple.

1.2 Historia

R fue desarrollado en los 90's por Ross Ihaka y Robert Gentleman (Figura 1), del departamento de Estadística de la Universidad de Auckland. **R** es una implementación "open-source" del lenguaje S (Creado por John Chambers a principios de los 70's), que también es la base del sistema S-Plus (entorno comercial). El desarrollo actual de **R** es responsabilidad del [R Development Core Team](#). Además, **R** dispone de una comunidad de desarrolladores/usuarios detrás que se dedican constantemente a la mejora y a la ampliación de las funcionalidades y capacidades del programa. Nosotros mismos podemos ser desarrolladores de **R**!!



R: A Language for Data Analysis and Graphics

ROSS IHAKA and ROBERT GENTLEMAN

In this article we discuss our experience designing and implementing a statistical computing language. In developing this new language, we sought to combine what we felt were useful features from two existing computer languages. We feel that the new language provides advantages in the areas of portability, computational efficiency, memory management, and scoping.

Key Words: Computer language; Statistical computing.

Ross Ihaka is Senior Lecturer, and Robert Gentleman is Senior Lecturer, Department of Statistics, University of Auckland, Private Bag 92019, Auckland, New Zealand. e-mail: ihaka@stat.auckland.ac.nz.

©1996 American Statistical Association, Institute of Mathematical Statistics, and Interface Foundation of North America
Journal of Computational and Graphical Statistics, Volume 5, Number 3, Pages 299-314

Figura 1: Creadores de R (Ihaka y Gentleman 1996)

Foto obtenida de (<https://www.stat.auckland.ac.nz/~ihaka/downloads/the-r-project.pdf>)

1.3 Distribuciones de R

R se distribuye para:

- Windows
- Linux
- MacOS
- Unix

Los programas fuentes, binarios y la documentación de R, así como una lista de las preguntas más frecuentes acerca de R, se encuentran en el sitio de CRAN *Comprehensive R Archive Network* (<http://cran.r-project.org/mirrors.html>).

1.4 Instalando R

Los pasos para instalar **R** son:

1. Entrar a <http://cran.r-project.org/mirrors.html>
2. Elejir uno de los servidores
3. En la sección de “*Download and Install R*”, entrar a la liga de Linux, MacOS o Windows según sea el caso.
4. Descargar
5. Iniciar instalación

Una vez instalado R, podemos iniciar R, dando doble clic en el ícono de R (Figura 2).



Figura 2: Icono de R

1.5 ¿Qué es RStudio?

RStudio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para R, que facilita la tarea de uso interactivo de **R** y la programación de scripts. En este sentido, RStudio nos ofrece:

1. Visualización de los gráficos integrada en el mismo entorno
2. Acceso cómodo a la ayuda electrónica sobre R
3. Un editor que nos permite guardar nuestro trabajo
4. Una consola para la ejecución de código.
5. Editor de sintaxis que apoya la ejecución de código
6. Auto completado de código y sangría inteligente
7. Ejecución de código de **R** directamente desde el editor de código fuente.

1.5.1 Distribuciones

RStudio está disponible en versiones *open source* y comercial, para escritorio:

- Windows
- Mac
- Linux)

o desde web:

- RStudio Server
- RStudio Server Pro.

RStudio dispone de [hojas de consulta rápida](#), que facilitan el aprendizaje y el uso de algunos de sus paquetes.

1.5.2 Instalación

1. Instalar R, si no se encuentra previamente instalado. Ver la sección [Instalando R](#).
2. Ir a <https://www.rstudio.com>.
3. Seleccionar Download RStudio.
4. Descargar RStudio, dependiendo de nuestro sistema operativo (Windows, Mac o Linux).
5. Iniciar instalación.

Una vez instalado, puedes iniciar RStudio, dando doble clic en el ícono de RStudio (Figura 3).



Figura 3: Icono de RStudio

Nota: En este meetup trabajaremos **R** desde el ambiente de trabajo de RStudio. Sin embargo, en el [Apéndice 1](#), se muestra una breve descripción de **R** desde línea de comandos, por ejemplo: [Iniciando R desde línea de comandos](#).

