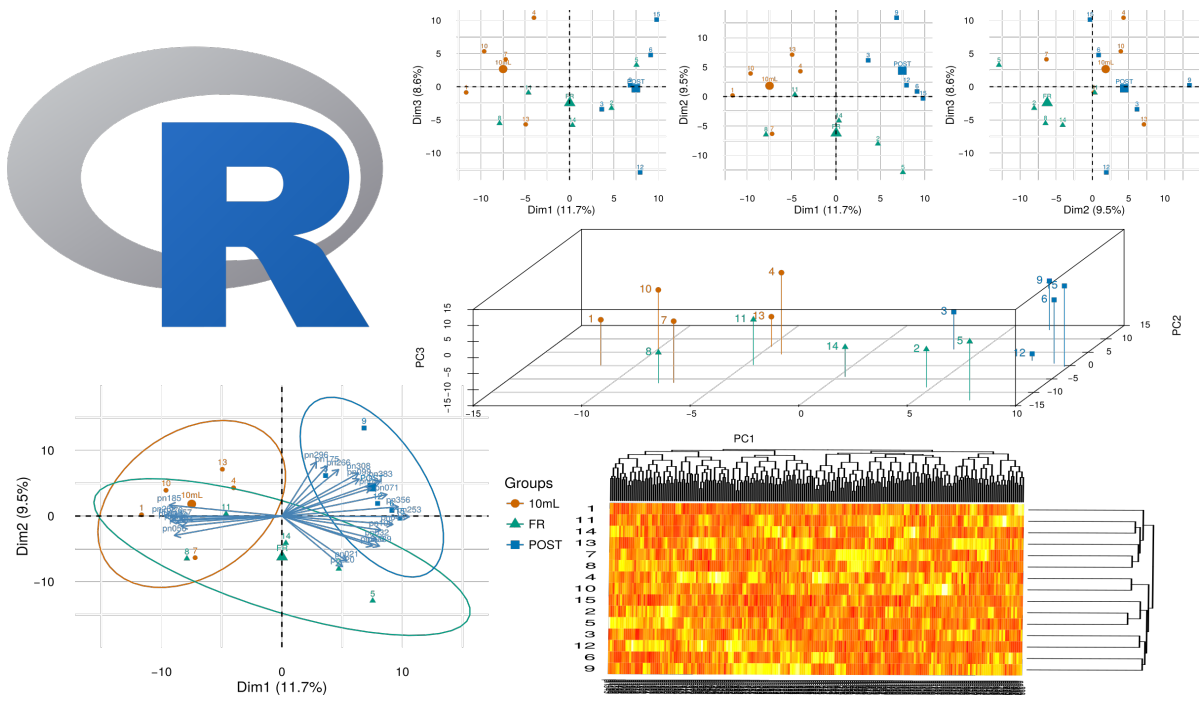
Análisis de datos y métodos estadísticos con R



Antonio Maurandi López y Aurora Gónzalez Vidal (Eds.)

edit.um
Ediciones de la Universidad de Murcia

Análisis de datos y métodos estadísticos con R

Análisis de datos y métodos estadísticos con R

Antonio Maurandi López y Aurora Gónzalez Vidal (Eds.)

Editum. Ediciones de la Universidad de Murcia

Cómo citar esta obra:
Maurandi López, A., y González Vidal, A.(Eds.). (2022). *Análisis de datos y métodos estadísticos con R*. Editum. Ediciones de la Universidad de Murcia.
https://doi.org/10.6018/editum.2967

DOI: 10.618/editum.2967

ISBN: 978-84-09-43065-9

Si el lector detecta algún error en el libro o bien quiere contactar con el autor, puede enviar un correo a publicaciones@um.es

Diseño y Maquetación: Antonio Maurandi López, Aurora González Vidal.
LaTeX

Relación de autores

Coordinación editorial y autoría:

Dr. Antonio Maurandi López
Dpto. de Didáctica de las Ciencias Matemáticas y Sociales
Fac. de Educación, Univ. de Murcia

Dra. Aurora Gónzalez Vidal
Dpto. de Ingeniería de la Información y las Comunicaciones
Fac. de Informática, Univ. de Murcia

Autorías de los capítulos:

Dra. Laura del Río Alonso
Dpto. de Sanidad Animal
Fac. de Veterinaria , Univ. de Murcia

Álvaro Hernández Vicente
Dpto. de Medicina
Fac. de Medicina, Univ. de Murcia

María Elvira Ferre Jaén
Facultativa de Estadística
Observatorio de Empleo, Univ. de Murcia

Dr. Francisco Javier Ibáñez López
Dpto. de Métodos de Investigación y Diagnóstico en Educación
Fac. de Educación , Univ. de Murcia

Dra. Mª. Francisca Carreño Fructuoso
Dpto. de Geología Minera
Centro tecnológico del mármol, piedra y materiales

Dr. Fernándo Pérez Sanz
Investigador Colaborador
Instituto Murciano de Investigación Biosanitaria Virgen de la Arrixaca

Dr. José Ant. Palazón Ferrando
Dpto. de Ecología e Hidrología
Fac. de Biología , Univ. de Murcia

Índice general

Índice de figuras

Índice de tablas

Prólogo

Este libro recopila y da forma a parte del trabajo de muchos años que hemos realizado un grupo de colegas, amigos, compañeros de la Universidad de Murcia (UMU) que tenemos un nexo común que es el entendimiento de R como *lingua franca* que conecta las ciencias y humanidades con el análisis de datos.

Todo comienza cuando el profesor José Antonio Palazón organiza las primeras Jornadas de Usuarios de R en 2008 en Murcia e involucra a Antonio Maurandi, en aquellos tiempos responsable de la Sección de Apoyo Estadístico (SAE) de Servicio de Apoyo a la Investigación (SAI) de la Universidad de Murcia. Esto supone un hito para el SAE, ya que conlleva el abandono progresivo de programas propietarios como SPSS y el salto definitivo a R y GNU/Linux en todos los trabajos, implicando evidentemente un cambio en a filosofía de trabajo de toda la sección. El SAE se convirtió en un hervidero de ideas y en el germen de un fructífero grupo de trabajo que comenzó con los estudiantes del Grado de Matemáticas que hacían sus prácticas curriculares en el SAE. Los y las estudiantes comenzaban como prácticas curriculares, se reenganchaban como no-curriculares, hubo becas propias y finalmente algunas disfrutaron de contratos laborales del ministerio. Fue una época preciosa, mucho trabajo pero también mucha diversión y crecimiento personal.

Yo fui una de esas estudiantes que llegó terminando la carrera al SAE y salió como estudiante de doctorado. En el SAE atendíamos trabajos de *ciencia de datos* para investigación de muy diversas áreas de conocimiento: medicina, ciencias experimentales, veterinaria, educación, filologías, bellas artes,... la oficina llegó a ser un *"servicio de guardia"* para muchos grupos de investigación.

En 2014, Antonio Maurandi integra a tod@s est@s estudiantes como pro-

fesor@s del la tercera edición del curso `Fundamentos Estadísticos para Investigación. Introducción a R`, comúnmente conocido como el *"curso FEIR"*. El FEIR fue un curso de estudios propios de la Universidad de Murcia que tuvo un éxito rotundo, generó listas de correo y una pequeña comunidad de *eRReRos*. Parte del profesorado de este curso son autores de capítulos del presente libro. La experiencia fue muy positiva y enriquecedora, trabajar en un entorno colaborando, compartiendo, intergeneracional, interdisciplinar: matemáticos, biólogos, veterinarios... fue muy divertido. Fue *conocimiento abierto en esencia pura*, maestros mezclados con aprendices, profesores/as y los que hasta hacía bien poco éramos aún estudiantes, juntos en una dinámica que desembocó en lo que de broma atinamos a denominar el `00Rteam`. La filosofía de este grupo puede leerse en su web https://gauss.inf.um.es/00rteam.html.

En el año 2018 esta pequeña comunidad de amantes de R se conformó como asociación, denominada "Usuarios Murcia R" (UMUR), y nos animamos a albergar las "X Jornadas de R" otra vez en la Universidad de Murcia, ya la comunidad de *eRReRos* no era tan pequeña como diez años atrás. La asociación UMUR, a día de hoy sigue fomentando el avance del conocimiento y uso del lenguaje de programación R y el desarrollo de la profesión en todas sus vertientes, especialmente la investigadora, docente y empresarial (https://gauss.inf.um.es/umur/).

Este divertido grupo de gente del que hablamos ha impartido numerosos cursos sobre R y su filosofía a través de la Escuela Internacional de Doctorado (EIDUM), desde la *"Introducción a R"*, *Investigación reproducible*, a cursos avanzados de *"Análisis Mutlivariante"*, ha participado en asignaturas de varios másteres, y de él han surgido, por el momento, al menos cinco tesis doctorales relacionadas con el análisis de datos. Creemos que una Universidad debería ser así, que el conocimiento y la colaboración fueran la esencia de nuestro día a día, que *el mérito del maestr@ fuera que sus alumn@s llegaran más lejos que él mismo*.

Volviendo a este libro, debido a los numerosos cursos y talleres que hemos impartido, de forma orgánica, se han generado materiales abiertos que han estado en uso y revisión durante años y ahora, habiendo seleccionado algunos de ellos y ampliado otros, ha nacido este libro. Esperamos que este material le sea útil y lo disfrute tanto como nosotros hemos disfrutando creándolo.

Aurora González Vidal

Introducción

Este libro es un compendio de apuntes del curso "Fundamentos Estadísticos para Investigación. Introducción a R", cuya primera edición fue en el curso 2011/2012 y al que se incorporaron nuevos profesores en el curso 2014/15, cuando formalizamos los materiales, *"tema a tema"*. Después, hasta la fecha de hoy hemos seguido actualizando el material a través de numerosos curso que hemos impartido, los autores de este libro, para la Escuela Internacional de Doctorado de la UMU. Así que creemos que es un trabajo maduro, pero no definitivo, seguiremos actualizándolo según creamos que hay mejores maneras de hacer las cosas y atentos a las corrientes que surjan en el *universo o ecosistema de R*, por eso al final de esta breve introducción facilitamos un repositorio de gitHub donde se irá incluyendo el material más actual.

El material está estructurado en 12 capítulos.

El **capítulo 1**, "Entorno de trabajo R. RStudio", es una breve introducción a R, desde lo más básico. Mucha gente aprende R de forma desordenada, sin embargo creemos que es interesante partir de una base ordenada y después ir saltando en función de lo que más interese, pero R como lenguaje tiene sus particularidades que es interesante tratar alguna vez despacio.

El **capítulo 2**, "Estadística descriptiva", se podría considerar continuación del primero, partiendo de definiciones formales aplicamos lo más básico de R para hacer estadística también básica: estadística descriptiva, dejando entender la potencia de hacerlo con R y deja la puerta abierta al capítulo siguiente.

En el **capítulo 3**, "Introducción a los contrastes", tratamos las bases de la estadística inferencial, intentamos que se entienda qué es un p-valor, que es un contraste y trabajamos los supuestos habituales de estos contrastes, normalidad,

homocedasticidad, etc... desde un punto de vista práctico, con muchos ejemplos y código, pero sin olvidar el fundamento teórico.

En el **capítulo 4**, "Comparación paramétrica de medias" y el **capítulo 5** "Contrastes no paramétricos"", trabajamos la comparación de dos o más grupos, y las diferentes técnicas y abordajes al problema según las hipótesis previas que podamos asumir o no, se trabaja desde un punto de vista práctico, proporcionando los necesarios *how to* en R pero sin olvidar el rigor estadístico de la cuestión.

En el **capítulo 6 y 7**, "Modelos de Regresión Lineal" y "Regresión logística", abordamos los modelos de regresión más habituales, lineal simple y múltiple y el modelo de regresión logística que es tan recurrido en ciencia experimental hoy en día.

En el **capítulo 8**, "Potencia Estadística", abordamos la pregunta *¿merece la pena hacer este estudio con estos "n" individuos que tengo?"*. Trabajamos con librería pwr y proporcionamos los *how to* para calcular en cada técnica los parámetros necesarios.

En el **capítulo 9**, "Miscelánea visual", hemos juntado varios conceptos: gráficos, aplicaciones visuales y tablas. Trabajamos diferentes formas de hacer gráficos en R, usamos aquí la librería "base" para hacer gráficos y el más potente ggplot, que es una sintaxis de gráficos completa y hoy en día ha desplazado a casi todas las otras formas de graficar en R. También en este capítulo hemos incluido una introducción a la librería Shiny para crear aplicaciones web con R. Terminamos con una introducción a la libreria kableExtra, extraordinaria librería que permite crear tablas para informes vistosos tanto en html como el pdf vía LaTeX.

En el **capítulo 10**, "Manipulación de datos", repasamos formas que podríamos llamar clásicas de manipulación de dataframes, y librerías que en las últimos tiempos se han ido imponiendo por su versatilidad e integración con otras librerías como ggplot, hablamos del universo Tidyverse.

En el **capítulo 11**, "Trabajar con Likert". El análisis de cuestionarios es una tarea muy habitual en ciencias sociales, así que hemos querido dedicar un capítulo completo a su análisis y validación, prestando especial atención a las escalas tipo Likert.

En el **capítulo 12**, "Análisis multivariante: principales técnicas", como su propio nombre indica tratamos las principales técnicas de Análisis multivariante: Clasificación y Ordenación.

Puedes encontrar materiales actualizados y los conjuntos de datos mencionados en este repositorio online:

https://github.com/auroragonzalez/ADMER.

