

Hands-on! Introducción al análisis de datos con R 12 y 13 de febrero de 2025

Beatriz Fernández Blanco
Maria Guaita Céspedes

Contents

1. ¿Por qué R?	2
1.1 Un poco de historia...	2
1.2 Ventajas de usar R	3
2. Utilizar R en RStudio	3
2.1 Instalación de R y RStudio	3
2.2 Exploramos RStudio	5
3. R como calculadora	7
4. Objetos en R	8
4.1 Concepto de variable	9
4.2 Concepto de objeto	10
5. Rutas y directorios	14
6. Operaciones con conjuntos de datos	15
6.1. Importar un conjunto de datos	15
6.2. Conocer la estructura de un conjunto de datos	16
6.3. Filtrar un conjunto de datos	23
6.4. Reordenar un conjunto de datos	35
6.5. Operaciones sencillas con columnas:	35
6.6. Buscar valores y reemplazarlos	36
6.7. El paquete dplyr	39

7. Breve análisis estadístico	41
7.1. Estadística descriptiva	43
7.2. Estadística inferencial	49
7.3. Edición de figuras:	56
8. Guardar la información	65
9. Buenas prácticas	66
10. Análisis de un conjunto de datos	66
11. Cómo seguir aprendiendo por tu cuenta	66

1. ¿Por qué R?

1.1 Un poco de historia...

Conocer cómo nació R es interesante para comprender sus características. R es un lenguaje de programación creado por Ross Ihaka y Robert Gentleman en los años 90, que eran estadísticos en la Universidad de Auckland (Nueva Zelanda) y querían crear un material mejor para dar el curso de introducción a la estadística a sus alumnos. Así, crearon el lenguaje R, basado en el lenguaje S. R siempre ha sido un proyecto de uso libre desde junio de 1995. El hecho de que fuera creado por estadísticos y no por ingenieros para el desarrollo de software como pasa con otros lenguajes de programación como C y Java, explica que uno de sus puntos fuertes sea la **interactividad** y la **visualización** de los datos.



Figure 1: Logo de R y sus creadores

El mantenimiento y desarrollo de R es realizado por el **R Development Core Team**, un equipo de especialistas en ciencias computacionales y estadística provenientes de diferentes instituciones y lugares alrededor del mundo. Este equipo mantiene la versión **base** de R que, como su nombre indica, es sobre la cual se crean otras implementaciones de R así como los paquetes que expanden su funcionalidad.

Referencias:

- <https://bookdown.org/jboscomendoza/r-principiantes4/un-poco-de-historia.html>
- <http://rafalab.dfci.harvard.edu/dsbook/getting-started.html#fn1>
- Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. Journal of computational and graphical statistics, 5(3), 299-314.

1.2 Ventajas de usar R

En esta sesión vamos a conocer R como una herramienta para visualizar y analizar datos. Probablemente a estas alturas hayáis empleado algunos programas para el análisis de datos, como puede ser Excel para hacer gráficos y SPSS o Graphpad para la estadística. Veamos algunas ventajas que nos proporciona R:

- Es de uso libre (forma parte del sistema GNU)
- Se puede usar en diferentes sistemas operativos: Windows, Linux, Mac.
- Hay una gran comunidad de usuarios de R activa y en continuo crecimiento y, por lo tanto, hay muchos recursos para aprender y resolver dudas.
- Ofrece muchas utilidades en el análisis de datos, en especial en cuanto a interactividad y visualización.
- En general, conocer un lenguaje de programación os facilitará el análisis de datos, sea cuál sea vuestro ámbito de especialización. Además, mejora otras habilidades transversales, como es el pensamiento lógico.

2. Utilizar R en RStudio

En esta sesión trabajaremos en RStudio en la nube a través de Posit Cloud. Únicamente necesitamos registrarnos con una cuenta de correo electrónico. Si bien es práctico para los objetivos de esta sesión, tened en cuenta que Posit Cloud tiene unos recursos limitados en la versión libre, de los cuales destaca que solo permite 25 horas de uso al mes. Lo ideal es trabajar en local, por lo que os dejamos las instrucciones.

2.1 Instalación de R y RStudio

Todo el código y documentación de R está almacenado en **CRAN** (Comprehensive R Archive Network), que es una red de servidores alrededor del mundo. Es decir, CRAN es el sitio oficial a través del cual descargaremos R.

cran.r-project.org

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux \(Debian, Fedora/Redhat, Ubuntu\)](#)
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

R for Windows

Subdirectories:

- [base](#)
- [contrib](#)
- [old.contrib](#)
- [Rtools](#)

Please do not download the source code for Windows.

You may also want to download:

- [Rtools](#)

Note: CRAN does not provide a Windows installer for Rtools.

R-4.2.2 for Windows

[Download R-4.2.2 for Windows](#) (76 megabytes, 64 bit)

[README on the Windows binary distribution](#)

[New features in this version](#)

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from [here](#).

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- Patches to this release are incorporated in the [r-patched snapshot build](#)
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#)
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is [~CRAN.MIRROR~bin/windows/base-release.html](#)

Last change: 2022-10-31

R-4.2.2-win.exe

Figure 2: Instalación de R

Una vez instalado R podemos abrir la consola para ver qué apariencia tiene. Aquí ya podríamos trabajar.

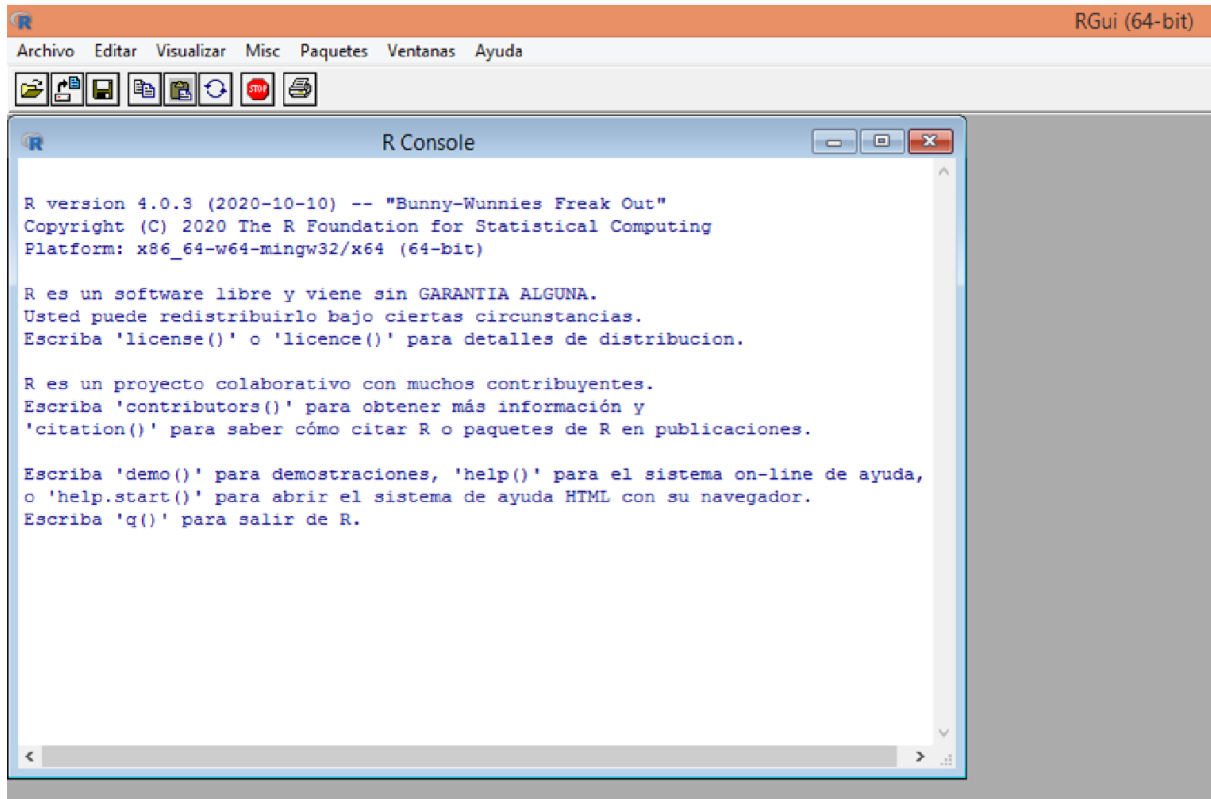


Figure 3: La consola de R

Sin embargo, vamos a aprender a usar R dentro de RStudio, que es una interfaz que nos da muchísimas más funcionalidades que trabajar solo con la consola. La descarga de RStudio se realiza desde Posit.

2.2 Exploramos RStudio

Cuando abrimos RStudio vemos varios paneles, vamos a ver qué es cada uno de ellos.

1. La consola A la izquierda tenemos la consola. La consola es el espacio donde R ejecuta las órdenes que le damos. El símbolo de “mayor que” se llama *prompt*, y significa que R está listo para que le demos una orden. En la consola podemos escribir la siguiente operación y pulsamos la tecla INTRO:

```
3+2
```

```
## [1] 5
```


posit.co/download/rstudio-desktop/			
 PRODUCTS SOLUTIONS LEARN & SUPPORT EXPLORE MORE			
OS	Download	Size	SHA-256
Windows 10/11	RSTUDIO-2022.12.0-353.EXE	202.77 MB	FD8EA4B4
macOS 11+	RSTUDIO-2022.12.0-353.DMG	365.71 MB	FD4BEBB5
Ubuntu 18+/Debian 10+	RSTUDIO-2022.12.0-353-AMD64.DEB	131.20 MB	23CAE58F
Ubuntu 22	RSTUDIO-2022.12.0-353-AMD64.DEB	131.95 MB	8BC3F84D
Fedora 19/Red Hat 7	RSTUDIO-2022.12.0-353-X86_64.RPM	145.99 MB	A717CDAD
OpenSUSE 15	RSTUDIO-2022.12.0-353-X86_64.RPM	131.50 MB	983E7D0C

Figure 4: Instalación de RStudio

2. Scripts Normalmente, en un análisis de datos necesitamos ejecutar varias tareas y finalmente queremos guardarlas para recuperar este análisis en un futuro. Esto no es posible en la consola; ya que las instrucciones que ejecutemos en ella sólo se almacenan durante la sesión de R. La forma de poder guardarlas es generando un **script** de R. Un script es un bloque de código. Vamos a crear nuestro primer script. Para ello, en la barra de herramientas pinchamos en File > New File > R Script. Hay otro tipo de documentos de R, como Markdown, muy útil para realizar informes. Vemos que se abre un nuevo panel como si fuera una hoja en blanco en la que iremos escribiendo nuestras líneas de código. Podemos ejecutar cualquiera de las operaciones anteriores:

3+2

```
## [1] 5
```

Para ejecutar el código que está escrito en el script pinchamos en “Run” o pulsamos la combinación de teclas CTRL+INTRO. Observad que el resultado se visualiza en la consola: tanto la instrucción como el resultado pero, ¡recordad!, lo que escribimos en la consola no se queda guardado. No tenemos que ejecutar línea por línea, sino que podemos ejecutar bloques de código seleccionando todas las líneas que queramos ejecutar.

3. El entorno En la parte derecha superior hay otro panel con varias pestañas. Destacaremos dos:
 - En la pestaña “History” podemos consultar las instrucciones que hemos ido ejecutando. Si pinchamos en alguna de ellas vemos que se copia en la consola. Otra forma de recuperar instrucciones ya ejecutadas es, situándonos en la consola, usar las flechas de hacia arriba y hacia abajo del teclado.
 - En la pestaña “Environment” aparecerán todos los elementos que forman parte del entorno de R. El entorno es el conjunto de objetos que están activos durante nuestra sesión de R. Ahora mismo en nuestro entorno únicamente están las funcionalidades de R base, pero veremos cómo va creciendo a lo largo de la sesión.
4. Gráficos y otros En la parte derecha inferior aparece otro panel con varias pestañas. Destacaremos tres:
 - Files. Aquí podemos navegar por las diferentes carpetas de nuestro ordenador.
 - Plots. Aquí se visualizan los gráficos que generamos.
 - Help. En esta pestaña obtendremos ayuda de R si se la pedimos.

3. R como calculadora

El uso más sencillo de R es como calculadora. Aquí tenemos algunas operaciones básicas.

```
3+2 # suma
```

```
## [1] 5
```

```
3-2 # resta
```

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

```
3*2 # multiplicación
```

```
## [1] 6
```

```
3/2 # división con decimales
```

```
## [1] 1.5
```

```
3/%2 # división entera
```

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

```
3**2 # potencia
```

```
## [1] 9
```

```
3^2 # potencia
```

```
## [1] 9
```

4. Objetos en R

Hasta ahora hemos visto el uso de R como calculadora. Sin embargo, las instrucciones que tendremos que ejecutar en un análisis de datos serán muchas otras: importar el conjunto de datos, filtrarlo, hacer operaciones con las variables de nuestro interés, hacer un gráfico, etc. Por ello, debemos conocer cómo interpreta y almacena R la información.

4.1 Concepto de variable

En primer lugar, tenemos que recordar que las instrucciones que se ejecutan en la consola no se guardan, pero necesitamos que R la recuerde y la tengamos accesible para seguir operando sobre ella.

Por ejemplo, hasta ahora hemos realizado operaciones sencillas, pero si pensamos en hacer varias operaciones, seguramente nos interese guardar el resultado de algunas de ellas. Podemos asignar el resultado $3+2$ a una variable que se llame “a”. La asignación de valores a una variable se puede hacer con la combinación “<-”, como si fuera una flecha, o con el signo “=”. Por convenio se prefiere la primera. Probad a ejecutar este código y veréis cómo quedan asignado el valor 5 a la variable “a”. En R no importa si ponemos espacio entre la variable y su valor, lo que puede ser distinto en otros lenguajes. Lo que sí importa es si escribimos un espacio entre “<” y “-”, ya que R interpreta otra cosa... Cada vez que hay un espacio, R interpreta que hay un elemento distinto.

```
a <- 3+2
a=3+2
a < -3 # ¡OJO!
```

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

Definir una variable es la forma en que R se guarda la información para tenerla disponible durante el rato que estamos trabajando en nuestro proyecto. Fijaos en el panel superior derecho: la variable que hemos creado ahora aparece en la pestaña “environment”. El “environment” o **entorno** es el conjunto de variables que están activos en R durante nuestra sesión. Podemos eliminar las variables del entorno pinchando en el símbolo de la escoba.

Las variables no tienen por qué ser números, también pueden ser palabras.

```
b <- "Hola"
c <- "Adios"
```

Las variables se pueden sobrescribir y reasignarles otro valor.

```
c<-4
```

Podemos nombrar a las variables como queramos pero teniendo en cuenta algunas reglas:

- **NUNCA** debemos nombrar una variable con un nombre de una variable que ya exista en R. Por ejemplo, no sería apropiado crear la variable `sum<-2+3`, porque `sum()` es una función que existe en R y podemos crear conflictos. Ante la duda podemos ejecutar el comando `help()` para consultar si existe una variable, (`help(sum)`); si no nos devuelve nada entonces es que esa variable no está predefinida en R.

- Las variables **no contienen espacios** porque si no R las interpretará como dos objetos distintos.
- Por convención se escriben en **minúscula** y donde querríamos poner un espacio pondremos un guión bajo (o un punto). Estas convenciones pueden ser distintas en otros lenguajes de programación.
- Intentad que los nombres de vuestras variables sean **representativos** de su significado, tanto por facilitaros vuestro trabajo como por si lo lee otra persona. Puede que creéis muchas variables y necesitaréis saber qué es cada una de ellas, sobre todo si volvéis a mirar vuestro script pasado un tiempo.

4.2 Concepto de objeto

La información que almacenamos en variables puede ser de distinto tipo. R tiene una idea preconcebida del tipo de información con el que va a trabajar. A cada tipo de información se le llama objeto. Cada objeto tiene asociadas una serie de operaciones o instrucciones que se le pueden aplicar. Cualquier tipo de objeto se puede asignar a una variable. Existen diferentes tipos de objetos en R base: vectores, listas, matrices, conjuntos de datos,... y otros objetos que son propios de determinados paquetes. Es difícil decidir qué tipo de objeto comenzar a explicar primero porque muchas veces la explicación de uno depende de otro. Normalmente se comienza por los vectores porque son el tipo de objeto más sencillo. Sin embargo, iremos en el orden que nos parece más didáctico a favor de esta sesión.

Funciones (*function*)

En primer lugar, queremos que sepáis que R cuenta con funciones para poder trabajar con otro tipo de objetos. Las funciones son objetos que podemos ver como una máquina a la que damos unos datos y nos devuelve un resultado. Por ejemplo, para obtener la raíz cuadrada de 2 podemos emplear la función `sqrt()`

```
sqrt(2)
```

```
## [1] 1.414214
```

Iremos viendo poco a poco varias funciones, pero nos gustaría resaltar dos por su utilidad:

- * La función `class()` nos dice el tipo de objeto. Como hemos dicho, cada tipo de objeto en R lleva asociadas unas operaciones que se pueden realizar y otras que no. A veces obtenemos errores porque estamos intentando aplicar una operación no permitida. Si sabemos el tipo de objeto con el que estamos trabajando, nos daremos cuenta de qué podemos y qué no podemos aplicarle. Si le preguntamos a R qué tipo de objeto es `sqrt`, nos dirá que es una función. Quizás ahora no le veáis la utilidad, pero hay clases de objetos más complicados y ocasiones en que tras muchas líneas de código ya no se recuerda de qué tipo son algunas variables.

```
class(sqrt)
```

```
## [1] "function"
```

- La función `help()` nos da información de las funciones. Si le pedimos ayuda sobre la función `sqrt`, vemos que nos dice que está implementada en R base y que calcula la raíz cuadrada. Además, nos informa de qué tipo de objeto podemos introducir como entrada y otros parámetros que se puedan modificar, así como ejemplos de cómo usarla.

```
help(sqrt)
```

```
## starting httpd help server ... done
```

Nosotros también podríamos crear nuestras propias funciones, aunque esto se escapa del objetivo de esta sesión.

