Fundamentos para el uso de R/RStudio

Alejandro Franco y Luis Carlos Ramos

28/Junio - 23/Julio

Bienvenid@s

Este documento aborda los temas que serán tratados durante el curso con título *Fundamentos para el uso de R/Rstudio*, de manera que el asistente tenga acceso facilitado a la información esencial. Los detalles de cada tema serán desarrollados durante las sesiones, así como las dudas que surjan de los ejercicios que aquí se proponen.

Especificaciones

1. Las letras en **negrita** y *crusiva* contienen links que podrían ser de interés para su visita, como ***este***, que contiene archivos de guías rápidas para las aplicaciones más comunes de R y los programas con los que se le asocia.
2. **IMPORTANTE:** así es como se mostrará una indicación a seguir, no la pierdas de vista.
3. Un renglón con sangría indica el inicio de un ejercicio extraclase que se propone para reforzar el tema.

R/RStudio

R es un lenguaje de programación que tiene como precedente al lenguaje S y actualmente es usado para aprender estadística, enseñarla, hacer proyecciones en mapas, trabajar con imágenes, procesar datos genómicos, manipulando datos, obtener gráficos, estadísticos y simulaciones en una misma aplicación. Se distribuye como un software gratuito y de código abierto, compatible con los sistemas operativos Windows, macOS y Linux. El software para el uso del lenguaje se descarga directamente del CRAN (*Comprehensive R Archive Network*) en ***esta página web***, que mostrará la siguiente ventana, donde debe descargarse el ejecutable de acuerdo al sistema operativo y características del equipo.

Al terminar de ejecutar la instalación y abrir el programa, se mostrará la interfaz gráfica (GUI, *graphical user interface*) de R (Fig. 2), que puede resultar un poco abrumadora si es la primera vez que utilizamos un software de programación. Para hacer más dinámico el manejo del lenguaje, se hace uso de un entorno de programación, o IDE (Integrated Development Environment). En R las tres plataformas gráficas principales utilizadas en el manejo de R son ***RStudio***, ***RCommander*** y ***RKWard***, sin embargo existen más. El que se utiliza con más frecuencia es RStudio y es el que será usado durante el curso.

IMPORTANTE: El orden de ejecución de los programas es importante, primero debe ejecutarse R y después RStudio, para un funcionamiento adecuado.