Antonio Maurandi López

1 Entorno de trabajo R. RStudio

Antonio Maurandi López

1.1. ¿Qué es R?

R es un potente lenguaje orientado a objetos y destinado al análisis estadístico y la representación de datos. Se trata de software libre que permite su utilización libre y gratuitamente. La comunidad científica internacional lo ha elegido como la lingua franca del análisis de datos. Y tiene una gran implantación en universidades y cada vez más en mundo empresarial.

Otra definición: *R es un paquete estadístico de última generación al mismo tiempo que un lenguaje de programación*.

1.1.1. ¿Por qué emplear R?

Estadística se puede hacer con miles de programas estadísticos, incluso con hojas de cálculo e incluso los más osados con lápiz y papel (aunque la exactitud

Cómo citar este capítulo: Maurandi-López, A. (2022). Entorno de trabajo R. RStudio. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (7-60). Editum. Ediciones de la Universidad de Murcia.

de las máquinas es una potencia que no podemos desdeñar: cálculo de p-valores, estadísticos exactos...).

¿Qué tiene R que tanto nos gusta?:

- Es libre. Se distribuye bajo licencia GNU, lo cual significa que lo puedes utilizar y ¡mejorar!
- Es multiplataforma, hay versiones para Linux, Windows, Mac, iPhone... ¡web!
- Se puede analizar en R cualquier tipo de datos.
- Es potente. Es muy potente.
- Su capacidad gráfica difícilmente es superada por ningún otro paquete estadístico.
- Es compatible con 'todos' los formatos de datos (`.csv`, `.xls`, `.sav`, `.sas`...)
- Es ampliable, si quieres añadir algo: ¡empaquétalo!
- Hay miles de técnicas estadísticas implementadas, cada día hay más.
- ...

Además en los últimos años, junto el *boom* de la ciencia de datos a nivel mundial, ha demostrado ser el estándar más avanzado y versátil para el análisis de de datos, como decíamos anteriormente, la la lingua franca del análisis de datos. Se ha generalizado su uso desde un ambiente académico a todos los sectores empresariales. Además grandes apuestan por su uso con desarrollos basados en R, es el caso de RStudio.

1.1.2. Instalación de R

La instalación difiere para cada sistema operativo; en Windows es un ejecutable, en GNU/Linux se hace habitualmente ejecutando algunos comandos desde la consola...

La última versión, cuando se elabora este documento, es la R 4.1.2 que está disponible para su descarga desde la página web de CRAN: http://cran.es.r-

project.org/

1.1.2. En Windows

Se hace desde un ejecutable que bajamos desde "Download R 4.1.2 for Windows (62 megabytes, 32/64 bit)" y lo instalamos ejecutándolo; un vídeo de la instalación puede verse https://youtu.be/0jlMXPMoiOg.

1.1.2. En GNU/Linux

Dependiendo de la distribución el procedimiento puede variar, a modo de ejemplo indicamos el procedimiento para la instalación en una ditribución Ubuntu:

```
sudo apt-get update
sudo apt-get install r-base
```

Para más detalles sobre el procedimiento ver CRAN o esté sitio web: www.linuxize.com.

1.1.2. En Mac OX

Tanto la información para la intalación como el fichero necesarios están disponibles aquí: http://cran.es.r-project.org/bin/macosx/

1.1.2. Comprobando la instalación

Lanzar R y salir:

1. En GNU/Linux: R
2. En Windows: doble clic en el icono "R"
3. Para salir de la aplicación ejecutamos en ambos S.O.:

```
q()  # para salir
```

1.1.3. Trabajando con R

Tradicionalmente con R se trabaja en una consola y con *scripts*. Hoy en día ha dejado de ser tan usual esta forma de trabajo y trabajamos a través de interfaces de usuario (GUI) más avanzadas como RStudio, las cuales presentan "facilidades" para el trabajo a las que rápidamente nos acostumbramos, veremos RStudio más adelante por ser la GUI más popular. No obstante sigue siendo muy importante tener cierto manejo de la consola para obtener el máximo partido de R.

1.1.3. Ayuda en R

¿Cómo obtenemos ayuda en R? Veamos la siguiente tabla 1.1:

Función	Acción
`help.start()`	Ayuda general
`help(mean)` o `?mean`	Función de ayuda
`help.search("mean")`	Ayuda online
`RSiteSearch("mean")`	
`apropos("mean", mode ="function")`	Lista todas las funciones que contienen `mean` en el nombre
`data()`	Muestra los conjuntos de datos de ejemplo que hay disponibles

Tabla 1.1: *Funciones de ayuda en R*

Otra fuente de ayuda son las listas de correo, `R-help-es`: https://stat.ethz.ch/mailman/listinfo/r-help-es. Es una lista de distribución donde uno puede preguntar y responder dudas que surgen al trabajar con R. Esta lista surgió muy cerca de la Universidad de Murcia y gracias a ella se han realizado ya más de 10 jornadas de usuarios de R en español; la primera en Murcia, la segunda en Oviedo y las sucesivas en Madrid, Barcelona, Zaragoza y Santiago de Compostela, la décima volvió a ser en Murcia. Las jornadas de organizan desde la Asociación R Hispano, para saber más de estas jornadas visita: www.r-es.org.

1.1.3. Espacio y directorio de trabajo

El *workspace* es el espacio de trabajo en que se incluyen todos los objetos definidos por el usuario (ya veremos qué son estos objetos, que incluyen variables, vectores, *dataframes*...), se almacena en memoria intermedia mientras trabajas con R.

Cuando termina una sesión de R, el propio R te pregunta si quieres guardar el *workspace* para usos futuros. Este espacio, "workspace", se recarga al volver a iniciar la sesión. El directorio de trabajo o *working directory* es el directorio donde por defecto "lee" R. También es donde guardará el *workspace* al finalizar la sesión y donde buscará un workspace guardado al inicio. Si quieres que R lea un fichero que no esté en *directory* hay que especificar la ruta completa.

Funciones para manejar el *workspace*, tabla 1.2:

Tabla 1.2: *Funciones para el manejo del workspace*

Función	Acción
`getwd()`	Muestra el wd: *working directory*
`setwd("midirectorio")`	Ajusta el wd al especificado
`ls()` o `dir()`	Lista lo que hay en el `wd`
`history()`	Muestra los últimos comandos ejecutados
`savehistory()`	Guarda el historial de comandos, por defecto en `.RHistory`
`loadhistory()`	Carga el historial de comandos
`save.image("mywspace.R")`	Guarda los objetos del *workspace*, por defecto en `.RData`
`load("mywspace.R")`	Carga el *workspace* `mywspace.R`

1.1.3. Cosas varias: asignaciones, variables, comentarios, inputs...

Todo lo precedido por almohadillas # R lo considera un comentario y no lo *interpreta*. En R empleamos el operador <- para hacer asignaciones, y una variable se crea *al vuelo*, esto es, en el mismo instante en el que la asignas. Es más, en R **no puedes declarar variables con anterioridad y dejarlas vacías** como sí ocurre

con otros lenguajes de programación.

Así:

```
mivariable <- 7  # asigna el número 7 a una variable
```

Si quiero ver qué valor toma "mi variable" escribo:

```
mivariable
```

```
[1] 7
```

Nota: Cualquier asignación a una variable crea o reutiliza un objeto de R.

Si guardo el *workspace* , ahora, quedará guardada mi nueva variable en él.

1.1.3. Una sesión "tipo" en R

Veamos expresiones que habitualmente se usan una sesión de R.

```
> getwd()
```

```
[1] "C:/Documents and Settings/Administrador/Mis documentos"
```

```
> setwd( "C:/CursoR_UMU" ) # ¡¡Ojo con la barra!!
```

```
set.seed( 1 )
x <- runif( 20 )
x
```

```
 [1] 0.2655 0.3721 0.5729 0.9082 0.2017 0.8984 0.9447 0.6608 0.6291
[10] 0.0618 0.2060 0.1766 0.6870 0.3841 0.7698 0.4977 0.7176 0.9919
[19] 0.3800 0.7774
```

```
summary( x )
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.062   0.345   0.601   0.555   0.772   0.992
```

```
hist( x )
```

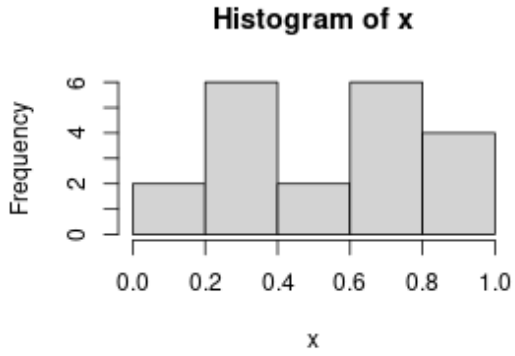


Figura 1.1: *Ejemplo de gráfico sencillo: Histograma.*

Ahora cerramos la sesión guardando, explícitamente, el historial de expresiones y el *work space* actual (*objetos*).

```
> savehistory()
> save.image()
> q()
```

1.1.3. R como calculadora

Podemos emplear R como una calculadora de la siguiente manera:

```
45 + 23
```

```
[1] 68
```

```
100 / 4
```

```
[1] 25
```

```
sqrt( 25 )
```

```
[1] 5
```

```
# usando variables
x <- 100 / 4
```

```
5
```

```
[1] 5
```

hay definidas algunas constantes...

```
pi
```

```
[1] 3.14
```

```
r    <- 5
area <- 2 * pi + r
area
```

```
[1] 11.3
```

```
c( 25, 100, 2, 3 ) * 5
```

```
[1] 125 500  10  15
```

Nota: Cambia el número de decimales por defecto con `options( digits = 12 )` y para saber en que está establecido usa `getOption( "digits" )`.

```
old <- getOption( "digits" ) # guardamos la opcion actual
getOption( "digits" )
```

```
[1] 3
```

```
options( digits = 12 )  # cambiamos el número de digitos
getOption( "digits" )
```

```
[1] 12
```

```
options( digits = old ) # lo devolvemos al estado inicial
getOption( "digits" )
```

```
[1] 3
```

1.1.3. En R trabajamos con scripts

Normalmente una sesión de trabajo partirá de otra previa, y lo que hicimos previamente lo dejamos almacenado en un fichero de de intuiciones (*comandos*) o *script*. Esta forma de trabajar es muy potente pues nos posibilita, además de repetir un análisis tal cual se hizo (*bitácora* de trabajo), recuperarlo para mejorarlo, ampliarlo, corregirlo o adaptarlo a otros datos. Para cargar un fichero con instrucciones, un *script*, empleamos la función `source()`.

```
source( "fichero_de_comandos.R" )
```

Un fichero que crease un histograma de una curva normal debería de contener, por ejemplo, estas instrucciones:

```
x <- rnorm( 100 )  # crea 100 observaciones aleatorias
                   # de una normal tipificada
hist( x )
```

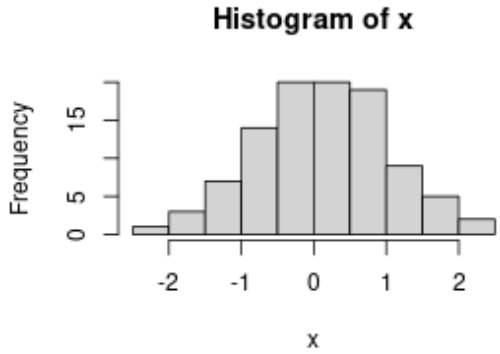


Figura 1.2: *Gráfico histograma creado desde un script.*

Así que creamos un fichero de texto plano con el contenido anterior, lo guardamos con el nombre `script.R` (en realidad da lo mismo la extensión, es una convención). Ejecutamos:

```
source( "script.R" )
```

> **Ejercicio**: Ejecutar mediante el comando `source()` el fichero `10A-ejSource.R`

1.1.3. Ampliar la funcionalidad de R. Packages

Con la instalación simple de R tenemos muchísimas posibilidades, no obstante existen multitud de módulos opcionales que llamamos paquetes, *packages*. Los paquetes son colecciones de funciones y datos.

El directorio de tu PC donde se almacenan los *packages* es denominado `library`.

```
.libPaths()
[1] "C:/Archivos de programa/R/R-4.2/library"
```

o

```
.libPaths()
[1] "/home/usr/R/x86-64-pc-linux-gnu-library/4.2"
[2] "/usr/local/lib/R/site-library"
[3] "/usr/lib/R/site-library"
[4] "/usr/lib/R/library"
```

Para saber qué paquetes tienes instalados empleamos la función `library()`. *No es lo mismo instalar que cargar un paquete*. La instrucción `search()` nos informa de que paquetes están instalados y cargados en el sistema listos para usarse.

Instalamos un paquete con la instrucción `install.packages("nombre_paquete")`, sólo instalamos una vez cada paquete, cargamos el paquete con la instrucción `library("nombre_paquete")` y tendremos que ejecutar este comando en cada sesión en que queramos emplear dicho paquete.

```
install.packages("foreign")
```

```
--- Please select a CRAN mirror for use in this session ---
HTTPS CRAN mirror
Selection: 39
probando la URL 'https://ftp.cixug.es/CRAN/src/contrib/foreign_0.8-67.tar.gz'
Content type 'application/x-gzip' length 334175 bytes (326 KB)
==================================================
downloaded 326 KB
```

```
....
....
installing to /home/amaurandi/R/x86_64-pc-linux-gnu-library/3.3/foreign/libs
...
* DONE (foreign)

The downloaded source packages are in
    `/tmp/Rtmp3wi1AY/downloaded_packages'
```

Los paquetes disponibles están en CRAN, específicamente en http://cran.r-project.org/.