2 Primeras nociones, un tour por RStudio

2.1 Ambiente de trabajo

RStudio por defecto tiene cuatro paneles de trabajo (Figura 4), estos paneles pueden estar ordenados de diferente manera, por lo tanto la descripción que se muestra a continuación está basada en la posición que tienen estos paneles en la figura 4. Por otra parte, a veces el panel 2 no aparece, porque no tenemos activo ningún script.

1. El panel inferior izquierdo es una consola de R en la que se puede escribir, ejecutar código, así como mostrar resultados del código que se va ejecutando.
2. El panel superior izquierdo es un editor de código, en este panel se abren los archivos en pestañas.

3. El panel superior derecho es para mostrar las variables en el entorno y un histórico de comandos ejecutados.
4. El panel inferior derecho contiene varias pestañas, entre ellas destacan files (muestra el directorio de archivos), plots (muestra las gráficas según se van ejecutando) y help (muestra la página de ayuda de las funciones cuando se soliciten).

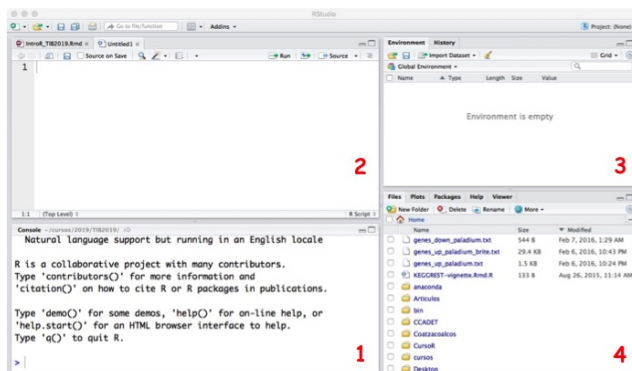


Figura 4: Ambiente de trabajo de RStudio

2.2 Consola

La **consola** de R, panel inferior izquierdo, es **donde nuestro código es interpretado y ejecutado**. Podemos escribir código directamente en la consola y R nos dará el resultado de lo que pidamos allí mismo. Esta es la razón por la que se dice que **R permite el uso interactivo**.

Cuando hablamos de ejecutar nos referimos a pedir que R realice algo, es decir, le estamos dando una instrucción. Por otra parte, cuando decimos que R nos devuelve algo, es que ha realizado algo que le hemos pedido y nos está dando una salida. Por ejemplo, si escribimos la siguiente instrucción en la consola y damos “*enter*”, estamos pidiendo que se ejecute la operación:

```
5 + 8
## [1] 13
```

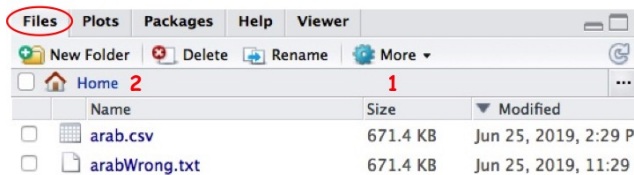
2.3 Movindose entre los directorios

El directorio de trabajo de R, corresponde a la ruta desde donde se inicia R. Cuando queremos trabajar con archivos, R los busca por defecto en el directorio de trabajo. Para visualizar archivos, cambiar o consultar el directorio de trabajo, seleccionamos la pestaña **Files**, que se encuentra en el panel inferior derecho de nuestro ambiente de trabajo (Figura 4, panel 4). En esa pestaña encontraremos diversos botones y un administrador de archivos (Figura 5A).

1. **More**. Funciones relacionadas con archivos o el directorio de trabajo.
2. **Home**. Para moverse a nuestro directorio raíz, que no necesariamente tiene que ser el mismo del directorio de trabajo.

La Figura 5B, corresponde al menú que se despliega con **More**

A)



B)

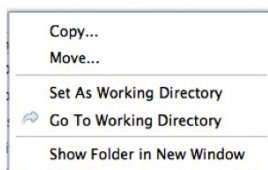


Figura 5: Menú de File

2.3.1 ¿Cuál es mi directorio de trabajo?