Vectores (*numeric*, *character*)

Los vectores son objetos en los que podemos almacenar más de un valor, tanto números (*numeric*) como caracteres (*character*). Los vectores se crean con la función `c()`, que significa concatenar, separando los elementos con una coma.

```
numeros <- c(3,2,5,10,1,12)
arboles<-c("pino","castaño", "manzano", "roble")
```

Los valores de un vector son del mismo tipo. Por ejemplo, si creo un vector que tiene números y caracteres, R interpreta los números como caracteres.

```
numeros.caracteres<-c(1,2,"pino","castaño")
class(numeros.caracteres)
```

```
## [1] "character"
```

Para acceder a los elementos de un vector empleamos los corchetes. Los elementos en R empiezan a contarse desde el 1 (otros lenguajes empiezan en 0). Así accedemos al segundo elemento:

```
arboles[2]
```

```
## [1] "castaño"
```

Este índice que R reconoce lo podemos usar para ordenar los datos. Vamos a decirle que queremos ver el vector “arboles” pero que nos muestre primero el tercer elemento, luego el primero y finalmente el segundo.

```
arboles[c(3,1,2)]
```

```
## [1] "manzano" "pino"      "castaño"
```

También tenemos algunas funciones en R para ordenar vectores. La función `sort()` nos devuelve los números ordenados de menor a mayor. Si indicamos el parámetro `decreasing=TRUE`, entonces los devuelve de mayor a menor.

```
sort(numeros)
```

```
## [1] 1 2 3 5 10 12
```

```
sort(numeros, decreasing = TRUE)
```

```
## [1] 12 10 5 3 2 1
```

La función `order()` funciona igual que `sort()`, pero en vez de devolvernos los números directamente, nos devuelve el puesto que ocupan en el vector.

```
order(numeros)
```

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

```
order(numeros, decreasing = TRUE)
```

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

Curiosamente, las funciones `order()` y `sort()` también se pueden aplicar en vectores de caracteres, que nos devuelven ordenados alfabéticamente.

```
sort(arboles)
```

```
## [1] "castaño" "manzano" "pino"     "roble"
```

```
order(arboles)
```

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

Conjuntos de datos (*data.frame*)

Ya hemos visto los dos tipos de objetos básicos para poder explicar el tipo de objeto que más nos interesa hoy: los conjuntos de datos. Los conjuntos de datos son tablas con un número de filas y de columnas. Se crean con la función `data.frame()`, indicando el nombre de cada variable (cada columna) y los valores que contiene (es decir, las observaciones para cada una).

```
df<-data.frame(meses=c("enero","febrero","marzo","abril","mayo","junio",
                        "julio","agosto","septiembre","octubre",
                        "noviembre","diciembre"),
               dias=c(31,28,31,30,31,30,31,31,30,31,30,31))
```

Podemos acceder a las columnas del `data.frame` con el signo del dólar.

```
df$meses
```

```
## [1] "enero"      "febrero"    "marzo"      "abril"      "mayo"
## [6] "junio"      "julio"      "agosto"    "septiembre" "octubre"
## [11] "noviembre"  "diciembre"
```

```
df$dias
```

```
## [1] 31 28 31 30 31 30 31 31 30 31 30 31
```

También podemos acceder con los corchetes, teniendo en cuenta que los `data.frame` tienen dos dimensiones: filas y columnas. Por lo tanto, indicamos primero las filas que queremos seleccionar y luego las columnas, separando los números por una coma. Si no ponemos ningún número, R interpretará que lo queremos todo.

```
df[,] # nos devuelve todas las filas y todas las columnas
```

```
##      meses dias
## 1      enero  31
## 2    febrero  28
## 3      marzo  31
## 4      abril  30
## 5       mayo  31
## 6      junio  30
## 7      julio  31
## 8      agosto  31
## 9  septiembre  30
## 10     octubre  31
## 11  noviembre  30
## 12  diciembre  31
```

```
df[,1] # nos devuelve todas las filas y la primera columna
```

```
## [1] "enero"      "febrero"    "marzo"      "abril"      "mayo"
## [6] "junio"      "julio"      "agosto"     "septiembre" "octubre"
## [11] "noviembre"  "diciembre"
```

```
df[1,] # nos devuelve la primera fila y todas las columnas
```

```
## meses dias
## 1 enero 31
```

```
df[, "dias"] # nos devuelve todas las filas y la columna "dias"
```

```
## [1] 31 28 31 30 31 30 31 31 30 31 30 31
```

5. Rutas y directorios

Una cosa importante para importar y exportar datos en R es saber indicarle en qué lugar de nuestro ordenador están o queremos guardar nuestros datos.

Directorio de trabajo

El directorio de trabajo es lugar en nuestro ordenador (la carpeta) donde estamos trabajando y donde, si no indicamos otra cosa, R buscará y escribirá información cuando se lo pidamos. Para saber cuál es nuestro directorio de trabajo usamos la función `getwd()` (*get working directory*).

```
getwd()
```

```
## [1] "C:/Users/bea_f/OneDrive/Escritorio/CIB2025/Taller R"
```

Rutas completas y relativas

La función `getwd()` nos devuelve la ruta de carpetas que tendríamos que ir abriendo desde el disco C hasta nuestra carpeta de trabajo. Esto se llama una **ruta completa**. Podemos saber qué archivos hay en nuestro directorio con la función `list.files()`

```
list.files()
```

```
## [1] "BCA_assay.xlsx"
## [2] "BCA_assay_04-12-23.csv"
## [3] "BCA_assay_04-12-23.xlsx"
## [4] "Figuras.pptx"
## [5] "Figure2.png"
## [6] "Figure3.png"
## [7] "Figure4.png"
## [8] "Figure5.png"
## [9] "Figure6.png"
## [10] "Figure7.png"
## [11] "HistoriaR.png"
## [12] "IntroduccionBioestadisticaUCOPress.pdf"
## [13] "R_CIB2024.pdf"
## [14] "R_CIB2025.pdf"
## [15] "R_CIB2025.R"
## [16] "R_CIB2025.Rmd"
## [17] "R_CIB2025_Bea.Rmd"
## [18] "tmb_mskcc_2018_clinical_data.tsv"
## [19] "trackdown.R"
```

Esta función nos devuelve los archivos solo con su nombre, en forma de **ruta relativa** al directorio en el que estamos. Las rutas relativas se expresan respecto a un directorio, que puede ser el actual u otro, pero no respecto al disco.

Podemos cambiar nuestro directorio de trabajo con la función `setwd()`, indicando la ruta de la carpeta deseada, sea en forma completa o relativa. Aquí es muy útil el tabulador para ir autocompletando sin necesidad de que nos acordemos del orden de las carpetas.

6. Operaciones con conjuntos de datos

Ahora que sabemos un poco de cómo funciona R vamos a seguir profundizando en los conjuntos de datos, que creemos que es lo que más podéis usar en esta etapa (TFG, TFM, ect.). Vamos a ver ejemplos de las operaciones más comunes que nos puede interesar hacer.

6.1. Importar un conjunto de datos

1. Importar datos de Excel (.xlsx) Las hojas de cálculo como Excel son muy prácticas para guardar datos y seguramente así lo hagáis. Para leer datos de R en Excel usamos el paquete `readxl`. Si no está instalado hay que instalarlo y luego cargar el paquete al entorno. Os proporcionamos unos datos de absorbancia de un ensayo BCA para medir concentración de proteínas. Tenemos las funciones `readxl()` y `readxlsx()`. En este caso usamos la función `readxls()` porque nuestro archivo es .xlsx.

```
# install.packages("readxl") # Descomentar esta línea para instalarlo
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.3.1
```

```
bca<-read_xlsx("BCA_assay.xlsx",sheet=1,skip = 1)
```

2. Importar datos delimitados por separadores En general, cualquier archivo que contenga datos tiene los valores organizados en filas y dentro de cada fila los valores separados por algún carácter (separador) para indicar que pertenecen a diferentes columnas. Este separador puede ser un espacio (.txt), una coma, (.csv) o un tabulador (.tsv) En general, la función `read.delim()` detecta bien el tipo de separador que se ha empleado para delimitar las columnas. También podemos emplear la función `read.table()`, aunque en esta sí tenemos que indicarle cuál es el separador.

```
tmb<-read.delim("tmb_mskcc_2018_clinical_data.tsv")
tmb<-read.table("tmb_mskcc_2018_clinical_data.tsv",header=TRUE,sep="\t")

# read.csv para archivos separados por ","
help("read.csv")
```

Además hay paquetes fuera de R base que también sirven para leer archivos, como **readr** y **openslx**.

6.2. Conocer la estructura de un conjunto de datos

Lo primero que tenemos que conocer de nuestro conjunto de datos es el número de observaciones (filas) y variables (columnas), así como el tipo de datos que contiene. Esto lo podemos obtener con la función `str()`.

```
nrow(bca)
```

```
## [1] 11
```

```
ncol(bca)
```

```
## [1] 4
```



```
colnames(bca)
```

```
## [1] "nombre"      "medida1"      "medida2"      "concentracion"
```

```
head(bca)
```

```
## # A tibble: 6 x 4
##   nombre medida1 medida2 concentracion
##   <chr>    <dbl>    <dbl> <chr>
## 1 A      0.824    0.771 2000
## 2 B      0.639    0.589 1500
## 3 C      0.477    0.468 1000
## 4 D      0.372    0.373 750
## 5 E      0.272    0.302 500
## 6 F      0.215    0.222 250
```

```
tail(bca)
```

```
## # A tibble: 6 x 4
##   nombre    medida1 medida2 concentracion
##   <chr>    <dbl>    <dbl> <chr>
## 1 F      0.215    0.222 250
## 2 G      0.162    0.162 125
## 3 H      0.12     0.13  25
## 4 I      0.114    0.11   0
## 5 muestra1 0.241    0.247 NA
## 6 muestra2 0.338    0.24  NA
```

```
# Funciones de resumen:
```

```
str(bca)
```

```
## tibble [11 x 4] (S3: tbl_df/tbl/data.frame)
## $ nombre      : chr [1:11] "A" "B" "C" "D" ...
## $ medida1     : num [1:11] 0.824 0.639 0.477 0.372 0.272 0.215 0.162 0.12 0.114 0.2
## $ medida2     : num [1:11] 0.771 0.589 0.468 0.373 0.302 0.222 0.162 0.13 0.11 0.24
## $ concentracion: chr [1:11] "2000" "1500" "1000" "750" ...
```

```
summary(bca)
```

```
##      nombre          medida1          medida2      concentracion
## Length:11      Min.   :0.1140      Min.   :0.1100      Length:11
## Class :character 1st Qu.:0.1885      1st Qu.:0.1920      Class :character
## Mode  :character Median :0.2720      Median :0.2470      Mode  :character
##                  Mean  :0.3431      Mean   :0.3285
##                  3rd Qu.:0.4245      3rd Qu.:0.4205
##                  Max.   :0.8240      Max.   :0.7710
```

Acceder a los datos: data.frame[filas, columna]

Acceder a las columnas:

```
bca$nombre
```

```
## [1] "A"      "B"      "C"      "D"      "E"      "F"
## [7] "G"      "H"      "I"      "muestra1" "muestra2"
```

```
bca$medida1
```

```
## [1] 0.824 0.639 0.477 0.372 0.272 0.215 0.162 0.120 0.114 0.241 0.338
```

```
bca$medida2
```

```
## [1] 0.771 0.589 0.468 0.373 0.302 0.222 0.162 0.130 0.110 0.247 0.240
```

Acceder a una fila concreta:

```
bca[6,]
```

```
## # A tibble: 1 x 4
##   nombre medida1 medida2 concentracion
##   <chr>    <dbl>    <dbl> <chr>
## 1 F      0.215    0.222 250
```

Acceder a un valor específico:

```
bca[3,2]
```

```
## # A tibble: 1 x 1
##   medida1
##   <dbl>
## 1 0.477
```

```
# Acceder a un subconjunto de datos:
```

```
bca[1:3,1:3]
```

```
## # A tibble: 3 x 3
##   nombre medida1 medida2
##   <chr>    <dbl>   <dbl>
## 1 A      0.824    0.771
## 2 B      0.639    0.589
## 3 C      0.477    0.468
```

```
bca[1:3, c(1,4)] # acceder por posición
```

```
## # A tibble: 3 x 2
##   nombre concentracion
##   <chr>   <chr>
## 1 A     2000
## 2 B     1500
## 3 C     1000
```

```
bca[1:3, c("nombre", "concentracion")] # Acceder por nombre
```

```
## # A tibble: 3 x 2
##   nombre concentracion
##   <chr>   <chr>
## 1 A     2000
## 2 B     1500
## 3 C     1000
```

```
bca[1:3, c("concentracion", "nombre")] # Reordenar columnas
```

```
## # A tibble: 3 x 2
##   concentracion nombre
##   <chr>         <chr>
## 1 2000         A
## 2 1500         B
## 3 1000         C
```

Conocer las variables categoricas y cuantos registros tengo de cada categoria/grupo: Con un conjunto de datos más grande:

```
colnames(tmb)
```

```
## [1] "Study.ID"
## [2] "Patient.ID"
## [3] "Sample.ID"
## [4] "Age.at.Which.Sequencing.was.Reported..Days."
## [5] "Age.Group.at.Diagnosis.in.Years"
## [6] "Cancer.Type"
## [7] "Cancer.Type.Detailed"
## [8] "Drug.Type"
## [9] "Gene.Panel"
## [10] "Institute.Source"
## [11] "Metastatic.Site"
## [12] "Mutation.Count"
## [13] "Oncotree.Code"
## [14] "Overall.Survival..Months."
## [15] "Overall.Survival.Status"
## [16] "Primary.Tumor.Site"
## [17] "Sample.Class"
## [18] "Number.of.Samples.Per.Patient"
## [19] "Sample.coverage"
## [20] "Sample.Type"
## [21] "Sex"
## [22] "Somatic.Status"
## [23] "TMB..nonsynonymous."
## [24] "Tumor.Purity"
```

```
str(tmb) #1661 observaciones, 24 variables
```

```
## 'data.frame':    1661 obs. of  24 variables:
## $ Study.ID          : chr  "tmb_mskcc_2018" "tmb_mskcc_2018"
## $ Patient.ID        : chr  "P-0000057" "P-0000062" "P-0000062"
## $ Sample.ID         : chr  "P-0000057-T01-IM3" "P-0000062-T01-IM3"
## $ Age.at.Which.Sequencing.was.Reported..Days.: int  41 80 62 66 61 63 47 44 67 60 ...
## $ Age.Group.at.Diagnosis.in.Years           : chr  "31-50" ">71" "61-70" "61-70" ...
## $ Cancer.Type                               : chr  "Breast Cancer" "Esophagogastric Cancer"
## $ Cancer.Type.Detailed                     : chr  "Breast Mixed Ductal and Lobular"
## $ Drug.Type                                : chr  "PD-1/PDL-1" "PD-1/PDL-1" "PD-1/PDL-1"
## $ Gene.Panel                                : chr  "IMPACT341" "IMPACT341" "IMPACT341"
## $ Institute.Source                          : chr  "MSKCC" "MSKCC" "MSKCC" "MSKCC"
## $ Metastatic.Site                          : chr  NA NA NA NA ...
## $ Mutation.Count                           : int  5 6 13 10 12 12 6 3 5 8 ...
## $ Oncotree.Code                            : chr  "MDLC" "GEJ" "BLCA" "BLCA" ...
```

```
## $ Overall.Survival..Months.      : int  0 1 42 43 57 12 18 4 1 8 ...
## $ Overall.Survival.Status        : chr   "1:DECEASED" "1:DECEASED" "0:LIV
## $ Primary.Tumor.Site             : chr   "Breast" "Esophagus" "Bladder" "
## $ Sample.Class                   : chr   "Tumor" "Tumor" "Tumor" "Tumor"
## $ Number.of.Samples.Per.Patient  : int   1 1 1 1 1 1 1 1 1 1 ...
## $ Sample.coverage                : int   835 1176 900 795 905 783 997 506
## $ Sample.Type                    : chr   "Primary" "Primary" "Primary" "P
## $ Sex                            : chr   "Female" "Male" "Male" "Male" ..
## $ Somatic.Status                 : chr   "Matched" "Matched" "Matched" "M
## $ TMB..nonsynonymous.            : num   5.55 6.65 15.53 9.98 13.31 ...
## $ Tumor.Purity                   : chr   "25" "30" "70" "30" ...
```

```
summary(tmb)
```

```
##      Study.ID      Patient.ID      Sample.ID
## Length:1661      Length:1661      Length:1661
## Class :character Class :character Class :character
## Mode  :character Mode  :character Mode  :character
##
##
##
## Age.at.Which.Sequencing.was.Reported..Days. Age.Group.at.Diagnosis.in.Years
## Min.      :15.00                               Length:1661
## 1st Qu.:53.00                               Class :character
## Median :63.00                               Mode  :character
## Mean    :61.41
## 3rd Qu.:71.00
## Max.    :90.00
## NA's     :1
## Cancer.Type      Cancer.Type.Detailed Drug.Type      Gene.Panel
## Length:1661      Length:1661          Length:1661      Length:1661
## Class :character Class :character      Class :character      Class :character
## Mode  :character Mode  :character      Mode  :character      Mode  :character
##
##
##
## Institute.Source  Metastatic.Site  Mutation.Count  Oncotree.Code
## Length:1661      Length:1661          Min.      : 1.00      Length:1661
## Class :character Class :character      1st Qu.: 4.00      Class :character
## Mode  :character Mode  :character      Median   : 6.00      Mode  :character
##                                     Mean      : 12.61
##                                     3rd Qu.: 12.00
```

```

##                                     Max.      :213.00
##                                     NA's       :51
## Overall.Survival..Months. Overall.Survival.Status Primary.Tumor.Site
## Min.      : 0.00                Length:1661                Length:1661
## 1st Qu.: 4.00                  Class :character          Class :character
## Median :11.00                 Mode  :character          Mode  :character
## Mean    :14.08
## 3rd Qu.:20.00
## Max.    :80.00
##
## Sample.Class      Number.of.Samples.Per.Patient Sample.coverage
## Length:1661      Min.      :1                Min.      : 56.0
## Class :character 1st Qu.:1                1st Qu.: 581.0
## Mode  :character Median :1                Median : 744.0
##                      Mean  :1                Mean   : 736.1
##                      3rd Qu.:1              3rd Qu.: 891.0
##                      Max.    :1              Max.    :1744.0
##
## Sample.Type      Sex      Somatic.Status      TMB..nonsynonymous.
## Length:1661      Length:1661      Length:1661      Min.      : 0.000
## Class :character Class :character      Class :character 1st Qu.: 2.936
## Mode  :character Mode  :character      Mode  :character Median : 5.872
##                      Mean   : 11.562
##                      3rd Qu.: 11.092
##                      Max.    :207.489
##
## Tumor.Purity
## Length:1661
## Class :character
## Mode  :character
##
##
##
##