Normalmente son necesarios más paquetes que los que vienen por defecto, así que siempre se está instalando paquetes nuevos. A lo largo de este documento veremos algunos de los más usuales.

Para obtener información sobre un paquete empleamos la función `help( package = "nombre_del_paquete" )`.

Nota: ¡En el momento de escribir este manual existen 18921 paquetes! Seguro que si lo comprobamos ahora mismo habrá más.

Ejercicio: Visitar la web de CRAN http://cran.r-project.org/ y consultar los últimos paquetes añadidos/modificados.

Nota. Los *Task View* son colecciones de paquetes para trabajos concretos o para áreas de trabajo concretas. Instalándose una *Task View* automáticamente conjuntos enteros de paquetes se instalan sin tener que ir instalando paquete a paquete.

```
> To automatically install these views, the ctv package needs to be
> installed, e.g., via
> install.packages("ctv")
> library("ctv")
> and then the views can be installed
> via `install.views` or `update.views` (which first assesses which of
> the packages are already installed and up-to-\>date), e.g.,
> install.views("Econometrics")
> or
```

```
> update.views("Econometrics")
```

1.1.3. Otra *sesión tipo* de R

Probemos a comunicarnos con R y ver su versatilidad e interactividad con un ejemplo donde de forma intuitiva emplearemos estructuras de datos que no hemos introducido aún y algunas funciones usuales.

```
peso <- c( 4.4, 5.4, 6.4, 3.2, 7.5, 3, 6.1, 3.1, 6.1, 7, 3.4 )
        # c() concatena y crea un vector
edad <- c( 1, 2, 3, 4, 5, 6, 7, 8, 9, 1 )
edad
```

```
 [1] 1 2 3 4 5 6 7 8 9 1
```

```
peso
```

```
 [1] 4.4 5.4 6.4 3.2 7.5 3.0 6.1 3.1 6.1 7.0 3.4
```

```
mean( peso )  # función media
```

```
[1] 5.05
```

```
mean( edad )
```

```
[1] 4.6
```

```
length( edad )
```

```
[1] 10
```

```
length( peso )  # función longitud/dimensión
```

```
[1] 11
```

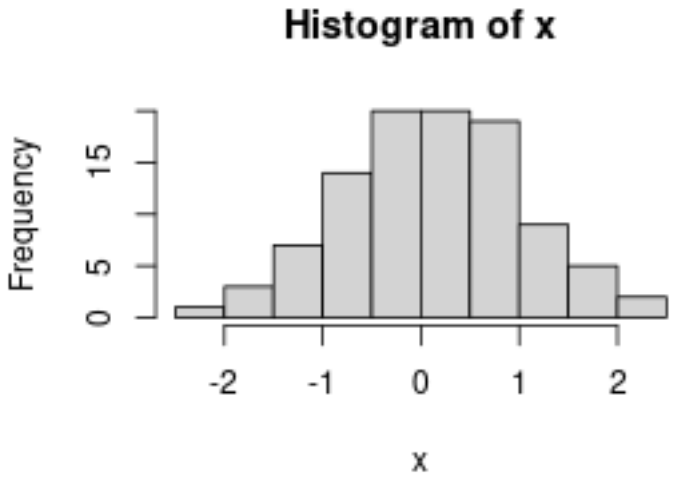
```
edad <- c( edad, 5)  # Nota le añadimos a edad una observación más
length( edad )
```

```
[1] 11
```

```
cor( peso, edad )
```

```
[1] -0.188
```

```
plot( edad, peso )
```




```
q()
```

1.1.4. RStudio

1.1.4. Una GUI para R multiplataforma

Una de las cosas que menos gustan de R es que *se ve diferente en Windows que en Linux*. Algo que suele molestar a los usuarios más experimentados en R es utilizar programas en que los menús te limiten (algo parecido le pasa a los usuarios de GNU/Linux). En general, el usuario de R no quiere que los menús de la aplicación le sugieran qué hacer con sus datos, más bien simplemente quiere *decirle* al software qué hacer con los datos, es él el que manda. Esto se hace mediante la combinación de expresiones y gracias a un poco de formación, experiencia y tesón.

Una programa que soluciona en gran medida estos problemas es **RStudio**

http://rstudio.org/.

RStudio es una GUI, *Graphical user interface*, para R programada en `C#`, multiplataforma (Windows, Linux y Mac). Que aúna todos los entornos y asume la filosofía de los expresiones, pero aportando algunas 'ayudas' que hacen más llevadero el día a día.

Otra definición de RStudio: es un entorno libre y de código abierto para el desarrollo integrado (IDE) de R. Se puede ejecutar en el escritorio (Windows, Mac o Linux) o incluso a través de Internet mediante el servidor RStudio.

Entre otras cosas encontramos que RStudio:

- Nos permite abrir varios *scripts* a la vez
- Nos permite ejecutar trozos de código con sólo marcarlo en los *scripts*
- Nos muestra el *workspace*
- Nos muestra el historial
- Nos muestra los objetos del *workspace*
- Integra la ayuda
- Integra la gestión de librerías
- Permite intercalar un lenguaje de marcas, `markdown`, con código de R, con el paquete Rmarkdown y knitr.
- Genera salidas en HTML, LaTeX y otros formatos.
- Facilita enormemente el trabajo reproducible y por lo tanto la investigación reproducible.
- Se integra con Git y con SVN.
- etc.

Actualmente RStudio es un programa muy desarrollado con un historial amplio de versiones estables y ofrece una gran integración con ficheros en diversos formatos: R scripts (.R), markdown (md), Rmarkdown (.Rmd), LaTeX (.tex), .. que posibilitan desarrollar cómodamente investigación reproducirle gracias al paquete `knitr` de Yihui Xie [2]. Puede ampliar información sobre este tema en el libro de Christopher Gandrud [3]. La gran gran facilidad para generar documentos dinámicos con RStudio y `knitr` ha hecho que RStudio se convierta en la GUI por excelencia para R. Este documento está editado con RStudio y procesado con los paquetes `knitr` y `rmarkdown` aunque el último toque se lo hemos dado trabajando directamente con LaTeX y TeXstudio.

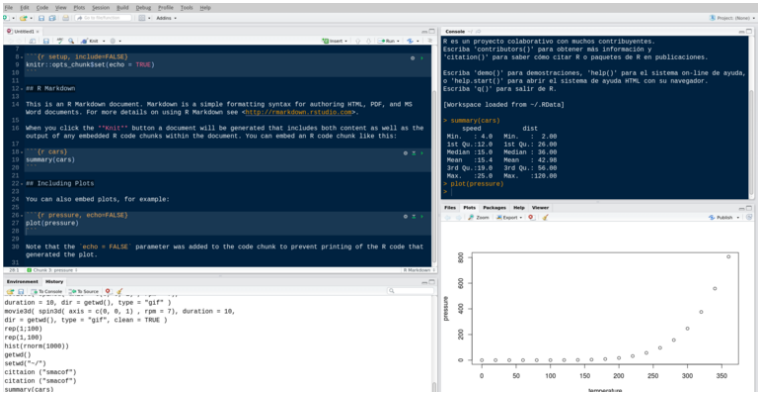


Figura 1.3: *RStudio*

1.2. Estructuras de datos en R

Trabajaremos con ejemplos y fijándonos en los diferentes tipos de datos y de estructuras que R tiene para albergarlos.

Un *data set* es (normalmente) un "conjunto de datos" que está ordenado por filas y columnas, y donde cada fila representa una observación y cada columna una variable. Dependiendo de las áreas de conocimiento recibe distinto nombre: tabla de datos, matriz de datos,...

Importamos un fichero de datos desde un `csv` (no nos preocupemos, de momento, en los aspectos relacionados con el cómo importar datos).

```
read.table( "10A-ejemplo01.csv",
            sep = ';',
            head = T )
```

```
  paciente   admision edad diabetes status
1        1 11/11/2010   23   tipo 1  bueno
2        2 11/01/2009   56   tipo 1  bueno
3        3 15/03/2012   78   tipo 2   malo
4        4 28/04/2008   34   tipo 1  bueno
5        5 09/08/2011   23   tipo 2 mejora
6        6 24/04/2012   12   tipo 1  bueno
7        7 08/08/2008   56   tipo 2  bueno
```

```
>
```

Hay diversos tipos de variables, una forma de clasificarlas podría ser esta: *datos nominales*, *ordinales* y *numéricos*.

En el ejemplo anterior:

- Son variables numéricas: `paciente` y `edad`
- Son variables ordinales: `status`
- Son variables nominales: `diabetes`

Nota. Para almacenar los datos R cuenta, como todos los lenguajes de programación, de una gran variedad de estructuras de datos. Estas van de lo más sencillo a las estructuras más complejas.

1.2.1. Tipos de objetos

1.2.1. Vectores

Son matrices de una dimensión que solamente pueden contener valores homogéneos, ya sean numéricos, alfanuméricos o valores lógicos. Emplearemos la función `c()` para construir vectores (función *combine*).

```
x <- c( 1, 2, 3, 4, 5, 6, 7, 8 )
y <- c( "juan", "pepe", "iñaky", "amparito"
        ,"mariano", "juancar", "fulano", "elefante" )
z <- c( TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE )
```

Podemos acceder a los elementos un vector usando los corchetes, que indican subíndice, así, x_3, x_6, ...

```
y[ 3 ]
```

```
[1] "iñaky"
```

```
y[ 6 ]
```

```
[1] "juancar"
```

```
y[ 8 ]
```

```
[1] "elefante"
```

```
x[ 1 ]
```

```
[1] 1
```

Es importante ver que un subíndice puede ser una expresión y por lo tanto un conjunto de valores:

```
y[ c( 6, 8 ) ]
```

```
[1] "juancar"  "elefante"
```

Nota: En R los *índices (numeraciones)* comienzan en el 1, no en el 0 como ocurre en muchos lenguajes de programación.

Extra: El operador ":" genera secuencias.

```
c( 2:6 )
```

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

```
c( 1:3 )
```

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

Nota: Podemos crear un vector de un cierto tipo y dejarlo vacío para ir *llenándolo* después en un bucle por ejemplo. Se puede hacer con las instrucciones `v<-vector('numeric')` o `v<-vector('character')`.

1.2.1. Matrices

Una matriz es un vector con un atributo adicional (`dim`), que a su vez es un vector numérico de longitud 2 que define el número de filas y columnas. Se crean con la función `matrix()`.

```
matrix( data = NA, nrow = 2, ncol = 2, byrow = F, dimnames = NULL )
```

```
     [,1] [,2]
[1,]   NA   NA
```

```
[2,]   NA   NA
```

```
matrix( data = 1:4, nrow = 2, ncol = 2, byrow = F, dimnames = NULL )
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
matrix( data = 5, nrow = 2, ncol = 2, byrow = F, dimnames = NULL )
```

```
     [,1] [,2]
[1,]    5    5
[2,]    5    5
```

```
matrix( data = 1:6, nrow = 2, ncol = 2, byrow = F, dimnames = NULL )
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
matrix( data = 1:6, nrow = 2, ncol = 3, byrow = F, dimnames = NULL )
```

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

```
x <- matrix( data = 1:6, nrow = 2, ncol = 3, byrow = F, dimnames = NULL )
dim( x )
```

```
[1] 2 3
```

Las matrices son muy versátiles, podemos convertir vectores en matrices.

```
x <- 1:15
x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
dim( x )
```

```
NULL
```

```
dim( x ) <- c( 5, 3 )
```

```
x
```

```
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

```
x <- matrix( 1:10, nrow = 2 ) #¡no tenemos que especificar las columnas!
x
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Los **arrays** son otro tipo similar a las matrices pero pueden tener más de dos dimensiones. Nos nos detenemos en ellos.

