Introducción al lenguaje R, Rstudio y R Markdown

PID_00293461

Marta Casals Fontanet
Alícia Vila Grifo

Tiempo mínimo de dedicación recomendado: 5 horas

Universitat Oberta de Catalunya

Marta Casals Fontanet

Estadística de formación y profesión. Licenciada en ITM. Máster en Educación y nuevas tecnologías. Profesora de Matemáticas en secundaria y universidad. Profesora colaboradora en el máster de Bioinformática y bioestadística de la UOC y la UB.

Alicia Vila Grifo

Licenciada en Matemáticas por la Universidad de Valencia y profesora consultora de Probabilidad, Estadística y Análisis de Datos en diferentes estudios de la UOC. Actualmente es profesora funcionaria de Informática en el Departamento de Educación de la Generalitat de Cataluña, en los ámbitos de programación y bases de datos. También participa en la coordinación de proyectos de aplicaciones web y de análisis de datos.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por los profesores: Álvaro Leitao Rodríguez y David Cantón Fabà

Cómo citar este recurso de aprendizaje con el estilo Harvard:

Casals, M. y Vila A. (2024). *Introducción al lenguaje R, Rstudio y R Markdown.* [Recurso de aprendizaje textual]. 1.a ed. Fundació Universitat Oberta de Catalunya (FUOC).

Primera edición: febrero 2024

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Marta Casals Fontanet, Alícia Vila Grifo
Producción: FUOC

Índice

1. Introducción al LAB 1

El **LAB1** es un recurso didáctico complementario de la asignatura **Software para el análisis de datos (SAD)** del máster interuniversitario de Bioinformática y Bioestadística de la Universitat Oberta de Catalunya (UOC) y la Universitat de Barcelona (UB).

Este **LAB1** forma parte de un conjunto de laboratorios prácticos que conjugan contenidos teóricos con ejemplos, ejercicios y casos prácticos reales del ámbito de conocimiento del máster.

El **LAB1** explica los conceptos básicos introductorios de programación en el lenguaje **R** y su aplicación al ámbito biosanitario. En realidad, aunque **R** es un lenguaje de programación, no suele utilizarse para desarrollar grandes proyectos sino más bien para realizar paquetes de funciones o bloques de programas que puedan ser reutilizados con un objetivo específico. Por ello, las aplicaciones más habituales suelen estar dirigidas a la extracción, manipulación, tratamiento, análisis e interpretación de conjuntos de datos con fines específicos.

En este **LAB1**, se introduce al estudiante al lenguaje de programación en **R** y al entorno de desarrollo **RStudio**. También introduciremos el lenguaje de marcado **R Markdown** para poder generar documentos en **R** que sean reproducibles y se puedan descargar en diferentes formatos.

2. Instrucciones del LAB1

El **LAB1** contiene una serie de apartados con introducciones teóricas, ejemplos y casos prácticos, además de otros ejercicios que se propondrán para que sean resueltos por el estudiante. La temporización y las pautas para trabajar este laboratorio serán indicadas en el aula de la asignatura. También, incluye la propuesta de solución de los diversos ejercicios y casos prácticos, que servirá para que el estudiante pueda autoevaluar las soluciones de los ejercicios realizados por él mismo.

En ocasiones, para realizar los ejercicios, se utilizarán conjuntos de datos de paquetes propios de **RStudio**, como es el caso de *biopsy*, *anorexia* y *birthwt* del paquete *MASS*, *iris*, *airquality* del paquete **datasets**, así como también otros conjuntos de datos abiertos de repositorios externos. Todos los ejercicios se realizarán en el entorno de desarrollo integrado **RStudio** (https://cran.rstudio.com/) y se usará **R** como lenguaje de programación para la informática estadística y los gráficos.

3. Contenidos del LAB1

En este **LAB1** trabajaremos los siguientes contenidos:

- Instalación y entorno de trabajo de **R** y **RStudio**.
- Interfaz de **R** y **RStudio**.
- Introducción al lenguaje **R Markdown**.

- Importación y exportación de archivos en **R** y **RStudio**.
- Gestión de paquetes en **R** y **RStudio**.
- Tipos de datos, operadores y estructuras de datos en **R**.
- Conjuntos de datos.
- Visualización de los datos.
- Ejercicios de práctica y casos prácticos en **R**.

4. Instalación de R y RStudio

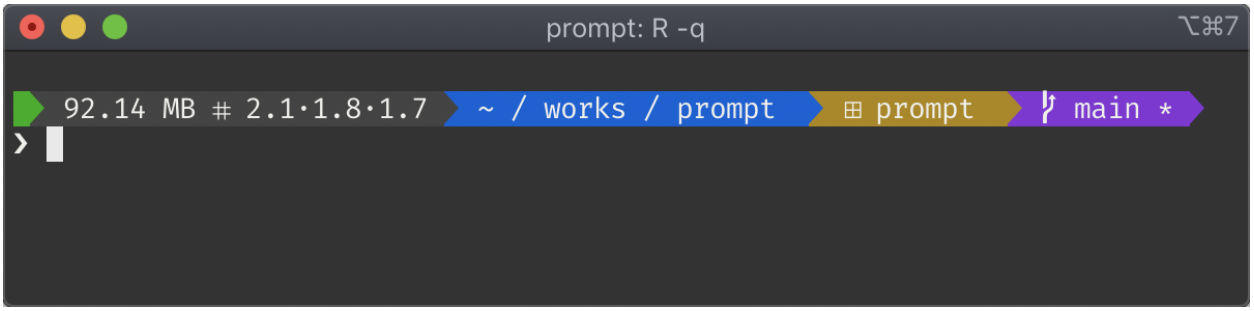
R es un lenguaje de programación centrado en el ámbito del análisis estadístico y el manejo de los datos. Originado a partir del lenguaje **S**, actualmente es uno de los lenguajes de programación más utilizados en los campos de biomedicina, bioinformática y minería de datos y posee la flexibilidad de disponer de un gran número de paquetes que proporcionan funcionalidades adicionales a nivel de cálculos y gráficos. **R** forma parte del sistema GNU y se compila y ejecuta en diferentes plataformas.

La web oficial de descarga de **R** es download R program. Es conveniente **acceder a las fuentes oficiales** para realizar la descarga del programa, ya que aquí encontraréis las versiones y la documentación de **R** actualizadas.

Para descargar **R**, seleccionad la versión en función de vuestra plataforma de trabajo:

- Descarga **R** para Linux.
- Descarga **R** para (Mac) OS X.
- Descarga **R** para Windows.

Una vez instalado el programa **R**, ya podemos empezar a trabajar. Si desde la consola general de vuestro ordenador escribís **R**, aparecerá un símbolo *prompt* (>), donde podéis empezar a escribir instrucciones en **R**.

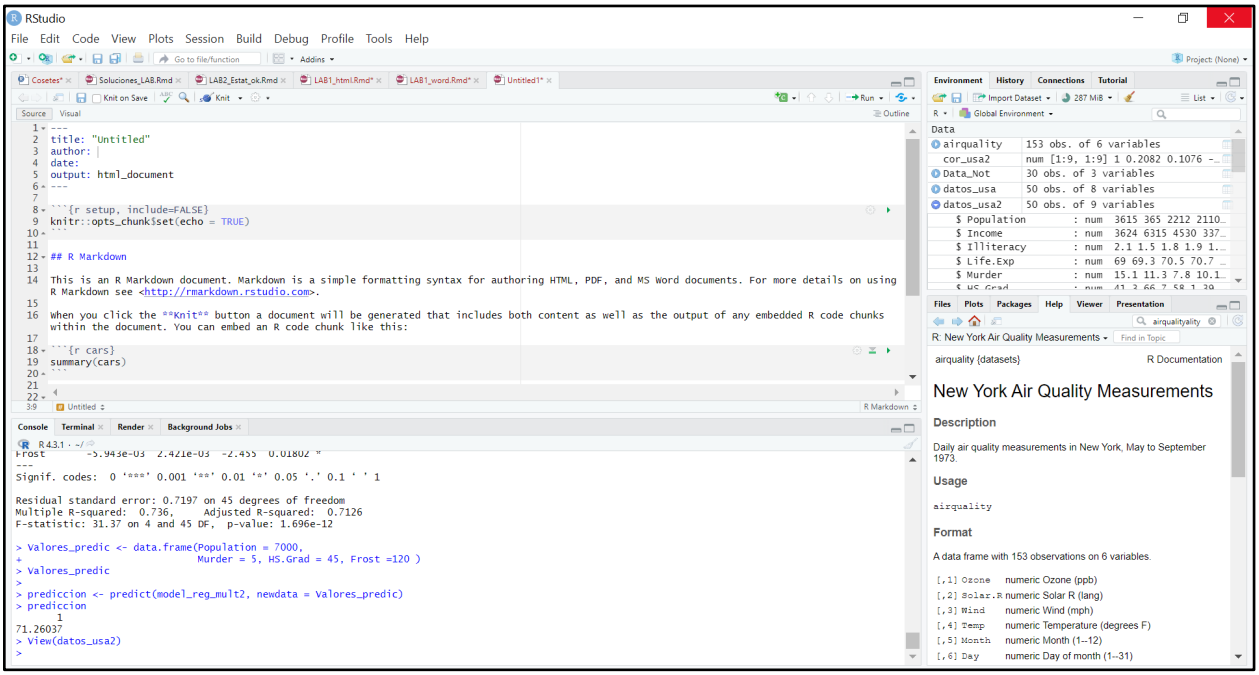


Para hacer más fácil el manejo del lenguaje **R** y llegar a generar todo tipo de documentos, os aconsejamos descargar el entorno de desarrollo **RStudio**, ya que es muy cómodo, sencillo de usar y nos ayuda en la generación de documentos para investigación que sean reproducibles.

En la web oficial de **RStudio** podéis descargar el programa (download RStudio). En esta página también tenéis opción de descargar el programa **R** sin problemas.

5. Interfaz de R y RStudio

La interfaz de **RStudio** se divide en cuatro ventanas (en la siguiente imagen), las de la izquierda corresponden al espacio de trabajo y las de la derecha ofrecen información diversa de los datos con los que estamos trabajando, paquetes instalados e información general de **R**.



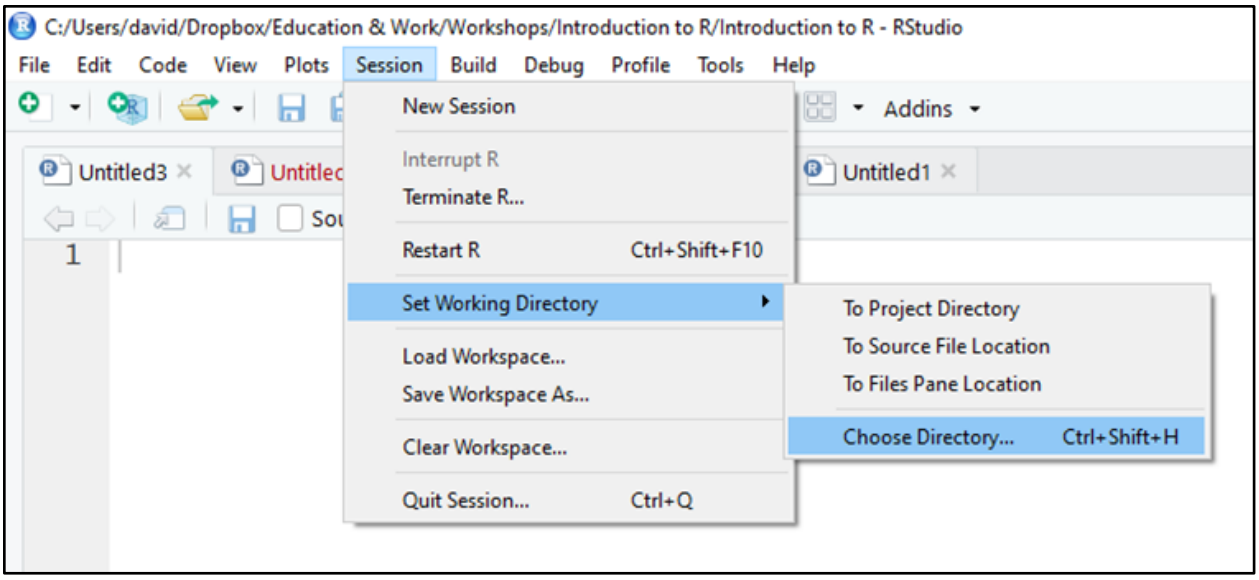
Para empezar a trabajar con el programa **R** desde el entorno **RStudio**, es suficiente con abrir el programa y seleccionar ***File > New File > R Script***.

A partir de este momento, visualizaremos las cuatro ventanas del entorno de **RStudio** y, una vez escrito el código en el espacio de trabajo y ciclando ***Run*** o bien ***Ctrl+R***, queda ejecutado el código en **R**. El resultado lo podremos visualizar en la ventana inferior, que corresponde también al trabajo en modo consola. El script realizado se guardará desde ***File > Save As*** obteniendo un archivo en formato «***.R***».

Antes de empezar a trabajar, **es necesario definir la ubicación de los archivos de trabajo**, ya que, si no hacemos este paso al inicio, tendremos problemas para ejecutar bien nuestros ficheros.

Es suficiente con ir a ***Session > Set working directory > Choose Directory*** o usar las teclas ***CTRL + shift + H***.

La función *setwd()* permite cambiar la ubicación de los archivos desde la consola en cualquier momento.

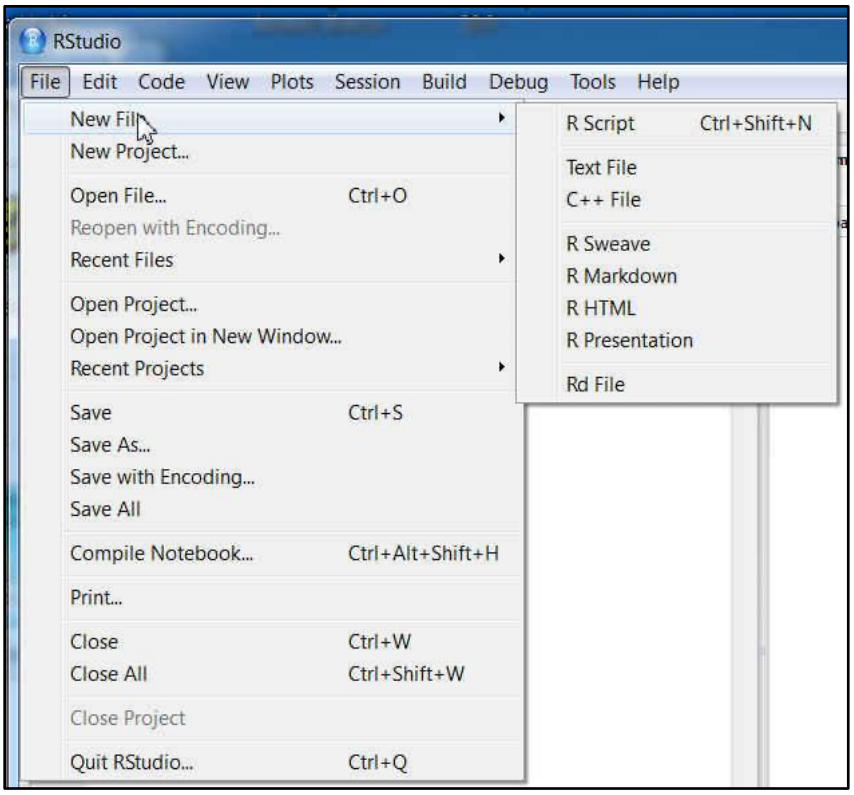


Es muy útil y recomendable utilizar la ayuda del programa que encontraréis en la pestaña **Help** del panel derecho si tenéis cualquier duda.