Para movernos al directorio de trabajo, seleccionamos **Go To Working Directory** del menú **More** (Figura 5B) y en automático nos llevará al directorio de trabajo. También desde línea de comandos podemos escribir:

```
getwd()
```

Y como respuesta tendremos la ruta del directorio de trabajo.

2.3.2 ¿Cómo cambio de directorio?

Para cambiar de directorio de trabajo:

1. Nos movemos a la carpeta que queremos definir como el directorio de trabajo, por medio del administrador de archivos.
2. Seleccionamos **Set As Working Directory** del menú **More**.

Desde línea de comandos podemos usar la instrucción `setwd()` y en los parentesis ponemos la dirección hacia donde queremos movernos.

```
setwd("/home/usuario_X/R")
```

Si consultamos nuevamente cuál es el directorio de trabajo, veremos qué cambió.

```
getwd()
```

```
[1] "/home/usuario_X/R"
```

2.3.3 Consultando el contenido de un directorio

Para consultar el contenido de un directorio, nos movemos al directorio de interés, por medio del administrador de archivos o desde línea de comandos usamos la instrucción `dir()` y en los parentesis ponemos la ruta del directorio que queremos consultar. Si dejamos los parentesis vacíos, nos mostrará el contenido del directorio de trabajo. por ejemplo:

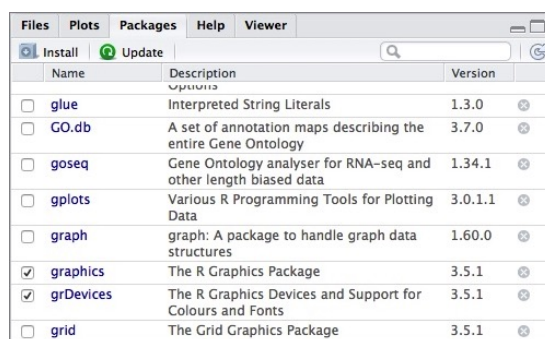
```
dir()
```

2.4 Paquetes de R

Las funciones de R se agrupan en paquetes (*packages*). Los paquetes que contienen las funciones más habituales se incluyen por defecto en la distribución de R, y el resto se encuentran disponibles en [CRAN](https://cran.r-project.org/), o en algunos otros repositorios como es el caso de Bioconductor (<https://www.bioconductor.org>).

2.4.1 Visualizando los paquetes instalados

Para visualizar los paquetes que tenemos instalados seleccionamos la pestaña de **Packages**, que se encuentra en el panel inferior derecho de nuestro ambiente de trabajo (Figura 4, panel 4). Los paquetes que se encuentran seleccionados (Figura 6), son aquellos que están cargados y listos para usarse. Para utilizar un paquete instalado, es necesario cargarlo. Al iniciar R se cargan por defecto los paquetes básicos, por ejemplo: `graphics` y `stats`, entre otros.

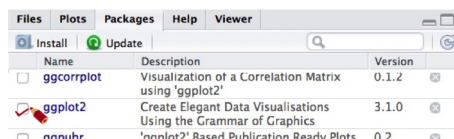


Name	Description	Version
<input type="checkbox"/> glue	Interpreted String Literals	1.3.0
<input type="checkbox"/> GO.db	A set of annotation maps describing the entire Gene Ontology	3.7.0
<input type="checkbox"/> goseq	Gene Ontology analyser for RNA-seq and other length biased data	1.34.1
<input type="checkbox"/> gplots	Various R Programming Tools for Plotting Data	3.0.1.1
<input type="checkbox"/> graph	graph: A package to handle graph data structures	1.60.0
<input checked="" type="checkbox"/> graphics	The R Graphics Package	3.5.1
<input checked="" type="checkbox"/> grDevices	The R Graphics Devices and Support for Colours and Fonts	3.5.1
<input type="checkbox"/> grid	The Grid Graphics Package	3.5.1

Figura 6: Paquetes instalados

2.4.2 Cargando un paquete

Para cargar un paquete en particular se selecciona la casilla del paquete que queremos utilizar. Por ejemplo, si queremos usar el paquete `ggplot2`, marcamos la casilla correspondiente a este paquete (Figura 7).