1.2.1. Dataframes

Un *dataframe* (a veces se traduce como *'marco de datos'*) es una generalización de las matrices donde cada columna puede contener tipos de datos distintos al resto de columnas, manteniendo la misma longitud. Es lo que más se parece a una tabla de datos de SPSS o SAS, o de cualquier paquete estadístico estándar. Se crean con la función `data.frame()`.

```
nombre <- c( "juan", "pp", "iñaky", "amparo",
             "mariano", "juancar", "fulano", "elefante" )
edad <- c( 23, 24, 45, 67, 32, 56, 78, 45 )
peso <- c( 34, 34, 56, 78, 34, 56, 76, 87 )
length( nombre )
```

```
[1] 8
```

```
length( edad )
```

```
[1] 8
```

```
length( peso )
```

```
[1] 8
```

```
caseid <- c( 1:length( peso ) )
df     <- data.frame( caseid, nombre, edad, peso )
df
```

```
  caseid   nombre edad peso
1      1     juan   23   34
2      2       pp   24   34
3      3    iñaky   45   56
4      4   amparo   67   78
5      5  mariano   32   34
6      6  juancar   56   56
7      7   fulano   78   76
8      8 elefante   45   87
```

Seleccionar columnas concretas de un *dataframe* con corchetes [ ].

```
df[ 1:2 ]
```

```
  caseid   nombre
1      1     juan
2      2       pp
3      3    iñaky
4      4   amparo
5      5  mariano
6      6  juancar
7      7   fulano
8      8 elefante
```

```
df[ c( 1, 3 ) ]
```

```
  caseid edad
1      1   23
2      2   24
3      3   45
4      4   67
5      5   32
6      6   56
```

```
7       7   78
8       8   45
```

Prueba con `df["nombre"]`

Para acceder al vector que forma una de las columnas de un *dataframe* también usamos el operador $.

```
df$nombre
```

```
[1] "juan"     "pp"       "iñaky"    "amparo"   "mariano"  "juancar"
[7] "fulano"   "elefante"
```

```
df$peso
```

```
[1] 34 34 56 78 34 56 76 87
```

Ampliar un *dataframe*:

```
diabetes <- c( "Tipo1", "Tipo1", "Tipo2", "Tipo2",
               "Tipo1", "Tipo1", "Tipo2",  "Tipo1" )
estado   <- c( "bueno", "malo", "bueno", "bueno",
               "bueno", "malo", "bueno", "malo" )
length( diabetes )
```

```
[1] 8
```

```
length( estado )
```

```
[1] 8
```

```
df <- data.frame( df, diabetes, estado )
df
```

```
  caseid   nombre edad peso diabetes estado
1      1     juan   23   34    Tipo1  bueno
2      2       pp   24   34    Tipo1   malo
3      3    iñaky   45   56    Tipo2  bueno
4      4   amparo   67   78    Tipo2  bueno
5      5  mariano   32   34    Tipo1  bueno
6      6  juancar   56   56    Tipo1   malo
7      7   fulano   78   76    Tipo2  bueno
```

```
8      8 elefante   45   87    Tipo1   malo
```

Para conocer el número de filas de un dataframe podemos usar la función `nrow()`, `ncol()` nos devuelve el número de columnas.

```
nrow( df )
```

```
[1] 8
```

```
ncol( df )
```

```
[1] 6
```

Extra: Crear un tabla de frecuencias. Explorar la función `cbind()` para *ampliar* dataframes.

```
table( df$diabetes, df$estado )
```

```
        bueno malo
  Tipo1     2    3
  Tipo2     3    0
```

Las funciones `attach()`, `detach()` sirven para acceder más *fácilmente* a un dataframes ; *si bien no es aconsejable su uso*.

```
df$peso
```

```
[1] 34 34 56 78 34 56 76 87
```

```
attach( df )
```

```
The following objects are masked _by_ .GlobalEnv:

    caseid, diabetes, edad, estado, nombre, peso
```

```
peso
```

```
[1] 34 34 56 78 34 56 76 87
```

```
detach( df )
```

Nota: si ya existe un objeto con ese nombre puede llevarnos a confusión. Una alternativa al attach es el `with()`.

```
with( df, { +peso } )
```

```
[1] 34 34 56 78 34 56 76 87
```

En nuestro ejemplo estamos empleando la variable `caseid` para identificar las observaciones de forma unívoca. Podemos emplear la opción `rownames` en la función `data.frame()` con tal de usar una variable para cuestiones como etiquetar casos, etc.

```
df <- data.frame( caseid,
                  nombre,
                  edad,
                  peso,
                  diabetes,
                  estado,
                  row.names = caseid )
```

Nota: Por motivos de legibilidad y mantenibilidad del código no es recomendable emplear las funciones `attach()`, `detach()` y `with()`.

1.2.2. Factores

Las variables pueden ser clasificadas como *nominales*, *ordinales* o *numéricas*. Las variables nominales son variables categóricas sin un orden definido, como *diabetes* en nuestro ejemplo anterior.

```
df$diabetes
```

```
[1] "Tipo1" "Tipo1" "Tipo2" "Tipo2" "Tipo1" "Tipo1" "Tipo2" "Tipo1"
```

Las variables ordinales, por ejemplo `estatus`, implican orden pero no cantidad.

```
df$estado
```

```
[1] "bueno" "malo"  "bueno" "bueno" "bueno" "malo"  "bueno" "malo"
```

En R prestamos un interés especial a estas variables nominales y ordinales y las llamamos *factores*. Los factores son cruciales porque van a determinar cómo se analizarán los datos y cómo se mostrarán en gráficos. Para convertir un vector en un factor empleamos la función `factor()`.

```
estado   <- c( "bueno", "malo", "bueno", "bueno",
               "bueno", "malo", "bueno", "malo" )
diabetes <- c( "Tipo1", "Tipo1", "Tipo2", "Tipo2",
               "Tipo1", "Tipo1", "Tipo2",  "Tipo1" )
str( diabetes )
```

```
 chr [1:8] "Tipo1" "Tipo1" "Tipo2" "Tipo2" "Tipo1" "Tipo1" ...
```

```
str( estado )
```

```
 chr [1:8] "bueno" "malo" "bueno" "bueno" "bueno" "malo" "bueno" ...
```

```
diabetes <- factor( diabetes )
str( diabetes )
```

```
 Factor w/ 2 levels "Tipo1","Tipo2": 1 1 2 2 1 1 2 1
```

```
estado <- factor( estado )
str( estado )
```

```
Factor w/ 2 levels "bueno","malo": 1 2 1 1 1 2 1 2
```

Nota: La función `str()` nos muestra la estructura de un objeto. Observa que antes de hacer a diabetes un factor era un vector de `chars`.

Un repaso algo más extenso, sin ser muy largo, a las estructuras de datos y operadores se puede hacer siguiendo el *Curso básico de R* de F. Carmona y A. Berihuete, alojado en la web de la Universidad de Cádiz [4].

1.3. Entrada de datos

R es 'compatible' con prácticamente cualquier formato de datos.

Figura 1.4: *Entrada de datos.*

1.3.1. Desde el teclado

Si tenemos un *dataframe* y queremos editarlo *"a mano"* usamos la función `edit()`.

```
df <- data.frame( edad = numeric(   0 ),
                  sexo = character( 0 ),
                  peso = numeric(   0 ) )
df
```

```
[1] edad sexo peso
<0 rows> (or 0-length row.names)
```

Nota: lo hemos creado *vacío*. Ahora lo editaremos con `edit()`.

```
df <- edit( df )
```

Eliminar, borrar un objeto se puede hacer con `rm()`

```
rm( df )
```

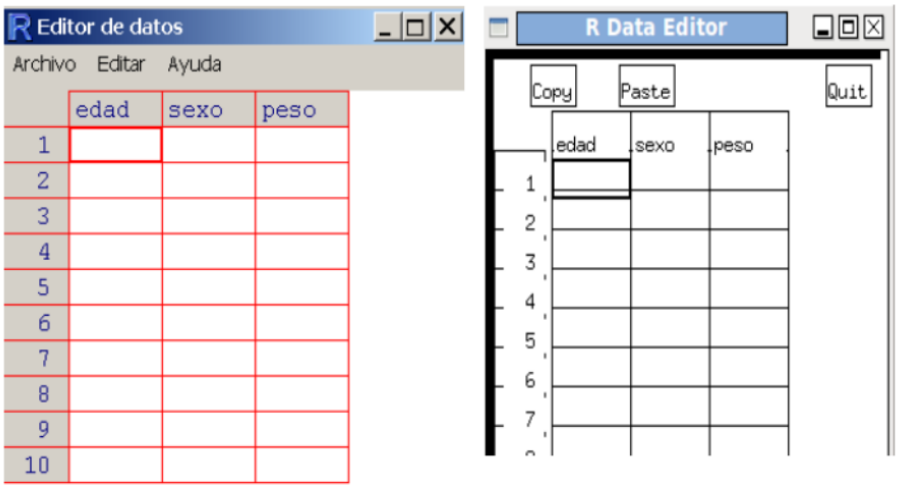


Figura 1.5: *Editores visuales de datos:* `edit( df )`.

1.3.2. Importar datos desde ficheros de texto

La función más importante para importar datos, al menos una de las más usadas es `read.table()`, automáticamente te convierte los datos en un *dataframe*.

```
df <- read.table( file,
                  header = logical_value,
                  sep = "delimiter",
                  row.names = "name" )
```

- `header` será TRUE o FALSE según la primera fila del fichero ASCII represente el nombre de las variables.

- sep es el delimitador, es decir, el separador de campos. Por defecto es el espacio " " pero se puede especificar lo que sea:
  sep=".", sep=",", sep="" (tabulación)...
- row.names es un parámetro opcional para especificar una o varias variables de identificador de casos.

```
df <- read.table( "10A-ejemplo01.csv",
                  sep = ";",
                  head = T )
```

Por defecto las variables de tipo carácter se convierten a factores, si queremos que no lo haga habrá que indicarle stringsAsFactor = FALSE como opción.

Hay muchas más opciones, mira la ayuda help(read.table). Algunas muy importantes tratan el tema de los valores faltantes (na.strings = "NA") , o de la separación de decimales para números (dec = ".").

1.3.3. Desde una hoja de cálculo

Lo mejor para leer ficheros *excel* es guardar éstos en formato *csv*, es decir, texto delimitado por comas. También podemos leer el *xls* directamente con el paquete RODBC.

```
> #install.packages("RODBC")
--- Please select a CRAN mirror for use in this session ---
probando la URL
'http://cran.es.r-project.org/bin/windows/contrib/2.14/RODBC_1.3-5.zip'
Content type 'application/zip' length 753058 bytes (735 Kb)
URL abierta
downloaded 735 Kb

package `RODBC' successfully unpacked and MD5 sums checked

The downloaded packages are in
        C:\WINDOWS\Temp\RtmpecjIRj\downloaded_packages
> library("RODBC")
Mensajes de aviso perdidos
```

```
package `RODBC' was built under R version 2.14.2
> channel <- odbcConnectExcel ("10A-ejemplo01.xls")
> df3<-sqlFetch(channel, "sheet1")
> df3
  paciente   admision edad diabetes status
1        1 2010-11-11   23   tipo 1  bueno
2        2 2009-01-11   56   tipo 1  bueno
3        3 2012-03-15   78   tipo 2   malo
4        4 2008-04-28   34   tipo 1  bueno
5        5 2011-08-09   23   tipo 2 mejora
6        6 2012-04-24   12   tipo 1  bueno
7        7 2008-08-08   56   tipo 2  bueno
8        8 2099-01-21   88   tipo 1   malo
> odbcClose(channel)
```

Otras funciones que te pueden interesar: `read.xlsx()`

```
library( xlsx )
ficheroexcel <- "c:/ficheroexcel.xlsx"
mydataframe  <- read.xlsx( ficheroexcel, 1 )
# La instrucción anterior lee la primera hoja
# del fichero ficheroexcel.xlsx
```

Otra librería muy interesante para trabajar con hojas de cálculo `openxlsx`.

1.3.4. Desde SPSS

Emplearemos la función `spss.get()` del paquete *Hmisc* (se requiere el paquete *foreign*).

```
# install.packages("foreign")
# install.packages("Hmisc")
library( Hmisc )
df4 <- spss.get( "10A-mydata.sav", use.value.labels = TRUE )

head( df4 )

  id   sexo     fechnac educ         catlab salario salini tiempemp
```

```
1  1 Hombre 11654150400    15      Directivo   57000  27000       98
2  2 Hombre 11852956800    16 Administrativo   40200  18750       98
3  3  Mujer 10943337600    12 Administrativo   21450  12000       98
4  4  Mujer 11502518400     8 Administrativo   21900  13200       98
5  5 Hombre 11749363200    15 Administrativo   45000  21000       98
6  6 Hombre 11860819200    15 Administrativo   32100  13500       98
  expprev minoría
1     144      No
2      36      No
3     381      No
4     190      No
5     138      No
6      67      No
```

1.3.5. Desde SAS

Pues emplear las funciones `read.ssd()` del paquete *foreign* y `sas.get()` del *Hmisc*. Aunque si estos ficheros son posteriores a SAS 9.1 no funcionarán; en ese caso deberás exportar los datos desde *SAS* a *csv* y emplear la función `read.table()` (*que siempre es mejor, más que ir aprendiendo tanta función*).

```
SAS :
proc export data=mydata
outfile="mydata.csv"
dbms=csv;
run;
```

1.3.6. Desde R

```
mydata <- read.table( "mydata.csv", header = TRUE, sep = "," )
```

1.3.7. Anotaciones

Al inicio de un análisis no necesitamos muchos detalles más que los vistos hasta ahora, pero para interpretar los resultados y que con el tiempo no dejen de

tener sentido conviene 'anotarlos'.

- **Etiquetas de variables**
  Para una variable cualquiera, por ejemplo *"edad"*, podríamos querer que tuviera una etiqueta (según la nomenclatura de *spss*), que fuera *"Edad el día de ingreso (años)"*

```
names( df )[ 3 ] <- "Edad el día de ingreso (años)"
names( df )
```

```
[1] "paciente"                      "admision"
[3] "Edad el día de ingreso (años)" "diabetes"
[5] "status"
```

- **Etiquetas de valores**
  Podemos tener un factor codificado, por ejemplo `sexo`, donde 1 es *'Masculino'* y 2 *'Femenino'*. Podemos crear unas etiquetas con el siguiente código.
  Primero le añadimos a nuestro ejemplo una variable más (una columna) de *"unos"* y *"doses"* representando el sexo codificado.

```
sexo    <- c( 1, 1, 1, 1, 2, 2, 2 )
df      <- data.frame( df, sexo )
df$sexo <- factor( df$sexo )
```

Comprobamos la estructura del *dataframe*:.

```
str( df )
```

```
'data.frame':   7 obs. of  6 variables:
 $ paciente                    : int  1 2 3 4 5 6 7
 $ admision                    : chr  "11/11/2010" "11/01/2009" "15/03/2012" "28/04/2008" ...
 $ Edad.el.día.de.ingreso..años.: int  23 56 78 34 23 12 56
 $ diabetes                    : chr  "tipo 1" "tipo 1" "tipo 2" "tipo1" ...
 $ status                      : chr  "bueno" "bueno" "malo" "bueno" ...
 $ sexo                        : Factor w/ 2 levels "1","2": 1 1 1 1 2 2 2
```

Creamos las etiquetas

```
df$sexo <- factor( df$sexo,
                   levels = c( 1, 2 ),
                   labels = c( "masculino", "femenino" )
                  )
```

Ejercicio. Mira ahora la estructura del *dataframe* con `str(df)`.