También podéis visualizar la ayuda sobre una determinada función precediendo esta de un interrogante (?), como, por ejemplo:

```
?print
```

En el margen superior de **RStudio** encontraréis menús contextuales que os pueden ayudar en cualquiera de vuestros trabajos.



Al trabajar con **R** y **R Markdown**, serán de mucha utilidad los siguientes menús:

- **File**: encontraréis todas las opciones de los documentos. Por ejemplo, crear, guardar, renombrar, etc.
- **Edit**: podréis editar el texto, buscar y remplazar palabras, deshacer acciones de texto, cortar, copiar y enganchar, etc.
- **View**: hallaréis todas las opciones de visualización que podáis necesitar.
- **Code**: encontraréis las opciones más directas con el espacio de trabajo. Por ejemplo, ejecutar líneas, trozos de código, buscar funciones y objetos de **R** de forma fácil y rápida, etc.
- **Plot**: se usa para todo aquello que necesitéis hacer relativo a los gráficos ya generados como, por ejemplo, guardarlos en algún formato, buscar algún gráfico específico, eliminar gráficos, copiarlos en el cortapapeles, etc.
- **Session**: en este menú podréis configurar el directorio, interrumpir, limpiar o terminar una sesión.
- **Tools**: sobre todo se usa para la instalación y manipulación de paquetes de R.
- **Help**: se encuentran los comandos para buscar cualquier tipo de ayuda del programa, funciones, etc.

En el ámbito estadístico, es muy habitual redactar un informe en el que se necesite intercalar código y también texto; para ello, existe una herramienta llamada **R Markdown**, que permite generar un documento claro y amigable en diversos formatos (docx, html, pdf).

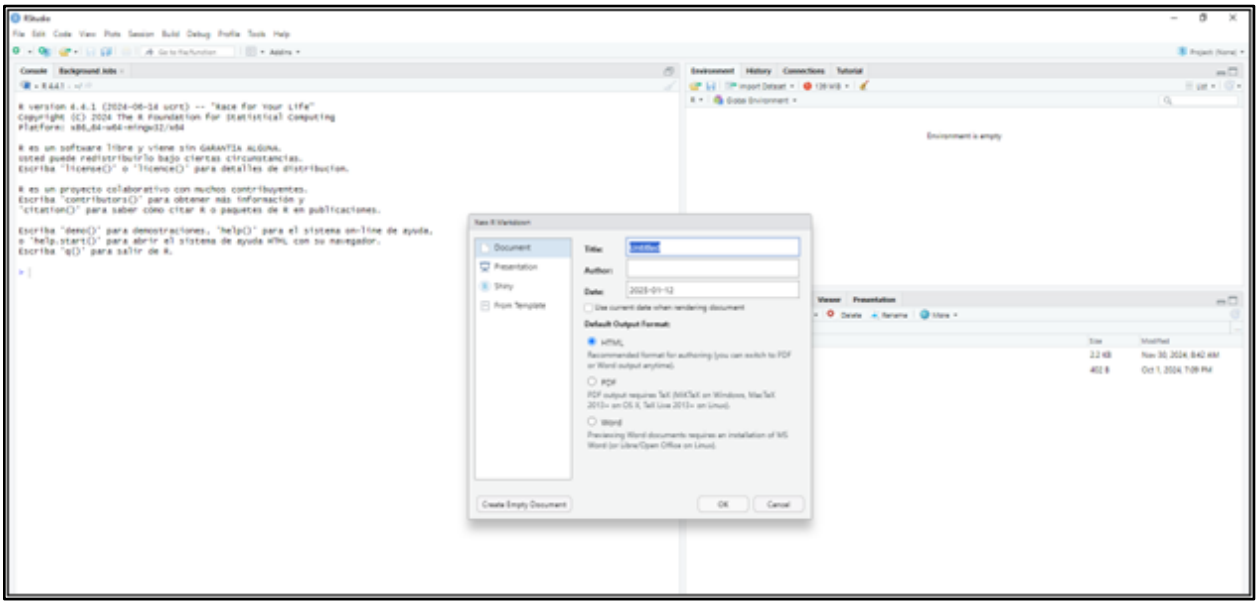
Por ejemplo, este **LAB1** se ha editado en **R Markdown** y se ha generado un documento «*.doc*» y «*.html*».

Si queremos generar un nuevo documento usando **R Markdown**, es suficiente con acceder a ***File > New File > R Markdown*** y, sobre la base de unas reglas de edición específicas, podréis redactar el documento.

¡IMPORTANTE!: os recomendamos encarecidamente generar las prácticas o los laboratorios de esta asignatura en R Markdown, ya que las PECs se deben entregar en R Markdown.

6. Introducción a R Markdown

Una vez generado un archivo nuevo ***File > New File > R Markdown*** y seleccionado el formato, encontraremos un documento con un archivo como el de la imagen siguiente con tres clases de elementos diferentes:



- **La cabecera**: como vemos, la plantilla del documento comienza con una cabecera limitada por tres guiones (- - -) por encima y por debajo, donde figuran el título del documento, el autor, la fecha y el formato de salida (html en este caso). En esta cabecera se pueden incluir otras instrucciones para especificar otros formatos de salida, el aspecto de la salida (colores, tamaño de letra), etc.
- **Los *chunks***: son las cajitas grises que contienen código **R**. Estas cajas están enmarcadas por tres acentos graves (```) al inicio y al final. En la primera línea de la caja, junto a los tres acentos y entre llaves se puede asignar un nombre a cada *chunk*, así como diversas opciones sobre el comportamiento de este. Así, por ejemplo, la opción **echo=TRUE** indica que el contenido del *chunk* se muestra en la salida, y **echo=FALSE** que no se muestra.
- **El texto**: se escribe directamente en el editor sobre el fondo blanco y le podremos dar formato.

En definitiva, en este tipo de documentos, podemos insertar títulos, enlaces, imágenes, tablas y generar código de programación **ejecutable y reproducible en R**.

Posteriormente, es suficiente con clicar el botón **Knit** de la barra de menú de la primera ventana y automáticamente se generará el documento en el formato definido previamente.

Es importante tener en cuenta que, inicialmente, para utilizar este tipo de formatos se requiere instalar paquetes adicionales que os pedirán por defecto la primera vez que ejecutéis un fichero. En **RStudio**, este formato tendrá extensión «*.Rmd*».

Elementos principales para poder generar un documento básico en **R Markdown**:

1) ***Encabezados***
2) ***Citas***
3) ***Listas***
4) ***Tablas***
5) ***Enlaces e imágenes***
6) ***Elementos para remarcar el texto***
7) ***Código ejecutable en R***
8) ***Fórmulas matemáticas y estadísticas***
9) ***Exportación de documentos en R Markdown***

6.1. Encabezados

Las almohadillas (#) servirán en Markdown para crear encabezados (títulos de sección o párrafo). Deberéis usarlos añadiendo uno por cada nivel de la siguiente forma:

```
# Encabezado 1
## Encabezado 2
### Encabezado 3
#### Encabezado 4
##### Encabezado 5
###### Encabezado 6
```

Encabezado 1

Encabezado 2

Encabezado 3

Encabezado 4

Encabezado 5

Encabezado 6

6.2. Citas

Las citas se generan usando el carácter *mayor que* (>) al comienzo del bloque de texto.

```
> La verdadera educación consiste en obtener lo mejor de uno mismo - Mahatma Gandhi.
```

> ***La verdadera educación consiste en obtener lo mejor de uno mismo - Mahatma Gandhi.***

Si la cita en cuestión se compone de varios párrafos, deberéis añadir el mismo símbolo > al comienzo de cada uno de ellos.

```
 > La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y,
 > por regla general, pueden ser expresadas en un lenguaje comprensible para todos - Albert Einstein.
```

> La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general, pueden ser expresadas en un lenguaje comprensible para todos - Albert Einstein.

6.3. Listas

Generar listas en **R Markdown** es mucho más sencillo que en **HTML**. Para ello podéis usar asteriscos (*), guiones (-) o símbolos de suma (+). Al igual que ocurre con las listas desordenadas, también podréis anidarlas o combinarlas.

```
1. Elemento 1
2. Elemento 2
  1. Elemento 1 dentro del segundo nivel
        *Elemento 1 del tercer nivel
        *Elemento 2 del tercer nivel
   2. Elemento 2 dentro del segundo nivel
```

1. Elemento 1
2. Elemento 2
    a. Elemento 1 dentro del segundo nivel
        - Elemento 1 del tercer nivel
        - Elemento 2 del tercer nivel
    b. Elemento 2 dentro del segundo nivel

6.4. Tablas

La generación de tablas de contenido en **R Markdown** es relativamente sencilla. Encontramos dos opciones sencillas para crear tablas editables. En primer lugar, podemos generar una tabla de **forma manual** usando una codificación específica que veremos a continuación.

Por otro lado, podemos usar un Generador de Tablas que nos permitirá generar un código que podremos copiar directamente al documento.

La estructura general de una tabla sería la siguiente:

```
| Left-Aligned  | Center Aligned  | Right Aligned |
|:------------- |:---------------:| -------------:|
| Row 1         | Cell 2          | Cell 3        |
| Row 2         | Cell 5          | Cell 6        |
| Row 3         | Cell 8          | Cell 9        |
```

Por ejemplo, queremos una tabla de tres filas con cuatro columnas en que los nombres de las columnas estén centrados. Podremos hacer lo siguiente:

```
| Título 1 | Título 2 | Título 3 | Título 4|
|:-------- |:--------:| --------:| -------:|
| 1        |  2       |  3       | 11      |
| 4        |  5       |  6       | 21      |
| 6        |  55      | 56       | 23      |
```

El resultado de nuestra tabla sería el siguiente:

Título 1	Título 2	Título 3	Título 4
1	2	3	11
4	5	6	21
6	55	56	23

Siempre se podrán modificar filas, columnas, títulos y cualquier formato al gusto de cada editor.

6.5. Enlaces e imágenes

6.5.1. Enlaces en línea

Para generar un enlace debemos usar **corchetes ([])** para poner el nombre que queremos que se vea y **paréntesis ()** para el enlace concreto, tal como se indica a continuación:

```
[Enlace UOC](https://uoc.edu/)
```

Enlace UOC

6.5.2. Imágenes

Para incluir una imagen en el documento, se hace como los enlaces y añadiendo un signo de exclamación (!) delante del código.

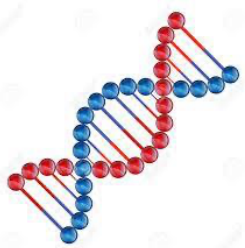
```
#![Texto opcional](/ruta/a/la/imagen.jpg){width=width height=height}
```

Podemos añadir medidas de amplitud y altura de la imagen con los elementos ***width*** **y** ***height***. También podréis añadir un título alternativo entrecomillándolo al final de la ruta. Esto sería el título que se muestra al dejar el cursor del ratón sobre la imagen.

Por ejemplo, escribiremos:

```
![](ADN.jpg "ADN"){width="100"}
```

y visualizaremos la siguiente imagen:



6.6. Elementos para remarcar el texto

Los elementos de énfasis más usados en párrafos de texto son la **negrita**, la *cursiva* y el subrayado. En **R Markdown** realzar el texto es muy sencillo.

- **Cursiva**: se usa un asterisco (*) delante y detrás del texto que hay que remarcar:

```
*letra en cursiva*
```

letra en cursiva

- **Negrita**: se usan dos asteriscos (**) delante y detrás del texto que hay que remarcar:

```
**letra en negrita**
```

letra en negrita

- **Subrayado**: se recomienda no subrayar en **R Markdown** para no confundir con los hipervínculos.
- **Tachado**: se usan dos virgulillas (~~) delante y detrás del texto que hay que remarcar.

```
~~letra tachada~~
```

~~letra tachada~~

6.7. Código ejecutable en R

Una de las funcionalidades más importantes de **RStudio** es incorporar código de **R** en los documentos. Para insertar código ejecutable en **R**, deberemos usar **chunks** o trozos de código. Para insertar un *chunk* en el documento, podemos ir al menú principal ***Code > Insert Chunk*** o usar las teclas **ALT + CTRL + I**. También podemos usar el 

que se encuentra en la parte superior de la ventana de edición de **RStudio**.

Las diferentes opciones de los *chunks* se pueden poner después del encabezado del fragmento de código.

- **eval**: si escribimos la opción *eval = TRUE*, se evalúa el fragmento de código para su ejecución. Si escribimos *FALSE* no se evaluará.
- **error**: si escribimos *error = TRUE*, indica que sí se muestren los mensajes de error, en caso de que el código contenga algún problema en el diseño o su evaluación.
- **echo**: si escribimos la opción *echo = TRUE*, se verá el código en el documento. En caso contrario, solo el resultado.
- **include**: si escribimos *include = FALSE*, evitaremos que el código y los resultados aparezcan en el archivo terminado. **R Markdown** aún ejecuta el código en el fragmento y los resultados pueden ser utilizados por otros fragmentos.
- **message**: *message = FALSE* evita que los mensajes generados por el código aparezcan en el archivo terminado.
- **warning**: *warning = FALSE* evita que las advertencias generadas por el código aparezcan en el archivo terminado.
- **fig.cap**: si escribimos *fig.cap = "..."*, agregaremos un título a los resultados gráficos.

Si queremos poner más de una opción en un mismo trozo de código, lo podremos hacer separando cada elemento por comas. Lo veremos en ejemplos más adelante.

6.8. Fórmulas matemáticas y estadísticas

El lenguaje **Markdown** ha heredado el potencial de **LaTeX** para generar fórmulas matemáticas, pero lo ha adaptado generando una tipografía adecuada y elegante. Para añadir fórmulas matemáticas solo tenemos que introducir un código que empieza siempre por un $ y usamos lenguaje **LaTeX** a continuación:

- Para las fórmulas o ecuaciones en una misma línea (*inline equations*), se pone el código entre dos dólares: `$ código $`
- Para las fórmulas o ecuaciones entre líneas (*display equations*), se pone el código entre cuatro dólares: `$$ código $$`

A continuación, se muestra un ejemplo de código para qué veáis cómo funciona:

```
$\( \left[\overline{X} -z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \overline{X}+z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \right] \)$
```

La fórmula matemática anterior se podrá visualizar de la siguiente forma:

$$\left[\overline{X} - z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \overline{X} + z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}\right]$$

Para generar ecuaciones matemáticas con **LaTeX**, podéis usar un editor en línea como Numberempire o Codecogs.