Name	Description	Version
<input type="checkbox"/> ggcorrplot	Visualization of a Correlation Matrix using 'ggplot2'	0.1.2
<input checked="" type="checkbox"/> ggplot2	Create Elegant Data Visualisations Using the Grammar of Graphics	3.1.0
<input type="checkbox"/> ggpubr	'ggplot2' Based Publication Ready Plots	0.2

Figura 7: Cargando un paquete

2.4.3 Visualizando las funciones de un paquete determinado

Para visualizar las funciones contenidas en un paquete, daremos `click` en el paquete que queremos consultar y posteriormente nos mostrará la documentación del paquete. Por ejemplo si damos `click` en `ggplot2` nos mostrará sus funciones (Figura 8).

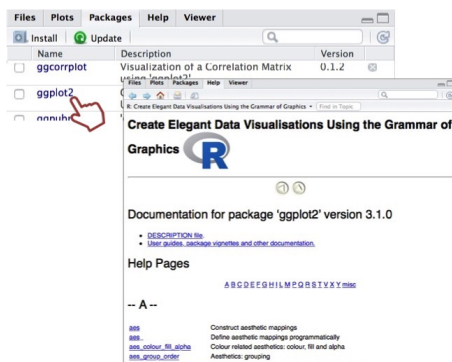


Figura 8: Visualización de funciones de un paquete

2.4.4 Instalando paquetes

Para instalar un paquete, seleccionamos **Install** que se encuentra en la pestaña de paquetes. En automático se abrirá una ventana en la cual escribiremos el nombre del paquete que queremos instalar, por ejemplo: `knitr`. La casilla donde escribimos el nombre del paquete, cuenta con la función de autocompletado, por lo tanto si hay algún valor coincidente (que comience por los caracteres escritos) te los mostrará (Figura 9). Es importante marcar la casilla **Install dependencies**, para que se instalen todas las dependencias, es decir, programas o paquetes que requiere el paquete que estamos instalando. Posteriormente, daremos `click` en el botón **Install** e iniciará el proceso de instalación.

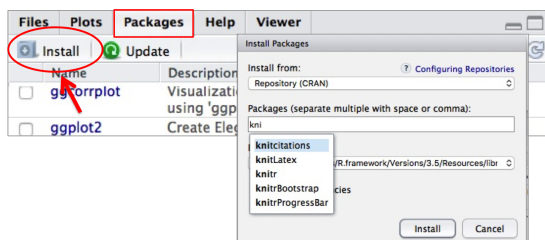


Figura 9: Instalación del paquete `knitr`

Nota: En el [Apéndice 1](#) se muestran algunos comandos para el [Manejo de paquetes desde línea de comandos](#).

2.5 Consultando la ayuda

Para consultar la ayuda seleccionamos la pestaña **Help**, que se encuentra en el panel inferior derecho de nuestro ambiente de trabajo (Figura 4, panel 4). En esa pestaña encontraremos varios botones y campos para llenado (Figura 10).

1. Campo de búsqueda
2. Ayuda de R. Documentación general de R y RStudio
3. Mostrar Ayuda en una ventana emergente

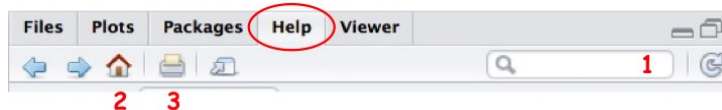


Figura 10: Menú de Ayuda

Para solicitar ayuda de un comando en particular, escribimos el comando en el campo de búsqueda, damos `enter` y en automático aparecerá la ayuda. Por ejemplo, podemos solicitar ayuda de `matrix` (Figura 11)