```

```
table(tmb$Sex) # Distribución de sexos
```

```

##
## Female   Male
##    627    1034

```

```
table(tmb$Cancer.Type) # Tipos de cancer recogidos
```

```
##
```

```
##          Bladder Cancer          Breast Cancer
##                215                44
## Cancer of Unknown Primary    Colorectal Cancer
##                88                110
##      Esophagogastric Cancer          Glioma
##                126                117
##      Head and Neck Cancer          Melanoma
##                139                320
## Non-Small Cell Lung Cancer    Renal Cell Carcinoma
##                350                151
## Skin Cancer, Non-Melanoma
##                1
```

```
table(tmb$Sample.Type) # Tipos de muestras: al diagnostico o metastásicas
```

```
##
## Metastasis    Primary
##      930      731
```

```
table(tmb$Drug.Type) # Tipos de fármaco
```

```
##
##      Combo      CTLA4 PD-1/PDL-1
##      255      99      1307
```

Añadir columnas:

```
# todas las filas tendrán el valor asignado
bca$fecha <- "12/02/2025"
bca$factor_correccion <- 0.5

# especificar el valor de cada fila
valores <- c(0.1,0.5,1.6,1.80,1.76,0.90, 1.60, 1.81,1.74, 2.4, 0.9)
bca$factor_correccion <- valores
```

6.3. Filtrar un conjunto de datos

- Operadores lógicos Los operadores lógicos nos sirven para hacerle preguntas a R y que nos responda con verdadero o falso.

```
# IMPORTANTE ESCRIBIR DOS VECES "="
1==2 ## ¿1 es igual a 2?
```

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

```
1!=2 ## ¿1 es diferente de 2?
```

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

```
1<0 ## ¿1 es menor que 0?
```

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

```
1>0 ## ¿1 es mayor que 0?
```

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

Filtros básicos:

```
# Filtrar conjunto de datos
bca[bca$nombre == "muestra1",]
```

```
## # A tibble: 1 x 6
##   nombre   medida1 medida2 concentracion fecha      factor_correccion
##   <chr>     <dbl>   <dbl> <chr>          <chr>          <dbl>
## 1 muestra1  0.241    0.247 NA           12/02/2025      2.4
```

```
bca[bca$nombre == "muestra2",]
```

```
## # A tibble: 1 x 6
##   nombre   medida1 medida2 concentracion fecha      factor_correccion
##   <chr>     <dbl>   <dbl> <chr>          <chr>          <dbl>
## 1 muestra2  0.338    0.24 NA           12/02/2025      0.9
```

```
bca[bca$medida1 > 0.5,] # Importante: la "," indica que queremos todas las columnas.
```

```
## # A tibble: 2 x 6
##   nombre medida1 medida2 concentracion fecha      factor_correccion
##   <chr>     <dbl>   <dbl> <chr>          <chr>          <dbl>
## 1 A         0.824    0.771 2000          12/02/2025      0.1
## 2 B         0.639    0.589 1500          12/02/2025      0.5
```



```
# Obtener valores de una columna
bca$nombre[bca$medida1 >0.5] # Nota: Ya seleccionamos una columna,
```

```
## [1] "A" "B"
```

```
# no hace falta la ",".
```

Filtros combinados:

```
# Muestras de cancer de mama
tmb_mama<-tmb[tmb$Cancer.Type == "Breast Cancer",]

# Filtras columnas:
tmb_col<-tmb[tmb$Cancer.Type == "Breast Cancer",
             c("Sample.ID", "Patient.ID", "Sample.Type", "Mutation.Count")]
tmb_filt  <- tmb[tmb$Cancer.Type == "Breast Cancer",
                c("Sample.ID", "Patient.ID", "Sample.Type", "Mutation.Count")]
tmb_filt
```

```
##           Sample.ID Patient.ID Sample.Type Mutation.Count
## 1    P-0000057-T01-IM3 P-0000057      Primary           5
## 17   P-0000247-T01-IM3 P-0000247    Metastasis          2
## 25   P-0000392-T01-IM3 P-0000392    Metastasis          3
## 26   P-0000422-T01-IM3 P-0000422    Metastasis          6
## 29   P-0000447-T01-IM3 P-0000447      Primary          3
## 34   P-0000547-T01-IM3 P-0000547    Metastasis          6
## 41   P-0000638-T01-IM3 P-0000638    Metastasis         NA
## 50   P-0000704-T01-IM3 P-0000704    Metastasis          9
## 97   P-0001312-T01-IM3 P-0001312    Metastasis          2
## 135  P-0001785-T02-IM3 P-0001785      Primary          2
## 156  P-0002041-T01-IM3 P-0002041    Metastasis         NA
## 210  P-0002626-T01-IM3 P-0002626    Metastasis         NA
## 230  P-0002789-T01-IM3 P-0002789    Metastasis          2
## 269  P-0003224-T01-IM5 P-0003224    Metastasis         15
## 272  P-0003241-T01-IM5 P-0003241      Primary          3
## 274  P-0003265-T01-IM5 P-0003265    Metastasis          4
## 276  P-0003273-T01-IM5 P-0003273    Metastasis          1
## 461  P-0005131-T01-IM5 P-0005131    Metastasis          2
## 470  P-0005176-T01-IM5 P-0005176    Metastasis          2
## 485  P-0005274-T01-IM5 P-0005274    Metastasis          6
## 527  P-0005712-T01-IM5 P-0005712      Primary          4
## 549  P-0005855-T01-IM5 P-0005855    Metastasis          6
## 589  P-0006227-T01-IM5 P-0006227    Metastasis          7
```

## 657	P-0006842-T01-IM5	P-0006842	Metastasis	6
## 676	P-0007014-T01-IM5	P-0007014	Primary	3
## 693	P-0007127-T01-IM5	P-0007127	Metastasis	4
## 706	P-0007349-T01-IM5	P-0007349	Primary	5
## 818	P-0008571-T01-IM5	P-0008571	Metastasis	2
## 890	P-0009297-T01-IM5	P-0009297	Metastasis	7
## 902	P-0009364-T01-IM5	P-0009364	Metastasis	28
## 921	P-0009498-T01-IM5	P-0009498	Metastasis	5
## 942	P-0009727-T01-IM5	P-0009727	Primary	1
## 994	P-0010252-T01-IM5	P-0010252	Metastasis	5
## 1026	P-0010669-T01-IM5	P-0010669	Primary	4
## 1037	P-0010800-T01-IM5	P-0010800	Metastasis	4
## 1055	P-0010994-T01-IM5	P-0010994	Metastasis	4
## 1096	P-0011552-T01-IM5	P-0011552	Metastasis	6
## 1142	P-0012803-T01-IM5	P-0012803	Metastasis	5
## 1145	P-0012825-T02-IM6	P-0012825	Metastasis	1
## 1178	P-0013210-T01-IM5	P-0013210	Metastasis	3
## 1319	P-0015445-T01-IM6	P-0015445	Primary	3
## 1344	P-0015713-T01-IM6	P-0015713	Metastasis	1
## 1356	P-0015905-T01-IM6	P-0015905	Primary	5
## 1387	P-0016226-T01-IM6	P-0016226	Metastasis	4

```
str(tmb_filt)
```

```
## 'data.frame': 44 obs. of 4 variables:
## $ Sample.ID : chr "P-0000057-T01-IM3" "P-0000247-T01-IM3" "P-0000392-T01-IM3" "
## $ Patient.ID : chr "P-0000057" "P-0000247" "P-0000392" "P-0000422" ...
## $ Sample.Type : chr "Primary" "Metastasis" "Metastasis" "Metastasis" ...
## $ Mutation.Count: int 5 2 3 6 3 6 NA 9 2 2 ...
```

```
# Muestras metastasicas de cancer de mama:
```

```
tmb_mama_metastasis<-tmb[tmb$Cancer.Type == "Breast Cancer" &
                        tmb$Sample.Type == "Metastasis",]
```

```
# Muestras de varios tipos de cancer: filtro por listado de valores
table(tmb$Cancer.Type) # Tipos de cancer recogidos
```

##		
##	Bladder Cancer	Breast Cancer
##	215	44
##	Cancer of Unknown Primary	Colorectal Cancer
##	88	110
##	Esophagogastric Cancer	Glioma

```
##                               126                               117
##      Head and Neck Cancer                               Melanoma
##                               139                               320
## Non-Small Cell Lung Cancer      Renal Cell Carcinoma
##                               350                               151
## Skin Cancer, Non-Melanoma
##                               1
```

```
tmb_cancers_filt<-tmb[tmb$Cancer.Type %in%
  c("Breast Cancer", "Melanoma", "Skin Cancer, Non-Melanoma"),] # dataframe
tmb_id_cancer<-tmb$Sample.ID[tmb$Cancer.Type %in%
  c("Breast Cancer", "Melanoma", "Skin Cancer, Non-Melanoma")] # listado
table(tmb$Age.Group.at.Diagnosis.in.Years)
```

```
##
##   <30   >71  31-50  50-60  61-70
##    50   413   283   416   499
```

```
tmb_id_age<-tmb$Sample.ID[tmb$Age.Group.at.Diagnosis.in.Years %in%
  c("31-50", "50-60")]
```

```
subset(tmb, Cancer.Type == "Breast Cancer")
```

```
##      Study.ID Patient.ID      Sample.ID
## 1 tmb_mskcc_2018 P-0000057 P-0000057-T01-IM3
## 17 tmb_mskcc_2018 P-0000247 P-0000247-T01-IM3
## 25 tmb_mskcc_2018 P-0000392 P-0000392-T01-IM3
## 26 tmb_mskcc_2018 P-0000422 P-0000422-T01-IM3
## 29 tmb_mskcc_2018 P-0000447 P-0000447-T01-IM3
## 34 tmb_mskcc_2018 P-0000547 P-0000547-T01-IM3
## 41 tmb_mskcc_2018 P-0000638 P-0000638-T01-IM3
## 50 tmb_mskcc_2018 P-0000704 P-0000704-T01-IM3
## 97 tmb_mskcc_2018 P-0001312 P-0001312-T01-IM3
## 135 tmb_mskcc_2018 P-0001785 P-0001785-T02-IM3
## 156 tmb_mskcc_2018 P-0002041 P-0002041-T01-IM3
## 210 tmb_mskcc_2018 P-0002626 P-0002626-T01-IM3
## 230 tmb_mskcc_2018 P-0002789 P-0002789-T01-IM3
## 269 tmb_mskcc_2018 P-0003224 P-0003224-T01-IM5
## 272 tmb_mskcc_2018 P-0003241 P-0003241-T01-IM5
## 274 tmb_mskcc_2018 P-0003265 P-0003265-T01-IM5
## 276 tmb_mskcc_2018 P-0003273 P-0003273-T01-IM5
## 461 tmb_mskcc_2018 P-0005131 P-0005131-T01-IM5
```

## 470	tmb_mskcc_2018	P-0005176	P-0005176-T01-IM5
## 485	tmb_mskcc_2018	P-0005274	P-0005274-T01-IM5
## 527	tmb_mskcc_2018	P-0005712	P-0005712-T01-IM5
## 549	tmb_mskcc_2018	P-0005855	P-0005855-T01-IM5
## 589	tmb_mskcc_2018	P-0006227	P-0006227-T01-IM5
## 657	tmb_mskcc_2018	P-0006842	P-0006842-T01-IM5
## 676	tmb_mskcc_2018	P-0007014	P-0007014-T01-IM5
## 693	tmb_mskcc_2018	P-0007127	P-0007127-T01-IM5
## 706	tmb_mskcc_2018	P-0007349	P-0007349-T01-IM5
## 818	tmb_mskcc_2018	P-0008571	P-0008571-T01-IM5
## 890	tmb_mskcc_2018	P-0009297	P-0009297-T01-IM5
## 902	tmb_mskcc_2018	P-0009364	P-0009364-T01-IM5
## 921	tmb_mskcc_2018	P-0009498	P-0009498-T01-IM5
## 942	tmb_mskcc_2018	P-0009727	P-0009727-T01-IM5
## 994	tmb_mskcc_2018	P-0010252	P-0010252-T01-IM5
## 1026	tmb_mskcc_2018	P-0010669	P-0010669-T01-IM5
## 1037	tmb_mskcc_2018	P-0010800	P-0010800-T01-IM5
## 1055	tmb_mskcc_2018	P-0010994	P-0010994-T01-IM5
## 1096	tmb_mskcc_2018	P-0011552	P-0011552-T01-IM5
## 1142	tmb_mskcc_2018	P-0012803	P-0012803-T01-IM5
## 1145	tmb_mskcc_2018	P-0012825	P-0012825-T02-IM6
## 1178	tmb_mskcc_2018	P-0013210	P-0013210-T01-IM5
## 1319	tmb_mskcc_2018	P-0015445	P-0015445-T01-IM6
## 1344	tmb_mskcc_2018	P-0015713	P-0015713-T01-IM6
## 1356	tmb_mskcc_2018	P-0015905	P-0015905-T01-IM6
## 1387	tmb_mskcc_2018	P-0016226	P-0016226-T01-IM6
##	Age.at.Which.Sequencing.was.Reported..Days.		
## 1			41
## 17			50
## 25			40
## 26			52
## 29			36
## 34			62
## 41			35
## 50			49
## 97			58
## 135			43
## 156			62
## 210			59
## 230			43
## 269			60
## 272			49
## 274			58
## 276			39
## 461			68

## 470	31
## 485	69
## 527	43
## 549	57
## 589	43
## 657	44
## 676	44
## 693	73
## 706	32
## 818	50
## 890	40
## 902	63
## 921	54
## 942	44
## 994	53
## 1026	66
## 1037	64
## 1055	52
## 1096	73
## 1142	50
## 1145	37
## 1178	59
## 1319	60
## 1344	33
## 1356	52
## 1387	54
##	Age.Group.at.Diagnosis.in.Years Cancer.Type
## 1	31-50 Breast Cancer
## 17	50-60 Breast Cancer
## 25	31-50 Breast Cancer
## 26	50-60 Breast Cancer
## 29	31-50 Breast Cancer
## 34	61-70 Breast Cancer
## 41	31-50 Breast Cancer
## 50	31-50 Breast Cancer
## 97	50-60 Breast Cancer
## 135	31-50 Breast Cancer
## 156	61-70 Breast Cancer
## 210	50-60 Breast Cancer
## 230	31-50 Breast Cancer
## 269	50-60 Breast Cancer
## 272	31-50 Breast Cancer
## 274	50-60 Breast Cancer
## 276	31-50 Breast Cancer
## 461	61-70 Breast Cancer

## 470	31-50 Breast Cancer
## 485	61-70 Breast Cancer
## 527	31-50 Breast Cancer
## 549	50-60 Breast Cancer
## 589	31-50 Breast Cancer
## 657	31-50 Breast Cancer
## 676	31-50 Breast Cancer
## 693	>71 Breast Cancer
## 706	31-50 Breast Cancer
## 818	31-50 Breast Cancer
## 890	31-50 Breast Cancer
## 902	61-70 Breast Cancer
## 921	50-60 Breast Cancer
## 942	31-50 Breast Cancer
## 994	50-60 Breast Cancer
## 1026	61-70 Breast Cancer
## 1037	61-70 Breast Cancer
## 1055	50-60 Breast Cancer
## 1096	>71 Breast Cancer
## 1142	31-50 Breast Cancer
## 1145	31-50 Breast Cancer
## 1178	50-60 Breast Cancer
## 1319	50-60 Breast Cancer
## 1344	31-50 Breast Cancer
## 1356	50-60 Breast Cancer
## 1387	50-60 Breast Cancer
##	Cancer.Type.Detailed Drug.Type Gene.Panel
## 1	Breast Mixed Ductal and Lobular Carcinoma PD-1/PDL-1 IMPACT341
## 17	Breast Invasive Ductal Carcinoma Combo IMPACT410
## 25	Breast Invasive Ductal Carcinoma PD-1/PDL-1 IMPACT341
## 26	Breast Invasive Lobular Carcinoma CTLA4 IMPACT341
## 29	Breast Invasive Ductal Carcinoma Combo IMPACT341
## 34	Breast Invasive Lobular Carcinoma PD-1/PDL-1 IMPACT341
## 41	Breast Invasive Ductal Carcinoma CTLA4 IMPACT341
## 50	Breast Invasive Ductal Carcinoma CTLA4 IMPACT341
## 97	Breast Invasive Ductal Carcinoma PD-1/PDL-1 IMPACT341
## 135	Breast Invasive Ductal Carcinoma PD-1/PDL-1 IMPACT410
## 156	Breast Invasive Ductal Carcinoma PD-1/PDL-1 IMPACT341
## 210	Breast Invasive Ductal Carcinoma CTLA4 IMPACT341
## 230	Breast Invasive Ductal Carcinoma CTLA4 IMPACT341
## 269	Breast Invasive Ductal Carcinoma CTLA4 IMPACT410
## 272	Breast Mixed Ductal and Lobular Carcinoma Combo IMPACT410
## 274	Breast Invasive Ductal Carcinoma CTLA4 IMPACT410
## 276	Breast Invasive Ductal Carcinoma CTLA4 IMPACT410
## 461	Breast Invasive Ductal Carcinoma CTLA4 IMPACT410