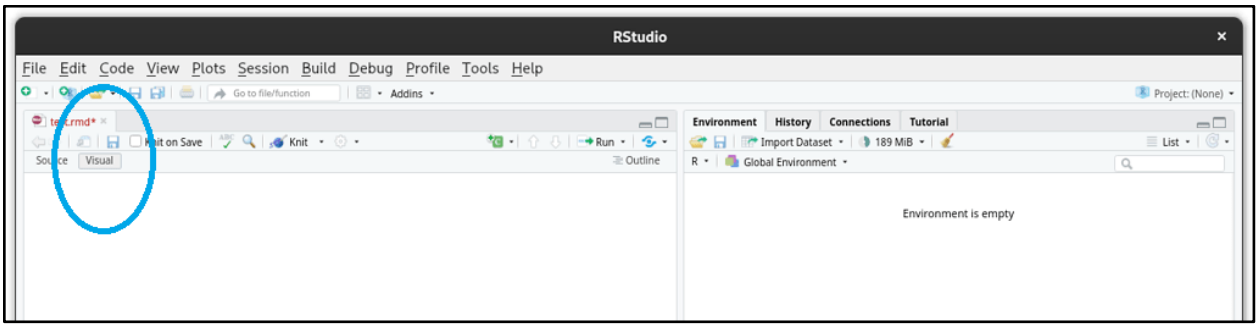
6.9. Exportación de documentos en R Markdown

Desde la versión 0.98.932 de **RStudio** es muy sencillo generar documentos **R Markdown** que puedan exportarse. Estos documentos se pueden dar en tres formatos diferentes:

1) **html**: para ser visualizados a través de navegadores web. Existe la posibilidad de que el documento se exporte como una página web o como una presentación html (similar a PowerPoint).
2) **doc**: para ser directamente editados con **Microsoft Word** o **LibreOffice Writer**.
3) **pdf**: para esta última opción es preciso que el ordenador del usuario disponga de una instalación válida de **LaTeX**.

LaTeX es un completo (y complejo) sistema de edición de textos de código abierto que puede descargarse libremente para **Windows** (MiKTeX), para **Mac** (MacTeX) y para Linux (TeX Live), si bien lo habitual es que en los sistemas Linux TeX Live venga ya instalado por defecto. Podéis descargar MiKTeX en el siguiente enlace: https://miktex.org/download.

Nota: antes de exportar un documento, lo podemos previsualizar y hacer cambios en él pulsando el botón *Visual*:



Ejemplo 1:

Creamos un documento en **R Markdown** que se exporte a una página **HTML** y que contenga un párrafo de texto en negrita y uno en cursiva, una imagen y un enlace. Para hacerlo generamos el código que veréis en la siguiente imagen:

El resultado de nuestra página es el siguiente:

6.10. RPubs

Es importante explicar en este laboratorio cómo funciona **RPubs**, ya que puede ser de mucha utilidad para estudiantes y profesorado.

RPubs es un servidor con cientos de ejercicios y prácticas para **R** en todos los idiomas. También hay una galería de ejemplos de gráficos que se pueden obtener con **R**. Incluye el código reproducible utilizado para cada uno. La página web es: https://rpubs.com/

7. Gestión de paquetes en R y RStudio

R contiene un conjunto estándar de **paquetes** que son colecciones de funciones en R, datos y código abierto, compilado para un determinado propósito. El directorio donde se almacenan los paquetes se llama biblioteca (*library*) y, en diversas ocasiones, será necesario instalar paquetes adicionales según las funcionalidades que necesitemos utilizar. Una vez instalados, deben cargarse en la sesión para ser utilizados.

Para instalar paquetes en R, deberemos colocarnos en la *consola de RStudio* y escribir el comando ***> install.packages("nombre del paquete")*** y clicar **INTRO**, también podemos ir al menú ***Tools > Install packages*** o también, por último, podemos ir a la ventana inferior derecha y seleccionar el botón **Install**. Es conveniente seleccionar la opción **Install dependencies** que aparece, ya que en ocasiones algunos paquetes requieren la previa instalación de otros.

En la imagen anterior, visualizamos los tres métodos de instalación de paquetes. Además de la instalación, las siguientes funciones aportan información acerca de qué paquetes tenemos instalados:

```
sessionInfo() #Informa de la versión de R y de los paquetes cargados
```

```
## R version 4.3.1 (2023-06-16 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=Catalan_Spain.utf8  LC_CTYPE=Catalan_Spain.utf8
## [3] LC_MONETARY=Catalan_Spain.utf8 LC_NUMERIC=C
## [5] LC_TIME=Catalan_Spain.utf8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.3.1    fastmap_1.1.1     cli_3.6.1         tools_4.3.1
##  [5] htmltools_0.5.5   rstudioapi_0.15.0 yaml_2.3.7        rmarkdown_2.23
##  [9] knitr_1.43        xfun_0.39         digest_0.6.33     rlang_1.1.1
## [13] evaluate_0.21
```

```
library() #Informa de los paquetes instalados en RStudio y de la ruta de
ubicación
.libPaths() #Informa de la ruta de Library
```

```
## [1] "C:/Users/mcasal8/AppData/Local/Programs/R/R-4.3.1/library"
```

```r
search() #Lista de los paquetes cargados
```

```
## [1] ".GlobalEnv"        "package:stats"     "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"         "package:base"
```

A continuación, explicaremos brevemente unos cuantos paquetes, a modo de ejemplo, que tienen mucha utilidad para los estadísticos y analistas de datos:

- **dplyr**: es uno de los paquetes **R** más utilizados para tareas de ciencia de datos y aprendizaje automático.
- **ggplot2**: uno de los paquetes **R** más populares y ampliamente utilizados para la visualización de datos y el análisis exploratorio de datos. Con este paquete se pueden crear visualizaciones de datos interactivas.
- **KernLab**: este paquete se utiliza para regresión, clasificación, reducción de dimensionalidad, detección de anomalías y agrupamiento.
- **Brillante**: es un paquete **R** que se utiliza para crear una aplicación web interactiva para la ciencia de datos.

Ejemplo 2:

Uno de los paquetes más utilizados para trabajar con conjuntos de datos es el paquete *MASS* y también vioplot. Para instalarlos, haríamos lo siguiente:

```r
install.packages(c("MASS", "vioplot")) #instalamos los paquetes
```

Si además queremos visualizar unos datos concretos del paquete *iris*:

```r
library(knitr)
library("datasets")
data("iris") #cargamos los datos a RStudio
kable(head(iris)) #mostramos la información del conjunto de datos
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Si queremos también acceder a la documentación de un paquete, por ejemplo, **stats**:

```r
packageDescription("stats")
```

```
## Package: stats
## Version: 4.3.1
## Priority: base
```

```
## Title: The R Stats Package
## Author: R Core Team and contributors worldwide
## Maintainer: R Core Team <do-use-Contact-address@r-project.org>
## Contact: R-help mailing list <r-help@r-project.org>
## Description: R statistical functions.
## License: Part of R 4.3.1
## Imports: utils, grDevices, graphics
## Suggests: MASS, Matrix, SuppDists, methods, stats4
## NeedsCompilation: yes
## Built: R 4.3.1; x86_64-w64-mingw32; 2023-06-16 07:34:01 UTC; windows
##
## -- File: C:/Users/mcasal8/AppData/Local/Programs/R/R-
4.3.1/library/stats/Meta/package.rds
```

Otras funcionalidades para trabajar con paquetes de **R** son las siguientes:

Comando	**Utilización**
remove.packages()	Eliminar un paquete
old.packages()	Comprobar si los paquetes están actualizados
update.packages()	Actualizar todos los paquetes
install.packages()	Para actualizar solo un paquete

7.1. Repositorios

Un repositorio es el lugar donde están alojados los paquetes y desde el cual podemos descargarlos. Aunque nosotros dispongamos de un repositorio local, los paquetes son accesibles en línea para todo el mundo.

Los repositorios más populares de paquetes **R** son los siguientes:

CRAN: es el repositorio oficial compuesto de un conjunto de servidores web y ftp mantenidos por la comunidad **R**.

Para instalar paquetes en **CRAN** deberemos usar el código descrito anteriormente:

```
install.packages("datasets", repos="http://cran.rediris.es/")
```

Bioconductor: se trata de un repositorio específico para bioinformática. Tiene una comunidad muy activa que proporciona numerosas conferencias y encuentros a lo largo del año.

Para instalar paquetes como *GenomicFeatures* o *AnnotationBi* en **Bioconductor**:

```
source("https://bioconductor.org/biocLite.R")
biocLite(c("GenomicFeatures","AnnotationBi" ))
```

GitHub: a pesar de que no es específico para R, **GitHub** es con toda seguridad el repositorio más popular para la publicación de proyectos abiertos.

8. Importación de datos en R y RStudio

Para realizar nuestros estudios, necesitaremos utilizar datos externos que procederán de diversos tipos de archivos y que necesitaremos importar a **R** para poder trabajar con ellos.

En este **LAB1** estudiaremos las instrucciones y herramientas de **R** para poder leer los formatos de archivos más comunes. Para ello, utilizaremos el comando *read()* seguido por el formato del tipo de archivo que hay que leer, por ejemplo, **read.csv** si el formato del archivo es csv o *read.SAS()* si el formato es un archivo de SAS. Recordad que debéis importar la librería *readr*.

En la siguiente tabla podéis ver un resumen:

Datos	**Comando**	**Extensión**
Texto	read.txt	.txt
CSV	read.csv	.csv
Excel	read_excel	.xls
SAS	read.Sas	.sa7bdata

Si ejecutamos *help(read)*, podemos visualizar todos los tipos que existen y la sintaxis de esta instrucción.

```
help(read)
```

Ejemplo 3:

Imaginemos que queremos importar un documento «*.csv*»; podremos usar el siguiente código con un fichero:

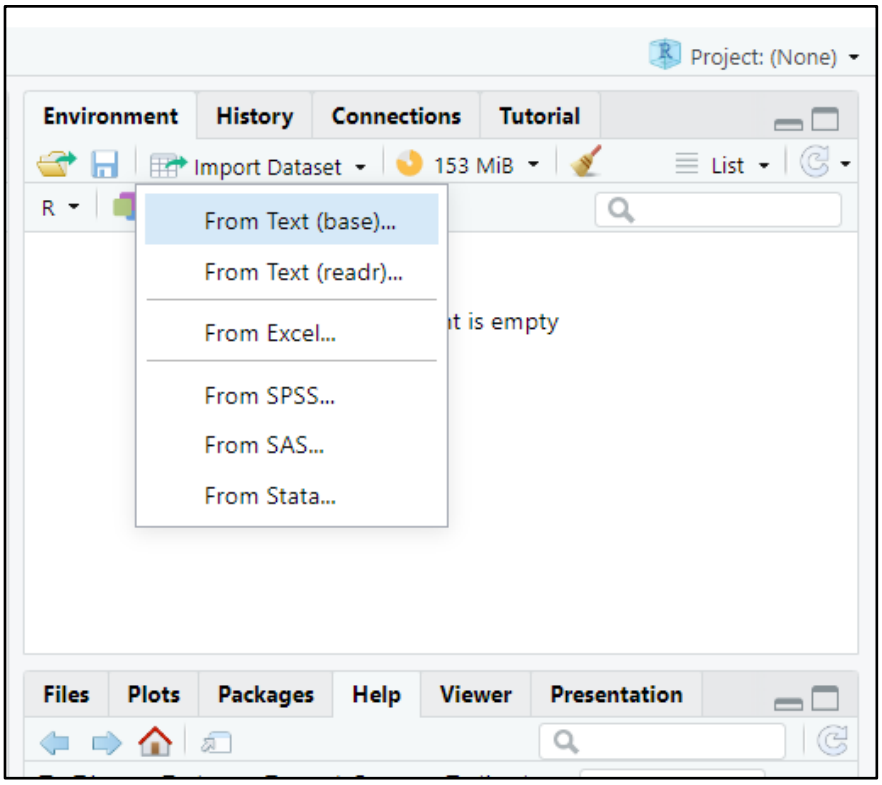
```
library(readr)
datasets <-
read_csv("C:/Users/mcasal8/Desktop/LAB1/CovidDeaths/CovidDeaths.csv")
View(CovidDeaths)
```

O un conjunto de datos en un fichero .xls con este código:

```
install.packages("readxl")
library(readxl)
dataset <- read_excel("C:/Users/mcasal8/Desktop/LAB1/CovidDeaths.xls")
View(dataset)
```

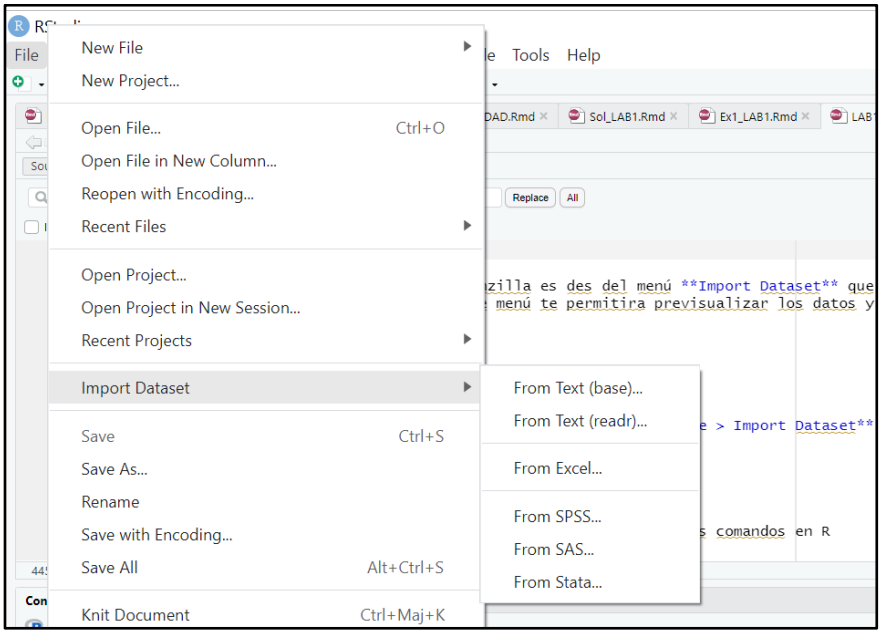
Nota: Podéis probar este código con vuestros propios ficheros.

Otra manera de importar **datasets** de forma sencilla es desde el menú **Import Dataset** que se encuentra en la ventana superior derecha de **RStudio**. Este menú os permitirá previsualizar los datos y ver si se importan de forma correcta.



Una vez abierto, hay que seleccionar las opciones específicas de importación.

Por último, también podremos importar desde el menú de ***RStudio File > Import Dataset*** en cualquier formato.



9. Exportación de datos en R

Para exportar datos en diferentes formatos podemos usar los diferentes comandos en R.

Los más comunes son los siguientes:

- Exportar datos a archivos de tipo **.txt** usando: *write.table(mydata,"ruta_archivo/ mydatasalida.txt",sep="")*

```
write.table("data","/home/mcasal8/Desktop/SAD_LAB1/Data1.txt",sep="")
```

- Exportar datos a archivos de tipo **.csv** usando: *write.csv(mydata, "ruta_archivo/ mydatasalida.csv")*

```
write.csv("data", file="/home/mcasal8/Desktop/SAD_LAB1/Data2.csv")
```

- Exportar datos a archivos **.xlsx** usando: *write.xlsx(mydata, "ruta_archivo/ mydatasalida.xlsx")*