1.3.8. Algunas funciones útiles

Tabla 1.3: *Algunas funciones útiles.*

Función	Acción
`length(obj)`	Número de componentes, elementos
`dim(obj)`	Dimensión de un objeto
`str(obj)`	Estructura de un objeto
`class(obj)`	Clase (`class`) o tipo de objeto
`names(obj)`	Nombres de los componentes de un objeto
`c(obj,obj,...)`	Combina objetos en un vector
`head(obj)`	Lista la primera parte de un objeto
`tail(obj)`	Lista la última parte (cola) de un objeto
`ls()`	Lista los objetos actuales
`rm(obj)`	Borra un objeto
`newobj <- edit(obj)`	Edita un objeto y lo guarda
`fix(obj)`	Edita sobre un objeto ya creado

1.4. Manipulación básica de objetos en R

Trabajaremos desde ahora con la *GUI RStudio*, veremos que todo es más sencillo y procuraremos coger un poco de práctica en trabajo sobre *scripts*, que es como la mayor parte de los especialistas trabajan. Pronto veremos que es lo más cómodo.

Vamos a crear un *data set* y trabajaremos con él a lo largo del capítulo. El fichero con el *script* de creación del `dataframe` inicial lo puedes encontrar en `manipulacion-basica.R`

```
# getwd()
manager <- c( 1:5 )
date    <- c( "10/11/08", "10/12/08", "10/13/08", "10/14/08", "10/15/08" )
country <- c( "US", "US", "UK", "UK", "UK" )
```

```
gender  <- c( "M", "F", "F", NA, "F" )
age     <- c ( NA, 45, 25, 39, 99 )
q1      <- c( 5, 3, 3, 3, 2 )
q2      <- c( 5, 5, 5, NA, 2 )
q3      <- c( 5, 5, 2, NA, 1 )
df     <- data.frame( manager, date, country, gender, age, q1, q2, q3,
                      stringsAsFactors = FALSE )
df
```

```
  manager     date country gender age q1 q2 q3
1       1 10/11/08      US      M  NA  5  5  5
2       2 10/12/08      US      F  45  3  5  5
3       3 10/13/08      UK      F  25  3  5  2
4       4 10/14/08      UK   <NA>  39  3 NA NA
5       5 10/15/08      UK      F  99  2  2  1
```

	manager	date	country	gender	age	q1	q2	q3
1	1	10/11/08	US	M	NA	5	5	5
2	2	10/12/08	US	F	45	3	5	5
3	3	10/13/08	UK	F	25	3	5	2
4	4	10/14/08	UK	NA	39	3	NA	NA
5	5	10/15/08	UK	F	99	2	2	1

Figura 1.6: *Vista del navegador de objetos.*

1.4.1. Operadores aritméticos

Si queremos crear una nueva variable usando operadores aritméticos es tan sencillo como saberse los operadores.

Tabla 1.4: *Operadores aritméticos.*

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación

Operador	Descripción
*	Multiplicación
/	División
^ ó **	Potencias
%%	Módulo (x mod y) 5%%2 es 1
%/%	División entera 5%/%2 es 2

Así si queremos crear una variable q1mas2 que sea q1 + q2

```
q1masq2 <- q1 + q2
```

Un método interesante para crear nuevas variables y que automáticamente se incluyan en el *dataframe* es el empleo de la función `transform()`.

```
# creamos variables con la función transform
df <- transform( df,
                 sumx  = q1 + q2,
                 meanx = ( q1 + q2 ) / 2 )
df
```

```
  manager     date country gender age q1 q2 q3 sumx meanx
1       1 10/11/08      US      M  NA  5  5  5   10     5
2       2 10/12/08      US      F  45  3  5  5    8     4
3       3 10/13/08      UK      F  25  3  5  2    8     4
4       4 10/14/08      UK   <NA>  39  3 NA NA   NA    NA
5       5 10/15/08      UK      F  99  2  2  1    4     2
```

Otra forma de incluir variables automáticamente en un dataframe es definiendolas *al vuelo* como parte del mismo:

```
df$nueva <- df$q1 * 2
df
```

```
  manager     date country gender age q1 q2 q3 sumx meanx nueva
1       1 10/11/08      US      M  NA  5  5  5   10     5    10
2       2 10/12/08      US      F  45  3  5  5    8     4     6
3       3 10/13/08      UK      F  25  3  5  2    8     4     6
4       4 10/14/08      UK   <NA>  39  3 NA NA   NA    NA     6
```

```
5        5 10/15/08       UK       F  99  2  2  1    4      2      4
```

1.4.2. Recodificación de variables

Para recodificar variables es interesante hablar, en primer lugar, de los operadores lógicos, pues a menudo recodificamos según una regla lógica.

Tabla 1.5: *Operadores lógicos.*

Operador	Descripción
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Exactamente igual a
==	Exactamente igual a
!=	No igual a/que
!x	Diferente de x
x | y	x o y
x & y	x e y
isTRUE(x)	Evalúa si x es una expesión verdadera

```
# Creamos la variable agecat (categoría de edad),
# le asignamos el código 'anciano' si age>75,
# si está entre 44 y 77 le asignamos 'maduro' y
# si está° por debajo de 44 'joven'.
df$agecat[ df$age > 75 ]               <- "anciano"
df$agecat[ df$age <= 75 & df$age > 44 ] <- "maduro"
df$agecat[ df$age <= 44 ]              <- "joven"
df
```

```
  manager     date country gender age q1 q2 q3 sumx meanx nueva
1       1 10/11/08      US      M  NA  5  5  5   10     5    10
2       2 10/12/08      US      F  45  3  5  5    8     4     6
3       3 10/13/08      UK      F  25  3  5  2    8     4     6
```

```
4       4 10/14/08      UK    <NA>  39  3 NA NA   NA    NA     6
5       5 10/15/08      UK       F  99  2  2  1    4     2     4
   agecat
1    <NA>
2  maduro
3   joven
4   joven
5 anciano
```

La expresión `variable[ condición ] <- expresión` hará sólo la asignación cuando la condición sea TRUE.

1.4.3. Renombrar variables

Para renombrar variables lo más burdo sería recurrir a la función `fix( df )` (edición de la tabla) sustituyendo *"a mano"* los nombres. En contra se suele emplear el comando `rename`; en RStudio podemos editar los *data frames* clicando en la descripción del objeto, en la pestaña `Environment`.

```
# install.packages( 'reshape' ) # renombrar variables
library( reshape )

df <- rename( df,
    c(manager = "manID", date = "testdate") )

# Otra opción es con la función names(),
# sin necesidad de más paquetes que los básicos.
names( df )
```

```
 [1] "manID"    "testdate" "country"  "gender"   "age"      "q1"
 [7] "q2"       "q3"       "sumx"     "meanx"    "nueva"    "agecat"
```

```
names( df )[ 3 ] <- "pais"

# del mismo modo
names( df )[ 6:8 ] <- c( "it1", "it2", "it3" )
df
```

```
  manID testdate pais gender age it1 it2 it3 sumx meanx nueva  agecat
1     1 10/11/08   US      M  NA   5   5   5   10     5    10   <NA>
2     2 10/12/08   US      F  45   3   5   5    8     4     6 maduro
3     3 10/13/08   UK      F  25   3   5   2    8     4     6  joven
4     4 10/14/08   UK   <NA>  39   3  NA  NA   NA    NA     6  joven
5     5 10/15/08   UK      F  99   2   2   1    4     2     4 anciano
```

Nota: Fíjate en que hemos comentado la instalación del paquete, `install.packages()`, pues sólo es necesaria si no has instalado previamente la librería, y que la carga de la librería, `library()` sólo es necesaria cuando en la sesión de trabajo la necesitamos por primera vez.

1.4.4. Valores perdidos (missing values)

El tratamiento de datos perdidos, más conocidos como *missing* es fundamental para la correcta interpretación de un análisis.

- En R valores imposibles, *por ejemplo el resultando de* $\sqrt{-9}$, se representan por el código: `NaN` (*Not a Number*).
- Los valores perdidos (*simplemente, hemos perdido algún dato y no sabemos qué valor toma*) por el código: `NA` (*Not available*).

Hay muchas funciones para identificar estos valores, la más usada es `is.na()`.

```
# missings
y <- c( 1, 2, 3, NA )
is.na( y )  # ¿cuáles son NA?,
```

```
[1] FALSE FALSE FALSE  TRUE
```

```
# is.na() devuelve un objeto del mismo tamaño que el que recibe.
is.na( df[ , 1:10 ] )
```

```
     manID testdate  pais gender   age   it1   it2   it3  sumx meanx
[1,] FALSE    FALSE FALSE  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
[2,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
[3,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[4,] FALSE    FALSE FALSE   TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
[5,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

1.4.4. Recodificar valores a missing

Para recodificar valores faltantes podemos emplear las mismas técnicas de recodificación que ya hemos visto.

```
# recodificar
df$age[ is.na( df$age ) ] <- 99
# observa qué ocurre con 'is.na(df)'
is.na( df[ , 1:10 ] )
```

```
     manID testdate  pais gender   age   it1   it2   it3  sumx meanx
[1,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[2,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[3,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[4,] FALSE    FALSE FALSE   TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
[5,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
# ahora devolvemos el df a su estado original.
df$age[ df$age == 99 ] <- NA
is.na( df[ , 1:10 ] )
```

```
     manID testdate  pais gender   age   it1   it2   it3  sumx meanx
[1,] FALSE    FALSE FALSE  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
[2,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[3,] FALSE    FALSE FALSE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[4,] FALSE    FALSE FALSE   TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
[5,] FALSE    FALSE FALSE  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
```

1.4.4. Excluir valores missing del análisis

En muchas funciones se incorpora un argumento para excluir estos valores. Emplearemos la opción `na.rm = TRUE` para que **no** considere los valores faltantes.

```
# excluir los missings del análisis
x <- c( 1, 2, NA, 3 )
y <- x[ 1 ] + x[ 2 ] + x[ 3 ] + x[ 4 ]
z <- sum( x )
y
```

```
[1] NA
```

```
z
```

```
[1] NA
```

```
# ¡¡¡¡Ambos son NA!!!!
sum( x, na.rm = T )  # ¡Ahora no!
```

```
[1] 6
```

De una forma más general podemos emplear la función `na.omit()` que elimina cualquier fila de un *data frame* que tenga valores perdidos.

```
df <- na.omit( df )
df
```

```
  manID testdate pais gender age it1 it2 it3 sumx meanx nueva agecat
2     2 10/12/08   US      F  45   3   5   5    8     4     6 maduro
3     3 10/13/08   UK      F  25   3   5   2    8     4     6  joven
```

Nota: Si tenemos una tabla de datos con valores perdidos, pero trabajamos con un subconjunto de variables que no los presentan valores, no debemos eliminar *a priori* las observaciones incompletas, pues esto nos afectaría eliminando observaciones válidas.

1.4.5. Conversión de tipos

Muchas veces necesitamos que R entienda un número como un carácter, o viceversa. En R cuando añadimos a un vector numérico un elemento no numérico convierte todo el vector en no numérico automáticamente. Un resumen de las principales funciones de conversión de tipos puede verse en la tabla 1.6.

Lógico	Conversión
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`

Tabla 1.6: *Conversión de tipos.*

```
# conversión de tipos.
a <- c( 1, 2, 3 )
a
```

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

```
is.numeric( a )
```

```
[1] TRUE
```

```
is.vector( a )
```

```
[1] TRUE
```

```
a <- as.character( a )
a
```

```
[1] "1" "2" "3"
```

```
is.numeric( a )
```

```
[1] FALSE
```

```
is.vector( a )
```

```
[1] TRUE
```

1.4.6. Ordenar datos

Para ordenar un vector es suficiente con la función `sort`; Pero para ordenar las filas de un *data frame* empelamos la función `order()`, que nos genera un índice de orden, por defecto ordena ascendentemente. Empleamos el signo *'menos'* para cambiar el sentido.

```
df
```

```
  manID testdate pais gender age it1 it2 it3 sumx meanx nueva agecat
2     2 10/12/08   US      F  45   3   5   5    8     4     6 maduro
3     3 10/13/08   UK      F  25   3   5   2    8     4     6  joven
```

```
df_ordenado <- df[ order( df$age ),  ]
df_ordenado
```

```
  manID testdate pais gender age it1 it2 it3 sumx meanx nueva agecat
3     3 10/13/08   UK      F  25   3   5   2    8     4     6  joven
2     2 10/12/08   US      F  45   3   5   5    8     4     6 maduro
```

```
df_ordenado2 <- df[ order( df$gender , -df$age ),  ]
df_ordenado2
```

```
  manID testdate pais gender age it1 it2 it3 sumx meanx nueva agecat
2     2 10/12/08   US      F  45   3   5   5    8     4     6 maduro
3     3 10/13/08   UK      F  25   3   5   2    8     4     6  joven
```

1.4.7. Ampliar/unir conjuntos de datos

Podemos ampliar un *data frame* añadiendo variables (columnas) o casos (filas). Y añadir columnas con las funciones `merge()` y `cbind()`.