## 470	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 485	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT410
## 527	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 549	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 589	Breast Invasive Lobular Carcinoma	PD-1/PDL-1	IMPACT410
## 657	Breast Invasive Ductal Carcinoma	Combo	IMPACT410
## 676	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 693	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 706	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 818	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT410
## 890	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT410
## 902	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT410
## 921	Breast Invasive Lobular Carcinoma	PD-1/PDL-1	IMPACT410
## 942	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 994	Breast Invasive Carcinoma, NOS	PD-1/PDL-1	IMPACT410
## 1026	Breast Invasive Ductal Carcinoma	CTLA4	IMPACT410
## 1037	Breast Invasive Lobular Carcinoma	CTLA4	IMPACT410
## 1055	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT410
## 1096	Breast Invasive Carcinoma, NOS	PD-1/PDL-1	IMPACT410
## 1142	Breast Invasive Cancer, NOS	CTLA4	IMPACT410
## 1145	Invasive Breast Carcinoma	PD-1/PDL-1	IMPACT468
## 1178	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT410
## 1319	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT468
## 1344	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT468
## 1356	Breast Invasive Ductal Carcinoma	PD-1/PDL-1	IMPACT468
## 1387	Breast Invasive Cancer, NOS	CTLA4	IMPACT468
##	Institute.Source Metastatic.Site Mutation.Count Oncotree.Code		
## 1	MSKCC	<NA>	5 MDLC
## 17	MSKCC	Liver	2 IDC
## 25	MSKCC	Pleura	3 IDC
## 26	MSKCC	Ovary	6 ILC
## 29	MSKCC	<NA>	3 IDC
## 34	MSKCC	Bone	6 ILC
## 41	MSKCC	Heart	NA IDC
## 50	MSKCC	Liver	9 IDC
## 97	MSKCC	Lymph Node	2 IDC
## 135	MSKCC	<NA>	2 IDC
## 156	MSKCC	Lymph Node	NA IDC
## 210	MSKCC	Lung	NA IDC
## 230	MSKCC	Chest Wall	2 IDC
## 269	MSKCC	Liver	15 IDC
## 272	MSKCC	<NA>	3 MDLC
## 274	MSKCC	Skin	4 IDC
## 276	MSKCC	Bone	1 IDC
## 461	MSKCC	Lung	2 IDC

## 470	MSKCC	Lymph Node	2	IDC
## 485	MSKCC	Chest Wall	6	IDC
## 527	MSKCC	<NA>	4	IDC
## 549	MSKCC	Bone	6	IDC
## 589	MSKCC	Ovary	7	ILC
## 657	MSKCC	Liver	6	IDC
## 676	MSKCC	<NA>	3	IDC
## 693	MSKCC	Lung	4	IDC
## 706	MSKCC	<NA>	5	IDC
## 818	MSKCC	Lymph Node	2	IDC
## 890	MSKCC	Skin	7	IDC
## 902	MSKCC	Lymph Node	28	IDC
## 921	MSKCC	Pleura	5	ILC
## 942	MSKCC	<NA>	1	IDC
## 994	MSKCC	Brain	5	BRCNOS
## 1026	MSKCC	<NA>	4	IDC
## 1037	MSKCC	Lymph Node	4	ILC
## 1055	MSKCC	Skin	4	IDC
## 1096	MSKCC	Bone	6	BRCNOS
## 1142	MSKCC	Lymph Node	5	BRCANOS
## 1145	MSKCC	Soft Tissue	1	BRCA
## 1178	MSKCC	Lung	3	IDC
## 1319	MSKCC	<NA>	3	IDC
## 1344	MSKCC	Liver	1	IDC
## 1356	MSKCC	<NA>	5	IDC
## 1387	MSKCC	Liver	4	BRCANOS
##	Overall.Survival..Months. Overall.Survival.Status Primary.Tumor.Site			
## 1		0	1:DECEASED	Breast
## 17		17	0:LIVING	Breast
## 25		3	1:DECEASED	Breast
## 26		1	1:DECEASED	Breast
## 29		12	1:DECEASED	Breast
## 34		3	1:DECEASED	Breast
## 41		1	1:DECEASED	Breast
## 50		1	1:DECEASED	Breast
## 97		3	1:DECEASED	Breast
## 135		13	0:LIVING	Breast
## 156		4	1:DECEASED	Breast
## 210		60	0:LIVING	Breast
## 230		9	1:DECEASED	Breast
## 269		28	0:LIVING	Breast
## 272		2	1:DECEASED	Breast
## 274		19	1:DECEASED	Breast
## 276		6	1:DECEASED	Breast
## 461		5	1:DECEASED	Breast

## 470	5	1:DECEASED	Breast
## 485	2	1:DECEASED	Breast
## 527	20	0:LIVING	Breast
## 549	3	1:DECEASED	Breast
## 589	4	1:DECEASED	Breast
## 657	9	1:DECEASED	Breast
## 676	5	1:DECEASED	Breast
## 693	15	1:DECEASED	Breast
## 706	14	1:DECEASED	Breast
## 818	15	0:LIVING	Breast
## 890	14	1:DECEASED	Breast
## 902	19	0:LIVING	Breast
## 921	13	0:LIVING	Breast
## 942	4	1:DECEASED	Breast
## 994	18	0:LIVING	Breast
## 1026	2	0:LIVING	Breast
## 1037	10	1:DECEASED	Breast
## 1055	4	1:DECEASED	Breast
## 1096	4	1:DECEASED	Breast
## 1142	3	0:LIVING	Breast
## 1145	14	0:LIVING	Breast
## 1178	3	1:DECEASED	Breast
## 1319	2	1:DECEASED	Breast
## 1344	9	0:LIVING	Breast
## 1356	1	1:DECEASED	Breast
## 1387	22	1:DECEASED	Breast
##	Sample.Class	Number.of.Samples.Per.Patient	Sample.coverage Sample.Type
## 1	Tumor	1	835 Primary
## 17	Tumor	1	967 Metastasis
## 25	Tumor	1	817 Metastasis
## 26	Tumor	1	572 Metastasis
## 29	Tumor	1	815 Primary
## 34	Tumor	1	711 Metastasis
## 41	Tumor	1	850 Metastasis
## 50	Tumor	1	477 Metastasis
## 97	Tumor	1	922 Metastasis
## 135	Tumor	1	782 Primary
## 156	Tumor	1	431 Metastasis
## 210	Tumor	1	186 Metastasis
## 230	Tumor	1	520 Metastasis
## 269	Tumor	1	528 Metastasis
## 272	Tumor	1	384 Primary
## 274	Tumor	1	833 Metastasis
## 276	Tumor	1	304 Metastasis
## 461	Tumor	1	909 Metastasis

## 470	Tumor	1	419	Metastasis
## 485	Tumor	1	685	Metastasis
## 527	Tumor	1	719	Primary
## 549	Tumor	1	772	Metastasis
## 589	Tumor	1	487	Metastasis
## 657	Tumor	1	961	Metastasis
## 676	Tumor	1	521	Primary
## 693	Tumor	1	601	Metastasis
## 706	Tumor	1	691	Primary
## 818	Tumor	1	1282	Metastasis
## 890	Tumor	1	670	Metastasis
## 902	Tumor	1	1086	Metastasis
## 921	Tumor	1	945	Metastasis
## 942	Tumor	1	616	Primary
## 994	Tumor	1	839	Metastasis
## 1026	Tumor	1	748	Primary
## 1037	Tumor	1	706	Metastasis
## 1055	Tumor	1	1010	Metastasis
## 1096	Tumor	1	913	Metastasis
## 1142	Tumor	1	790	Metastasis
## 1145	Tumor	1	1093	Metastasis
## 1178	Tumor	1	900	Metastasis
## 1319	Tumor	1	735	Primary
## 1344	Tumor	1	957	Metastasis
## 1356	Tumor	1	649	Primary
## 1387	Tumor	1	735	Metastasis
##	Sex	Somatic.Status	TMB..nonsynonymous.	Tumor.Purity
## 1	Female	Matched	5.5457765	25
## 17	Female	Matched	2.2183106	50
## 25	Female	Matched	3.3274659	40
## 26	Female	Unmatched	6.6549318	80
## 29	Female	Matched	3.3274659	60
## 34	Female	Matched	6.6549318	85
## 41	Female	Matched	0.0000000	30
## 50	Female	Matched	9.9823977	80
## 97	Female	Matched	2.2183106	60
## 135	Female	Matched	2.2183106	50
## 156	Female	Matched	0.0000000	10
## 210	Female	Matched	0.0000000	20
## 230	Female	Matched	2.2183106	30
## 269	Female	Matched	14.6807955	30
## 272	Female	Matched	2.9361591	70
## 274	Female	Matched	3.9148788	60
## 276	Female	Matched	0.9787197	60
## 461	Female	Matched	1.9574394	30

## 470	Female	Matched	1.9574394	50
## 485	Female	Matched	5.8723182	40
## 527	Female	Matched	3.9148788	70
## 549	Female	Matched	5.8723182	10
## 589	Female	Matched	6.8510379	70
## 657	Female	Matched	5.8723182	20
## 676	Female	Matched	2.9361591	40
## 693	Female	Matched	3.9148788	20
## 706	Female	Matched	4.8935985	60
## 818	Female	Matched	1.9574394	10
## 890	Female	Matched	8.8084773	50
## 902	Female	Matched	28.3828712	60
## 921	Female	Matched	4.8935985	40
## 942	Female	Matched	0.9787197	30
## 994	Female	Matched	4.8935985	80
## 1026	Female	Matched	3.9148788	60
## 1037	Female	Matched	3.9148788	30
## 1055	Female	Matched	3.9148788	30
## 1096	Female	Matched	5.8723182	10
## 1142	Female	Matched	4.8935985	60
## 1145	Female	Matched	0.8646981	50
## 1178	Female	Matched	2.9361591	50
## 1319	Female	Matched	2.5940943	40
## 1344	Female	Matched	0.8646981	40
## 1356	Female	Matched	4.3234905	30
## 1387	Female	Matched	3.4587924	60

6.4. Reordenar un conjunto de datos

Funcion order

```
tmb_ordenado <- tmb[order(tmb$Mutation.Count), ] # Orden ascendente

tmb_ordenado <- tmb[order(tmb$Mutation.Count, decreasing=TRUE),] # Descendente
head(tmb_ordenado$Mutation.Count)
```

```
## [1] 213 207 166 163 160 156
```

6.5. Operaciones sencillas con columnas:

```
bca$Conteo <- bca$medida1 + 10
bca$Conteo <- bca$medida1 -10
bca$Conteo <- bca$medida1/100
bca$Conteo <- bca$medida1 * bca$factor_correccion
```

Manejar valores nulos (NA). Tratar valores nulos, Cuando nos falta algún registro o dato:

```
any(is.na(bca))
```

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

```
colSums(is.na(bca)) # Nulos en la columna de concentracion
```

```
##          nombre          medida1          medida2    concentracion
##             0             0             0             0
##      fecha factor_correccion          Conteo
##             0             0             0
```

```
bca_aux <- bca
```

```
bca[is.na(bca)] <- 0 # Asignar un valor
any(is.na(bca))
```

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

```
bca_aux <- na.omit(bca_aux) # Eliminar estos registros
```

6.6. Buscar valores y reemplazarlos

Sustituir valores

```
# gsub
table(tmb$Cancer.Type)
```

```
##
##          Bladder Cancer          Breast Cancer
##             215             44
## Cancer of Unknown Primary    Colorectal Cancer
##             88             110
##      Esophagogastric Cancer          Glioma
```

```
##              126              117
##      Head and Neck Cancer              Melanoma
##              139              320
## Non-Small Cell Lung Cancer      Renal Cell Carcinoma
##              350              151
## Skin Cancer, Non-Melanoma
##              1
```

```
tmb$Cancer.Type <- gsub(pattern = "Cancer",
                        replacement = "Tumor",
                        x = tmb$Cancer.Type)
table(tmb$Cancer.Type)
```

```
##
##      Bladder Tumor      Breast Tumor      Colorectal Tumor
##              215              44              110
##      Esophagogastric Tumor      Glioma      Head and Neck Tumor
##              126              117              139
##              Melanoma Non-Small Cell Lung Tumor      Renal Cell Carcinoma
##              320              350              151
## Skin Tumor, Non-Melanoma      Tumor of Unknown Primary
##              1              88
```

```
tmb$Patient.ID[1:5]
```

```
## [1] "P-0000057" "P-0000062" "P-0000063" "P-0000071" "P-0000082"
```

```
tmb$Patient.ID <- gsub(pattern = "-",
                      replacement = ".",
                      x = tmb$Patient.ID)
tmb$Patient.ID[1:5]
```

```
## [1] "P.0000057" "P.0000062" "P.0000063" "P.0000071" "P.0000082"
```

Buscar valores: Cuando queremos saber la posición de un registro al trabajar con grandes volúmenes de información:

```
# grep: devuelve índices
# grepl: devuelve verdadero/falso
grep(pattern= "Glioma",
      x = tmb$Cancer.Type)
```

```
## [1] 14 56 57 61 78 103 104 105 106 107 117 130 142 147 152
## [16] 163 170 182 189 209 211 215 237 252 281 284 286 295 306 311
## [31] 334 338 384 386 432 435 440 446 463 481 495 511 520 551 554
## [46] 557 561 569 606 607 609 614 624 625 638 639 659 671 675 686
## [61] 692 705 720 741 755 778 788 789 800 801 849 875 903 907 914
## [76] 915 922 939 950 965 1003 1006 1014 1015 1020 1033 1065 1073 1081 1134
## [91] 1164 1177 1182 1183 1187 1195 1197 1199 1208 1210 1212 1218 1227 1231 1237
## [106] 1238 1249 1277 1306 1311 1313 1329 1333 1374 1390 1394 1421
```

```
glioma.df <- tmb[grepl(pattern= "Glioma",x = tmb$Cancer.Type),]
head(glioma.df)
```

```
##          Study.ID Patient.ID          Sample.ID
## 14  tmb_mskcc_2018 P.0000223 P-0000223-T01-IM3
## 56  tmb_mskcc_2018 P.0000749 P-0000749-T01-IM3
## 57  tmb_mskcc_2018 P.0000758 P-0000758-T01-IM3
## 61  tmb_mskcc_2018 P.0000818 P-0000818-T01-IM3
## 78  tmb_mskcc_2018 P.0001052 P-0001052-T01-IM3
## 103 tmb_mskcc_2018 P.0001388 P-0001388-T01-IM3
##      Age.at.Which.Sequencing.was.Reported..Days. Age.Group.at.Diagnosis.in.Years
## 14                                             29                               <30
## 56                                             28                               <30
## 57                                             52                               50-60
## 61                                             33                               31-50
## 78                                             63                               61-70
## 103                                            77                               >71
##      Cancer.Type      Cancer.Type.Detailed  Drug.Type Gene.Panel Institute.Source
## 14      Glioma    Anaplastic Astrocytoma PD-1/PDL-1  IMPACT341          MSKCC
## 56      Glioma    Anaplastic Astrocytoma PD-1/PDL-1  IMPACT468          MSKCC
## 57      Glioma Glioblastoma Multiforme PD-1/PDL-1  IMPACT410          MSKCC
## 61      Glioma Glioblastoma Multiforme PD-1/PDL-1  IMPACT341          MSKCC
## 78      Glioma Glioblastoma Multiforme PD-1/PDL-1  IMPACT410          MSKCC
## 103     Glioma Glioblastoma Multiforme PD-1/PDL-1  IMPACT341          MSKCC
##      Metastatic.Site Mutation.Count Oncotree.Code Overall.Survival..Months.
## 14      <NA>                5          AASTR                2
## 56      <NA>                8          AASTR                13
## 57      <NA>                4           GBM                14
## 61      <NA>                5           GBM                14
## 78      <NA>                5           GBM                 8
## 103     <NA>                6           GBM               16
##      Overall.Survival.Status Primary.Tumor.Site Sample.Class
## 14      1:DECEASED           Brain          Tumor
## 56      0:LIVING            Brain          Tumor
## 57      1:DECEASED           Brain          Tumor
```

```
## 61          1:DECEASED          Brain      Tumor
## 78          1:DECEASED          Brain      Tumor
## 103         1:DECEASED          Brain      Tumor
##      Number.of.Samples.Per.Patient Sample.coverage Sample.Type    Sex
## 14              1              1064    Primary Female
## 56              1              328    Primary   Male
## 57              1              243    Primary   Male
## 61              1              617    Primary   Male
## 78              1              350    Primary Female
## 103             1              541    Primary   Male
##      Somatic.Status TMB..nonsynonymous. Tumor.Purity
## 14      Matched      5.545777      80
## 56      Matched      8.873242      80
## 57      Matched      3.327466      90
## 61      Matched      4.436621      70
## 78      Matched      3.327466      70
## 103     Matched      5.545777      35
```

```
grepl_glioma<-grepl(pattern= "Glioma",
  x = tmb$Cancer.Type)

glioma.df <- tmb[grepl(pattern= "Glioma",x = tmb$Cancer.Type),]
```

6.7. El paquete dplyr

El paquete dplyr es muy útil para trabajar con conjuntos de datos, por lo que lo instalaremos si no lo está. Cheat Sheet: <https://nyu-cdsc.github.io/learningr/assets/data-transformation.pdf>

```
library(dplyr)  #

## Warning: package 'dplyr' was built under R version 4.3.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##      filter, lag

## The following objects are masked from 'package:base':
##
##      intersect, setdiff, setequal, union
```

```
# install.packages(dplyr)
```

Para los siguientes apartados veremos soluciones en R base y con funciones del paquete dplyr. Usar una u otra opción dependerá únicamente de vuestra preferencia.

```
tmb %>%  
  group_by(Sex) %>%  
  count() # Similar a la función table
```

```
## # A tibble: 2 x 2  
## # Groups:   Sex [2]  
##   Sex      n  
##   <chr> <int>  
## 1 Female  627  
## 2 Male   1034
```

```
# Aparece un NA  
tmb %>%  
  group_by(Cancer.Type) %>%  
  summarise(Promedio_tiempo = mean(Age.at.Which.Sequencing.was.Reported..Days.))
```

```
## # A tibble: 11 x 2  
##   Cancer.Type      Promedio_tiempo  
##   <chr>          <dbl>  
## 1 Bladder Tumor    67.4  
## 2 Breast Tumor     51  
## 3 Colorectal Tumor 54.2  
## 4 Esophagogastric Tumor 58.8  
## 5 Glioma           51.4  
## 6 Head and Neck Tumor 59.9  
## 7 Melanoma         63.0  
## 8 Non-Small Cell Lung Tumor NA  
## 9 Renal Cell Carcinoma 59.7  
## 10 Skin Tumor, Non-Melanoma 78  
## 11 Tumor of Unknown Primary 60.2
```

```
tmb %>%  
  group_by(Cancer.Type) %>%  
  summarise(Promedio_tiempo = mean(Age.at.Which.Sequencing.was.Reported..Days.,  
                                   na.rm = TRUE))
```



```
## # A tibble: 11 x 2
##   Cancer.Type      Promedio_tiempo
##   <chr>           <dbl>
## 1 Bladder Tumor      67.4
## 2 Breast Tumor       51
## 3 Colorectal Tumor   54.2
## 4 Esophagogastric Tumor 58.8
## 5 Glioma             51.4
## 6 Head and Neck Tumor 59.9
## 7 Melanoma           63.0
## 8 Non-Small Cell Lung Tumor 65.7
## 9 Renal Cell Carcinoma 59.7
## 10 Skin Tumor, Non-Melanoma 78
## 11 Tumor of Unknown Primary 60.2
```

```
tmb %>%
  filter(Cancer.Type == "Melanoma") %>%
  group_by(Sex) %>%
  summarise(Promedio_tiempo = mean(Age.at.Which.Sequencing.was.Reported..Days.,
                                   na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##   Sex      Promedio_tiempo
##   <chr>      <dbl>
## 1 Female      60.7
## 2 Male       64.3
```

Todas estas operaciones y más, recogidas en: <https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf>

7. Breve análisis estadístico

No vamos a entrar mucho en el terreno de la estadística, ya que es complejo y todo lo infinito que queramos. Sin embargo, hay una parte muy importante a la hora de hacer cualquier análisis, que es conocer el comportamiento de las variables (estadística descriptiva) antes de poder analizar la significancia de la relación entre ellas (estadística inferencial). El conjunto de datos con el que vamos a trabajar ahora lo hemos obtenido a través del portal **cBioPortal**. cBioPortal contiene datos de genómica en cáncer y su creación y mantenimiento es del Memorial Sloan Kettering Cancer Center. El conjunto de datos está disponible como “TMB and immunotherapy (MSK,Nat Genet 2019)” (https://www.cbioportal.org/study/summary?id=tmb_mskcc_2018). Podéis descargarlo desde el repositorio de github o en cBioPortal en la pestaña “Clinical Data” seleccionando todas las variables y descargando los datos en formato “.tsv”.