```
library(xlsx)
write.xlsx("data","/home/mcasal8/Desktop/SAD_LAB1/Data3.xlsx")
```

- Exportar datos a archivos de tipo **SPSS** usando: *write.foreing(mydata."ruta_archivo/ mydatasalida.spss", package="SPSS")*

```
library(foreing)
write.foreign("data","/home/mcasal8/Desktop/SAD_LAB1/Data4.spss",package="SPS
S")
```

- Exportar datos a archivos de tipo **SAS** usando: *write.foreing(mydata."ruta_archivo/ mydatasalida.sas", package="SAS")*

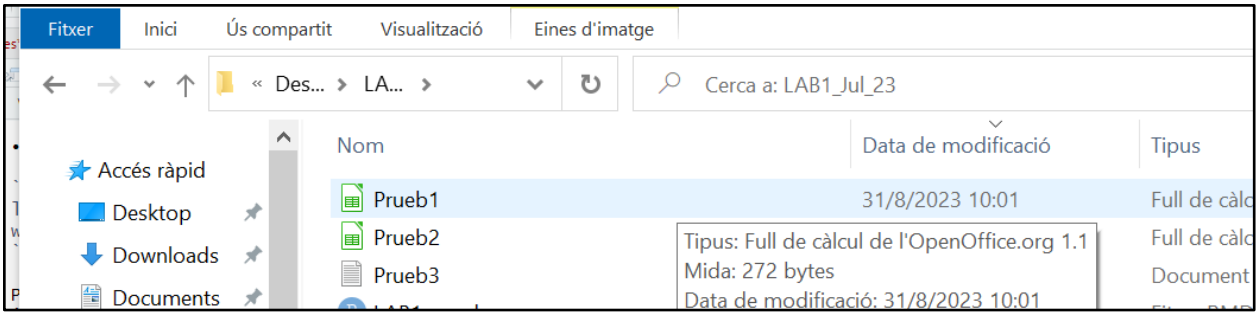
```
library(foreing)
write.foreign("data","/home/mcasal8/Desktop/SAD_LAB1/data5.sas",package="SAS"
)
```

Ejemplo 4:

Para ver cómo podemos hacer una exportación vamos a poner un ejemplo. Creamos un dataset inventado y ejecutamos las instrucciones:

```
flores = c(15,16,18,18,12,12,25,10,15,22,14,14,16,4,8,5,7,3,9,12) #flores
nacidas por m2
tratamiento =
c("s","s","s","s","s","s","s","s","s","s","n","n","n","n","n","n","n","n","n"
,"n")
dataflores =data.frame(tratamiento, flores)
write.csv(dataflores, file="Prueb1.csv") #exporto a un archivo .csv
write.csv(dataflores, file="Prueb2.csv", row.names = F) #elimino los números
de fila
write.table(dataflores, file="Prueb3.txt") #exporto a un archivo de texto
```

Los ficheros del resultado de la exportación los encontraremos en la carpeta de nuestro proyecto:



Por otro lado, si quisiéramos exportar un *output* o resultado como un resumen de una variable que sea de nuestro interés usaremos:

```
dataflores
resumen = summary(dataflores)
capture.output(resumen, file="resumen.doc")
```

10. Estructuras de datos en R

10.1. Tipología de datos

Los tipos de datos en **R** son los siguientes:

Tipos de datos	**Nombre**	**Expresión en R**
Datos numéricos	Numeric	num <- 1.0
Datos enteros	Integer	int <- -2
Datos lógicos	Logical	TRUE or FALSE
Factores	Factor	factor

10.2. Operadores

Los operadores principales de **R** son los siguientes:

- Operadores de asignación: ***<- (específico de R) o =***.

```
num1<-10 #asignamos el valor 10 al núm. 1
num2<-23 #asignamos el valor de 23 al núm. 2
num1= num2
num2

## [1] 23
```

- Operadores aritméticos: ***suma (+), resta (–), multiplicación (×), división (/), potencia (^), división entera (%%).***

```
num1<-10#asignamos valor a variable num1
num2<-20#asignamos valor a variable num2
num1+num2 #suma

## [1] 30

num2/num1 #división

## [1] 2

num2

## [1] 20
```

El uso más habitual de los operadores es en expresiones matemáticas de diversos tipos; por ejemplo, si queremos calcular el índice de masa corporal de un individuo, haríamos:

```
peso<-50 #asignamos el valor del peso en kg
altura<-1.60 #asignamos el valor de la altura en metros
IMC<-peso/((altura)^2) #valor del índice de masa corporal IMC
```

- Operadores relacionales: **menor que (<), menor o igual que (≤), mayor (>), mayor o igual que (≥), igual que (==), distinto de (!=)**.

```
num1==num2 #comprobamos si las variables son iguales

## [1] FALSE

num1<num2 #comparamos num1 y num2

## [1] TRUE

num1!=num2 #comprobamos si las variables son distintas

## [1] TRUE
```

- Operadores lógicos: **O (|), Y(&), negación (!)**.

```
bool1<-TRUE
bool2<-FALSE
bool1&bool2

## [1] FALSE

bool1|bool2

## [1] TRUE
```

10.3. Vectores

Son objetos elementales del lenguaje **R**. Los elementos que contiene son siempre del mismo tipo. Se genera con una función de concatenación.

```
vect <- c(1.5, 2, 3.5, 4) #crear un vector
vect #mostrar el vector

## [1] 1.5 2.0 3.5 4.0

is.vector(vect) #determinar si el objeto es un vector

## [1] TRUE
```

10.4. Secuencias numéricas

Se pueden generar secuencias numéricas de las siguientes formas:

```
1:22 #generamos una secuencia de números ordenados del 1 al 22.

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

seq1<- seq(0,20, by=0.5) #generamos una secuencia que va del 0 al 20 de 0.5
en 0.5.
seq1

## [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
## [16] 7.5 8.0 8.5 9.0 9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5
## [31] 15.0 15.5 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0
```

10.5. Matrices

Las matrices son estructuras bidimensionales organizadas por filas y columnas. Las podemos generar a partir de diferentes vectores como en este ejemplo:

```
data1 <- c(1:20) #creamos un vector a partir de una secuencia numérica
class(data1)

## [1] "integer"

data2 <- c(4,5) #creamos un vector con la dimensión de la matriz (núm. de
filas x núm. de columnas)
matriz <- matrix (data1, data2) #creamos la matriz
matriz #visualizamos la matriz

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
```

En otros laboratorios se explicará cómo operar con las matrices y sus funciones asociadas.

10.6. Listas

Las listas son colecciones ordenadas de elementos y datos de diferentes tipologías. A diferencia de los vectores y las matrices, pueden ser elementos diferentes y de diferentes longitudes. Las podemos crear de la siguiente forma:

```
vecta <- c(1, 2, 3, 4) #generamos un vector numérico
vectb <- c("a", "b", "c") #generamos un vector de texto
x <- matrix(1:12, ncol = 4) #generamos una matriz
lista <- list (vecta, vectb, x) #creamos una lista a partir de los elementos
anteriores que son de diferente tipología
lista #visualizamos la lista
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] "a" "b" "c"
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Si queremos llamar a algún elemento específico de la lista lo deberemos hacer con **corchetes simples [ ] o dobles [[ ]]**.

```
lista[1] #primer elemento de la lista
```

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

```
lista [[1]] # primera fila de la lista
```

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

```
lista [[3]][,1] #primera columna
```

```
## [1] 1 2 3
```

```
length(lista) #elementos de la lista
```

```
## [1] 3
```

```
length(lista[2])#elementos del segundo objeto de la lista
```

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

Para eliminar algunos elementos de una lista, podéis asignar el índice del elemento de la lista que queréis eliminar como ***NULL*** o indicar entre corchetes el índice con el operador "-". Si

deseáis eliminar varios elementos a la vez, podéis combinarlos con el comando *c()*, como se muestra a continuación:

```
lista
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] "a" "b" "c"
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
lista[[3]] <- NULL #eliminar el tercer elemento
lista[-3] #equivalente al anterior
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] "a" "b" "c"
```

```
#Eliminando el primer y segundo elementos a la vez
lista[-c(1, 2)]
```

```
## list()
```

10.7. Data frames

Los data frames o marcos de datos son estructuras de datos bidimensionales que se asemejan a una matriz, pero se diferencian en que pueden almacenar diferentes tipos de variables y valores.

11. Conjuntos de datos

En **R** trabajamos con conjuntos de datos; por un lado, podemos obtener **datasets** con la importación de ficheros. Pero en **R** también tenemos conjuntos de datos disponibles incluidos en paquetes que podemos usar y que podemos importar y analizar. Para ejemplificar estos contenidos, utilizaremos el conjunto de datos *birthwt* del paquete *MASS* de **R** que contiene datos de factores de riesgo asociados a los nacimientos de niños de bajo peso.

Ejemplo 5:

La descripción de este conjunto de datos puede consultarse desde la pestaña Información Paquete MASS

```
library (MASS) #cargamos el paquete MASS que tenemos previamente instalado
data("birthwt")#activamos los datos
View(birthwt) #muestra el conjunto de datos en formato tabla
dim (birthwt) #muestra el número de observaciones y el número de variables

## [1] 189  10

length(birthwt) #muestra el número de variables del conjunto

## [1] 10

head(birthwt, n=5) #muestra los primeros cinco registros

##    low age lwt race smoke ptl ht ui ftv  bwt
## 85   0  19 182    2     0   0  0  1   0 2523
## 86   0  33 155    3     0   0  0  0   3 2551
## 87   0  20 105    1     1   0  0  0   1 2557
## 88   0  21 108    1     1   0  0  1   2 2594
## 89   0  18 107    1     1   0  0  1   0 2600

names(birthwt) #muestra el nombre de las variables

##  [1] "low"   "age"   "lwt"   "race"  "smoke" "ptl"   "ht"    "ui"    "ftv"
## [10] "bwt"
```

Ejemplo 6:

Para acceder a datos dentro de un dataset, es parecido al acceso a los datos de una matriz. Veamos un ejemplo con un dataset generado por nosotros, al que llamaremos **MiDataf**:

```
genero <- c(1, 2, 1, 1, 1, 2, 2, 2, 1, 2)
edad <- c(24, 25, 26, 24, 25, 27, 21, 22, 25, 26)
fuma <- c("no", "sí", "no", "sí", "no", "no", "sí", "no", "no", "sí")
MiDataf <- data.frame(genero, edad, fuma)
MiDataf

##    genero edad fuma
## 1       1   24   no
## 2       2   25   sí
## 3       1   26   no
## 4       1   24   sí
## 5       1   25   no
## 6       2   27   no
## 7       2   21   sí
## 8       2   22   no
## 9       1   25   no
## 10      2   26   si
```

```
MiDataf[3:5, ] #datos de los elementos de las filas de la 3 a la 5
```

```
##   genero edad fuma
## 3      1   26   no
## 4      1   24   sí
## 5      1   25   no
```

```
MiDataf[,1] #datos de todos los elementos de la primera columna
```

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

Si queremos buscar alguna variable específica del conjunto de datos podemos usar:

```
MiDataf$genero #variable género
```

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

```
MiDataf$fuma #variable fuma
```

```
##  [1] "no" "sí" "no" "sí" "no" "no" "sí" "no" "no" "sí"
```

Si por ejemplo queremos hacer la media de la edad de los pacientes que no fuman:

```
mean(MiDataf$edad[MiDataf$fuma=="no"])
```

```
## [1] 24.83333
```

La forma anterior de acceder a los elementos de un data frame es bastante larga y puede llegar a ser tediosa, ya que debemos poner siempre el nombre del dataset y también el símbolo $; para poderlo hacer más eficiente, podemos usar las funciones *Attach()* y *Detach()*.

Estas funciones consisten básicamente en «enganchar» (*attach*) el contenido del data frame al entorno y así **R** busca los nombres de variables sin necesidad de más. De esta manera, se puede acceder a las variables de forma directa por su nombre. Si queremos desactivar este modo de trabajar, usaremos *detach()* y desengancharemos el data frame del entorno.

Ejemplo 7:

Podemos ver un ejemplo:

```
attach(MiDataf) #enganchamos el data frame
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     edad, fuma, genero
```

```
table(fuma, genero) #generamos una tabla cruzada con la variable fuma y la
variable género
```

```
##     genero
## fuma 1 2
##   no 4 2
##   sí 1 3
```

```
detach(MiDataf) #desenganchamos el data frame
```

El comando *with()* permite ejecutar una o varias instrucciones sobre las variables de un data frame accediendo a ellas solamente por su nombre, sin necesidad de utilizar attach y podemos hacer operaciones con ellas. Podemos ver un ejemplo:

```
with(MiDataf, {
edad_día = edad*365
edad_día
}) #usamos el comando with para calcular la edad en días y no en años

## [1] 8760 9125 9490 8760 9125 9855 7665 8030 9125 9490
```

11.1. Combinar conjuntos de datos

Si queremos combinar data frames podemos usar los comandos *rbind()* y *merge()* de la siguiente forma:

Ejemplo 8:

a) Si disponemos de dos data frames que tienen las mismas variables y datos diferentes, usaremos *rbind()*, como podéis observar en el siguiente ejemplo:

```
#creamos un primer data frame sobre enfermedades

Dataf_Enf1 = data.frame (enfermedad = c("diabetes", "colesterol",
"hipertensión", "hipotensión"), individuos= c("ind1", "ind2", "ind3",
"ind4"))
Dataf_Enf1 #observamos los elementos de Enf1

##     enfermedad individuos
## 1     diabetes       ind1
## 2   colesterol       ind2
## 3 hipertensión       ind3
## 4  hipotensión       ind4

#creamos un segundo data frame idéntico en variables

Dataf_Enf2 = data.frame (enfermedad = c("diabetes", "colesterol",
"hipertensión", "hipotensión"), individuos= c("ind21", "ind22", "ind23",
"ind24"))

Dataf_Enf2 #observamos los elementos de Enf2

##     enfermedad individuos
## 1     diabetes      ind21
## 2   colesterol      ind22
## 3 hipertensión      ind23
## 4  hipotensión      ind24

#combinamos los 2 data frames en 1
```

```
Dataf_Enf = rbind(Dataf_Enf1, Dataf_Enf2)
Dataf_Enf
```

```
##     enfermedad individuos
## 1     diabetes       ind1
## 2   colesterol       ind2
## 3 hipertensión       ind3
## 4  hipotensión       ind4
## 5     diabetes      ind21
## 6   colesterol      ind22
## 7 hipertensión      ind23
## 8  hipotensión      ind24
```

El inconveniente en la fusión de data frames con *rbind()* es que no podemos controlar las repeticiones.

b) Si los data frames tienen una estructura distinta, pero contienen variables en común que permiten identificar los mismos objetos en cada uno de los dos data frames, se pueden combinar con el comando *merge()*. Si queremos controlar las repeticiones, podemos usar la opción *all()*.