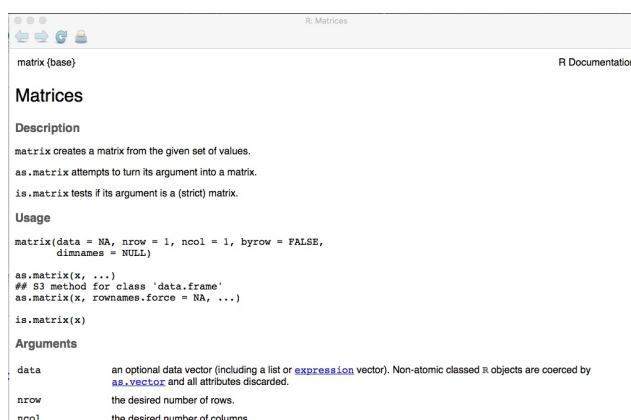


Figura 11: Menú de Ayuda

Como podemos observar en la Figura 11. La ayuda nos proporciona la descripción del comando consultado, sus argumentos y ejemplos (en algunos no hay ejemplos), entre otra información.

Además, podemos realizar búsquedas por aproximación, es decir, poniendo sólo una parte del término que busquemos o por un tema. Así, se desplegarán todas las páginas de ayuda en donde aparece dicho término. Por ejemplo, si buscamos el término “clustering” veremos, los paquetes en los que aparece ese término.

Nota: En el [Apéndice 1](#) se muestran algunos comandos para el [Consultando la ayuda desde línea de comandos](#).

2.6 Sección de visualización de gráficas

La pestaña **Plots**, permite la visualización de las gráficas, que se pueden generar con los diferentes paquetes de graficación (graphics, ggplot2, entre otros).

Para visualizar las gráficas, solo debemos escribir las instrucciones que la generan. Por ejemplo:

```
hist(runif(100))
```

Como podemos observar (Figura 12) la gráfica se genera en la sección de **Plots**. Si tenemos más de una gráfica las podemos recorrer por medio de las fechas que se encuentran en el menú abajo de la pestaña de **Plots**. En ese mismo menú, tenemos también la opción de

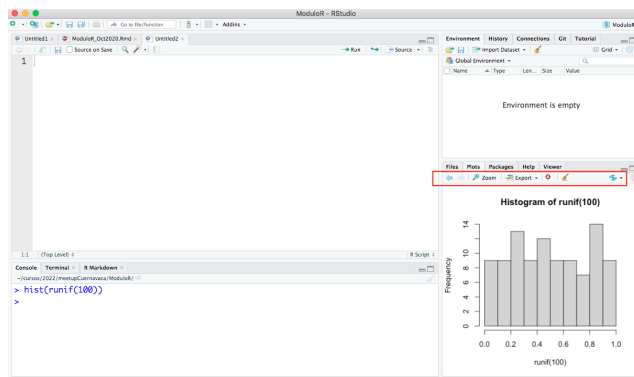


Figura 12: Menú de Plots

hacer **Zoom** a la imagen, lo cual nos abrirá una ventana emergente, con la gráfica en zoom. También podemos guardar la imagen por medio de la opción **Export**. Finalmente, en el menú de Plots, tenemos un tache (circulo rojo con una X blanca) y una escoba, el primero me permite eliminar la gráfica que se encuentra activa en el visualizador y la segunda elimina todas las gráficas del visualizador.

2.7 Environment

En el entorno de trabajo ("Environment") del programa, se muestran el conjunto de datos u objetos (variables, gráficos, funciones, entre otros) que se van generando y almacenando al ejecutar diferentes instrucciones. Para ver cómo funciona, crearemos dos objetos y los almacenaremos.

```
x=5
palabra="hola"
```

Como podemos observar en la sección **Environment** se muestran los objetos que acabamos de crear (Figura 13). Para guardar todos los objetos en un archivo, con el nombre y ruta que le especifiquemos, seleccionamos el ícono marcado con 1 en la Figura 13. Por ejemplo, guardaremos en el archivo "MisObjetos" (en el directorio de trabajo) los objetos creados hasta el momento.