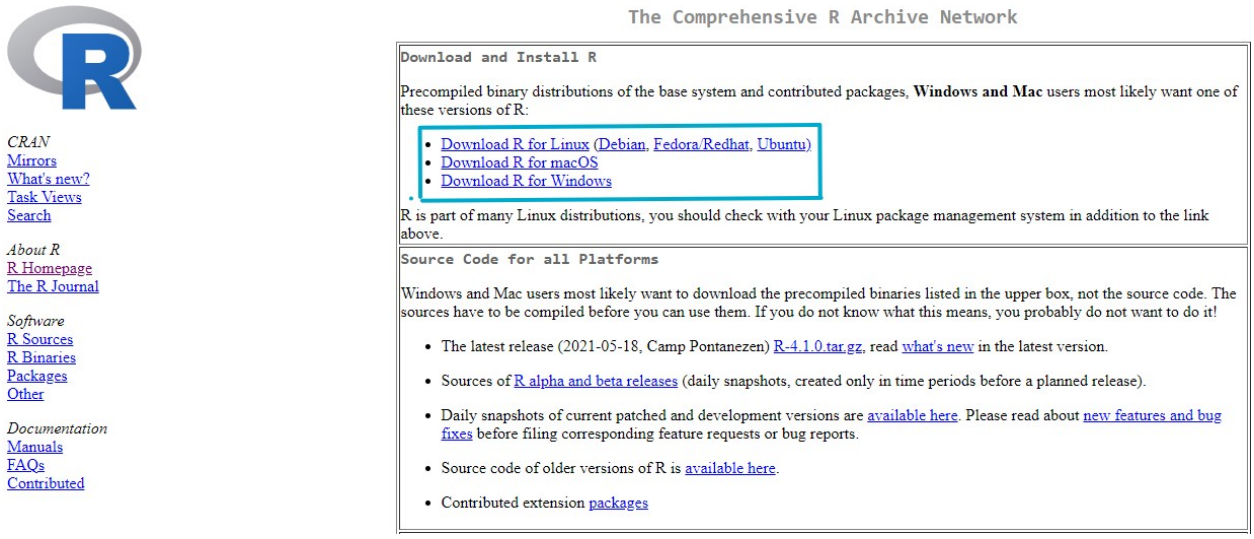
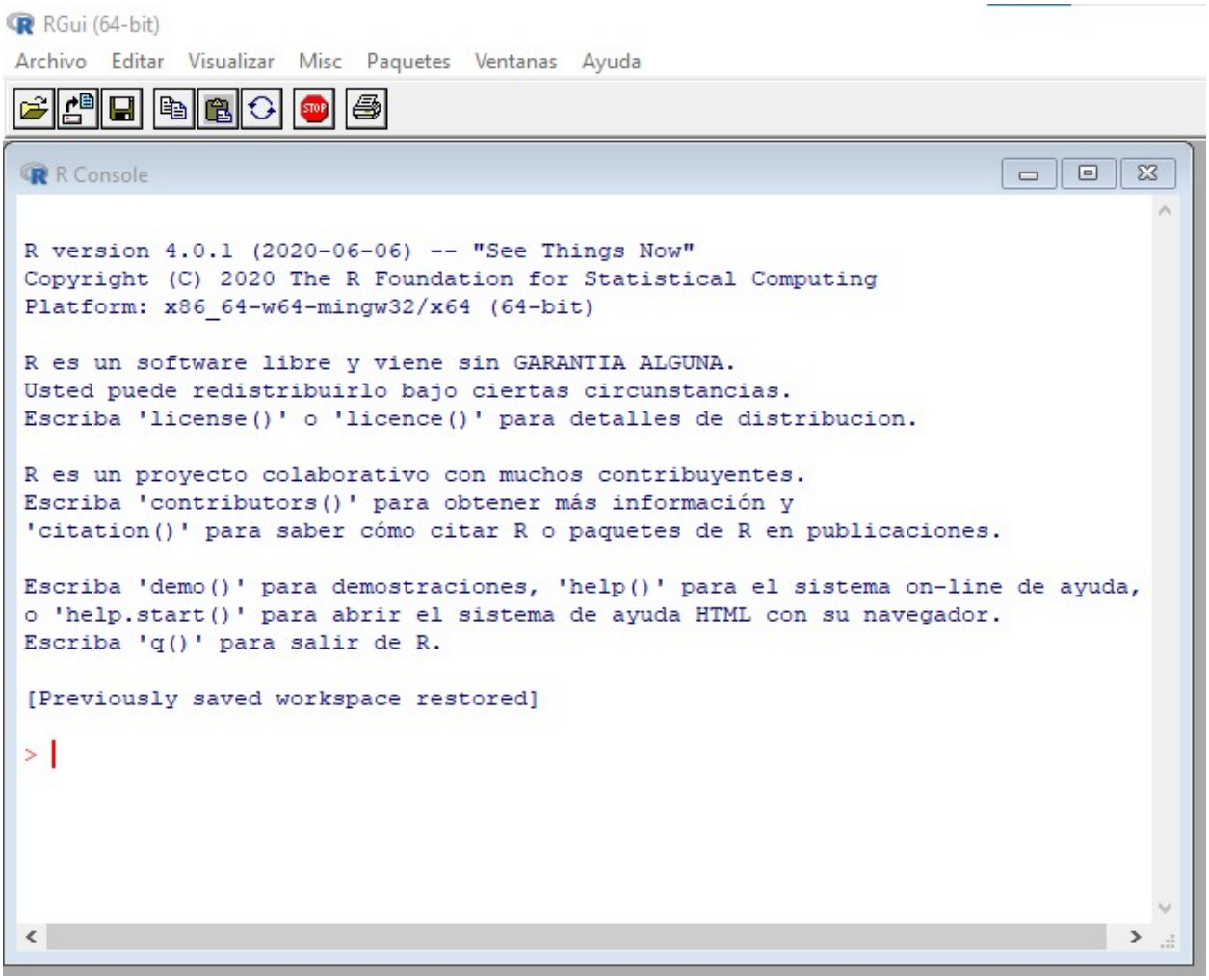


Figure 1: Ventana de la página web que muestra los ejecutables por sistema operativo



Es necesario entender como se compone el espacio de trabajo en RStudio para hacer uso eficiente de las facilidades que provee, la consola, ambiente, editor de código, las utilidades y el menú superior son las

partes principales en que se divide RStudio (Fig. 3).