Ejemplo 9:

Vamos a ver un ejemplo de uso del comando *merge()*. Creamos un data frame inventado sobre unos pacientes y su medicación:

```
Indiv1 <- c("I213", "I214", "I215", "I216", "I217")
Medic1 <- c("Paracetamol", "Ibuprofeno", "Aspirina", "Ibuprofeno",
"Paracetamol")
Past_día <- c(2, 3, 2, 2, 2)

df_medica1 <- data.frame (Indiv1, Medic1, Past_día)
df_medica1
```

```
##   Indiv1      Medic1 Past_día
## 1   I213 Paracetamol        2
## 2   I214  Ibuprofeno        3
## 3   I215    Aspirina        2
## 4   I216  Ibuprofeno        2
## 5   I217 Paracetamol        2
```

```
Indiv2 <- c("I213", "I214", "I215", "I216", "I217")
Medic2 <- c("Paracetamol", "Ibuprofeno", "Aspirina", "Ibuprofeno",
"Paracetamol")
Past_día <- c(2, 3, 2, 2, 2)

df_medica2 <- data.frame (Indiv2, Medic2, Past_día)
df_medica2
```

```
##   Indiv1      Medic1 Past_día
## 1   I213 Paracetamol        2
```

```
## 2   I214  Ibuprofeno        3
## 3   I215    Aspirina        2
## 4   I216  Ibuprofeno        2
## 5   I217 Paracetamol        2
```

```
merge(df_medica1, df_medica2)
```

```
##   Indiv1      Medic1 Past_día
## 1   I213 Paracetamol        2
## 2   I214  Ibuprofeno        3
## 3   I215    Aspirina        2
## 4   I216  Ibuprofeno        2
## 5   I217 Paracetamol        2
```

Si os fijáis en esta unión, veréis que *merge()* funciona fusionando los dos data frames anteriores. Para hacerlo, buscará las variables con el mismo nombre y no habrá repeticiones.

Ejemplo 10:

Vamos a crear otro ejemplo de data frame para poder explicar los diferentes tipos de unión que puede generar el comando *merge()*:

```
set.seed(999) #semilla aleatoria

médico_id <- 1:10 #generamos una variable con una lista ordenada para la id
de cada médico
médico_nombre <- c("Ona", "Jordi", "Oriol", "Pau", "Esther", "Xavi", "Jan",
"Marta", "Anna", "Abril") #variables con el nombre de cada profesional
médico_sal <- round(rnorm(10, mean = 1500, sd = 200)) #salario estimado
aleatorio que cobra cada profesional
médico_edad <- round(rnorm(10, mean = 50, sd = 8)) #variable aleatoria sobre
la edad
médico_espec <- c("Neuro", "Orto", "Gine", "Trauma",rep("General",
6))#especialidad de cada profesional

df_Med_1 <- data.frame(id = médico_id[1:8], nombre = médico_nombre[1:8],
                   salario_mensual = médico_sal[1:8])
df_Med_2 <- data.frame(id = médico_id[-5], nombre = médico_nombre[-5],
                   edad = médico_edad[-5], position = médico_espec[-5])

df_Med_1
```

```
##   id nombre salario_mensual
## 1  1    Ona            1444
## 2  2  Jordi            1237
## 3  3  Oriol            1659
## 4  4    Pau            1554
## 5  5 Esther            1445
## 6  6   Xavi            1387
## 7  7    Jan            1124
## 8  8  Marta            1247
```

```
df_Med_2
```

```
##   id nombre edad position
## 1  1    Ona   61    Neuro
## 2  2  Jordi   51     Orto
## 3  3  Oriol   58     Gine
## 4  4    Pau   51   Trauma
## 5  6   Xavi   39  General
## 6  7    Jan   51  General
## 7  8  Marta   51  General
## 8  9   Anna   57  General
## 9 10  Abril   33  General
```

Podemos unir los data frames con la función *merge()* de diferentes formas:

a) **Unión interna o Inner Join**

Esta es la unión más habitual y usada en la combinación de conjunto de datos. La hemos utilizado en el ejemplo anterior. Se trata de fusionar los dos conjuntos de datos en uno solo que contenga los elementos comunes. El comando *merge()* los unirá a partir de los nombres de las columnas (variables) que sean iguales.

```
merge(x=df_Med_1, y=df_Med_2) #escribir x o y es opcional en este comando.
```

```
##   id nombre salario_mensual edad position
## 1  1    Ona            1444   61    Neuro
## 2  2  Jordi            1237   51     Orto
## 3  3  Oriol            1659   58     Gine
## 4  4    Pau            1554   51   Trauma
## 5  6   Xavi            1387   39  General
## 6  7    Jan            1124   51  General
## 7  8  Marta            1247   51  General
```

b) **Combinación completa (externa) o Full (outer) Join**

Esta opción es la que genera una fusión completa, es decir, se combinan todas las columnas de los dos conjuntos de datos en uno solo para todos los elementos. Para hacerlo debemos establecer la opción *all=TRUE*:

```
merge (x=df_Med_1, y=df_Med_2, all = TRUE)
```

```
##    id nombre salario_mensual edad position
## 1   1    Ona            1444   61    Neuro
## 2   2  Jordi            1237   51     Orto
## 3   3  Oriol            1659   58     Gine
## 4   4    Pau            1554   51   Trauma
## 5   5 Esther            1445   NA     <NA>
## 6   6   Xavi            1387   39  General
## 7   7    Jan            1124   51  General
## 8   8  Marta            1247   51  General
## 9   9   Anna              NA   57  General
## 10 10  Abril              NA   33  General
```

c) **Unión izquierda (externa) o Left (outer) Join**

Esta opción en **R** consiste en unir todas las filas del primer data frame con los valores correspondientes del segundo. Para hacerlo, deberemos poner la opción *all.x = TRUE* de la siguiente forma:

```
Df_UniIzq <- merge(x=df_Med_1, y=df_Med_2, all.x= TRUE)
Df_UniIzq
```

```
##   id nombre salario_mensual edad position
## 1  1    Ona            1444   61    Neuro
## 2  2  Jordi            1237   51     Orto
## 3  3  Oriol            1659   58     Gine
## 4  4    Pau            1554   51   Trauma
## 5  5 Esther            1445   NA     <NA>
## 6  6   Xavi            1387   39  General
## 7  7    Jan            1124   51  General
## 8  8  Marta            1247   51  General
```

d) **Unión derecha (externa) o Right (outer) Join**

En este caso, la combinación se hará uniendo todas las filas del segundo data frame con las correspondientes del primero. Para hacerlo, deberemos poner la opción *all.y = TRUE* de la siguiente forma:

```
Df_UniDer <- merge(x=df_Med_1, y=df_Med_2, all.y= TRUE)
Df_UniDer
```

```
##   id nombre salario_mensual edad position
## 1  1    Ona            1444   61    Neuro
## 2  2  Jordi            1237   51     Orto
## 3  3  Oriol            1659   58     Gine
## 4  4    Pau            1554   51   Trauma
## 5  6   Xavi            1387   39  General
## 6  7    Jan            1124   51  General
## 7  8  Marta            1247   51  General
## 8  9   Anna              NA   57  General
## 9 10  Abril              NA   33  General
```

e) **Unión cruzada o Cross Join**

En este caso se realiza el producto cartesiano de los dos conjuntos de datos. Para hacerlo así, deberemos establecer como *NULL* la opción *by*. Se debe tener en cuenta que el resultado de este cruce genera muchos registros.

```
Df_Cruzado <- merge (x=df_Med_1, y=df_Med_2, by=NULL)
head(Df_Cruzado) #ponemos solo las primeras filas de la combinación
```

```
##   id.x nombre.x salario_mensual id.y nombre.y edad position
## 1    1      Ona            1444    1      Ona   61    Neuro
```

```
## 2    2    Jordi            1237    1      Ona    61    Neuro
## 3    3    Oriol            1659    1      Ona    61    Neuro
## 4    4      Pau            1554    1      Ona    61    Neuro
## 5    5   Esther            1445    1      Ona    61    Neuro
## 6    6     Xavi            1387    1      Ona    61    Neuro
```

11.2. Seleccionar y filtrar registros

En muchas ocasiones necesitaremos seleccionar datos o filtrar registros. Hay varias funciones que nos permiten filtrar además de los **corchetes []** (que hemos visto anteriormente) y son las siguientes:

a) La función *subset()* es un comando utilizado para seleccionar variables, observaciones o niveles de una variable.

Ejemplo 11:

Vamos a ver un ejemplo de su funcionamiento con diferentes opciones. Creamos un nuevo dataset:

```
Id<-
c("I1","I2","I3","I4","I5","I6","I7","I8","I9","I10","I11","I12","I13","I14",
"I15","I16","I17","I18","I19","I20","I21","I22")
Edad <- c(23,24,21,22,23,25,26,24,21,22,23,25,26,24,22,21,25,26,24,21,25,27)
Sexo <-c(1,2,1,1,1,2,2,2,1,2,1,2,2,2,1,1,1,2,2,2,1,2)
Peso <-c(76.5, 81.2, 79.3, 59.5, 67.3, 78.6, 67.9, 100.2, 97.8, 56.4, 65.4,
67.5, 87.4, 99.7, 87.6, 93.4, 65.4, 73.7, 85.1, 61.2, 54.8, 103.4)
Altura <-
c(165,154,178,165,164,175,182,165,178,165,158,183,184,164,189,167,182,179,165
,158,183,184)
Pacientes <- data.frame (Id, Edad, Sexo, Peso, Altura)
```

- Seleccionar un nuevo conjunto de datos llamado **Prueba1** con solo las variables **Id**, **Edad** y **Sexo**.
- Seleccionar todas las filas que tienen una edad mayor o igual que 24 años en un conjunto nuevo llamado **Prueba2**.
- Seleccionar todas las filas que tienen una edad menor que 25 años y no queremos incluir en la columna **Sexo** en un conjunto nuevo llamado **Prueba3**.
- Seleccionar un nuevo conjunto de datos llamado **Prueba4** con solo los registros que tienen una altura menor o igual que 165 cm o mayor que 175 cm.

```
Prueba1<- subset(Pacientes, select = c(Id,Edad,Sexo))  #apartado a
Prueba1

##     Id Edad Sexo
## 1   I1   23    1
## 2   I2   24    2
## 3   I3   21    1
```

```
## 4   I4   22    1
## 5   I5   23    1
## 6   I6   25    2
## 7   I7   26    2
## 8   I8   24    2
## 9   I9   21    1
## 10 I10   22    2
## 11 I11   23    1
## 12 I12   25    2
## 13 I13   26    2
## 14 I14   24    2
## 15 I15   22    1
## 16 I16   21    1
## 17 I17   25    1
## 18 I18   26    2
## 19 I19   24    2
## 20 I20   21    2
## 21 I21   25    1
## 22 I22   27    2
```

```
Prueba2<- subset(Pacientes, Edad >= 24) #apartado b
Prueba2
```

```
##     Id Edad Sexo  Peso Altura
## 2   I2   24    2  81.2    154
## 6   I6   25    2  78.6    175
## 7   I7   26    2  67.9    182
## 8   I8   24    2 100.2    165
## 12 I12   25    2  67.5    183
## 13 I13   26    2  87.4    184
## 14 I14   24    2  99.7    164
## 17 I17   25    1  65.4    182
## 18 I18   26    2  73.7    179
## 19 I19   24    2  85.1    165
## 21 I21   25    1  54.8    183
## 22 I22   27    2 103.4    184
```

```
Prueba3<- subset(Pacientes, Edad < 25, select = -c(Sexo))  #apartado c
Prueba3
```

```
##     Id Edad  Peso Altura
## 7   I7   26  67.9    182
## 13 I13   26  87.4    184
## 18 I18   26  73.7    179
## 22 I22   27 103.4    184
```

```
Prueba4<- subset(Pacientes, Altura <= 165 | Altura > 175)  #apartado d
Prueba4
```

```
##     Id Edad Sexo  Peso Altura
## 1   I1   23    1  76.5    165
```

```
## 2   I2   24    2  81.2    154
## 3   I3   21    1  79.3    178
## 4   I4   22    1  59.5    165
## 5   I5   23    1  67.3    164
## 7   I7   26    2  67.9    182
## 8   I8   24    2 100.2    165
## 9   I9   21    1  97.8    178
## 10 I10   22    2  56.4    165
## 11 I11   23    1  65.4    158
## 12 I12   25    2  67.5    183
## 13 I13   26    2  87.4    184
## 14 I14   24    2  99.7    164
## 15 I15   22    1  87.6    189
## 17 I17   25    1  65.4    182
## 18 I18   26    2  73.7    179
## 19 I19   24    2  85.1    165
## 20 I20   21    2  61.2    158
## 21 I21   25    1  54.8    183
## 22 I22   27    2 103.4    184
```

b) La función *sample()* se utiliza para tomar una muestra aleatoria de tamaño **n** de un conjunto de datos. El muestreo se puede hacer con reemplazo y sin reemplazo. Cada vez que llamamos sample, se generan muestras aleatorias diferentes pero si antes fijamos una semilla pseudoaleatoria con el comando *set.seed()*, obtendremos los mismos resultados cada vez que repitamos la instrucción.

Ejemplo 12:

A partir de la base de datos **Pacientes** generada anteriormente, tomamos una muestra aleatoria (sin reemplazo) de tamaño n=3 de nuestra base de datos:

```
set.seed(999)
f <- nrow(Pacientes)   #f es el tamaño del data frame a partir del número de
filas.
n <- 3                      #n es el tamaño de la nueva muestra aleatoria.
i <- sample(1:f, n, replace=FALSE)  #i serán las posiciones de las
observaciones aleatorias.
Prueba5 <- Pacientes[i,]            #La nueva muestra aleatoria
Prueba5

##   Id Edad Sexo Peso Altura
## 4 I4   22    1 59.5    165
## 7 I7   26    2 67.9    182
## 9 I9   21    1 97.8    178
```

a) Con la función *filter()* de la base de **R** podemos filtrar filas según una condición. Para poder usar la función **filter** deberéis instalar el paquete dplyr usando los comandos: "install.packages("dplyr")" y "library(dplyr)".

```
data(women)
Prueba6 <- filter(women, height > 58)
Prueba6
```

b) La librería **dplyr** usa un lenguaje específico para la manipulación y las operaciones con data frames. Una de estas estructuras es: **%>%** (el llamado operador pipe) que nos permite **concatenar múltiples operaciones** y **escribir una secuencia de operaciones de izquierda a derecha en una única línea**.

Otra opción es la función **filter**, que nos permitirá filtrar filas según una o varias condiciones.