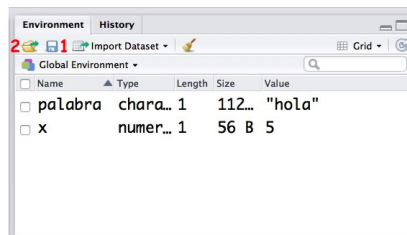


Figura 13: Ambiente de los objetos

Si nos movemos al directorio donde guardamos el archivo "MisObjetos", veremos que por defecto R coloca la extensión ".RData" al archivo.

Para cargar los objetos almacenados en el archivo MisObjetos, utilizamos el ícono marcado con 2 en la Figura 13 y seleccionamos el archivo que queremos cargar.

En el [Apéndice 1](#) se describe cómo guardar los objetos desde línea de comandos en la sección [Guardando y Cargando el Espacio de Trabajo desde línea de comandos](#)

2.8 History, comandos creados en una sesión

En la pestaña **History** del panel superior derecho (Figura 4, panel 3), se muestran todos los comandos que hemos escrito en la sesión (Figura 14). esta pestaña tiene a su vez un menú con diversas opciones (leer, guardar, borrar una instrucción o todas, mandar a consola o a un script)

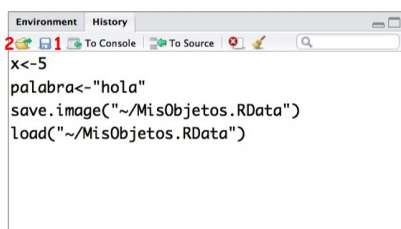


Figura 14: Historial de comndos

Para guardar los comandos en un archivo, con el nombre y ruta que le especifiquemos, seleccionamos el ícono marcado con 1 en la Figura 14. Por ejemplo, guardaremos en el archivo “MisComandos” (en el directorio de trabajo) los comandos utilizados hasta el momento.

Para cargar los comandos almacenados en el archivo MisComandos, utilizamos el ícono marcado con 2 en la Figura 14 y seleccionamos el archivo que queremos cargar.

En el [Apéndice 1](#) se describe cómo guardar el historial de comandos desde línea de comandos en la sección [Guardando y Cargando el Historial de Comandos desde línea de comandos](#).

Nota: El ambiente de trabajo de R, está constituido por los objetos e instrucciones creadas, durante la sesión

2.9 Saliendo de R

Para salir de RStudio seleccionamos **Quit RStudio** de **RStudio** o **Quit Session** en **File** en el menú principal (Figura 15).

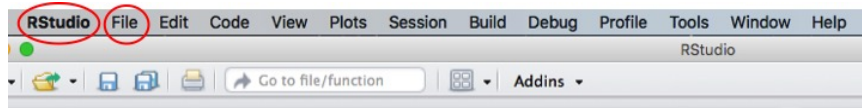


Figura 15: Menú Principal RStudio

También podemos escribir en la consola:

```
q()  
  
# 0 podemos escribir la palabra completa  
  
quit()
```

```
## Save workspace image? [y/n/c]:
```

Antes de cerrar la sesión, R nos preguntará si queremos almacenar el ambiente de trabajo, es decir, objetos y comandos creados en la sesión. Si le decimos que sí, entonces los objetos se guardarán en un archivo llamado `.RData`, que es de tipo binario y los comandos se almacenarán en el archivo `.Rhistory`, que es de tipo texto. Estos dos archivos se guardan en el directorio de trabajo y quedan como archivos ocultos. La siguiente vez que iniciemos una sesión de R en el mismo directorio de trabajo, el ambiente de trabajo y todos los objetos se restablecerán.

3 Apéndice 1

3.1 Iniciando R desde línea de comandos

Desde línea de comandos, en una terminal, escribir **R** y dar **enter**.

Automáticamente aparecerá en la pantalla información relacionada con la versión de R, los derechos reservados, cómo citar R, entre otras cosas más. Y al final aparecerá el símbolo de mayor que ">", que es el *prompt* de R.