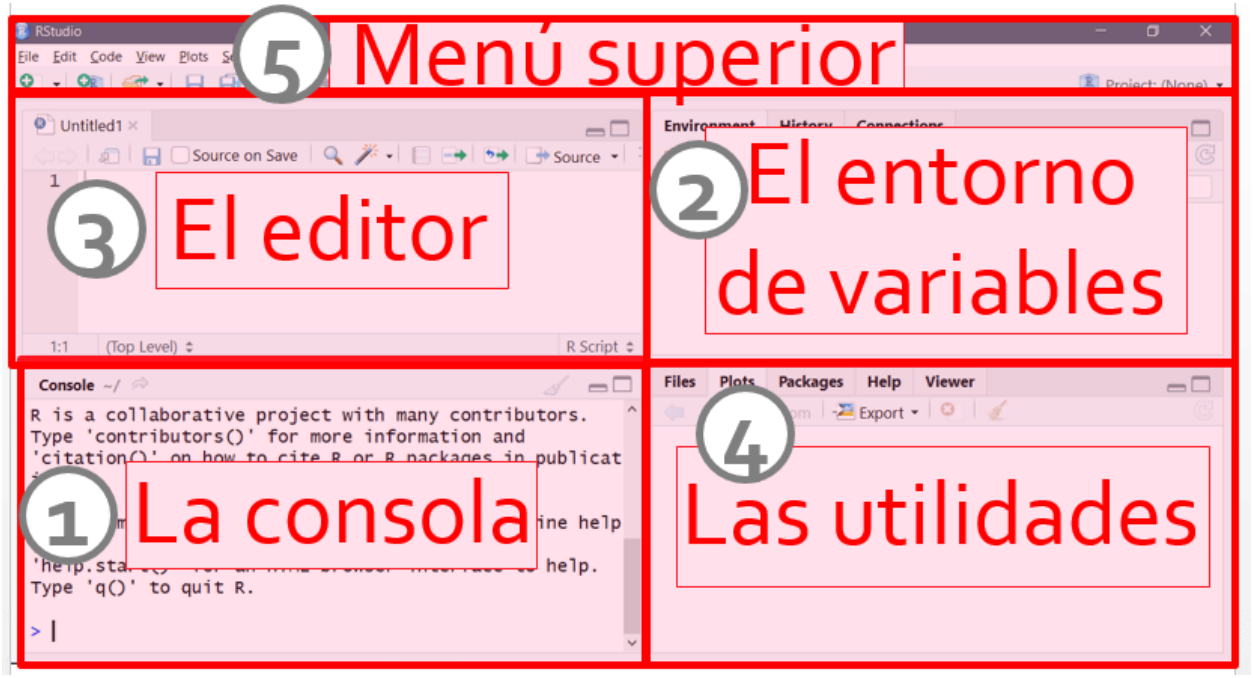


Figure 2: Divisiones principales de la ventana de RStudio

Control de versiones

Trabajar a través de un ***sistema de control de versiones*** facilita el intercambio y resguardo de información en una plataformal virtual, como los conocidos Microsoft OneDrive y Google drive, que al mismo tiempo que almacenan tus archivos, te permiten trabajar de manera remota y colaborativa en tiempo real. Para cumplir esa fucnión en R, se usa ***GitHub***, accede al enlace y crea una cuenta en la plataforma, asegurandote de registrar un correo que sí ocupes, pues será requerido más adelante. Antes de poder usar GitHub como un control de versiones es importante familiarizarse con los siguientes términos:

- **Repository:** equivalente a un 'folder/directorio'. Todos los archivos
- **Commit:** significa guardar las ediciones y cambios hechos. Git compara la versión anterior de todos los archivos en el 'repo'. Son 'el corazón' de los sistemas de control pues se observan todos los cambios que se han hecho con el tiempo
- **Push:** actualizar el repositorio con las ediciones hechas, así es posible tenerlos en línea
- **Pull:** actualizar la versión local del archivo para tener la última versión que alguien haya modificado (trabajo colaborativo)
- **Staging:** preparar un archivo para 'commit'. Cuando se tienen múltiples archivos modificados de un repositorio, es más eficiente 'stage' cada uno por separado para 'commit' con el comentario de cambio respectivo
- **Branch:** cuando un mismo archivo tiene dos copias de forma simultánea. Se crea una 'branch' cuando 'pull' el documento para su edición local