El código general para escribir estas opciones es el siguiente:

```
"data.frame" %>%  filter("condiciones") -> "Nuevo_nombre"   #Primera opción
"Nuevo_nombre" <- "data.frame" %>%  filter("condiciones")   #Segunda opción
```

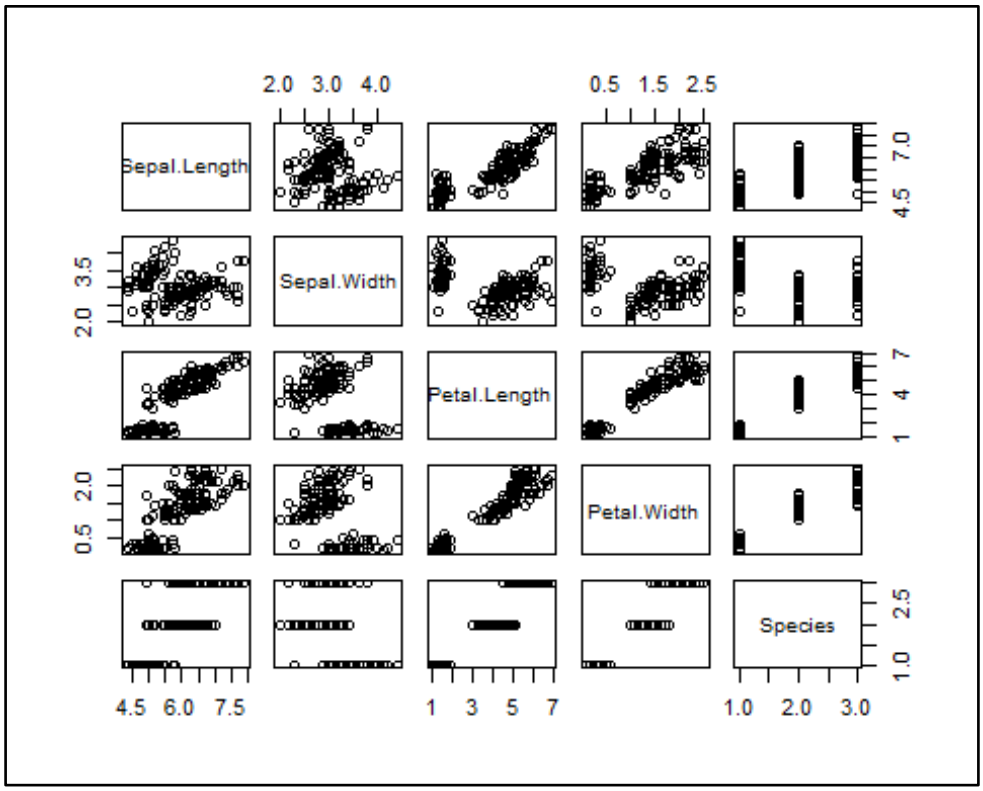
12. Visualización de los datos en R. Gráficas base

Vamos a hacer una pequeña introducción a los gráficos base de **R**, ya que en el **LAB2** podemos encontrar una explicación más extensa de cómo crear gráficos. Usaremos las funciones propias para generar algunos tipos de gráficos.

Ejemplo 13:

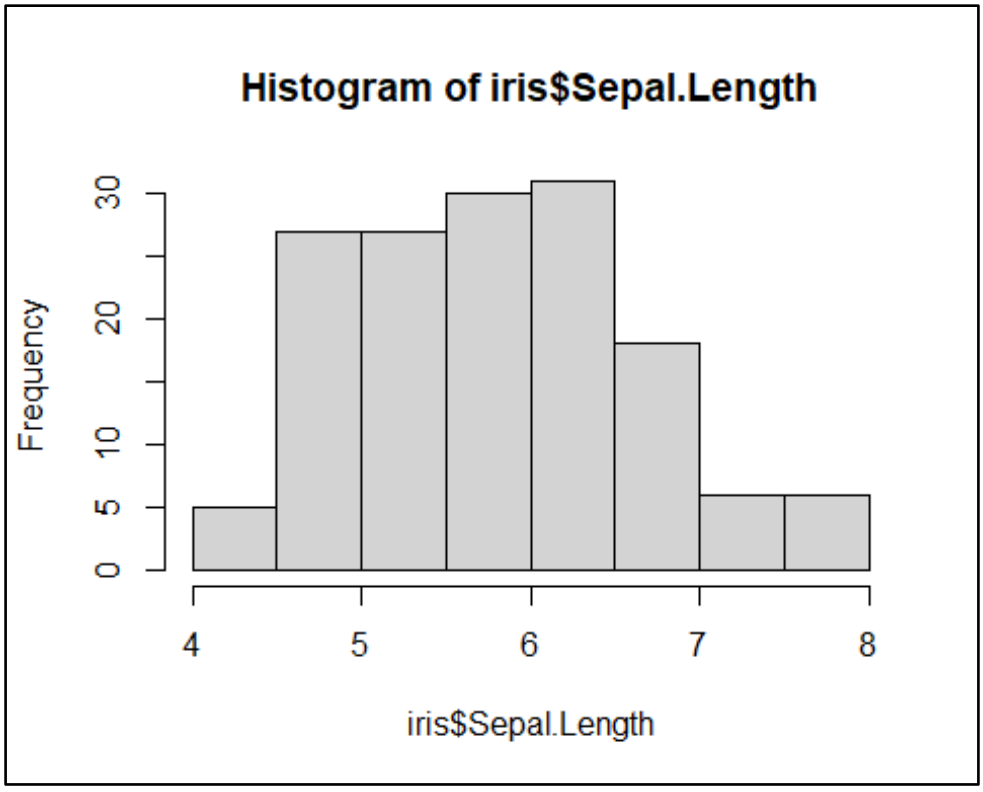
Para ejemplificar estas funciones, usaremos el paquete **Iris** que está incorporado en **R** y que se usa en infinidad de ejemplos de docencia. La forma más sencilla de generar gráficos en **R** es con la función *plot()*.

```
plot(iris)
```



Para generar histogramas podemos usar la función *hist()*

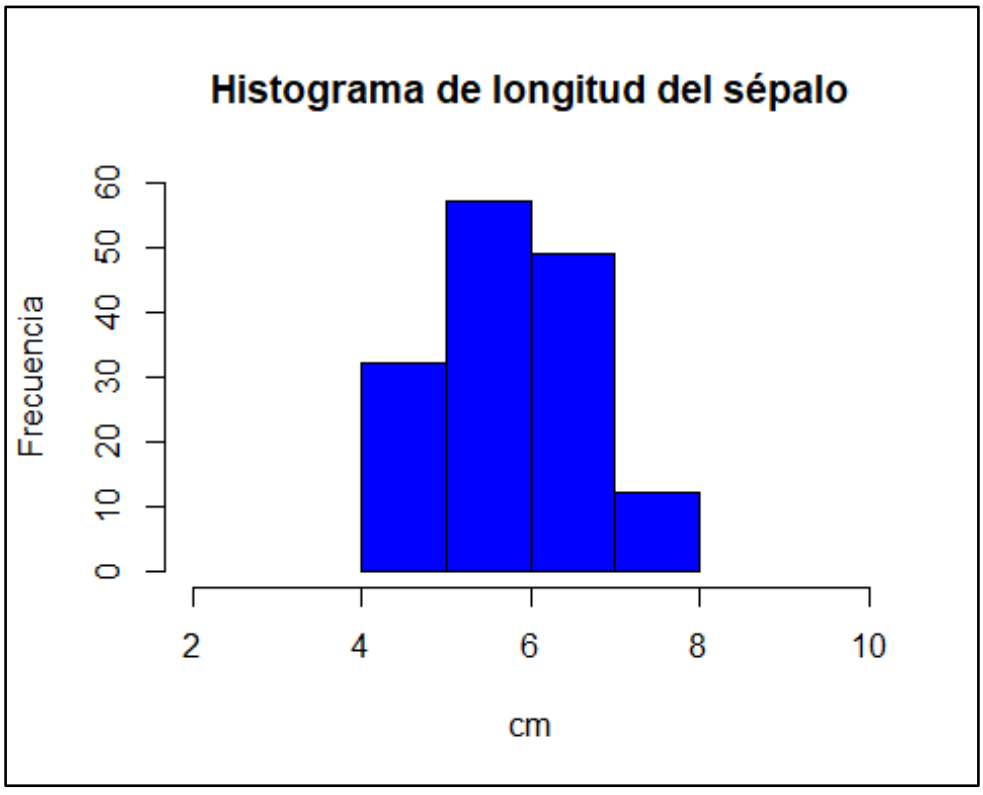
```
hist(iris$Sepal.Length)
```



Podemos incluir muchas opciones en el histograma como:

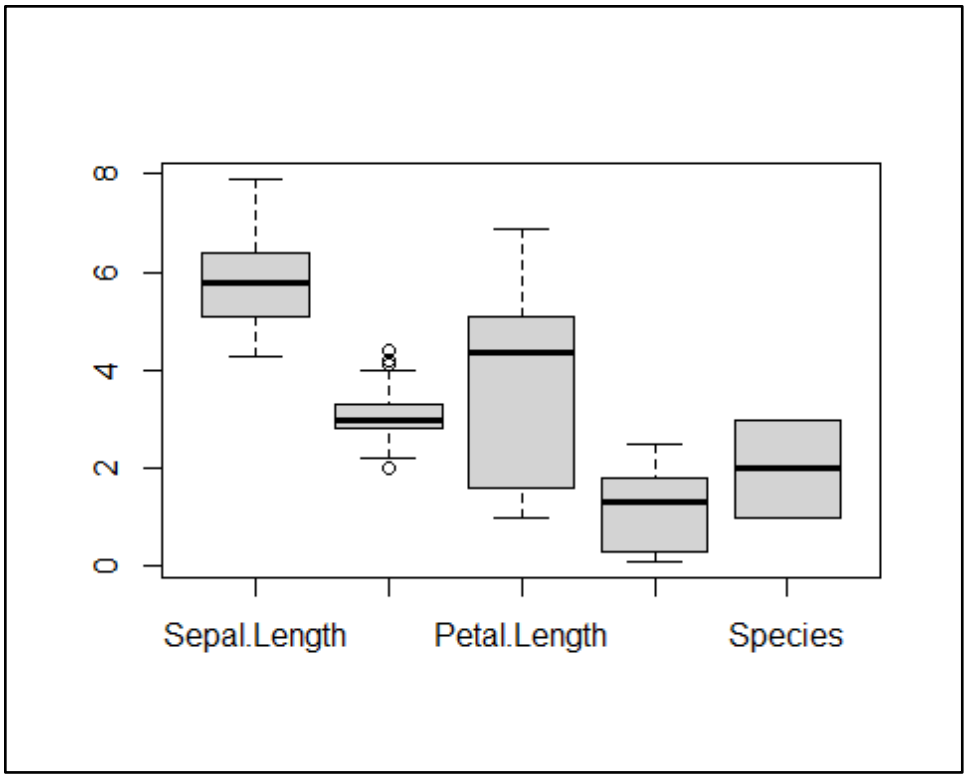
Opciones	Significado
main	Título de la tabla
xlab	Eje de las x
ylab	Eje de las y
breaks	Intervalos del histograma
col	Color del gráfico
abline	Línea superpuesta

```
hist(iris$Sepal.Length, breaks=c(4,5,6,7,8),
     main="Histograma de longitud del sépalo",
     xlab="cm", ylab="Frecuencia",
     xlim=c(2, 10), ylim=c(0, 60),
     col="blue") #creamos un histograma con títulos, intervalos, ejes y
especificamos el color azul
```

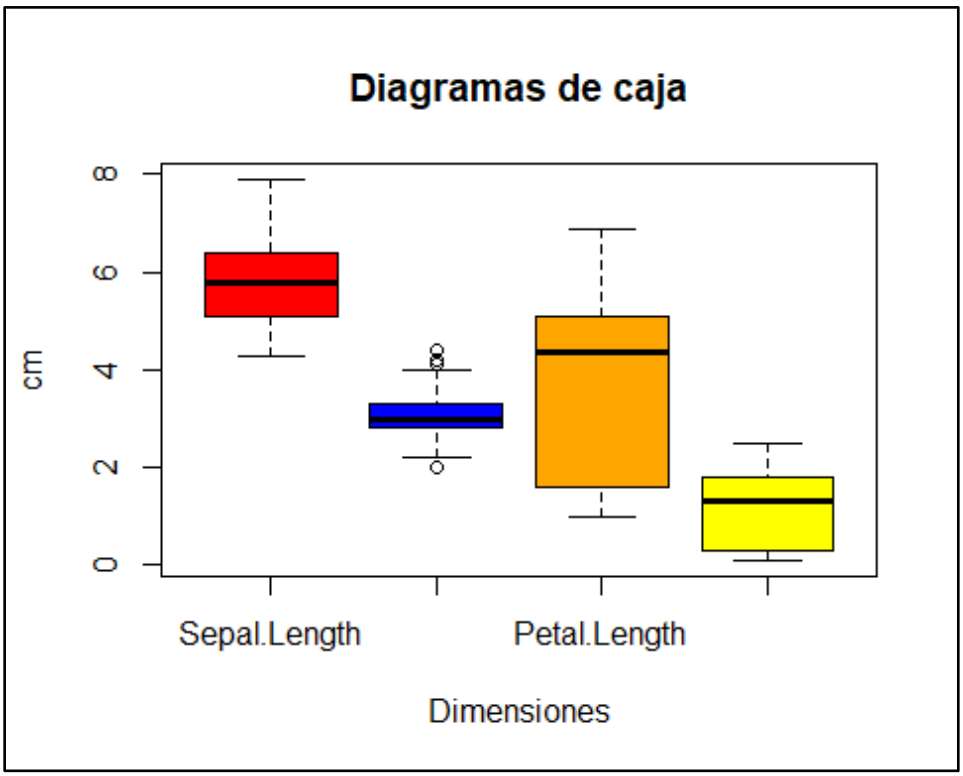


Por último, podemos también generar gráficos de caja y bigote que muestran el resumen de cinco de los números importantes del conjunto de datos (mínimo, el primer cuartil (Q1), la mediana (Q2), el tercer cuartil (Q3) y el valor máximo).

```
boxplot(iris) #genero el boxplot con todas las variables
```




```r
boxplot(iris[ ,-5],  main="Diagramas de caja",
        xlab="Dimensiones", ylab="cm",
        col=c("red", "blue", "orange", "yellow"))
```



13. Ejercicios y casos prácticos con R

Ejercicio 1:

Utilizando las funciones citadas en este Laboratorio, comprobad qué paquetes tenéis instalados en vuestra versión de **RStudio** e instalad el paquete *MASS* y el paquete ***Survival*** y comprobad la información que contienen.

Buscad información sobre el paquete *Rcmdr* (R Commander) desde la consola.

Ejercicio 2:

a) Importad un archivo de texto y buscad un *summary()* de tres variables que escojáis.
b) Importad un archivo «.csv» y buscad un *fivenum()* de dos variables que os parezcan relevantes para el estudio.

Ejercicio 3:

A partir del conjunto de datos *anorexia* del paquete *MASS*, que corresponden a los datos de cambio de peso de pacientes jóvenes con anorexia, mostrad los tipos de datos que contiene y comprobad si existen valores NA y NULL. Para la variable *Treat*, transformad los valores «*CBT*», «*Cont*» y «*FT*» en «*Cogn Beh Tr*», «*Contr*» y «*Fam Tr*», respectivamente.

Ejercicio 4:

a) Exportad los datos *biopsy* del paquete *MASS* a un archivo «.csv.»
b) Exportad los datos ***melanoma*** del paquete *MASS* a archivos de tres diferentes formatos y comprobad que se han creado los diferentes archivos en los formatos y las rutas especificados. Podéis generar una captura de pantalla de su ubicación en la carpeta.
c) Generad un resumen (**summary**) de la variable *age* de melanoma y guardad la salida que os aparece en un documento .doc
d) Buscad un data frame en algún repositorio de datos de Biomedicina, descargad un conjunto de datos en «.csv» e importad este fichero a un documento **R Markdown** usando el código o el menú de importación de **RStudio**.