```
dfA <- df  # nos creamos un par de dataframes aunque sean iguales
dfB <- df
df.total <- merge( dfA, dfB,
                   by = "manID" )

df.total
```

```
  manID testdate.x pais.x gender.x age.x it1.x it2.x it3.x sumx.x
1     2   10/12/08     US        F    45     3     5     5      8
2     3   10/13/08     UK        F    25     3     5     2      8
  meanx.x nueva.x agecat.x testdate.y pais.y gender.y age.y it1.y
1       4       6   maduro   10/12/08     US        F    45     3
2       4       6    joven   10/13/08     UK        F    25     3
  it2.y it3.y sumx.y meanx.y nueva.y agecat.y
1     5     5      8       4       6   maduro
2     5     2      8       4       6    joven
```

```
# unir matrices
A        <- matrix( 1:9, 3, 3 )
B        <- matrix( 1:9, 3, 3 )
AB.total <- cbind( A, B )
AB.total
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7    1    4    7
[2,]    2    5    8    2    5    8
[3,]    3    6    9    3    6    9
```

```
# unir matrices
lA        <- matrix( c( "a", "b", "c", "d" ), 2, 2 )
lB        <- matrix( c( "A", "B", "C", "D" ), 2, 2 )
lAB.total <- cbind( A, B )
lAB.total
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7    1    4    7
[2,]    2    5    8    2    5    8
[3,]    3    6    9    3    6    9
```

Para unir filas (observaciones) empleamos la función `rbind(dfA, dfB)`.

Nota: Ambos *data frames* han de tener las mismas columnas.

```
df.total2 <- rbind( dfA, dfB )
df.total2
```

```
   manID testdate pais gender age it1 it2 it3 sumx meanx nueva agecat
```

```
2       2 10/12/08   US       F  45   3   5   5   8      4      6 maduro
3       3 10/13/08   UK       F  25   3   5   2   8      4      6  joven
21      2 10/12/08   US       F  45   3   5   5   8      4      6 maduro
31      3 10/13/08   UK       F  25   3   5   2   8      4      6  joven
```

1.4.8. Seleccionar subconjuntos de un data frame

Muy a menudo necesitamos efectuar operaciones sobre una parte de un conjunto de datos, ya sea de un subconjunto de variables o de un subconjunto de observaciones. R es muy ágil en estas operaciones y hay muchas formas de acceder a observaciones o variables concretas.

Usaremos los corchetes para acceder a partes de el dataframe.

```
data.frame[índice de fila, índice de columna]
```

Nota: la regla es *"filas por columnas"*.

1.4.8. Seleccionar variables (columnas)

```
nuevo_df<- df[ , índices]
```

Observar que dejamos en blanco la primera posición entre los corchetes para referirnos a la columnas

```
# Seleccionar columnas de un dataframe
df <- data.frame( manager, date, country, gender, age, q1, q2, q3,
                  stringsAsFactors = FALSE )
df
```

```
  manager     date country gender age q1 q2 q3
1       1 10/11/08      US      M  NA  5  5  5
2       2 10/12/08      US      F  45  3  5  5
3       3 10/13/08      UK      F  25  3  5  2
4       4 10/14/08      UK   <NA>  39  3 NA NA
5       5 10/15/08      UK      F  99  2  2  1
```

```
new_df <- df[ , c( 6:8 ) ]
new_df
```

```
  q1 q2 q3
1  5  5  5
2  3  5  5
3  3  5  2
4  3 NA NA
5  2  2  1
```

```
new_df2 <- df[ , "country" ]
new_df2    # por el nombre de la 'columna'
```

```
[1] "US" "US" "UK" "UK" "UK"
```

```
new_df3 <- df[ , 3 ]
new_df3    # por el índice
```

```
[1] "US" "US" "UK" "UK" "UK"
```

```
# aunque más generalmente si especificamos el nombre no hace falta tratar
# con los índices
new_df4 <- df[ "country" ]
new_df4    # por el nombre de la 'columna'
```

```
  country
1      US
2      US
3      UK
4      UK
5      UK
```

```
str( new_df4 )
```

```
'data.frame':   5 obs. of  1 variable:
 $ country: chr  "US" "US" "UK" "UK" ...
```

```
str( new_df3 )
```

```
 chr [1:5] "US" "US" "UK" "UK" "UK"
```

Igualmente podríamos haber construido un vector de chars.

```
variables <- c( "country", "date")
new_df    <- df[ , variables ]`
```

Seleccionar para eliminar: Atención al signo '-' delante del índice.

```
# eliminamos las variables q3 y q4, que tienen índices 8 y 9
new_df5 <- df[ c( -8, -9 ) ]
new_df5
```

```
  manager     date country gender age q1 q2
1       1 10/11/08      US      M  NA  5  5
2       2 10/12/08      US      F  45  3  5
3       3 10/13/08      UK      F  25  3  5
4       4 10/14/08      UK   <NA>  39  3 NA
5       5 10/15/08      UK      F  99  2  2
```

1.4.8. Seleccionar observaciones (filas)

Haremos uso de las mismas reglas que para las columnas pero dejando vacía la segunda posición del corchete y jugando con los operadores lógicos.

```
nuevo_df<- df[índices, ]
```

```
# Seleccionar observaciones
df
```

```
  manager     date country gender age q1 q2 q3
1       1 10/11/08      US      M  NA  5  5  5
2       2 10/12/08      US      F  45  3  5  5
3       3 10/13/08      UK      F  25  3  5  2
4       4 10/14/08      UK   <NA>  39  3 NA NA
5       5 10/15/08      UK      F  99  2  2  1
```

```
# seleccionar las 3 primeras filas
df6 <- df[ 1:3, ]
df6
```

```
  manager     date country gender age q1 q2 q3
```

```
1        1 10/11/08       US       M  NA  5  5  5
2        2 10/12/08       US       F  45  3  5  5
3        3 10/13/08       UK       F  25  3  5  2
```

```
df7 <- df[ which( df$gender == "F" & df$country == "UK" ),  ]
df7
```

```
  manager     date country gender age q1 q2 q3
3       3 10/13/08       UK       F  25  3  5  2
5       5 10/15/08       UK       F  99  2  2  1
```

Una magnifico texto de referencia se puede encontrar en el texto de Joseph Adler, *R in a nutshell*[5]

1.5. Funciones

No vamos a detenernos mucho en los detalles de las funciones, en todo el libro trabajaremos con muchas de ellas y sus particularidades, sólo decir que hay miles y miles, agrupadas en paquetes, y cada día hay más paquetes que aporta la comunidad y que se encuentran en CRAN como ya mencionamos al principio del texto. La página *Rdocumentation*, en www.rdocumentation.org a la hora de redactar este documento recogía entre los distintos repositorios de R un total de 1.757.257 funciones.

1.5.1. Funciones matemáticas

Para empezar sí debemos de saber que existen un buen número de funciones matemáticas estándar, su sintaxis es muy parecida a otros lenguajes de programación.
Algunas son:

Tabla 1.7: *Funciones matemáticas.*

Función	Devuelve
`abs(x)`	Valor absoluto de `x`, `abs(-4)` devuelve 4
`sqrt(x)`	Raíz cuadrada de `x`, `sqrt(25)` devuelve 5

Función	Devuelve
`ceiling(x)`	Entero más pequeño mayor que `x`, `ceiling(4.6)` devuelve 5
`floor(x)`	Entero más grande no mayor que `x`, `floor(4.6)` devuelve 4
`trunc(x)`	Truncamiento de `x`
`round(x, digits=n)`	Redondea `x` a un número específico de decimales, por defecto `n=0`
`log(x,base=n)`	Logaritmos (`log` devuelve el logaritmo natural, log(exp(1)) devuelve 1

1.5.2. Funciones estadísticas

Las funciones estadísticas más habituales se recogen en la tabla siguiente.

Tabla 1.8: *Funciones estadísticas*

Función	Devuelve
`mean(x)`	Media
`median(x)`	Mediana
`sd(x)`	Desviación estándar
`var(x)`	Varianza
`sum(x)`	Suma
`cumsum(x)`	Suma acumulada
`range(x)`	Rango
`min(x)`	Mínimo
`max(x)`	Máximo
`table(x)`	Tabla de frecuencias de repetición de los valores de `x`
`scale(x,center=TRUE, scale=TRUE)`	Estandarizar, devuelve una varialbe con $\bar{x} = 0$ y $s^2 = 1$

Ejercicio: Probar estas funciones con `x<-c( 1, 2, 3, 4, 5, 6, 7, 8, 9 )`

Existen también funciones de densidad de probabilidad, test estadísticos... que veremos más adelante.

Las funciones de caracteres (`strings`, `chars`), que buscan, sustituyen, cuentan caracteres en cadenas, etc. nos las vamos a saltar también porque no son *"importantes""* para nuestros objetivos.

1.5.3. Miscelánea de funciones interesantes

`length()`

```
# Miscelánea de funciones interesantes medir objetos
x <- c( 1, 2, 3, 4, 5, 6, 7, 8, 9 )
length( x )
```

```
[1] 9
```

`seq()`

```
# generar secuencias con saltos
seq( 1, 10, 2 )     # seq(from, to, by)
```

```
[1] 1 3 5 7 9
```

```
seq( 1, 20, 3 )
```

```
[1]  1  4  7 10 13 16 19
```

`rep()`, `sort()`

```
# repetir
rep( 1, 5 )    # rep(ob, número de repet)

rep( c( "A", "B" ), 5 )

rep( 1:3, 2 )

# series repetidas y ordenadas
sort( rep( 1:3, 2 ) )
```

```
    cat()

# concatenar
cat( "hola", "amigo" )
```

```
hola amigo
```

```
cat( "hola",
    "amigo",
    "\n",
    "¿cómo estas?",
    file = "fichero-cocatenado.txt" ) # escribimos en un fichero
```

1.5.4. Aplicando funciones a objetos

Una funcionalidad que resulta muy útil al trabajar con R es que **podemos aplicar funciones a una gran cantidad de objetos**: vectores, *arrays*, matrices, *data frames*...

```
# aplicar funciones a objetos 'complejos'
y <- c( 1.23, 4.56, 7.89 )
round( y )
```

```
[1] 1 5 8
```

```
z <- matrix( runif( 12 ), 3 )
z
```

```
      [,1]  [,2]   [,3]  [,4]
[1,] 0.262 0.510 0.2576 0.854
[2,] 0.165 0.924 0.0465 0.347
[3,] 0.322 0.511 0.4179 0.131
```

```
log( z )
```

```
      [,1]    [,2]   [,3]   [,4]
[1,] -1.34 -0.6731 -1.356 -0.158
[2,] -1.80 -0.0791 -3.069 -1.058
[3,] -1.13 -0.6715 -0.873 -2.029
```

```
mean( z )  # devuelve un escalar
```

```
[1] 0.396
```

1.5.5. Aplicando funciones a los elementos de los objetos (apply)

En el último ejemplo `mean(z)` devuelve un escalar cuando le habíamos proporcionado una `matriz`, ¿qué hacemos si queremos las medias por filas?

Para eso en R podemos emplear la función `apply()`. Si quieres saber más prueba `?apply`.

```
apply( z, 1, mean )  # medias de las filas
```

```
[1] 0.471 0.371 0.346
```

```
apply( z, 2, mean )  # medias de las columnas
```

```
[1] 0.250 0.648 0.241 0.444
```

`apply(x, MARGIN, FUN)`

- `FUN`: cualquier función
- `MARGIN`: índice de la dimensión(1=fila, 2=columna) a la que se la quieres aplicar

Nota: `apply()` es una potentísima herramienta; existe una familia de funciones sejemantes `sapply`, `lapply`, `tapply`...

1.5.6. Control de flujo

Sobre control de flujo no vamos a tratar nada en este libro, sólo mencionaremos que como en cualquier lenguaje de programación existe y es fundamental para un trabajo avanzado. Lo mencionamos por si te animas a explorarlo y sacarle partido.

Aunque a continuación seguiremos hablando de funciones, dada su *naturaleza habitual dentro de los lenguajes de programación* nos referiremos a ellas también con el nombre de sentencias.

1.5.6. Sentencia for

Genera lo que se denomina un bucle de evaluación. Evaluará una o varias expresiones hasta que la variable no esté contenida en una secuencia dada:

```
for ( variable in secuencia ) { expresión/ones }
```

```
# FOR
x <- 0
x
```

```
[1] 0
```

```
# Pueden obviarse las llaves al contener una sola expresión
for ( i in 1:3 ) { x <- x + 1 }
x
```

```
[1] 3
```

```
for ( i in 1:2 ) { print( "¡¡Viva el software libre!!" ) }
```

```
[1] "¡¡Viva el software libre!!"
[1] "¡¡Viva el software libre!!"
```

1.5.6. Sentencias while

Un bucle `while` evalúa expresiones mientras se cumpla una condición.