- **Merge:** ediciones independientes del mismo archivo que son incorporadas a uno solo y unificado archivo.
- **Conflict:** más de una persona realizó un cambió en una misma 'sentencia', eso no puede ser 'merge' automáticamente, de tal forma que se hace manualmente
- **Clone:** hacer una copia de un repositorio existente y editarlo personalmente. Es cuando te integras a un equipo de trabajo
- **Fork:** copia personal de un repositorio ya existente.

  Realiza el ejercicio que se propone ***aquí*** para familiriazarte más rápido con los términos.

Para usar GitHub como control de versiones no basta con tener una cuenta, debe crearse un enlace entre la plataforma y el IDE de R, *RStudio*. Para ello, es necesario descargar y ejecutar ***GIT a través de este enlace***.

Tipos de objetos

Caracteres

Números complejos

Enteros

Vectores

Breviario: Obtener ayuda

R cuenta con documentación para resolver muchas de las dudas que puedas tener sobre el uso de una función o paquete, siendo primer aprendiz o alguien con más tiempo de experiencia. La siguiente tabla resume varias formas en las que puedes pedir ayuda.

FUNCIONES	USO
help () o ?	En la ventana de utilidades se abre una página de ayuda que despliega información. Se usa cuando recuerdas textualmente lo que deseas buscar
help.search()	Cuando no recuerdas el nombre exacto del objeto que deseas buscar
help.start()	Se despliega en utilidades para dirigirte a la página web que contenga la información
apropos()	Igual que help.search(), cuando se buscan coincidencias con una cadena de texto
example()	Al usarse en la consola provee un código con una serie de ejemplos en los que se debe usar
vignette()	Se usa con nombres textuales de objetos y despliega documentos PDF cuando están disponibles para el objeto
find()	Cuando se requiere saber el nombre del paquete de donde proviene la función
demo()	Cuando se requiere la demostración del uso de una función
history()	Para observar el historial de los comandos ejecutados en la sesión

Añadir un ejemplo por cada función para sólo un objeto

Breviario: Extracción tradicional de datos

Saber como es que se extraen elementos de un objeto es de suma importancia para el momento de hacer algunas operaciones, análisis y construcción de gráficos. A continuación se presenta una tabla con los operadores tradicionales de extracción.

Operador	Objeto
[ ]	Vectores,
[[ ]]	Matrices, data frames y listas
$	Matrices, data frames y listas

El operador *'[ ]'* e usa comunmente para extraer datos de un vector, las posibilidades son:

- Obtener un elemento o elementos de los que se conoce su posición

```
objeto <- c(9, 1, 6, 5, 8, 2, 6, 4, 8, 3)

extraccion_1 <- objeto [5]

extraccion_1
```

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

El resultado se debe a que el número 8 corresponde a la posición 5 dentro del vector.

- Extraer un conjunto de elementos que correspondan a una serie, ya sea que sean consecutivos o que correspondan a posiciones diferentes

```
extraccion_2 <- objeto [7:9]

extraccion_2
```

```
## [1] 6 4 8
```

Como te imaginarás, los valores '6, 4 y 8' corresponden a las posiciones '7, 8 y 9'. Aunque también se pueden extraer valores que correspondan a posiciones diferentes, usando un vector dentro de los corchetes para indicar las posiciones de los objetos que se desea extraer. Por ejemplo:

```
extraccion_3 <- objeto[c(1, 3, 8)]

extraccion_3
```

```
## [1] 9 6 4
```

Donde se indicaron las posiciones de los objetos dentro del vector, similar a los ejemplos anteriores.

- Un conjunto de valores que responden a una característica de interés

```
extraccion_4 <- objeto [objeto > 5]

extraccion_4
```

```
## [1] 9 6 8 6 8
```

La instrucción a R fue que nos diera todos los valores que fueran mayores a 5 dentro del vector *objeto*. De tal forma que el resultado es otro vector con esos valores, como se muestra arriba.