Esta es una lista de algunos repositorios de datos que podemos usar:

https://ouhsc.edu/bserdac/dthompso/web/statres.htm
https://guides.lib.berkeley.edu/publichealth/healthstatistics/rawdata

https://archive.ics.uci.edu/datasets

Ejercicio 5:

En el siguiente ejemplo veremos cómo utilizar diferentes operadores sobre el conjunto de datos *birthwt*, así como también algunas funciones que nos permiten obtener más información de las variables:

a) ¿Cuál es la edad máxima de las madres del conjunto de datos?
b) ¿Cuál es la edad mínima de las madres del conjunto de datos?
c) ¿Cuál es el rango de edad de las madres?
d) ¿Fumaba la madre cuyo recién nacido era el de menor peso?
e) ¿Cuánto pesó el recién nacido cuya madre tenía la edad máxima?
f) Listad los pesos de los recién nacidos, cuyas madres visitarán menos de dos veces al médico durante el primer trimestre.

Ejercicio 6:

A partir del conjunto de datos *anorexia* trabajado en apartados anteriores, cread una matriz que tenga como columnas los valores de Prewt y Postwt, y cada fila sean los valores correspondientes para cada posición.

Ejercicio 7:

Copia el código siguiente en tu consola para generar un data frame con veinticinco registros y seis variables, y responde a los siguientes apartados:

```
Identificador <-
c("I1","I2","I3","I4","I5","I6","I7","I8","I9","I10","I11","I12","I13","I14",
"I15","I16","I17","I18","I19","I20","I21","I22","I23","I24","I25")
Edad <-
c(23,24,21,22,23,25,26,24,21,22,23,25,26,24,22,21,25,26,24,21,25,27,26,22,29)
Sexo <-c(1,2,1,1,1,2,2,2,1,2,1,2,2,2,1,1,1,2,2,2,1,2,1,1,2) #1 para mujeres y
2 para hombres
Peso <-
c(76.5,81.2,79.3,59.5,67.3,78.6,67.9,100.2,97.8,56.4,65.4,67.5,87.4,99.7,87.6
,93.4,65.4,73.7,85.1,61.2,54.8,103.4,65.8,71.7,85.0)
Alt <-
c(165,154,178,165,164,175,182,165,178,165,158,183,184,164,189,167,182,179,165
,158,183,184,189,166,175) #altura en cm
Fuma <-
c("SÍ","NO","SÍ","SÍ","NO","NO","NO","SÍ","SÍ","SÍ","NO","NO","SÍ","SÍ","SÍ",
"SÍ","NO","NO","SÍ","SÍ","SÍ","NO","SÍ","NO","SÍ")

Trat_Pulmon <- data.frame(Identificador,Edad,Sexo,Peso,Alt,Fuma)
Trat_Pulmon
```

a) Seleccionad los registros con edad > 22.
b) Seleccionad el elemento 3 de la columna 4 del conjunto de datos (contando el identificador).
c) Usad el comando *subset()* para seleccionar todas las filas que tienen una edad menor que 27 años y sin incluir la columna **Alt**.

Ejercicio 8:

Incorporad el dataset ***ChickWeight*** que contiene información sobre el peso de 578 pollitos en gramos (*weight*), el tiempo desde la medición al nacer (*Time*), una variable identificadora de cada pollito (*Chick*) a partir del rango de peso y una variable factor con el tipo de dieta experimental que cada pollito recibió (*Diet*).

a) Incorporad el conjunto de datos ***ChickWeight*** del paquete **datasets** a vuestro entorno de trabajo.
b) Generad un gráfico de dispersión de la variable ***weight***.
c) Cread un diagrama de caja con la variable ***Time***.

Para más información sobre ***ChickWeight***: https://rdrr.io/r/datasets/ChickWeight.html

Ejercicio 9:

A partir del conjunto de datos *anorexia* del paquete *MASS*, cread otro data frame que se llame **anorexia_treat_df** formado por Treat y por un vector nuevo calculado a partir de la diferencia *Prewt-Postwt*. De esta manera, nos quedará un data frame que contenga el tipo de tratamiento y el valor del peso ganado o perdido después de haber realizado el tratamiento.

Seleccionad aquellos individuos que han ganado peso después del tratamiento y cread un nuevo conjunto llamado **anorexia_treat_C_df** que contenga solo los datos de aquellos que han seguido el tratamiento «Cont» y que han ganado peso después del tratamiento.

Ejercicio 10:

Entrad en RPubs y registraros. Crearos un perfil y subid un documento **R Markdown**. Los prerequisitos son tener instalado R y RStudio (v0.96.230 o más), y el paquete knitr (v0.5 o más).

Pasos que tenéis que seguir para publicar vuestro documento:

1) En **RStudio**, cread un documento **R Markdown**.
2) Generad el documento con ***Knit***.
3) En la ventana de previsualización, clicad el botón de publicar.

Como solución de vuestro ejercicio, copiad el enlace de vuestra página de prueba de **RPubs**.

Caso práctico:

Resolved los siguientes apartados:

a) Cread un conjunto de datos inventado con R. Debe contener treinta observaciones (quince para hombres y quince para mujeres) para seis variables con estas características:

Variable	Nombre	Características
Identificador	Id	carácter
Edad	Edad	numérica
Genero	Gene	2 valores 1 = mujer, 2 = hombre
Tratamiento	Trat	Factor. Tres tipos de tratamiento (A, B y C)
Peso	Peso	numérica (en kg)
Estatura	Alt	numérica (en cm)

b) Buscad información de vuestro conjunto de datos y de vuestras variables.

c) Cread una nueva variable a partir de alguna de las que tengamos. Por ejemplo, podéis calcular el IMC **(IMC = peso (kg)/ [estatura (m)]^2** e incluid la nueva variable en el conjunto de datos.

d) Cread dos data frames diferenciados para hombres y mujeres con dos nombres diferentes: **Df_Hombres** y **Df_Mujeres**.

e) Combinad de nuevo los dos ficheros anteriores y cread el primero de nuevo con el comando *rbind()*.

Solución a los ejercicios propuestos y casos prácticos con R

Solución del ejercicio 1:

```
library() #comprobamos qué paquetes tenemos instalados
install.packages("MASS")
install.packages("Survival")
??Rcmdr
```

Solución del ejercicio 2

```
#importación de un documento .txt escogido por vosotros
dataBreastCancer <-
read.csv("C:/Users/mcasal8/Desktop/SAD_LAB1_datos/dataBreastCancer.txt",
sep="")
#visualizamos el dataset por pantalla
View(dataBreastCancer)
#primeros registros del dataset
head(dataBreastCancer)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst
perimeter_worst
```

```
## 1     0.03003             0.006193        25.38         17.33
184.60
## 2     0.01389             0.003532        24.99         23.41
158.80
## 3     0.02250             0.004571        23.57         25.53
152.50
## 4     0.05963             0.009208        14.91         26.50
98.87
## 5     0.01756             0.005115        22.54         16.67
152.20
## 6     0.02165             0.005082        15.47         23.75
103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave.points_worst symmetry_worst fractal_dimension_worst
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440

#resumen estadístico del dataset (cortado)
summary(dataBreastCancer)

##        id              diagnosis          radius_mean      texture_mean
##  Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024   Mode  :character   Median :13.370   Median :18.84
##  Mean   : 30371831                      Mean   :14.127   Mean   :19.29

#importación de un documento .csv escogido por vosotros
library(readr)
wisc_bc_data <-
read_csv("C:/Users/mcasal8/Desktop/SAD_LAB1_datos/wisc_bc_data.csv")
View(wisc_bc_data)

#buscamos cinco estadísticos básicos de dos variables escogidas por nosotros
fivenum(wisc_bc_data$area_se)

## [1]   6.802  17.850  24.530  45.190 542.200

fivenum(wisc_bc_data$perimeter_se)

## [1]  0.757  1.606  2.287  3.357 21.980
```

Solución del ejercicio 3:

```
library("MASS")
#cargamos los datos de anorexia
data("anorexia")
#mostramos los primeros registros de anorexia
head(anorexia)
```

```
##   Treat Prewt Postwt
## 1  Cont  80.7   80.2
## 2  Cont  89.4   80.1
## 3  Cont  91.8   86.4
## 4  Cont  74.0   86.3
## 5  Cont  78.1   76.1
## 6  Cont  88.3   78.1
```

```
#mostramos cuántos valores perdidos hay
table(is.na(anorexia))
```

```
##
## FALSE
##   216
```

```
#mostramos si anorexia es NULL
table(is.null(anorexia))
```

```
##
## FALSE
##     1
```

```
#definimos etiquetas para los valores especificados
anorexia_F<- factor(anorexia$Treat,levels=c("CBT","Cont","FT"),labels=c("Cogn
Beh Tr","Contr","Fam Tr"))
anorexia_F
```

```
##  [1] Contr        Contr        Contr        Contr        Contr
##  [6] Contr        Contr        Contr        Contr        Contr
## [11] Contr        Contr        Contr        Contr        Contr
## [16] Contr        Contr        Contr        Contr        Contr
## [21] Contr        Contr        Contr        Contr        Contr
## [26] Contr        Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr
## [31] Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr
## [36] Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr
## [41] Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr
## [46] Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr
## [51] Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr Cogn Beh\nTr
## [56] Fam Tr       Fam Tr       Fam Tr       Fam Tr       Fam Tr
## [61] Fam Tr       Fam Tr       Fam Tr       Fam Tr       Fam Tr
## [66] Fam Tr       Fam Tr       Fam Tr       Fam Tr       Fam Tr
## [71] Fam Tr       Fam Tr
## Levels: Cogn Beh\nTr Contr Fam Tr
```

Solución del ejercicio 4:

```
library(MASS)
data("biopsy")
head("biopsy")

#exportamos el conjunto de datos biopsy al formato ".csv"

write.csv(biopsy, file="C:/Users/mcasal8/Desktop/LAB1_Jul_23/biopsy.csv")

data("Melanoma")

#exportamos el conjunto de datos Melanoma al formato ".csv", ".txt" y "xlsx"
write.csv(Melanoma, file="C:/Users/mcasal8/Desktop/LAB1_Jul_23/melanoma.csv")
write.table(Melanoma,"C:/Users/mcasal8/Desktop/LAB1_Jul_23/melanoma.txt")
library(xlsx)
write.xlsx(Melanoma,"C:/Users/mcasal8/Desktop/LAB1_Jul_23/melanoma.xlsx")
```

Solución del ejercicio 5:

```
#activamos el dataset birthw de la librería MASS
library(MASS) #cargamos el paquete MASS
data("birthwt") #definimos la estructura de datos
View(birthwt)
head(birthwt)

##    low age lwt race smoke ptl ht ui ftv  bwt
## 85   0  19 182    2     0   0  0  1   0 2523
## 86   0  33 155    3     0   0  0  0   3 2551
## 87   0  20 105    1     1   0  0  0   1 2557
## 88   0  21 108    1     1   0  0  1   2 2594
## 89   0  18 107    1     1   0  0  1   0 2600
## 91   0  21 124    3     0   0  0  0   0 2622

#respuestas de los apartados del ejercicio

max(birthwt$age) #apartado a)

## [1] 45

min(birthwt$age) #apartado b)

## [1] 14

rank<-max(birthwt$age)-min(birthwt$age) #apartado c)
rank

## [1] 31

birthwt$smoke[birthwt$bwt==min(birthwt$bwt)] #apartado d)

## [1] 1
```

```
birthwt$bwt[birthwt$age==max(birthwt$age)] #apartado e)
```

```
## [1] 4990
```

```
birthwt$bwt[birthwt$ftv<2] #apartado f)
```

```
##   [1] 2523 2557 2594 2600 2622 2637 2637 2663 2665 2722 2733 2751 2750
2769 2769
##  [16] 2778 2807 2821 2835 2836 2863 2877 2877 2906 2920 2920 2920 2920
2948 2948
##  [31] 2977 2977 2977 2977 2922 3005 3033 3042 3062 3062 3062 3062 3062
3090 3090
##  [46] 3090 3100 3104 3132 3147 3175 3175 3203 3203 3203 3225 3225 3232
3232 3234
##  [61] 3260 3274 3274 3317 3317 3317 3321 3331 3374 3374 3402 3416 3444
3459 3460
##  [76] 3473 3544 3487 3544 3572 3572 3586 3600 3614 3614 3629 3629 3637
3643 3651
##  [91] 3651 3651 3651 3699 3728 3756 3770 3770 3770 3790 3799 3827 3856
3860 3884
## [106] 3884 3912 3940 3941 3941 3969 3983 3997 3997 4054 4054 4111 4153
4167 4174
## [121] 4238 4593 4990  709 1021 1135 1330 1474 1588 1588 1701 1729 1790
1818 1885
## [136] 1893 1899 1928 1928 1928 1936 1970 2055 2055 2082 2084 2084 2100
2125 2187
## [151] 2187 2211 2225 2240 2240 2282 2296 2296 2325 2353 2353 2367 2381
2381 2381
## [166] 2410 2410 2410 2424 2438 2442 2466 2466 2466 2495 2495 2495
```

Solución del ejercicio 6:

```
library(MASS)
data(anorexia)
matr_anorexia<-matrix(c(anorexia$Prewt,anorexia$Postwt),ncol=2)
head(matr_anorexia)
```

```
##      [,1] [,2]
## [1,] 80.7 80.2
## [2,] 89.4 80.1
## [3,] 91.8 86.4
## [4,] 74.0 86.3
## [5,] 78.1 76.1
## [6,] 88.3 78.1
```