Este apartado servirá como ejemplo de cómo proceder a analizar un conjunto de datos. Exploramos la estructura del conjunto de datos con la función `str()`.

```
tmb<-read.delim("tmb_mskcc_2018_clinical_data.tsv")
str(tmb)
```

```
## 'data.frame':    1661 obs. of  24 variables:
## $ Study.ID          : chr  "tmb_mskcc_2018" "tmb_mskcc_2018"
## $ Patient.ID        : chr  "P-0000057" "P-0000062" "P-0000062"
## $ Sample.ID         : chr  "P-0000057-T01-IM3" "P-0000062-T01-IM3"
## $ Age.at.Which.Sequencing.was.Reported..Days. : int  41 80 62 66 61 63 47 44 67 60 ...
## $ Age.Group.at.Diagnosis.in.Years             : chr  "31-50" ">71" "61-70" "61-70" ...
## $ Cancer.Type                                   : chr  "Breast Cancer" "Esophagogastric Cancer"
## $ Cancer.Type.Detailed                         : chr  "Breast Mixed Ductal and Lobular"
## $ Drug.Type                                    : chr  "PD-1/PDL-1" "PD-1/PDL-1" "PD-1/PDL-1"
## $ Gene.Panel                                   : chr  "IMPACT341" "IMPACT341" "IMPACT341"
## $ Institute.Source                             : chr  "MSKCC" "MSKCC" "MSKCC" "MSKCC"
## $ Metastatic.Site                             : chr  NA NA NA NA ...
## $ Mutation.Count                              : int  5 6 13 10 12 12 6 3 5 8 ...
## $ Oncotree.Code                               : chr  "MDLC" "GEJ" "BLCA" "BLCA" ...
## $ Overall.Survival..Months.                   : int  0 1 42 43 57 12 18 4 1 8 ...
## $ Overall.Survival.Status                     : chr  "1:DECEASED" "1:DECEASED" "0:LIVING"
## $ Primary.Tumor.Site                         : chr  "Breast" "Esophagus" "Bladder"
## $ Sample.Class                               : chr  "Tumor" "Tumor" "Tumor" "Tumor"
## $ Number.of.Samples.Per.Patient              : int  1 1 1 1 1 1 1 1 1 1 ...
## $ Sample.coverage                            : int  835 1176 900 795 905 783 997 506
## $ Sample.Type                                : chr  "Primary" "Primary" "Primary" "Primary"
## $ Sex                                         : chr  "Female" "Male" "Male" "Male"
## $ Somatic.Status                             : chr  "Matched" "Matched" "Matched" "Matched"
## $ TMB..nonsynonymous.                       : num  5.55 6.65 15.53 9.98 13.31 ...
## $ Tumor.Purity                               : chr  "25" "30" "70" "30" ...
```

Vemos que tenemos 1661 observaciones para 24 variables, de las cuales unas son de tipo carácter y otras son de tipo numérico. Me llama la atención que hay una variable que identifica al paciente y otra la muestra del paciente, por lo que me pregunto, ¿hay algún paciente con más de una muestra? Esto lo puedo comprobar viendo si hay algún identificador del paciente repetido, con la función `duplicated()`.

```
id.duplicados<-duplicated(tmb$Patient.ID)
sum(id.duplicados)
```

```
## [1] 0
```

Una buena práctica es comprobar si hay algún valor faltante. En R estos valores se denominan *NA*. Para ello podemos usar la función `apply()`, indicando que sobre el objeto `df`, por filas, queremos saber si hay algún *NA*. Es decir, queremos saber si hay pacientes para los que no tenemos datos de alguna de las variables. Vamos a quedarnos únicamente con aquellas observaciones que están completas. A este nuevo conjunto de datos le vamos a llamar `tmb.complete`.

```
na.count<-rowSums(is.na(tmb))
tmb.complete<-tmb[na.count==0,]
```

7.1. Estadística descriptiva

En primer lugar, debemos caracterizar cada variable, describir qué valores tiene en nuestro estudio. La descripción de cada variable dependerá de si es de tipo numérico o categórico.

1. Variables de tipo numérico. Las variables de tipo numérico se describen con medidas de centralidad, como son la media y la mediana. Otros valores que podemos obtener son los valores mínimo y máximo. La función `summary()` nos permite obtener esta información para todas las variables, excepto para las de tipo carácter.

```
summary(tmb.complete)
```

```
##      Study.ID          Patient.ID          Sample.ID
## Length:850          Length:850          Length:850
## Class :character    Class :character    Class :character
## Mode  :character    Mode  :character    Mode  :character
##
##
##
## Age.at.Which.Sequencing.was.Reported..Days. Age.Group.at.Diagnosis.in.Years
## Min.      :20.00                                Length:850
## 1st Qu.:54.00                                Class :character
## Median :63.00                                Mode  :character
## Mean    :61.76
## 3rd Qu.:71.00
## Max.    :90.00
## Cancer.Type      Cancer.Type.Detailed Drug.Type          Gene.Panel
## Length:850      Length:850          Length:850          Length:850
## Class :character Class :character    Class :character    Class :character
## Mode  :character Mode  :character    Mode  :character    Mode  :character
##
##
##
```

```

## Institute.Source      Metastatic.Site      Mutation.Count      Oncotree.Code
## Length:850           Length:850           Min.      : 1.00      Length:850
## Class :character      Class :character      1st Qu.: 4.00      Class :character
## Mode  :character      Mode  :character      Median : 7.00      Mode  :character
##                                     Mean  : 13.13
##                                     3rd Qu.: 14.00
##                                     Max.  :163.00
## Overall.Survival..Months. Overall.Survival.Status Primary.Tumor.Site
## Min.      : 0.00           Length:850           Length:850
## 1st Qu.: 4.00           Class :character      Class :character
## Median :11.00           Mode  :character      Mode  :character
## Mean  :15.03
## 3rd Qu.:21.00
## Max.  :80.00
## Sample.Class           Number.of.Samples.Per.Patient Sample.coverage
## Length:850            Min.      :1           Min.      : 95.0
## Class :character      1st Qu.:1           1st Qu.: 591.5
## Mode  :character      Median :1           Median : 769.5
##                                     Mean  :1           Mean  : 756.0
##                                     3rd Qu.:1           3rd Qu.: 912.0
##                                     Max.  :1           Max.  :1744.0
## Sample.Type           Sex           Somatic.Status      TMB..nonsynonymous.
## Length:850           Length:850           Length:850           Min.      : 0.000
## Class :character      Class :character      Class :character      1st Qu.: 3.915
## Mode  :character      Mode  :character      Mode  :character      Median : 6.851
##                                     Mean  : 12.476
##                                     3rd Qu.: 13.310
##                                     Max.  :179.683
## Tumor.Purity
## Length:850
## Class :character
## Mode  :character
##
##
##

```

También podemos obtener estos valores para alguna variable de interés. Por ejemplo, vamos a explorar el número de mutaciones.

```
mean(tmb$Mutation.Count)
```

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

```
mean(tmb$Mutation.Count, na.rm = TRUE)
```

```
## [1] 12.61242
```

```
mean(tmb.complete$Mutation.Count)
```

```
## [1] 13.12588
```

También podemos caracterizar los valores mínimo y máximo.

```
min(tmb.complete$Mutation.Count)
```

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

```
max(tmb.complete$Mutation.Count)
```

```
## [1] 163
```

2. Variables de tipo categórico. Las variables de tipo categórico se definen por la frecuencia de sus valores. Podemos explorar la variable “Cancer.Type” con la función `table()`. Vemos que solo hay un paciente con cáncer de piel, por lo que probablemente eliminaremos este paciente en las pruebas estadísticas.

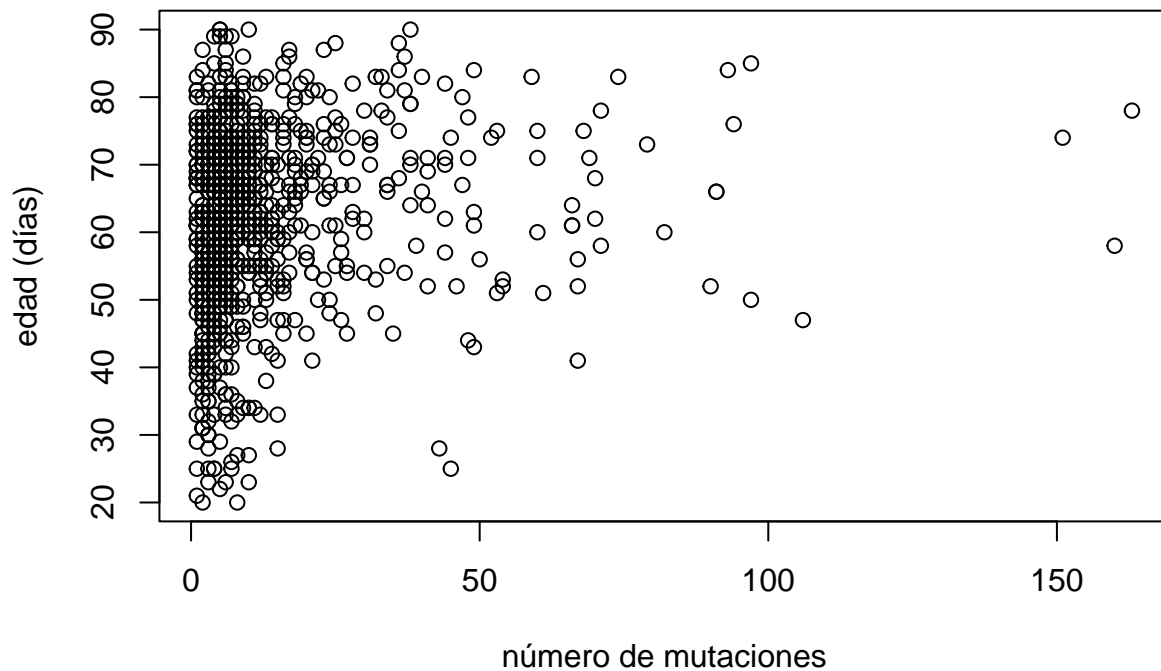
```
table(tmb.complete$Cancer.Type)
```

```
##
##          Bladder Cancer          Breast Cancer
##              87              30
## Cancer of Unknown Primary    Colorectal Cancer
##              64              55
##      Esophagogastric Cancer    Head and Neck Cancer
##              43              87
##              Melanoma Non-Small Cell Lung Cancer
##              248              170
##      Renal Cell Carcinoma    Skin Cancer, Non-Melanoma
##              65              1
```

3. Gráficos Representar la información de forma gráfica es un recurso muy útil para visualizar de forma inmediata su comportamiento

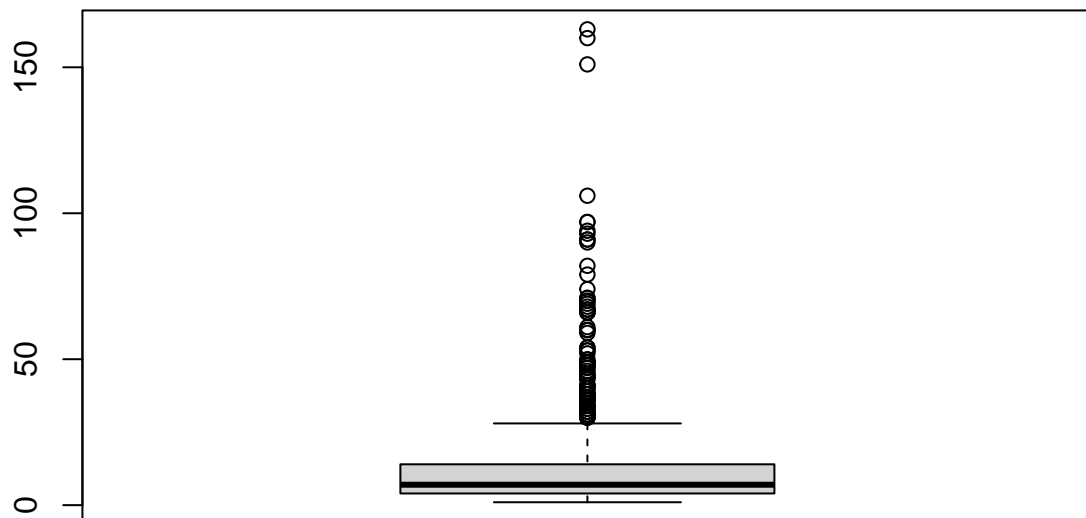
- Gráfico de dispersión. Es útil para explorar si existe alguna relación entre dos variables. Nos podemos preguntar si hay alguna relación entre el número de mutaciones y la edad del paciente. Fijaos que podemos cambiar el título de los ejes para que nos quede más presentable.

```
plot(tmb.complete$Mutation.Count,
     tmb.complete$Age.at.Which.Sequencing.was.Reported..Days.,
     xlab="número de mutaciones",
     ylab = "edad (días)")
```



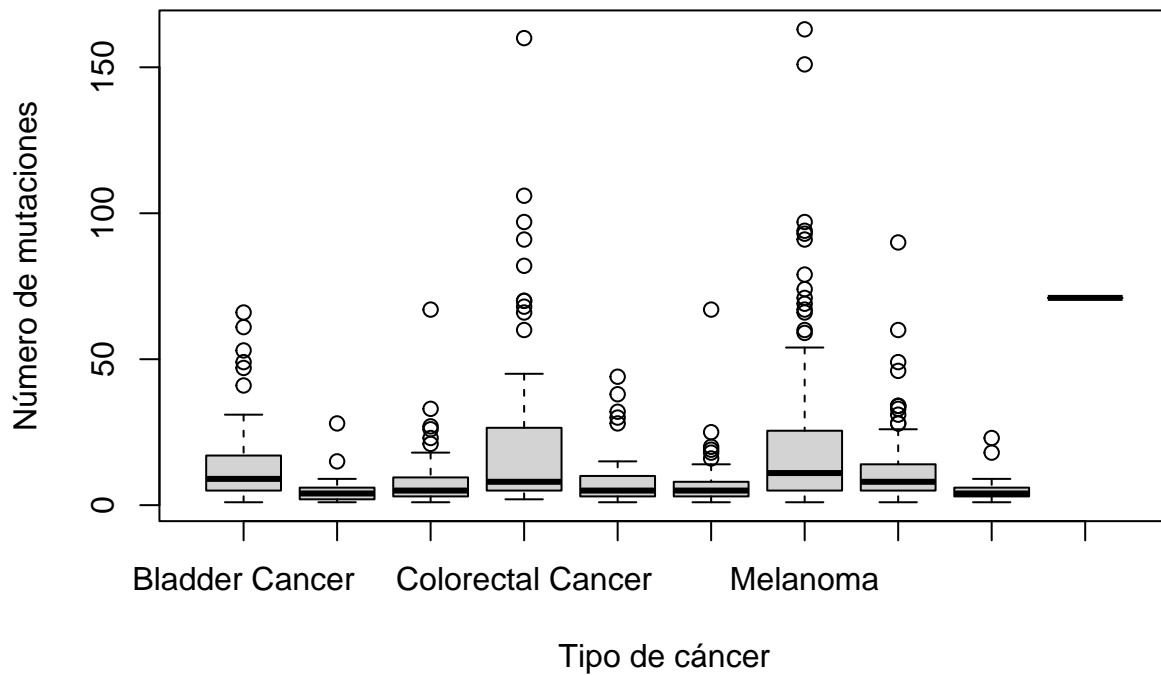
- Gráfico de cajas y bigotes Los gráficos de cajas y bigotes nos sirven para ver la dispersión y simetría de nuestros datos. Venos que la variable Mutation.Count tiene valores extremos.

```
boxplot(tmb.complete$Mutation.Count)
```



También lo podemos representar respecto a otra variable; por ejemplo, el número de mutaciones en los diferentes tipos de cáncer.