```
while (condición) sentencia
```

```
i <- 5
while ( i > 0 ) {
    print( "¡Perfecto mundo!" )
    i <- i - 1
}
```

```
[1] "¡Perfecto mundo!"
[1] "¡Perfecto mundo!"
[1] "¡Perfecto mundo!"
[1] "¡Perfecto mundo!"
[1] "¡Perfecto mundo!"
```

NOTA: En R, en la practica, `while` apenas se emplea siendo sustituidas por una combiandción `for` e `if`.

1.5.6. Sentencias if-else

Evalúa expresiones si se da una condición (`if`), si no se cumple evalúan expresión alternativa (`else`). La utilización de `else` es opcional.

```
if (condición) sentencia
ó
if (condición) sentencia1 else sentencia2
```

```
# IF-ELSE
if ( TRUE ) {
    print( "esta siempre se ejecuta" )
}

if ( FALSE ) {
    print( "esta nunca se ejecuta" )
    print( "pero nunca, nunca" )
} else {
    print( "¿lo ves?" )
    print( ";-)" )
}
```

Nota: Las sentencias `for`, `while` e `if-them` puede anidarse.

1.5.7. Funciones escritas por el usuario

Una de las mayores potencialidades de R es la posibilidad de escribir funciones *ad hoc*. Si además consideramos que estas se pueden empaquetar en

grupos y tener siempre disponibles para todas las sesiones nos podemos hacer una idea de lo potente y personalizable que puede llegar a ser R.

Un ejemplo muy sencillo:

```
# Definimos
euros <- function( pesetas ) pesetas/166.386

#Usamos
euros( 1000 )
```

La sintaxis, como vemos, es sencilla; partimos de un nombre, un procedimiento y un conjunto de argumentos:

```
mi_funcion <- function( arg1, arg2, ... ) {
    expresiones, para realizar el procedimiento
    return( objeto )
}

f.potencia <- function( num, exp = 2 ) {
    return( num * exp )
}
f.potencia( 5, 2 )

[1] 10

f.potencia( 5, 5 )

[1] 25

f.potencia( 5 )

[1] 10
```

Ejercicio: Crear una función que acepte dos argumentos, uno que sea un `char` con dos posibilidades `param` o `noparam`, que por defecto sea el valor "param". El segundo argumento que sea un vector numérico. Y que nos devuelva para "param", la media, la desviación típica y la varianza del vector. Y para "noparam", la mediana, los cuartiles 0.25 y 0.75, máximo y mínimo del vector.

Ejercicio: Crear una función que resuelva la ecuación de segundo grado. Y probarla para la ecuación $x^2 + 5x - 16 = 0$

```
# posible solución
e2grado <- function( a, b, c ) {
    discriminante <- ( b ^ 2 ) - ( 4 * a * c )
    solucion1 <- ( -b + sqrt( discriminante ) ) / ( 2 * a )
    solucion2 <- ( -b - sqrt( discriminante ) ) / ( 2 * a )
    c( solucion1, solucion2 )
}
e2grado( 1, 5, -16 )
```

```
[1]  2.22 -7.22
```

Podemos ahondar en muchos textos sobre la creación de funciones por el usuario, un texto sencillo es el de Emmanuel Paradis, *R para principiantes*[6], también el de R. Kabacoff, *R in Action*[7].

1.5.7. Cargar una función ad hoc al inicio de la sesión de R

Basta con crear un fichero que contenga la/s función/es o lo que deseemos calcular, a este fichero lo llamamos `script`. Lo cargamos al inicio de la sesión.

Nota: En Linux una forma sencilla de hacerlo es modificar el fichero `Rprofile.site` que por defecto se encuentra en `/usr/lib/R/etc/`. Modificaremos la función `.First` en `Rprofile.site`, cambiando única línea de código del fichero `.Firts`.

```
# .First <- function() cat("\n   Welcome to R!\n\n")
```

Por ejemplo para hacer que el *script* `fdescriptivos.r` se ejecute al inicio escribiremos:

```
.First <- function(){
 cat( "\n   Welcome to R!\n\n" )
 source( "/usr/lib/R/etc/fdescriptivos.R" )
                    }
```

2 Estadística descriptiva

Laura del Río Alonso

Repasamos en este capitulo, aunque muy superficialmente, conceptos estadísticos importantes; lo haremos de una manera intuitiva ya que nos será útil para encarar más adelante el concepto de contraste estadístico y *p-valor* más eficientemente.

2.1. Algunas definiciones

2.1.1. Población y muestra. Variables

- **Población**: Llamamos *población estadística*, *universo* o *colectivo* al conjunto de referencia sobre el cual van a recaer las observaciones.
- **Individuos**: Se llama *unidad estadística* o *individuo* a cada uno de los elementos que componen la población estadística. El individuo es un ente observable que no tiene por qué ser una persona, puede ser un objeto, un ser vivo, o incluso algo abstracto.

Cómo citar este capítulo: Del Rio, L. (2022). Estadística descriptiva. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (61-90). Editum. Ediciones de la Universidad de Murcia.

- **Muestra**: Es un subconjunto de elementos de la población. Se suelen tomar muestras cuando es difícil o costosa la observación de todos los elementos de la población estadística.
- **Censo**: Decimos que realizamos un censo cuando se observan todos los elementos de la población estadística.
- **Caracteres**: La observación del individuo la describimos mediante uno o más caracteres. El carácter es, por tanto, una cualidad o propiedad inherente en el individuo.

Tipos de caracteres:

- Cualitativos : aquellos que son categóricos, pero no son numéricos.
  - por ej. "color de los ojos", "profesión", "marca de coche"...
- Ordinales : aquellos que pueden ordenarse, pero no son numéricos.
  - por ej. "preguntas de encuesta sobre el grado de satisfacción de algo"
  - Mucho, poco, nada. Bueno, regular, malo...
- Cuantitativos : son numéricos.
  - por ej. "peso", "talla", "número de hijos", "número de libros leídos al mes"...

Modalidad o valor: Un carácter puede mostrar distintas modalidades o valores, es decir, son distintas manifestaciones o situaciones posibles que puede presentar un carácter estadístico. Las modalidades o valores son incompatibles y exhaustivos.

- Generalmente se utiliza el término *modalidad* cuando hablamos de caracteres cualitativos y el término *valor* cuando estudiamos caracteres cuantitativos.
- Por ejemplo: el carácter cualitativo "Estado Civil" puede adoptar las modalidades : casado, soltero, viudo. El carácter cuantitativo "Edad" puede tomar los valores : diez, once, doce, quince años...

Variable estadística: Al conjunto de los distintos valores numéricos que adopta un carácter cuantitativo se llama variable estadística.

Tipos de variables estadísticas :

- **Discretas**: Aquellas que toman valores aislados (números naturales), y que no pueden tomar ningún valor intermedio entre dos consecutivos fijados.
  - por ej. "núm. de goles marcados", "núm. de hijos", "núm. de discos comprados", "núm. de pulsaciones"...
- **Continuas**: Aquellas que toman infinitos valores (números reales) en un intervalo dado, de forma que pueden tomar cualquier valor intermedio, al menos teóricamente, en su rango de variación.
  - por ej. "talla", "peso", "presión sanguínea", "temperatura"...

Observación: Una observación es el conjunto de modalidades o valores de cada variable estadística medidos en un mismo individuo.

```
- p.ej. en una población de 100 individuos podemos estudiar,
  de forma individual, tres caracteres:

   - "edad : 18, 19, ...",
   - "sexo : Hombre, Mujer" y
   - "ha votado en las elecciones : Sí, No".

  Realizamos 100 observaciones con tres datos cada una, es decir,
  una de las observaciones puede ser (43, H, S).
```

2.1.2. Distribución de probabilidad y función de densidad de una variable aleatoria

Intuitivamente, una *variable aleatoria* puede tomarse como una cantidad cuyo valor no es fijo pero puede tomar diferentes valores; una *distribución de probabilidad* se usa para describir la probabilidad de que se den los diferentes valores (se denota usualmente por $F(x)$).

$$F_X(x) = P(x \leq X)$$

La distribución de probabilidad de una v.a. describe teóricamente la forma en que varían los resultados de un experimento aleatorio. Intuitivamente

se trataría de una lista de los resultados posibles de un experimento con las probabilidades que se esperarían ver asociadas con cada resultado.

La *función de densidad de probabilidad*, ***función de densidad***, o, simplemente, ***densidad de una variable aleatoria continua*** es una función, usualmente denominada $f(x)$ que describe la densidad de la probabilidad en cada punto del espacio de tal manera que la probabilidad de que la variable aleatoria tome un valor dentro de un determinado conjunto sea la integral de la función de densidad sobre dicho conjunto[8].

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

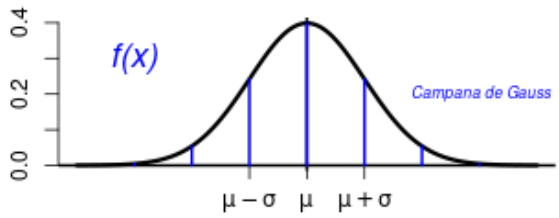


Figura 2.1: *Función de densidad de la distribución normal: campana de Gauss*

2.1.3. Parámetros y estadísticos

- **Parámetro**: Es una cantidad numérica calculada sobre una población.
  - La altura media de los individuos de un país
  - La idea es resumir toda la información que hay en la población en unos pocos números (parámetros).
- **Estadístico**: Ídem (cambiar población por muestra; es decir "Es una cantidad numérica calculada sobre una muestra".

- La altura media de los que estamos en este aula.
- ¿Somos una muestra (¿representativa?) de la población?

Si un estadístico se usa para aproximar un parámetro también se le suele llamar **estimador**.

2.2. Tipos de estadísticos

Los estadísticos se calculan, y estos estiman parámetros.

Hay diferentes tipos según las *cosas* que queramos saber de la distribución de una variable: posición, tendencia central, dispersión y forma de una distribución.

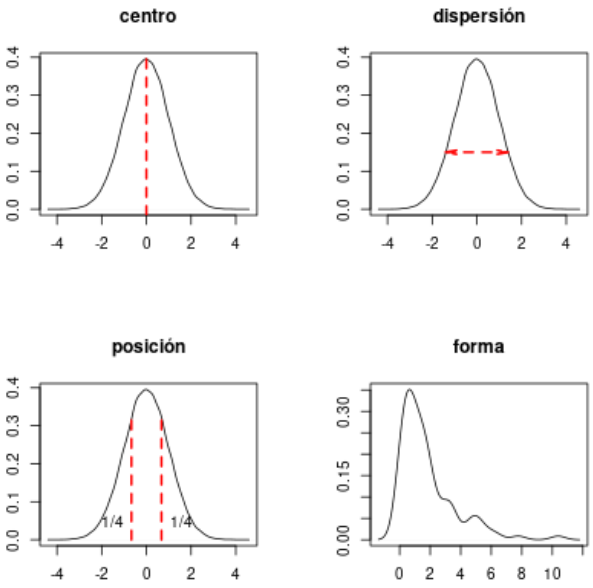


Figura 2.2: *Tipos de estadísticos.*

2.2.1. Medidas de posición

Dividen un conjunto ordenado de datos en grupos con la misma cantidad de individuos.

Las más populares: cuantiles, percentiles, cuartiles, deciles...

- Se define el cuantil de orden α como un valor de la variable por debajo del cual se encuentra una frecuencia acumuladada α.
- El percentil de orden k es el cuantil de orden $\frac{k}{100}$.

  Ejemplo: El 5% de los recién nacidos tiene un peso demasiado bajo. ¿Qué peso se considera "demasiado bajo"? Percentil 5 o cuantil 0.05.

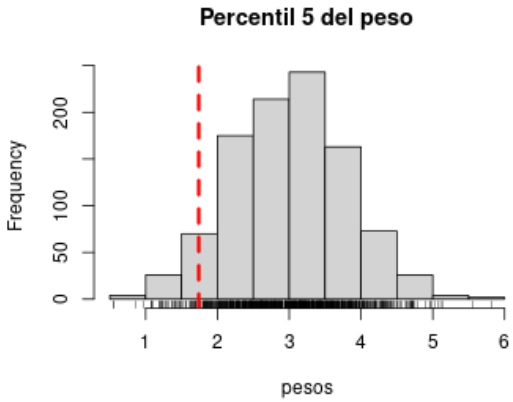


Figura 2.3: *Histograma con percentil 5.*

```
pesos <- rnorm( 1000, 3, 0.8 )
hist( pesos )

quantile( pesos, 0.05 )
```

```
  5%
1.71
```

Ejemplo: ¿Qué peso es superado sólo por el 25% de los individuos? Percentil 75, tercer cuartil o cuantil 0.75.

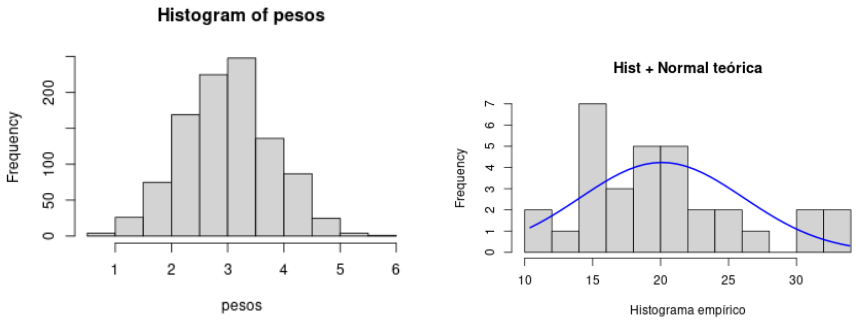


Figura 2.4: *a) Histograma para los datos de peso. b) Histograma y curva normal teórica asociada.*

```
quantile( pesos, 0.75 )
```

```
75%
3.5
```

2.2.2. Medidas de centralización o tendencia central

Indican valores con respecto a los que los datos 'parecen' agruparse.

- Media, mediana, moda...

Media

Media (*mean*). Es la media aritmética (promedio) de los valores de una variable. Suma de los valores dividido por el tamaño muestral.