A continuación se presentan las formas en las que se extraen datos a partir de listas, matrices y data frames. Los breviarios correspondientes serán impartidos en el momento que se aborden dichos objetos.

IMPORTANTE: la extracción de datos de un data frame y una matriz son equivalentes, a continuación solo se ejemplificará con un data frame.

Dando continuidad el uso del operador *'[ ]'*, también es usado en matrices y data frames, para un valor particular. Por ejemplo, si se requiere el dato de la columna 2 y la fila 1 del siguiente data frame:

wingcrd	tarsus	head	Wt
59.0	22.3	31.2	9.5
55.0	19.7	30.4	13.8
53.5	20.8	30.6	14.8
55.0	20.3	30.3	15.2
52.5	20.8	30.3	15.5
57.5	21.5	30.8	15.6
53.0	20.6	32.5	15.6
55.0	21.5	NA	15.7

Lo que debe hacerse usar los valores de las dimensiones, en primer término se usa la posición de la fila seguido de una coma y luego la posición de la columna, así:

```
morfometria[1, 2]
```

```
## [1] 22.3
```

Siguiendo la misma lógica, para extraer la o las columnas de una matríz entonces el primer espacio debe quedar vacío y colocarse una coma. De tal modo que para extraer las columnas necesarias será así:

```
morfometria[,3:4]
```

```
##   head   Wt
## 1 31.2  9.5
## 2 30.4 13.8
## 3 30.6 14.8
## 4 30.3 15.2
## 5 30.3 15.5
## 6 30.8 15.6
## 7 32.5 15.6
## 8   NA 15.7
```

El operador $*funcionaparausaroextraerunacolumnaespecíficadeunarreglobidimensional.Porejemplo, silacolumnaquesen$ y luego el nombre de la columna que se desea:

```
morfometria$wingcrd
```

```
## [1] 59.0 55.0 53.5 55.0 52.5 57.5 53.0 55.0
```

Finalmente pero no menos importante, el uso de *[[ ]]* se asocia más a listas, que son objetos que albergan distintos tipos de objetos. Para la demostración sobre el uso de este operador se combinaran los objetos de las demostraciones anteriores sobre extracción de datos, de tal forma que nuestra lista será la siguiente:

```
## $Vector
##  [1] 9 1 6 5 8 2 6 4 8 3
##
## $DataFrame
##   wingcrd tarsus head   Wt
## 1    59.0   22.3 31.2  9.5
## 2    55.0   19.7 30.4 13.8
## 3    53.5   20.8 30.6 14.8
## 4    55.0   20.3 30.3 15.2
## 5    52.5   20.8 30.3 15.5
## 6    57.5   21.5 30.8 15.6
## 7    53.0   20.6 32.5 15.6
## 8    55.0   21.5   NA 15.7
```

Si lo que necesitas extraer de esa lista sólo es el primer objeto *(Vector)*, entonces debe ponerse como cadena de caracteres dentro de los dos corchetes:

```
lista_extra[['Vector']]
```

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

La función del doble corchete para una lista puede ir más allá, puedes seleccionar algún elemento anidado en la lista. Es decir, si quisieramos la tercer columna del elemento *(DataFrame)* lo que debe hacerse es lo siguiente:

```
lista_extra[[c(2, 3)]]
```

```
## [1] 31.2 30.4 30.6 30.3 30.3 30.8 32.5   NA
```

Lo que se hizo fue indicar con un vector la posición del elemento de la lista y el lugar de la columna que en este caso se requería. Como lo habrás notado el ejemplo resultó en la columna *head* de nuestro arreglo bidimensional.

Lógicos

ALEX

Matrices

ALEX BREVIARIO Tipos de mensajes en R

Listas

ALEX

Data frame y arrays

Factores

ALEX

Funciones

ALEX

Valores especiales

Faltantes

Infinito

Tiempo (Fechas y horas)

Operaciones

Aritméticas

Lógicas

Relacionales

Paquetes

ALEX # Subconjuntos de datos: es un breviario

Subconjuntos de datos

A ver, este cambio...

Mañana lo hago el cambio

Extracción de datos

dplyr

ALEX BREVIARIO Operadores lógicos/booleanos # Sistemas de graficado

Graficado base

ggplot2

ALEX # Pruebas estadísticas **ALEX** # Funciones reciclables **LUIS**