3.2 Manejo de paquetes desde línea de comandos

```
# Para listar los paquetes instalados
library()

# Para cargar un paquete en particular, en este caso ggplot2
library(ggplot2)

# Para visualizar los paquetes que ya están cargados
search()
## [1] ".GlobalEnv"          "package:dplyr"        "package:kableExtra"
## [4] "package:knitr"        "package:BiocStyle"    "package:stats"
## [7] "package:graphics"    "package:grDevices"    "package:utils"
## [10] "package:datasets"
## [ reached getOption("max.print") -- omitted 3 entries ]

# Para visualizar las funciones del paquete colocado en la posición 6
ls(6)
## [1] "acf"                  "acf2AR"               "add.scope"
## [4] "add1"                 "addmargins"           "aggregate"
## [7] "aggregate.data.frame" "aggregate.ts"          "AIC"
## [10] "alias"
## [ reached getOption("max.print") -- omitted 439 entries ]
```

Para instalar un paquete utilizamos el comando:

```
install.packages("knitr")
```

En este caso se instalaría el paquete llamado `knitr`

3.3 Consultando la ayuda desde línea de comandos

Existen diversas maneras de obtener ayuda sobre los comandos de R, a continuación veremos las más frecuentes. Una manera clásica de obtener ayuda respecto a un comando es utilizando la palabra `help(comando)` o `?comando`, por ejemplo, para consultar la ayuda del comando `solve`:

```
help(solve)
```

```
?solve
```

```
## solve                                package:base                R Documentation
##
## Solve a System of Equations
##
## Description:
##
##   This generic function solves the equation 'a %*% x = b' for 'x',
##   where 'b' can be either a vector or a matrix.
##
## Usage:
##
##   solve(a, b, ...)
##
##   ## Default S3 method:
##   solve(a, b, tol, LINPACK = FALSE, ...)
##
## Arguments:
##
##   a: a square numeric or complex matrix containing the
##       coefficients of the linear system. Logical matrices are
##       coerced to numeric.
##
##   b: a numeric or complex vector or matrix giving the right-hand
```

Para solicitar ayuda de funciones con caracteres especiales o algunas palabras reservadas, es necesario poner entre comillas la función o palabra reservada, ya que si no se hace así obtendremos un mensaje de error, por ejemplo, `for` es una palabra reservada de R, consultemos la ayuda sin poner comillas.

```
help(for)
## Error: <text>:1:9: unexpected ')'
```

```
## 1: help(for)
```

```
##      ^
```

Pero si colocamos la palabra entre comillas, obtendremos la ayuda, sin problemas.

```
help("for")
```

```
## Control                                package:base                R Documentation
##
## Control Flow
##
## Description:
##
##   These are the basic control-flow constructs of the R language.
##   They function in much the same way as control statements in any
##   Algol-like language.  They are all reserved words.
```

3.3.1 Ayuda en general

Para abrir la ayuda general en un navegador (sólo si tenemos la ayuda en HTML instalada y tenemos conexión a red), utilizamos:

```
help.start()
```

3.3.2 Ayuda relacionada a un término

Podemos hacer búsquedas de información relacionada con un término en particular por medio de la función, `help.search`, la cual nos indicará los paquetes en donde se encuentra dicho término y una descripción muy pequeña del mismo. Por ejemplo:

```
help.search("clustering")
```

```
## Vignettes with name or keyword or title matching 'clustering' using
## fuzzy matching:
##
## graph::clusterGraph      clusterGraph and distGraph
## Concepts: clustering
##
## Type 'vignette("F00", package="PKG")' to inspect entries 'PKG::F00'.
##
##
## Help files with alias or concept or title matching 'clustering' using
## fuzzy matching:
##
## Biostrings::stringDist
##                               String Distance/Alignment Score Matrix
## Concepts: Clustering
## Heatplus::oldCutplot.dendrogram
##                               Plot Subtrees of a Dendrogram in Different
```

```
##                               Colors
## Concepts: Clustering
```