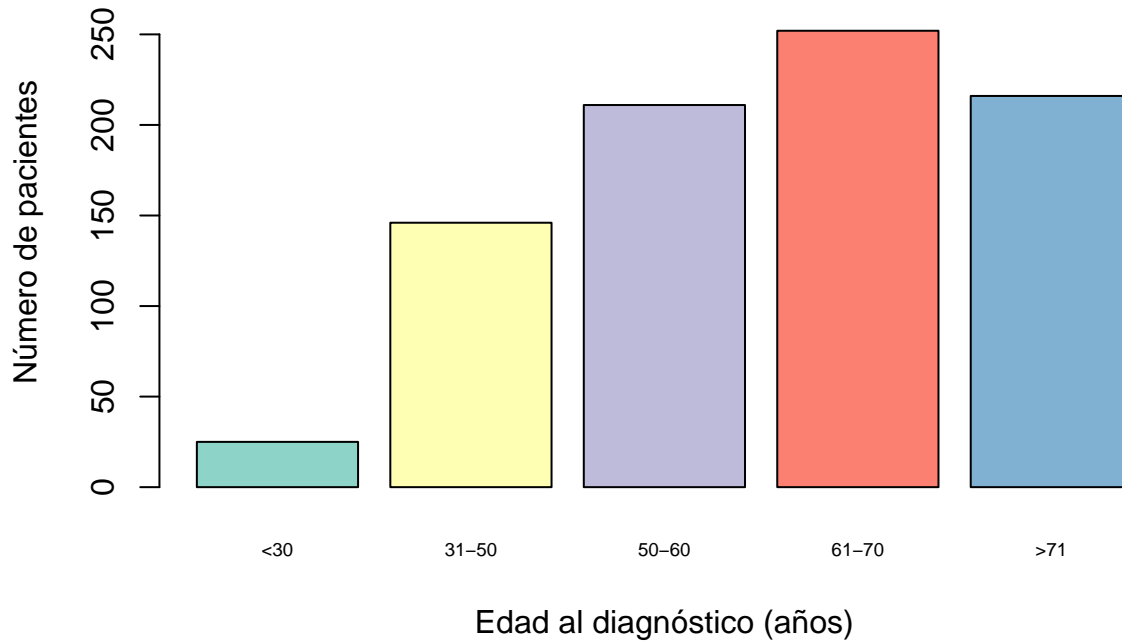
```
boxplot(tmb.complete$Mutation.Count ~tmb.complete$Cancer.Type,  
        xlab="Tipo de cáncer",  
        ylab="Número de mutaciones")
```



- Gráfico de barras. Nos sirve para representar la frecuencia de los valores de una variable. Podemos representar el grupo de edad de los pacientes. Vemos que el grupo más frecuente es el de 61-70 años.

```
library(RColorBrewer)
```

```
age.freq<-table(tmb.complete$Age.Group.at.Diagnosis.in.Years)[c(1,3,4,5,2)]
age.bar<-barplot(age.freq,
  col=brewer.pal(5, "Set3"),
  xlab="Edad al diagnóstico (años)",
  ylab="Número de pacientes",
  ylim = c(0,max(age.freq)+30),
  cex.names=0.6
)
```

7.2. Estadística inferencial

En un análisis de datos biológicos normalmente no queremos quedarnos en describir las variables que hemos estudiado, sino responder a una pregunta biológica, es decir, tenemos una hipótesis. Para contrastar nuestra hipótesis y ver si se cumple o no en base a los datos que hemos colectado, recurrimos a las pruebas estadísticas. Hay dos premisas que se tienen en cuenta para las variables numéricas, que son la normalidad y la igualdad de varianzas entre las variables a comparar.

1. Test de normalidad Una cuestión que tenemos que tener en cuenta para aplicar casi cualquier prueba estadística es saber si nuestros datos tienen una distribución normal, ya que las condiciones en que son válidas ciertas pruebas solo se aplican si los datos son normales. Para ello, podemos emplear pruebas como el test de Shapiro Wilks y pruebas gráficas.

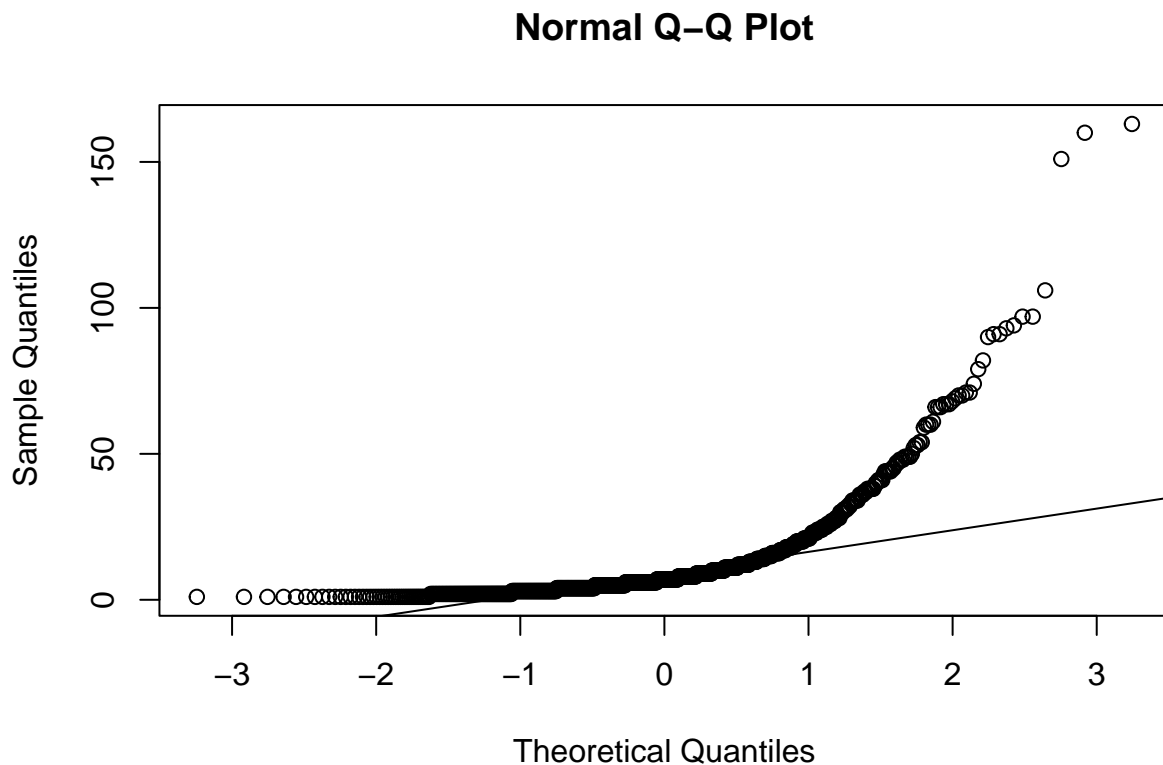
En el caso del test Shapiro Wilks, la hipótesis nula es que no hay diferencia entre la distribución de nuestros datos y una distribución normal. Por lo tanto, si el p valor está por encima del valor de significancia que hayamos establecido (que normalmente es del 5%), aceptaremos la hipótesis nula. En caso contrario, la rechazaremos y diremos que nuestros datos no siguen una distribución normal.

```
shapiro.test(tmb.complete$Mutation.Count)
```

```
##  
##  Shapiro-Wilk normality test  
##  
## data:  tmb.complete$Mutation.Count  
## W = 0.60519, p-value < 2.2e-16
```

También podemos recurrir a los gráficos de cuantiles, que son muy informativos. En él se representan los datos frente a los de una distribución normal, con una línea que los correlaciona. Solo rechazaremos la normalidad si los puntos se alejan mucho de la línea que relaciona los cuantiles con los cuantiles teóricos.

```
qqnorm(tmb.complete$Mutation.Count)  
qqline(tmb.complete$Mutation.Count)
```



En este caso, ambas pruebas nos indican que la variable `Mutation.Count` no sigue una distribución normal.

2. Test de igualdad de varianzas Otra de las premisas que se tienen que cumplir para aplicar ciertas pruebas es que las varianzas de los grupos sean iguales. Para ello podemos emplear el test de Levene usando la función `leveneTests()` del paquete `heplots`.

Por ejemplo, imaginad que quiero comparar si la varianza del número de mutaciones es igual entre hombres y mujeres. El test nos indica que las varianzas no son iguales.

```
library(heplots)
leveneTests(tmb.complete[, "Mutation.Count", drop=FALSE],
            tmb.complete$Sex)
```

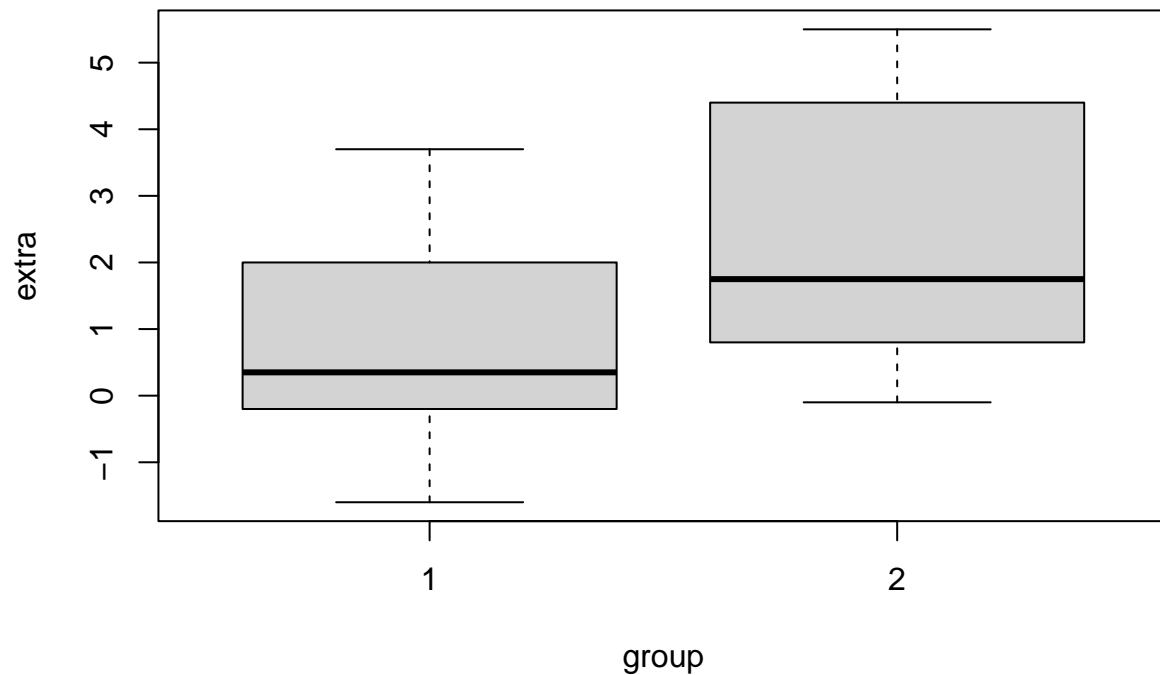
```
## Warning in leveneTest.default(x, group = group, center = center, ...): group
## coerced to factor.
```

```
## Levene's Tests for Homogeneity of Variance (center = median)
##
##              df1 df2 F value Pr(>F)
## Mutation.Count    1 848  5.2262 0.0225 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3. Comparación de medias para distribuciones normales (pruebas paramétricas)

- Comparación de dos grupos: T test. Como en nuestro conjunto de datos no tenemos variables que sigan una distribución normal, vamos a usar otros datos de ejemplo.

```
plot(extra ~ group, data = sleep)
```



```
leveneTests(sleep[,1, drop=FALSE], sleep$group)
```

```
## Levene's Tests for Homogeneity of Variance (center = median)
##
##      df1 df2 F value Pr(>F)
## extra   1  18  0.2482 0.6244
```

```
t.test(extra ~ group, data = sleep, var.equal=TRUE)
```

```
##
## Two Sample t-test
##
## data: extra by group
## t = -1.8608, df = 18, p-value = 0.07919
## alternative hypothesis: true difference in means between group 1 and group 2 is not e
## 95 percent confidence interval:
## -3.363874 0.203874
## sample estimates:
## mean in group 1 mean in group 2
##           0.75           2.33
```

- Comparación de más de dos grupos: ANOVA. Nos podemos preguntar si hay diferencias en el número de mutaciones entre los diferentes tipos de cáncer. El test de ANOVA nos dice que sí hay diferencias entre los distintos tipos de cáncer. En un sentido estricto, este test estaría mal aplicado porque la distribución de la variable no es normal, aunque hay autores que dicen que con más de 30 observaciones se puede aplicar un test paramétrico.

```
anova(lm(Mutation.Count ~ Cancer.Type, data = tmb.complete))
```

```
## Analysis of Variance Table
##
## Response: Mutation.Count
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Cancer.Type   9  31094   3454.9   11.883 < 2.2e-16 ***
## Residuals  840  244219    290.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. Comparación de medias para distribuciones no normales (pruebas no paramétricas)

- Comparación de dos grupos: test de Wilcoxon. Nos podemos preguntar si el número de mutaciones es diferente entre hombres y mujeres.

```
wilcox.test(tmb.complete$Mutation.Count[tmb.complete$Sex=="Female"],
            tmb.complete$Mutation.Count[tmb.complete$Sex=="Male"])
```

```
##
## Wilcoxon rank sum test with continuity correction
##
## data:  tmb.complete$Mutation.Count[tmb.complete$Sex == "Female"] and tmb.complete$Mut
## W = 84643, p-value = 0.7397
## alternative hypothesis: true location shift is not equal to 0
```

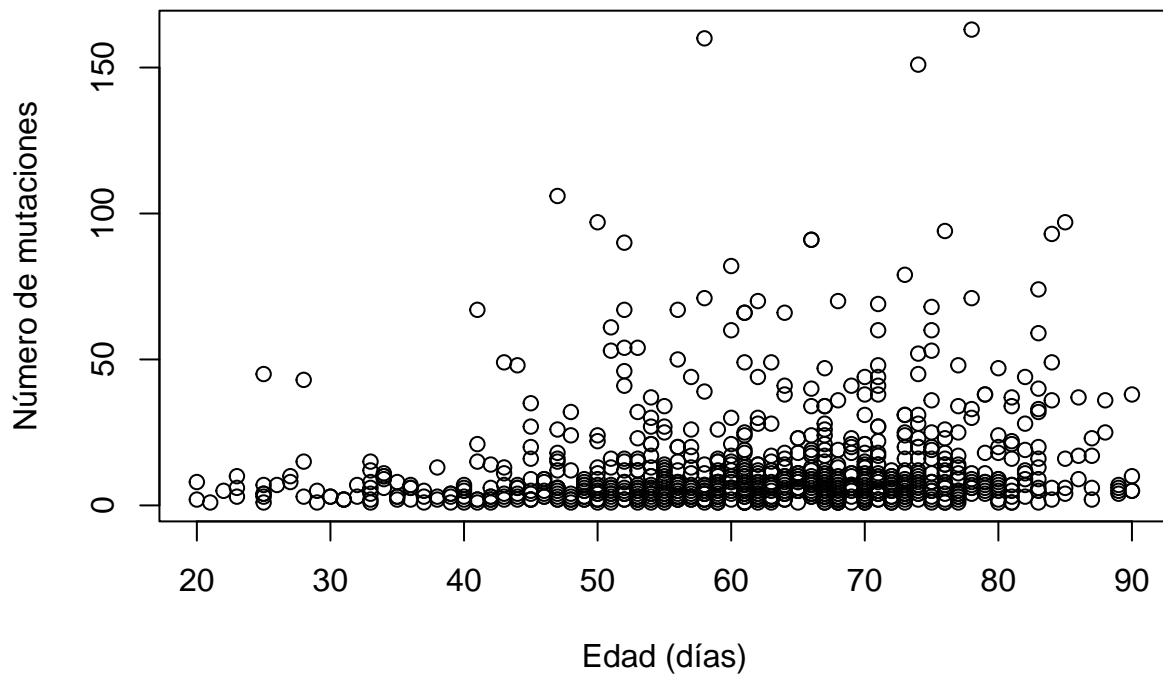
- Comparación de más de dos grupos: test de Kruskal Wallis. Vemos que el resultado del test no paramétrico nos lleva a la misma conclusión que el paramétrico: hay diferencias en el número de mutaciones entre los distintos tipos de cáncer.

```
kruskal.test(Mutation.Count ~ Cancer.Type, data = tmb.complete)
```

```
##
## Kruskal-Wallis rank sum test
##
## data:  Mutation.Count by Cancer.Type
## Kruskal-Wallis chi-squared = 102.16, df = 9, p-value < 2.2e-16
```

5. Correlación entre variables. Podemos emplear la función `plots()` para ver la relación entre variables y la función `cor()` para obtener el coeficiente de correlación. Vemos que el número de mutaciones y la edad no parecen tener correlación, y esto lo confirmamos con que el coeficiente de correlación es muy bajo.

```
plot(tmb.complete$Age.at.Which.Sequencing.was.Reported..Days.,  
     tmb.complete$Mutation.Count,  
     xlab="Edad (días)",  
     ylab = "Número de mutaciones")
```



```
cor(tmb.complete$Age.at.Which.Sequencing.was.Reported..Days.,  
    tmb.complete$Mutation.Count)
```

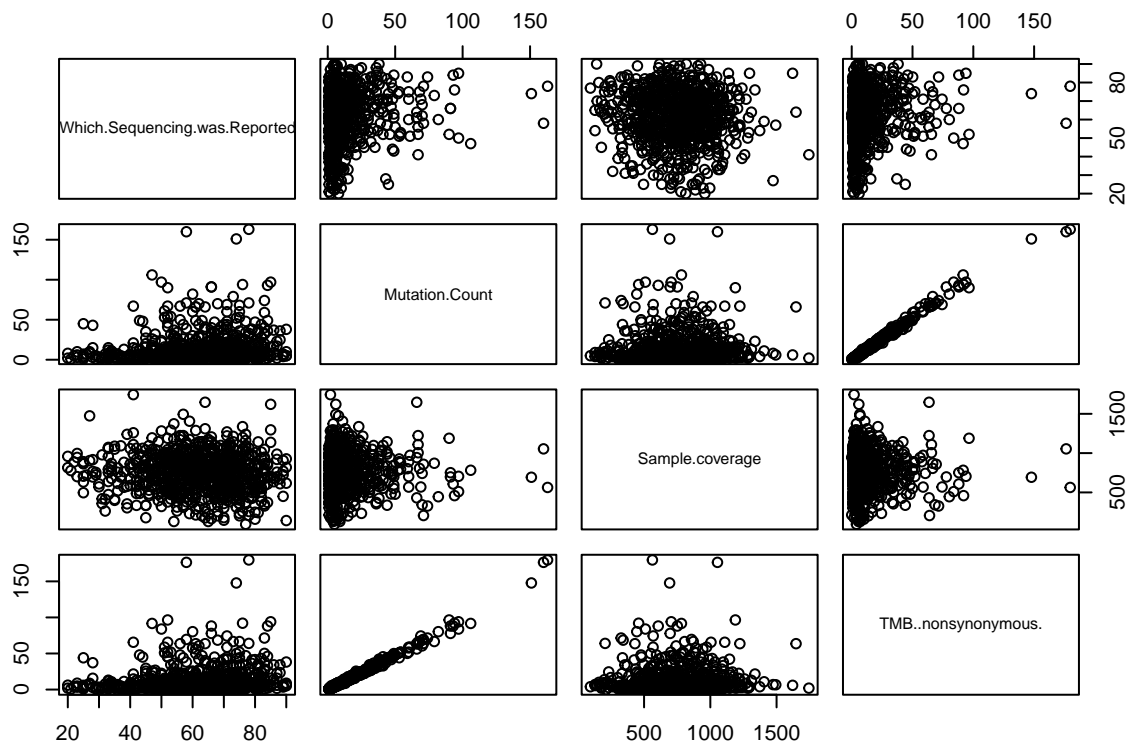
```
## [1] 0.1465951
```

La función `pairs()` nos permite obtener gráficos de dispersión para todas las variables dos a dos.

```

pairs(tmb.complete[,c("Age.at.Which.Sequencing.was.Reported..Days.",
                      "Mutation.Count",
                      "Sample.coverage",
                      "TMB..nonsynonymous."))]
)