Solución del ejercicio 7:

```
Identificador <-
c("I1","I2","I3","I4","I5","I6","I7","I8","I9","I10","I11","I12","I13","I14",
"I15","I16","I17","I18","I19","I20","I21","I22","I23","I24","I25")
Edad <-
c(23,24,21,22,23,25,26,24,21,22,23,25,26,24,22,21,25,26,24,21,25,27,26,22,29)
Sexo <-c(1,2,1,1,1,2,2,2,1,2,1,2,2,2,1,1,1,2,2,2,1,2,1,1,2)
Peso <-
c(76.5,81.2,79.3,59.5,67.3,78.6,67.9,100.2,97.8,56.4,65.4,67.5,87.4,99.7,87.6
,93.4,65.4,73.7,85.1,61.2,54.8,103.4,65.8,71.7,85.0)
Alt <-
c(165,154,178,165,164,175,182,165,178,165,158,183,184,164,189,167,182,179,165
,158,183,184,189,166,175) #altura en cm
Fuma <-
c("SÍ","NO","SÍ","SÍ","NO","NO","NO","SÍ","SÍ","SÍ","NO","NO","SÍ","SÍ","SÍ",
"SÍ","NO","NO","SÍ","SÍ","SÍ","NO","SÍ","NO","SÍ")

Trat_Pulmon <- data.frame(Identificador,Edad,Sexo,Peso,Alt,Fuma)
Trat_Pulmon
```

```
##    Identificador Edad Sexo  Peso Alt Fuma
## 1             I1   23    1  76.5 165   SÍ
## 2             I2   24    2  81.2 154   NO
## 3             I3   21    1  79.3 178   SÍ
## 4             I4   22    1  59.5 165   SÍ
## 5             I5   23    1  67.3 164   NO
## 6             I6   25    2  78.6 175   NO
## 7             I7   26    2  67.9 182   NO
## 8             I8   24    2 100.2 165   SÍ
## 9             I9   21    1  97.8 178   SÍ
## 10           I10   22    2  56.4 165   SÍ
## 11           I11   23    1  65.4 158   NO
## 12           I12   25    2  67.5 183   NO
## 13           I13   26    2  87.4 184   SÍ
## 14           I14   24    2  99.7 164   SÍ
## 15           I15   22    1  87.6 189   SÍ
## 16           I16   21    1  93.4 167   SÍ
## 17           I17   25    1  65.4 182   NO
## 18           I18   26    2  73.7 179   NO
## 19           I19   24    2  85.1 165   SÍ
## 20           I20   21    2  61.2 158   SÍ
## 21           I21   25    1  54.8 183   SÍ
## 22           I22   27    2 103.4 184   NO
## 23           I23   26    1  65.8 189   SÍ
## 24           I24   22    1  71.7 166   NO
## 25           I25   29    2  85.0 175   SÍ
```

```
selec1<- subset(Trat_Pulmon, Edad >22) #apartado a)
selec1
```

```
##    Identificador Edad Sexo  Peso Alt Fuma
## 1             I1   23    1  76.5 165   SÍ
## 2             I2   24    2  81.2 154   NO
## 5             I5   23    1  67.3 164   NO
## 6             I6   25    2  78.6 175   NO
## 7             I7   26    2  67.9 182   NO
## 8             I8   24    2 100.2 165   SÍ
## 11           I11   23    1  65.4 158   NO
## 12           I12   25    2  67.5 183   NO
## 13           I13   26    2  87.4 184   SÍ
## 14           I14   24    2  99.7 164   SÍ
## 17           I17   25    1  65.4 182   NO
## 18           I18   26    2  73.7 179   NO
## 19           I19   24    2  85.1 165   SÍ
## 21           I21   25    1  54.8 183   SÍ
## 22           I22   27    2 103.4 184   NO
## 23           I23   26    1  65.8 189   SÍ
## 25           I25   29    2  85.0 175   SI
```

```
selec2 <- Trat_Pulmon[3, 4] #apartado b)
selec2
```

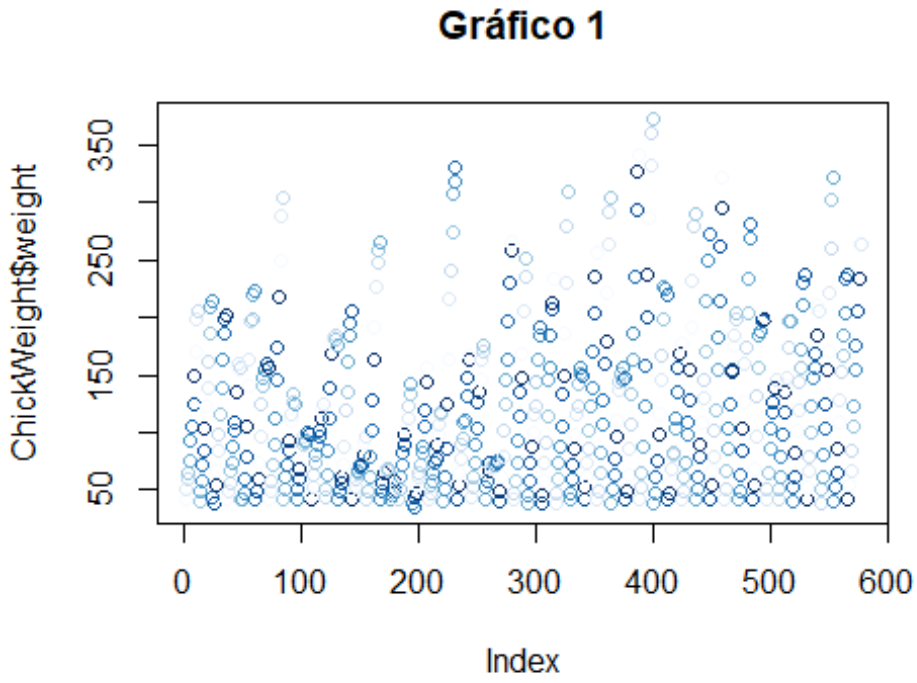
```
## [1] 79.3
```

```
selec3 <- subset(Trat_Pulmon, Edad < 27, select = -c(Alt))  #apartado c)
selec3
```

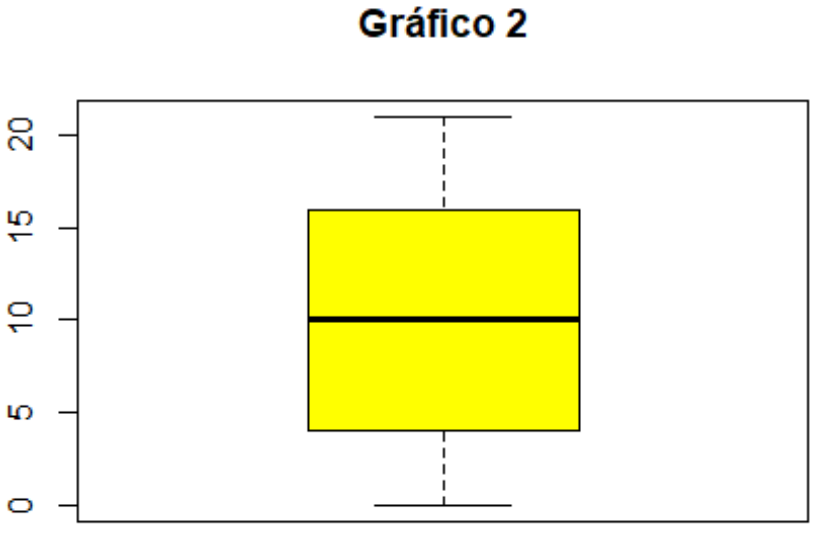
```
##    Identificador Edad Sexo  Peso Fuma
## 1             I1   23    1  76.5   SÍ
## 2             I2   24    2  81.2   NO
## 3             I3   21    1  79.3   SÍ
## 4             I4   22    1  59.5   SÍ
## 5             I5   23    1  67.3   NO
## 6             I6   25    2  78.6   NO
## 7             I7   26    2  67.9   NO
## 8             I8   24    2 100.2   SÍ
## 9             I9   21    1  97.8   SÍ
## 10           I10   22    2  56.4   SÍ
## 11           I11   23    1  65.4   NO
## 12           I12   25    2  67.5   NO
## 13           I13   26    2  87.4   SÍ
## 14           I14   24    2  99.7   SÍ
## 15           I15   22    1  87.6   SÍ
## 16           I16   21    1  93.4   SÍ
## 17           I17   25    1  65.4   NO
## 18           I18   26    2  73.7   NO
## 19           I19   24    2  85.1   SÍ
## 20           I20   21    2  61.2   SÍ
## 21           I21   25    1  54.8   SÍ
## 23           I23   26    1  65.8   SÍ
## 24           I24   22    1  71.7   NO
```

Solución del ejercicio 8:

```
library(datasets)
data.frame(ChickWeight)
head(ChickWeight)
plot(ChickWeight$weight, col=blues9, main="Gráfico 1")
```




```
boxplot(ChickWeight$Time, col="yellow", main="Gráfico 2")
```



Solución del ejercicio 9:

```
##cargamos los valores de anorexia
library(MASS)
post_pre_v<-c(anorexia$Postwt-anorexia$Prewt)
post_pre_v
```

```
##  [1]  -0.5  -9.3  -5.4  12.3  -2.0 -10.2 -12.2  11.6  -7.1   6.2  -0.2  -
9.2
## [13]   8.3   3.3  11.3   0.0  -1.0 -10.6  -4.6  -6.7   2.8   0.3   1.8
3.7
## [25]  15.9 -10.2   1.7   0.7  -0.1  -0.7  -3.5  14.9   3.5  17.1  -7.6
1.6
## [37]  11.7   6.1   1.1  -4.0  20.9  -9.1   2.1  -1.4   1.4  -0.3  -3.7  -
0.8
## [49]   2.4  12.6   1.9   3.9   0.1  15.4  -0.7  11.4  11.0   5.5   9.4
13.6
## [61]  -2.9  -0.1   7.4  21.5  -5.3  -3.8  13.4  13.1   9.0   3.9   5.7
10.7
```

```
#creamos el conjunto de datos a partir del vector del pre-post
anorexia_treat_df<-data.frame(anorexia,post_pre_v)
head(anorexia_treat_df)
```

```
##   Treat Prewt Postwt post_pre_v
## 1  Cont  80.7   80.2       -0.5
## 2  Cont  89.4   80.1       -9.3
## 3  Cont  91.8   86.4       -5.4
## 4  Cont  74.0   86.3       12.3
## 5  Cont  78.1   76.1       -2.0
## 6  Cont  88.3   78.1      -10.2
```

```
#seleccionamos los elementos que han seguido el tratamiento
anorexia_treat_C_df<-
subset(anorexia_treat_df,anorexia_treat_df$Treat=="Cont"&anorexia_treat_df$po
st_pre_v>0)
head(anorexia_treat_C_df)
```

```
##   Treat Prewt Postwt post_pre_v
## 1  Cont  80.7   80.2       -0.5
## 2  Cont  89.4   80.1       -9.3
## 3  Cont  91.8   86.4       -5.4
## 4  Cont  74.0   86.3       12.3
## 5  Cont  78.1   76.1       -2.0
## 6  Cont  88.3   78.1      -10.2
```

Solución del ejercicio 10:

Ejercicio opcional. Encontraréis ejemplos en: https://rpubs.com/

Solución caso práctico:

```
library(readr)
datos_invent <- read_csv("datos_invent.csv")

View(datos_invent)
summary(datos_invent)

##        Id              alt              peso             edad
genero
##  Min.   : 1.00   Min.   :116.0   Min.   : 58.00   Min.   :34.00   Min.
:1.0
##  1st Qu.: 8.25   1st Qu.:150.0   1st Qu.: 78.50   1st Qu.:60.25   1st
Qu.:1.0
##  Median :15.50   Median :158.0   Median : 95.00   Median :69.50   Median
:1.5
##  Mean   :15.50   Mean   :158.1   Mean   : 91.17   Mean   :67.67   Mean
:1.5
##  3rd Qu.:22.75   3rd Qu.:168.5   3rd Qu.:104.75   3rd Qu.:76.00   3rd
Qu.:2.0
##  Max.   :30.00   Max.   :200.0   Max.   :122.00   Max.   :92.00   Max.
:2.0
##      trat
##  Length:30
##  Class :character
##  Mode  :character
##
##
##

IMC <- datos_invent$peso/(datos_invent$alt/100)^2
IMC

##  [1] 36.16898 42.22222 42.22222 25.71101 41.86424 41.50230 40.56960
19.23018
##  [9] 41.86424 30.84442 45.62587 44.22186 43.58328 41.50230 15.41078
40.40404
## [17] 48.30559 36.28974 39.66942 40.29594 39.91457 22.50000 39.54840
45.29184
## [25] 40.15920 35.65134 40.89348 29.41176 38.86419 27.76621

dataIMC <- data.frame(datos_invent,IMC)
Df_Mujeres <- subset(dataIMC, datos_invent$genero==1)
Df_Mujeres

##    Id alt peso edad genero trat      IMC
## 1   1 144   75   85      1    A 36.16898
```

```
## 3   3 150   95   73      1    C 42.22222
## 5   5 153   98   71      1    B 41.86424
## 6   6 156  101   86      1    A 41.50230
## 9   9 153   98   70      1    B 41.86424
## 10 10 158   77   69      1    A 30.84442
## 11 11 142   92   70      1    C 45.62587
## 15 15 194   58   45      1    B 15.41078
## 17 17 116   65   60      1    C 48.30559
## 19 19 165  108   72      1    A 39.66942
## 20 20 174  122   77      1    C 40.29594
## 23 23 172  117   62      1    C 39.54840
## 24 24 117   62   45      1    B 45.29184
## 25 25 167  112   77      1    C 40.15920
## 29 29 169  111   67      1    B 38.86419
```

```
Df_Hombres <- subset(dataIMC, datos_invent$genero==2)
Df_Hombres
```

```
##    Id alt peso edad genero trat      IMC
## 2   2 150   95   73      2    B 42.22222
## 4   4 159   65   34      2    C 25.71101
## 7   7 157  100   63      2    C 40.56960
## 8   8 181   63   86      2    C 19.23018
## 12 12 137   83   64      2    B 44.22186
## 13 13 138   83   46      2    A 43.58328
## 14 14 156  101   84      2    C 41.50230
## 16 16 165  110   61      2    A 40.40404
## 18 18 166  100   54      2    B 36.28974
## 21 21 169  114   91      2    B 39.91457
## 22 22 200   90   92      2    A 22.50000
## 26 26 158   89   57      2    B 35.65134
## 27 27 161  106   73      2    A 40.89348
## 28 28 170   85   54      2    C 29.41176
## 30 30 147   60   69      2    C 27.76621
```

```
Df_Todo<-rbind(Df_Hombres, Df_Mujeres)
Df_Todo
```

```
##    Id alt peso edad genero trat      IMC
## 2   2 150   95   73      2    B 42.22222
## 4   4 159   65   34      2    C 25.71101
## 7   7 157  100   63      2    C 40.56960
## 8   8 181   63   86      2    C 19.23018
## 12 12 137   83   64      2    B 44.22186
## 13 13 138   83   46      2    A 43.58328
## 14 14 156  101   84      2    C 41.50230
## 16 16 165  110   61      2    A 40.40404
## 18 18 166  100   54      2    B 36.28974
## 21 21 169  114   91      2    B 39.91457
## 22 22 200   90   92      2    A 22.50000
## 26 26 158   89   57      2    B 35.65134
```

```
## 27 27 161  106   73      2    A 40.89348
## 28 28 170   85   54      2    C 29.41176
## 30 30 147   60   69      2    C 27.76621
## 1   1 144   75   85      1    A 36.16898
## 3   3 150   95   73      1    C 42.22222
## 5   5 153   98   71      1    B 41.86424
## 6   6 156  101   86      1    A 41.50230
## 9   9 153   98   70      1    B 41.86424
## 10 10 158   77   69      1    A 30.84442
## 11 11 142   92   70      1    C 45.62587
## 15 15 194   58   45      1    B 15.41078
## 17 17 116   65   60      1    C 48.30559
## 19 19 165  108   72      1    A 39.66942
## 20 20 174  122   77      1    C 40.29594
## 23 23 172  117   62      1    C 39.54840
## 24 24 117   62   45      1    B 45.29184
## 25 25 167  112   77      1    C 40.15920
## 29 29 169  111   67      1    B 38.86419
```

Información adicional

Algunos de los recursos materiales utilizados y recomendados para trabajar este **LAB1** son los siguientes:

Página principal de R Markdown

Guía de R Markdown

R Markdown: The Definitive Guide

Trucos de R Markdown en RStudio

R Data for Science

The R Book