3.3.3 Ayuda sobre ejemplos de uso de una función

La función `example`, nos muestra ejemplos de ejecución de alguna función en particular, siempre y cuando la función que busquemos tenga en su descripción ejemplos.

```
example("data.frame")
##
## dt.frm> L3 <- LETTERS[1:3]
##
## dt.frm> fac <- sample(L3, 10, replace = TRUE)
##
## dt.frm> (d <- data.frame(x = 1, y = 1:10, fac = fac))
##      x y fac
## [ reached 'max' / getOption("max.print") -- omitted 10 rows ]
##
## dt.frm> ## The "same" with automatic column names:
## dt.frm> data.frame(1, 1:10, sample(L3, 10, replace = TRUE))
##      X1 X1.10 sample.L3..10..replace...TRUE.
## [ reached 'max' / getOption("max.print") -- omitted 10 rows ]
##
## dt.frm> is.data.frame(d)
## [1] TRUE
##
## dt.frm> ## do not convert to factor, using I() :
## dt.frm> (dd <- cbind(d, char = I(letters[1:10])))
##      x y fac char
## [ reached 'max' / getOption("max.print") -- omitted 10 rows ]
##
## dt.frm> rbind(class = sapply(dd, class), mode = sapply(dd, mode))
##      x      y      fac      char
## [ reached getOption("max.print") -- omitted 2 rows ]
##
## dt.frm> stopifnot(1:10 == row.names(d)) # {coercion}
##
## dt.frm> (d0 <- d[, FALSE]) # data frame with 0 columns and 10 rows
## data frame with 0 columns and 10 rows
##
## dt.frm> (d.0 <- d[FALSE, ]) # <0 rows> data frame (3 named cols)
## [1] x
## [ reached getOption("max.print") -- omitted 2 entries ]
## <0 rows> (or 0-length row.names)
##
## dt.frm> (d00 <- d0[FALSE, ]) # data frame with 0 columns and 0 rows
## data frame with 0 columns and 0 rows
```


3.3.4 Ayuda sobre argumentos de una función

También se puede solicitar ayuda reacionada a los argumentos de una determinada función.

```
args(data.frame)
## function (... , row.names = NULL, check.rows = FALSE, check.names = TRUE,
##      fix.empty.names = TRUE, stringsAsFactors = default.stringsAsFactors())
## NULL
```

3.3.5 Búsquedas aproximadas

Finalmente, una alternativa para realizar búsquedas por aaproximación, es decir, poniendo sólo una parte del término que busacamos, es con el comando **apropos**.

```
apropos("solve")
## [1] "backsolve"      "forwardsolve"   "qr.solve"       "solve"
## [5] "solve.default" "solve.qr"
```

3.4 Guardando y Cargando el Espacio de Trabajo desde línea de comandos

Para almacenar y cargar los objetos que se crearon en una sesión de trabajo, utilizamos las funciones **save.image** y **load**.

```
save.image(file = "archivo.Rdata")
```

Recordemos que con la función **dir()** podemos consultar los archivos del directorio de trabajo, para verificar que se creó el archivo **archivo.Rdata**.

Para almacenar el archivo en un directorio diferente al de trabajo, se requiere especificar la ruta.

Si queremos llamar a los objetos almacenados en el archivo **archivo.Rdata**, utilizamos:

```
load(file = "archivo.Rdata")
```

Si el archivo no se encuentra en el directorio de trabajo, es necesario especificar la ruta en donde se encuentra.

3.4.1 Almacenando, en un archivo, objetos específicos creados en una sesión

Cuando queremos guardar uno o varios objetos creados en una sesión, usamos el comando **save**.

```
save(objeto, file = "archivo.Rdata")
save(objeto1,objeto2, objetoX, file = "archivo.Rdata")
```

Para cargar estos archivos utilizamos nuevamente la función **Load**.

3.5 Guardando y Cargando el Historial de Comandos desde línea de comandos

Si queremos guardar la historia de los comandos utilizados en un momento dado, lo hacemos con el comando `savehistory`. Por ejemplo:

```
savehistory(file = "archivo.Rhistory")
```

Utilicemos `dir()` para verificar que el archivo se creó

Para cargar el archivo `archivo.Rhistory` utilizamos la instrucción `loadhistory`.

```
loadhistory(file = "archivo.Rhistory")
```

Ihaka, Ross, y Robert Gentleman. 1996. «R: A Language For Data Analysis and Graphics.» *Journal of Computational and Graphical Statistics* 5 (3): 299, 314.