```



6. Test para variables categóricas Nos podemos preguntar si el cáncer de vejiga (bladder) es más frecuente en hombres o en mujeres. Para ello emplearemos el test de chi cuadrado con la función `chisq.test()`, que nos devuelve que sí es significativa la diferencia en el número de casos de cáncer de vejiga entre hombres y mujeres.

```

table(tmb.complete$Cancer.Type,tmb.complete$Sex)

```

```

##
##
##      Female Male
## Bladder Cancer      19   68
## Breast Cancer       30    0
## Cancer of Unknown Primary    32   32
## Colorectal Cancer      19   36
## Esophagogastric Cancer     14   29

```

```
##   Head and Neck Cancer           16   71
##   Melanoma                      87  161
##   Non-Small Cell Lung Cancer    96   74
##   Renal Cell Carcinoma          17   48
##   Skin Cancer, Non-Melanoma      0    1
```

```
chisq.test(table(tmb.complete$Sex[tmb.complete$Cancer.Type=="Bladder Cancer"]))
```

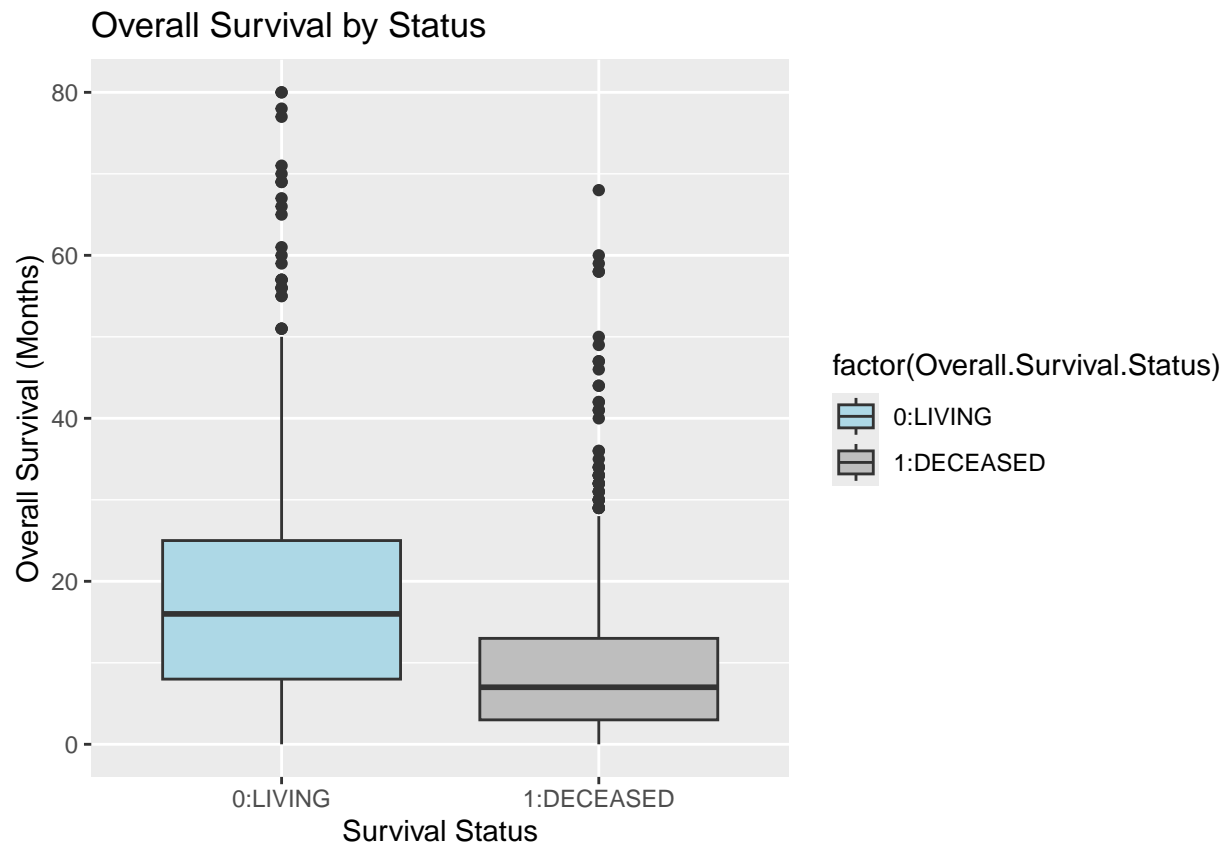
```
##
## Chi-squared test for given probabilities
##
## data:  table(tmb.complete$Sex[tmb.complete$Cancer.Type == "Bladder Cancer"])
## X-squared = 27.598, df = 1, p-value = 1.494e-07
```

7.3. Edición de figuras:

Opciones de personalización

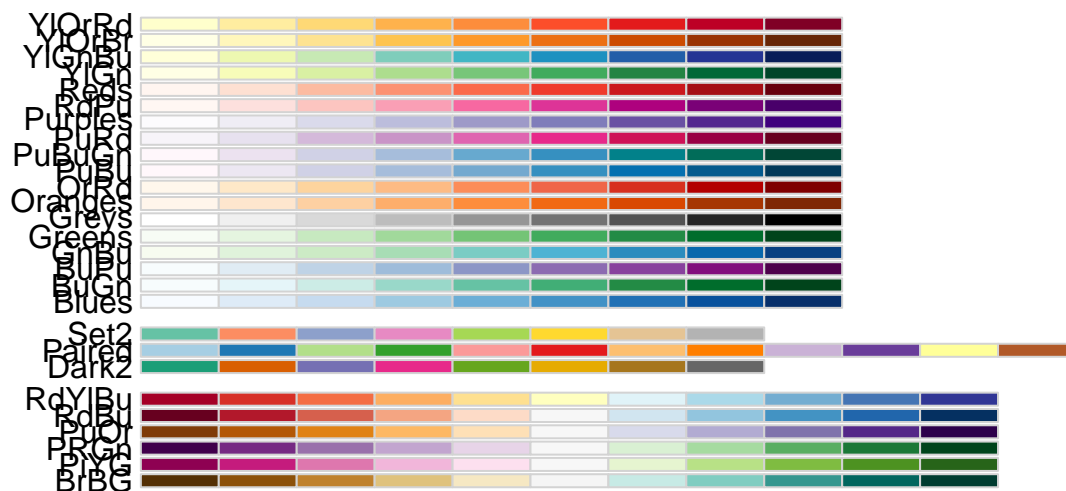
- Añadir colores manualmente: <https://r-charts.com/es/colores/>

```
ggplot(data = tmb, aes(x = Overall.Survival.Status, y = Overall.Survival..Months., fill
  geom_boxplot() +
  scale_fill_manual(values = c("0:LIVING" = "lightblue", "1:DECEASED" = "grey")) +
  labs(title = "Overall Survival by Status",
    x = "Survival Status",
    y = "Overall Survival (Months)"))
```

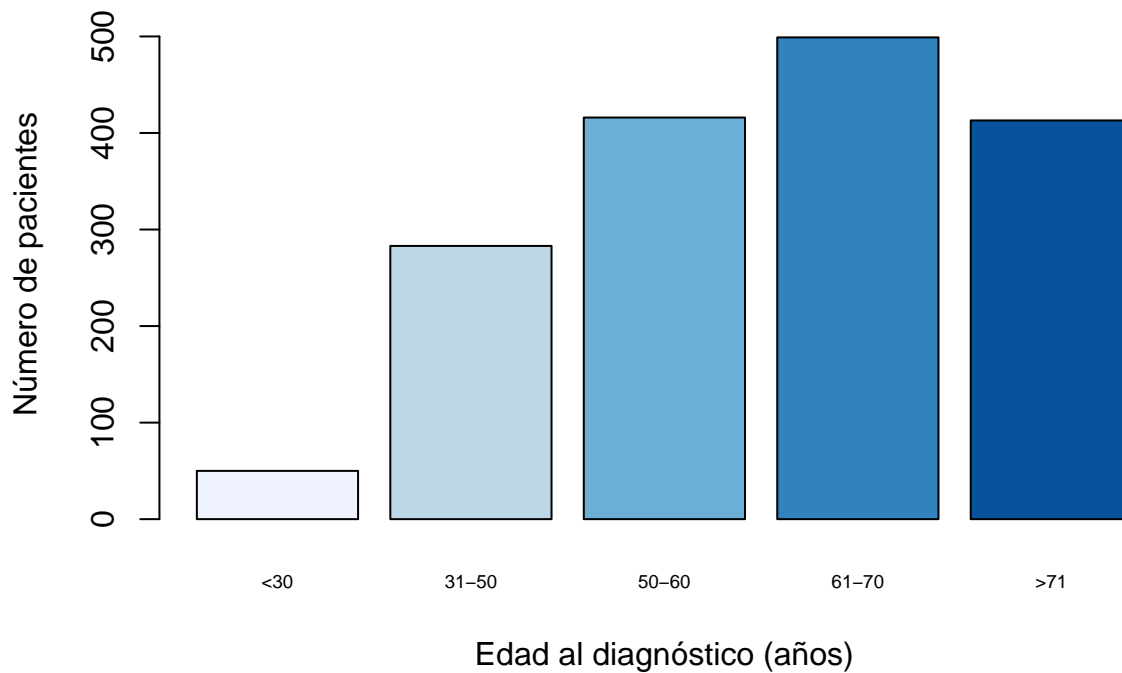



Paletas de colores

```
display.brewer.all(colorblindFriendly = TRUE)
```



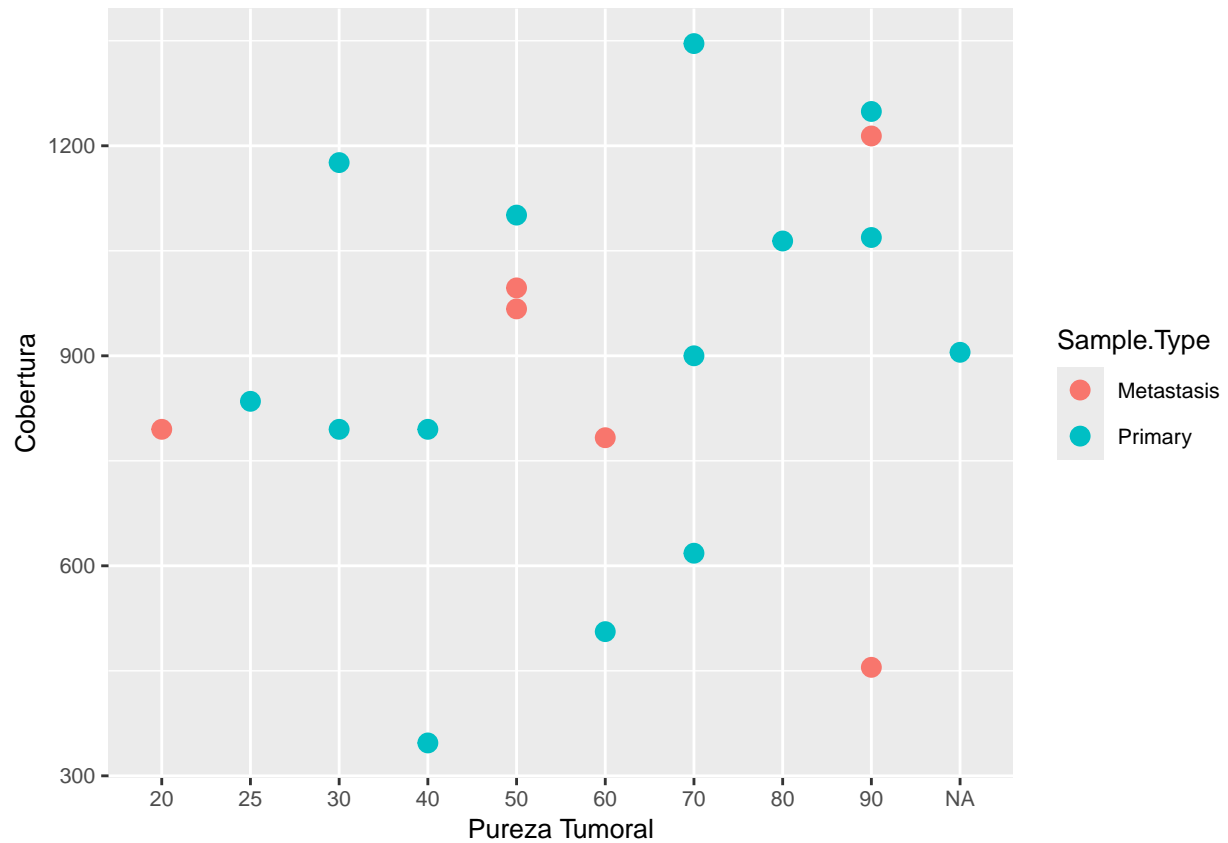
```
age.freq<-table(tmb$Age.Group.at.Diagnosis.in.Years)[c(1,3,4,5,2)]
age.bar<-barplot(age.freq,
                  col=brewer.pal(5, "Blues"),
                  xlab="Edad al diagnóstico (años)",
                  ylab="Número de pacientes",
                  ylim = c(0,max(age.freq)+30),
                  cex.names=0.6
)
```



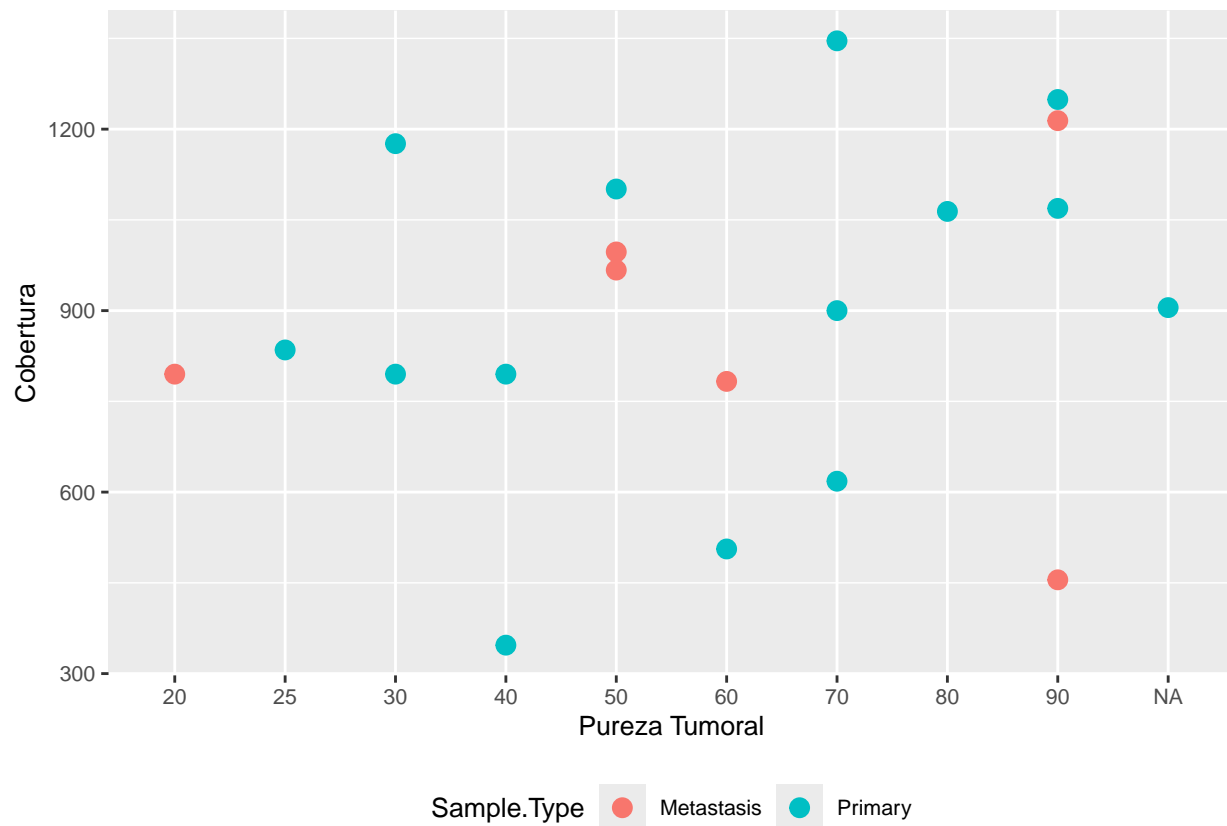
```
# brewer.pal(5, "RdBu"); brewer.pal(5, "Set3"); brewer.pal(5, "Pastel2")
```

Posiciones de la leyenda: `theme(text=element_text(size=10), legend.position = "bottom")`

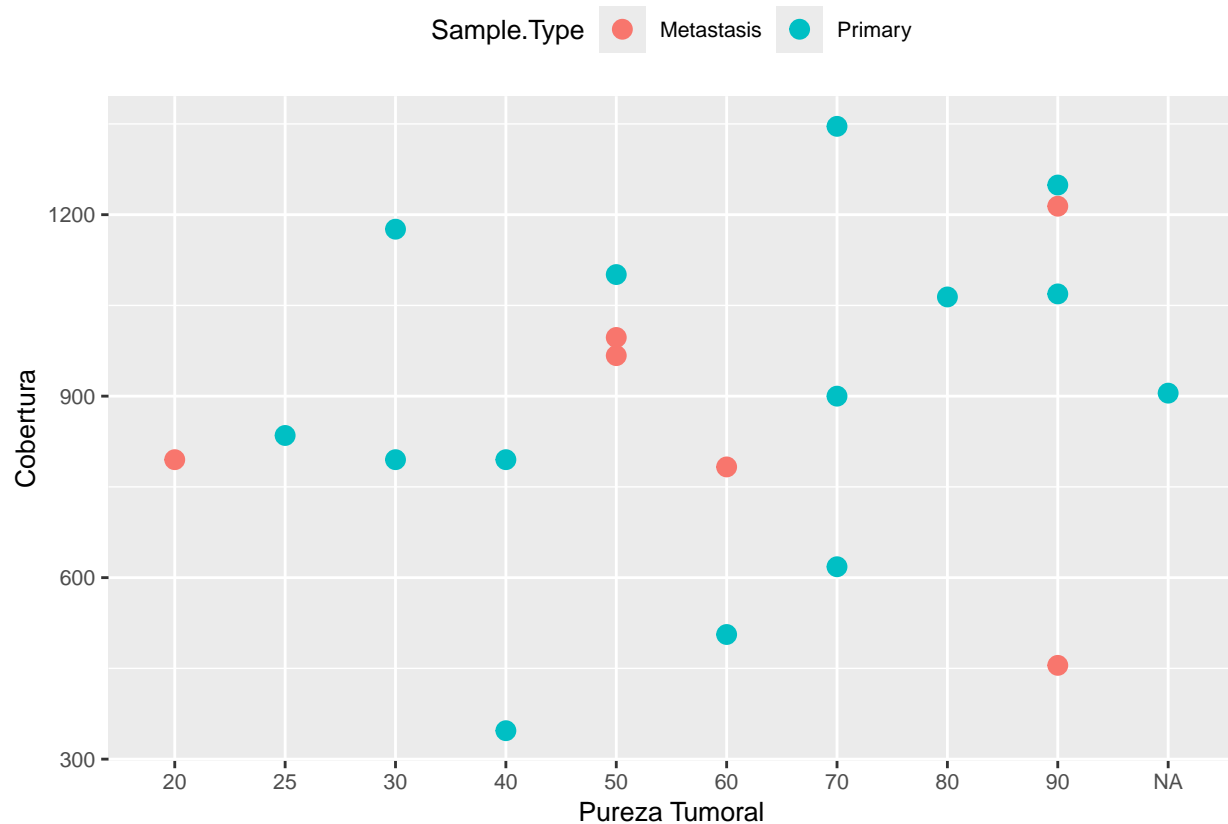
```
# Añadir leyenda con color: por defecto a la derecha
ggplot(data=tmb[1:20,],
       aes(y=Sample.coverage, x=Tumor.Purity, color=Sample.Type)) +
  geom_point(size=3) + # capa de puntos tamaño 3
  theme(text=element_text(size=10))+
  labs(y="Cobertura",x="Pureza Tumoral")
```



```
# Posiciones de la leyenda
ggplot(data=tmb[1:20,],
       aes(y=Sample.coverage, x=Tumor.Purity, color=Sample.Type)) +
  geom_point(size=3) +
  theme(text=element_text(size=10), legend.position = "bottom")+
  labs(y="Cobertura",x="Pureza Tumoral")
```

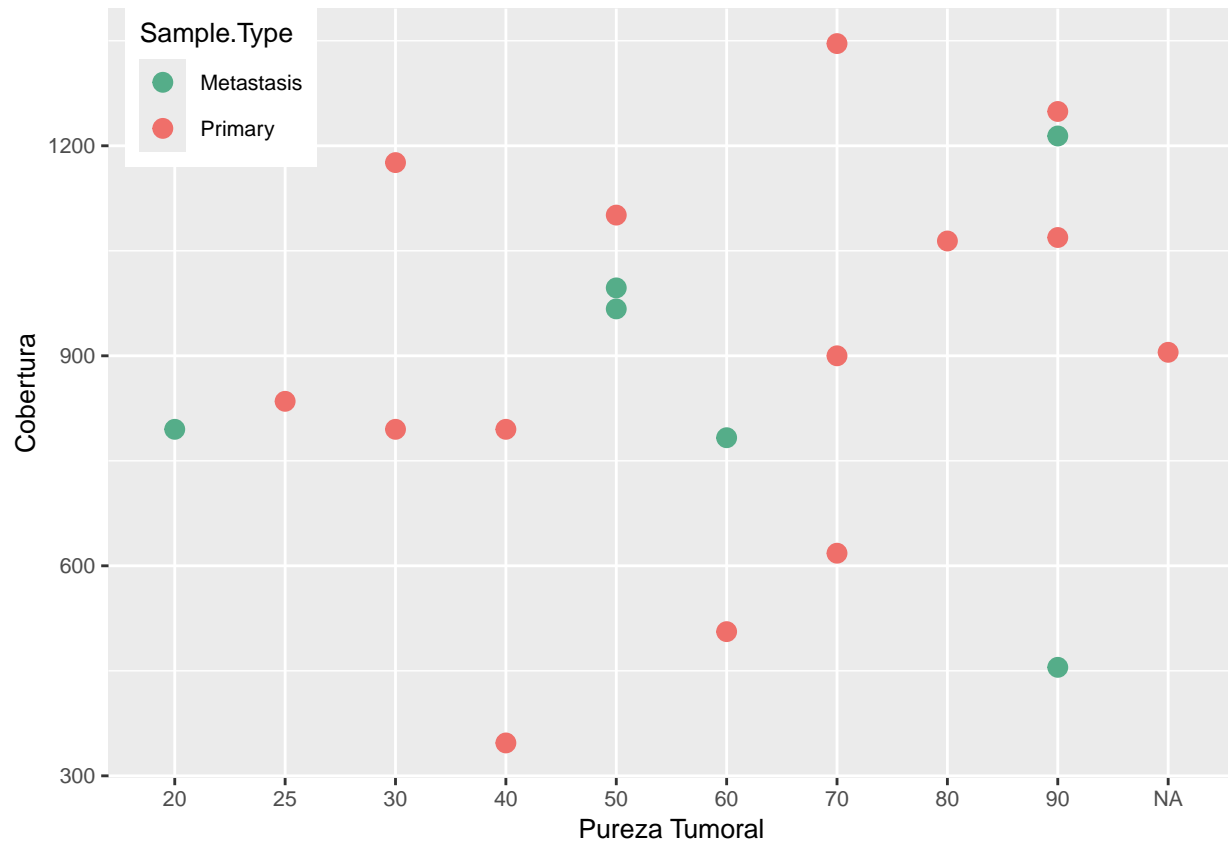


```
ggplot(data=tmb[1:20,],
       aes(y=Sample.coverage, x=Tumor.Purity, color=Sample.Type)) +
  geom_point(size=3) +
  theme(text=element_text(size=10), legend.position = "top")+
  labs(y="Cobertura",x="Pureza Tumoral")
```



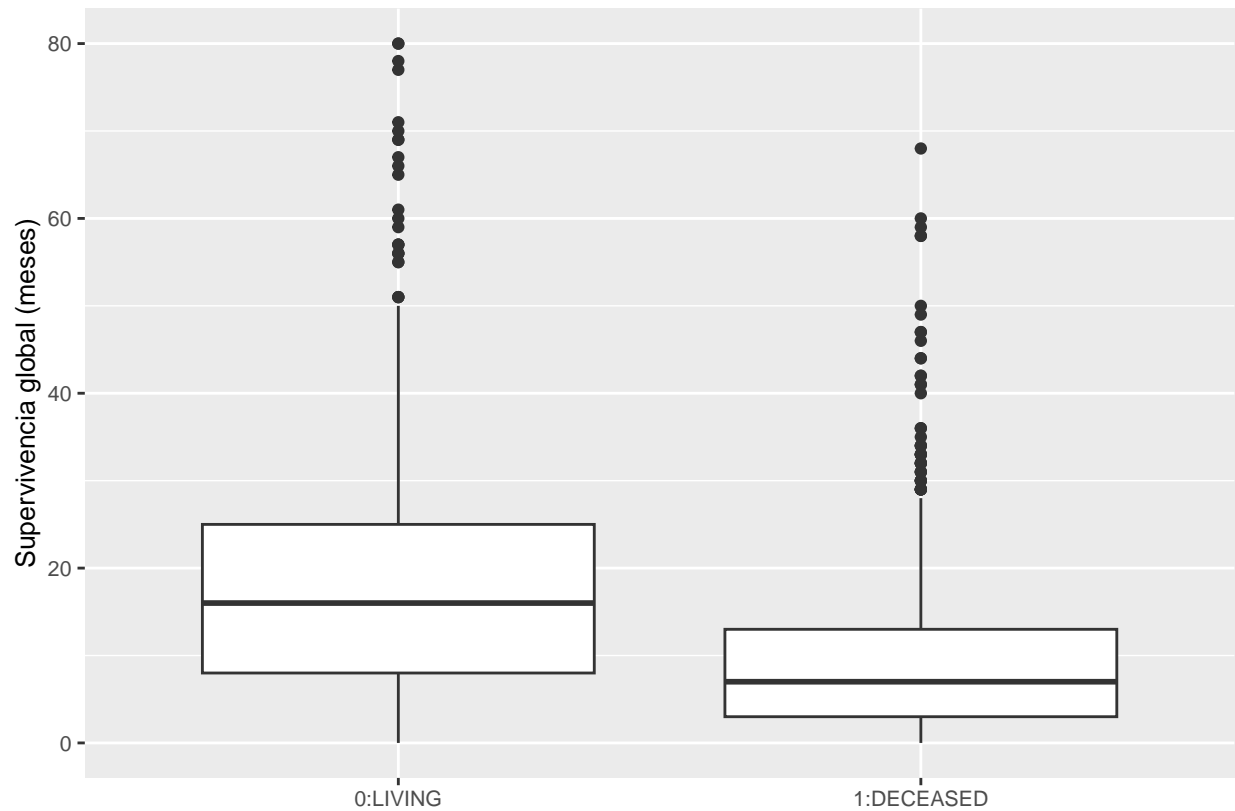
```
cols <- c("#55AD89", "#EF6F6A") ## cambiamos el color
ggplot(data=tmb[1:20,],
  aes(y=Sample.coverage, x=Tumor.Purity, color=Sample.Type)) +
  geom_point(size=3) +
  scale_color_manual(values = cols) +
  theme(text=element_text(size=10), legend.position = c(0.1, 0.9))+ # coordenadas de leyenda
  labs(y="Cobertura",x="Pureza Tumoral")
```

```
## Warning: A numeric 'legend.position' argument in 'theme()' was deprecated in ggplot2
## 3.5.0.
## i Please use the 'legend.position.inside' argument of 'theme()' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

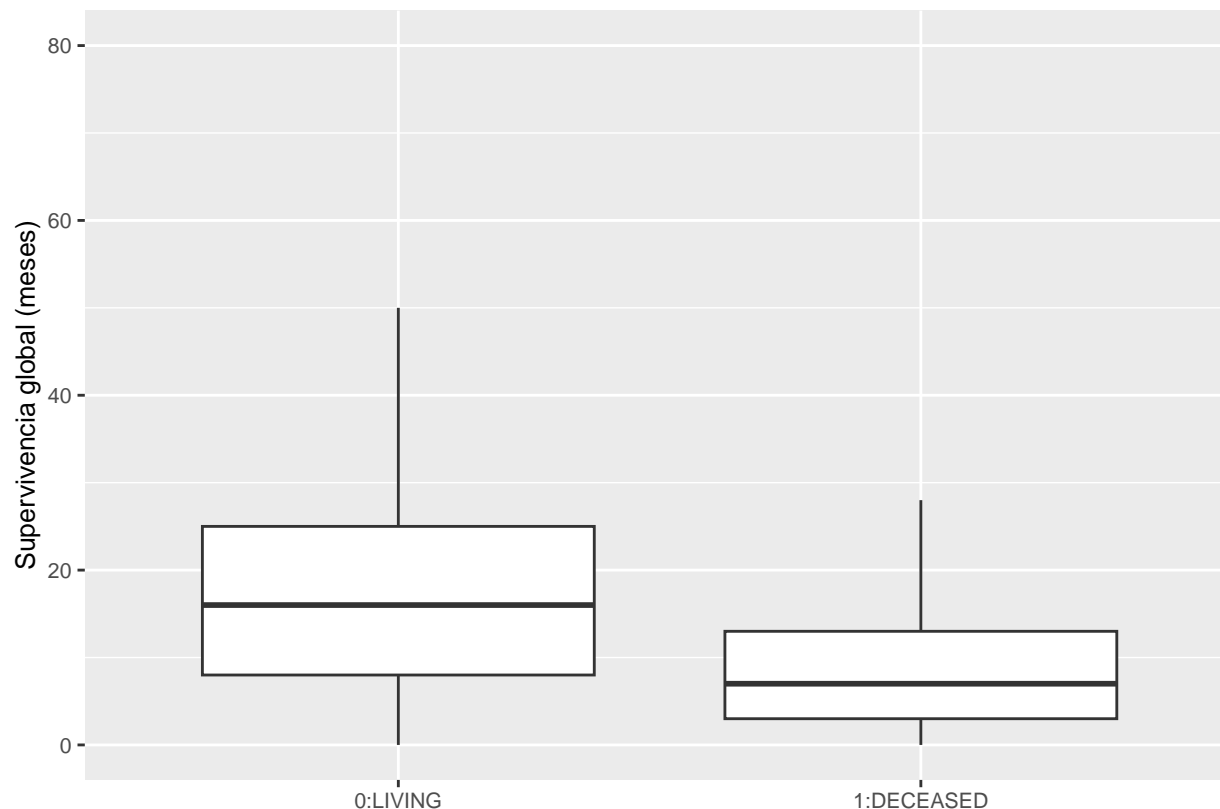


Eliminar outliers:

```
ggplot(data=tmb,
       aes(x=Overall.Survival.Status,y=Overall.Survival..Months.))+
  geom_boxplot() +
  labs(y="Supervivencia global (meses)",x=" ") +
  theme(text=element_text(size=10))
```



```
ggplot(data=tmb,  
       aes(x=Overall.Survival.Status,y=Overall.Survival..Months.))+  
  geom_boxplot(outlier.shape = NA) +  
  labs(y="Supervivencia global (meses)",x=" ") +  
  theme(text=element_text(size=10))
```

Guardar gráficos en calidad: ggsave()

```
# ruta_file_name = ""
# ggsave(ruta_file_name,
#       plot=last_plot(),
#       device = "png")
help(ggsave)
```

8. Guardar la información

Probablemente durante nuestro análisis hayamos modificado nuestro conjunto de datos, generado gráficos, etc. y queramos guardarlos.

Para guardar un conjunto de datos podemos emplear la función `write.table`, indicando el conjunto de datos que queremos guardar y la ruta donde lo queremos escribir, incluyendo el nombre y extensión del fichero.

```
# write.table()
```

Los gráficos generados se pueden guardar con la función `ggsave()`. También podemos pinchar en el desplegable “Export” y guardarlo en el formato que queramos o copiarlo en el portapapeles.

9. Buenas prácticas

- Escribir comentarios en el código. Los comentarios son líneas que van precedidas con la almohadilla “#” y que R no ejecuta. Sirven para dejar notas sobre lo que hace nuestro código o para dejar como borrador líneas de código que aún no queremos eliminar. Lo hacemos con Alt Gr + 3 o con la combinación de teclas Ctrl + MAYUS + C si sobre una selección de líneas.
- No usar espacios al nombrar archivos (mejor Mi_archivo.R que Mi archivo.R). También para las carpetas sería una buena costumbre.
- Usar nombres de variables representativos.
- Guardar el trabajo frecuentemente. Ctrl + S.

10. Análisis de un conjunto de datos

Es vuestro turno para trabajar con los datos. Os dejamos varias opciones. * Podéis seguir trabajando sobre el conjunto de datos “tmb_mskcc_2018_clinical_data.tsv”, “AML_cohort.tsv”

* R base incluye conjuntos de datos sobre los que trabajar, como el conjunto ToothGrowth. Estos conjuntos de datos se cargan con la función data(). * Vuestros propios datos.

11. Cómo seguir aprendiendo por tu cuenta

Un recurso muy sencillo, gratis y eficaz es swirl, que es un paquete para aprender R dentro de R. La instalación de swirl se realiza con install.packages(). Podéis encontrar las instrucciones en este enlace.

```
# install.packages("swirl")  
# library(swirl)  
# swirl()
```

También hay mucho material disponible en internet. Algunas recomendaciones:

- “An introduction to R” (31/10/2022). <https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>
- Los libros de Rafael Irizarry, profesor de bioestadística en la Universidad de Harvard. Uno de sus libros es Introduction to Data Science, disponible gratuitamente en Leanpub (<https://leanpub.com/datasciencebook>). También hay la versión en español (<http://rafalab.dfci.harvard.edu/dslibro/>). Echad también un ojo a sus cursos en edX (<https://www.edx.org/es/bio/rafael-irizarry>). Aunque son de pago si queréis el certificado, el material está accesible gratuitamente durante bastante tiempo.
- Cursos gratuitos en Datacamp: <https://www.datacamp.com/courses/free-introduction-to-r>

- Esta página es muy resolutive en temas de estadística: <http://www.sthda.com/english/wiki/what-is-r-and-why-learning-r-programming>

La forma de aprender a usar R es usándolo. Buscad algo que os motive. Un buen ejercicio puede ser usar datos de vuestros proyectos que sean sencillos y podáis comparar vuestros análisis con R y con la herramienta que hayáis empleado hasta ahora. En vuestra cuenta de Posit Cloud ahora tenéis guardado lo que habéis hecho en esta sesión. Os animamos a que instaléis R y RStudio en casa, a que os entretengáis un rato a repasar lo que hemos visto, a que os salgan errores, a intentar solucionarlo,... y a que poco a poco os vayáis animando a utilizarlo para analizar vuestros propios datos. Veréis que cuando os deis cuenta de que R es como un folio en blanco en el que podéis pintar con absoluta libertad, ¡no querréis volver a usar otra cosa!