```
mean( pesos )
```

```
[1] 3
```

```
x <- c( 1, 2, 3, 4, 5 )
mean(x)
```

```
[1] 3
```

```
media <- ( 1 + 2 + 3 + 4 + 5 ) / 5
```

```
media
```

```
[1] 3
```

- La media es conveniente cuando los datos se concentran simétricamente con respecto a ese valor. Es muy sensible a valores extremos (en estos casos hay otras 'medias', menos intuitivas pero que pueden ser útiles: media geométrica, ponderada...)

  La media representa el centro de gravedad de los datos.

Mediana

Mediana ('median'). Es un valor que divide a las observaciones en dos grupos con el mismo número de individuos (percentil 50). Si el número de datos es par, se elige la media de los dos datos centrales.

- Mediana de 1, 2, 4, **5**, 6, 6, 8 es 5
- Mediana de 1, 2, 4, **5, 6**, 6, 8, 9 es $\frac{5+6}{2} = 5.5$
- Es conveniente cuando los datos son asimétricos. No es sensible a valores extremos.
- Mediana de 1, 2, 4, 5, 6, 6, 800 es 5. ¡La media es 117, 7!

```
median( pesos )
```

```
[1] 3
```

```
x <- c( 1, 2, 4, 5, 6, 6, 8 )
median( x )
```

```
[1] 5
```

```
y <- c( 1, 2, 4, 5, 6, 6, 8, 9 )
z <- c( 1, 2, 4, 5, 6, 6, 800 )
median( z )
```

```
[1] 5
```

Moda

Es el valor donde la distribución de frecuencia alcanza un máximo. Puede haber más de uno (2 modas: *bimodal*, 3: *trimodal* ...)

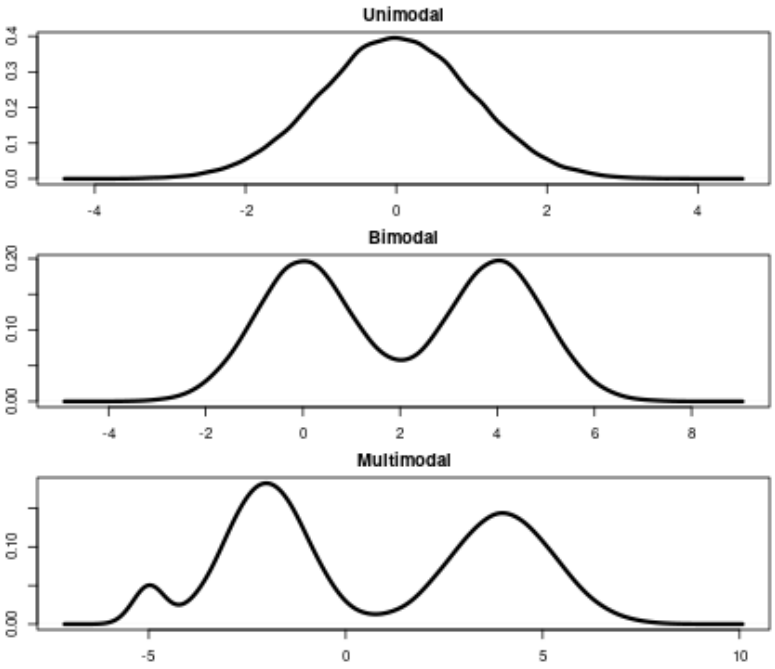


Figura 2.5: *Ejempo de distrbución unimodal, la moda está en el intervalo 4–4.5.*

```
# paquete "PrettyR" o crear una función
# install.packages( 'prettyR', dependencies = T )
library( "prettyR" )
help( package = "prettyR" )

# Otra forma, crearnos una función x discreta
modad <- function( x ) as.numeric( names( which.max( table( x ) ) ) )
# no funciona si hay más de una moda o si es constante!

modad( c( 1, 1, 3, 4, 5, 6, 6, 6) )
```

```
[1] 6
```

```
modac <- function( x ) {
    dd <- density( x )
    dd$x[ which.max( dd$y )]
}
modad( z )
```

```
[1] 6
```

Nota: Un inciso sobre el cálculo de medias

- Datos sin agrupar: $x_1, x_2, \ldots, x_7 : \bar{x} = \dfrac{\sum_i x_i}{n}$
- Datos organizados en tabla: Si está en intervalos usar como x_i las marcas de clase. Si no ignorar la columna de intervalos: $\bar{x} = \dfrac{\sum_i x_i n_i}{n}$

Tabla 2.1: *Datos organizados en tabla*

Intervalo	marca de clase	frecuencia
$L_0 - L_1$	x_1	n_1
$L_1 - L_2$	x_2	n_2
$L_2 - L_3$	x_3	n_3
$\ldots$	$\ldots$	$\ldots$
$L_{k-1} - L_k$	x_k	n_k

2.2.3. Medidas de dispersión

Indican la mayor o menor concentración de los datos con respecto a las medidas de centralización. Los más usuales son: la desviación típica, el coeficiente de variación, el rango, la varianza...

2.2.3. Fuentes de variabilidad

Imaginemos que los estudiantes de Bioestadística reciben diferentes calificaciones en la asignatura (variabilidad). ¿A qué puede deberse?

Lo que vemos es que hay diferencias individuales en el conocimiento de la materia. ¿Podría haber otras razones (fuentes de variabilidad)?
Supongamos, por ejemplo, que todos los alumnos poseen el mismo nivel de conocimiento. ¿Las notas serían las mismas en todos? Seguramente no.

Causas:

- Dormir poco el día del examen.
- Diferencias individuales en la habilidad para hacer un examen.
- El examen no es una medida perfecta del conocimiento.
- Variabilidad por error de medida.
- En alguna pregunta difícil, se duda entre varias opciones, y al azar se elige la mala.
- Variabilidad por azar, aleatoriedad.

2.2.3. Estadísticos de dispersión

- **Amplitud o Rango (*range*)** Diferencia entre observaciones extremas.
- **Rango intercuartílico** (*interquartile range*, IQR): Es la distancia entre primer y tercer cuartil.
  Rango intercuartílico: $P_{75} - P_{25} = Q_3 - Q_1$
- **Varianza** s^2
  Mide el promedio de las desviaciones (al cuadrado) de las observaciones con respecto a la media (*variance*):

- Es sensible a valores extremos (alejados de la media).
- Sus unidades son el cuadrado de las de la variable.

$$s^2 = \frac{1}{n}(x_i - \bar{x})^2$$

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} - 1$$

- **Desviación típica** (*standard deviation*, SD, sd...)
  Es la raíz cuadrada de la varianza.

  - Tiene las misma dimensionalidad (unidades) que la variable. Versión 'estética' de la varianza.

$$s = \sqrt{s^2}$$

- **Coeficiente de variación**
  Es la razón entre la desviación típica y la media.

  - Mide la desviación típica en forma de "qué tamaño tiene con respecto a la media"
  - También se la denomina variabilidad relativa.
  - Es una cantidad adimensional. Interesante para comparar la variabilidad de diferentes variables.
  - Se expresa como porcentaje.
  - Si el peso tiene CV=30% y la altura tiene CV=10%, los individuos presentan más dispersión en peso que en altura.

$$CV = \frac{s}{\bar{x}} \cdot 100$$

```
# Dispersión
pesos <- rnorm( 1000, 3, 0.8 )
range( pesos )
```

```
[1] 0.14 5.92
```

```
IQR <- ( quantile( pesos, 0.75, names = F ) -
         quantile( pesos, 0.25, names = F ) )
IQR
```

```
[1] 1.08
```

```
var( pesos )
```

```
[1] 0.667
```

```
sd( pesos )
```

```
[1] 0.816
```

```
CV <- sd( pesos ) / mean( pesos )
CV
```

```
[1] 0.272
```

2.2.4. Medidas de forma

Las medidas de forma son la *asimetría* y la *curtosis*.

- **Asimetría**
  - Las discrepancias entre las medidas de centralización son indicación de asimetría.
  - Coeficiente de asimetría (positiva o negativa).
  - Distribución simétrica (asimetría nula).
- **Curtosis o apuntamiento** La curtosis nos indica el grado de apuntamiento (aplastamiento) de una distribución con respecto a la distribución normal o gaussiana. Es una medida adimensional.
  - Platicúrtica (aplanada): curtosis < 0
  - Mesocúrtica (como la normal): curtosis= 0
  - Leptocúrtica (apuntada): curtosis > 0
- **Regla aproximativa** (para ambos estadísticos)
  - Curtosis o coeficiente de asimetría entre -1 y 1, es generalmente considerada una muy ligera desviación de la normalidad.
  - Entre -2 y 2 tampoco es malo del todo, según el caso.

2.2.5. Error típico de la media (SEM)

Error típico de la media: Cuando estimamos la media a partir de una muestra de un determinado tamaño (n) los valores que toma la media en las diferentes muestras varía. A la desviación típica de los valores que toma el estadístico se le denomina error típico de la media[9].

Da una idea de la variabilidad del estadístico.

Nota: No de la distribución de la variable.

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

$$s_{\bar{x}} = \frac{\widehat{s}}{\sqrt{n}}$$

Nota: Un 'error' muy frecuente de los investigadores es mostrar en los gráficos medias y errores típicos de la media en lugar de la desviación típica y, además, utilizar barras para representar los valores medios.

- Pero ¿es un error o es intencionado?
- ¿Por qué solemos mostrar gráficos de barras y errores típicos de la media (SEM)?

Martin Krzywinski y Naomi Altman hacen una interesante discusión sobre que barras de error emplear en Nature Methods *Points of significance* [10].

2.3. Gráficos útiles: un repaso rápido

2.3.0. Diagramas de cajas (boxplot)

Un diagrama de caja, según John Tukey (1977), es un gráfico, basado en cuartiles, mediante el cual se visualiza un conjunto de datos. Está compuesto por un rectángulo (*caja*) y dos brazos (*bigotes*); también reciben el nombre de *diagramas de cajas y bigotes*.

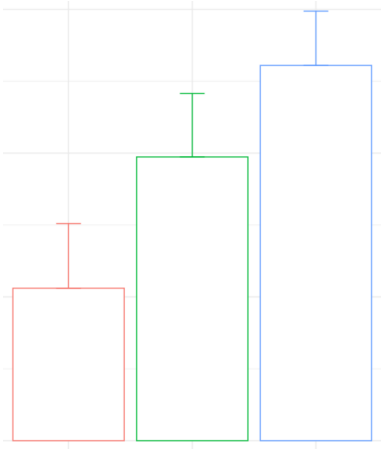


Figura 2.6: *Gráficos de medias y errores típicos.*

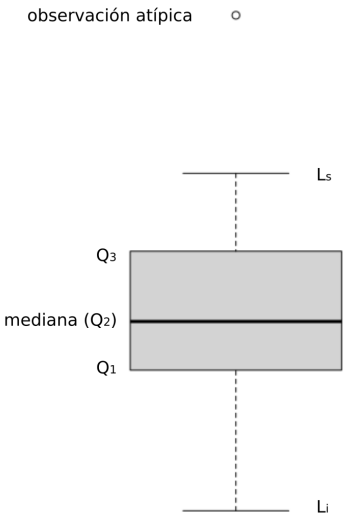


Figura 2.7: *Elementos de un* `boxplot`.

Es un gráfico que suministra información sobre los valores mínimo y máximo, los cuartiles Q1, Q2 o mediana y Q3, y sobre la existencia de valores atípicos y simetría de la distribución, ver figura 2.7.

$$\begin{aligned} L_i &= Q_1 - 1.5 \cdot IRQ \\ L_s &= Q_3 - 1.5 \cdot IRQ \\ IQR &= Q_3 - Q_1 \end{aligned}$$

Los valores atípicos son los inferiores a L_i y los superiores a L_s

```
boxplot( pesos )
```

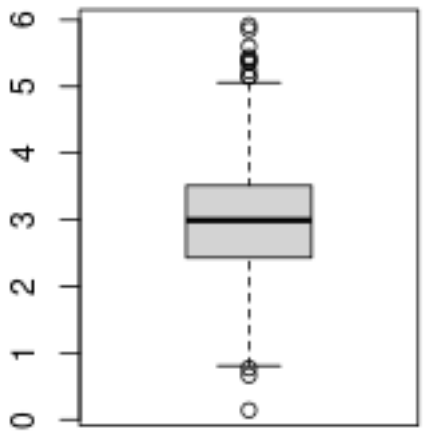


Figura 2.8: *Descripción de la variable* `peso` *mediante un* `barplot`.

```
p1 <- rnorm( 1000, 3, 0.8 )
p2 <- rnorm( 1000, 2, 0.5 )
p <- c(p1, p2)
```

```
grupo <- c( rep( "M", 1000 ), rep( "H", 1000 ) )
df <- data.frame( p, grupo )
str( df )
```

```
'data.frame':   2000 obs. of  2 variables:
 $ p    : num  3.49 2.68 3.84 3.48 3.81 ...
 $ grupo: chr  "M" "M" "M" "M" ...
```

```
head( df )
```

```
     p grupo
1 3.49     M
2 2.68     M
3 3.84     M
4 3.48     M
5 3.81     M
6 3.49     M
```

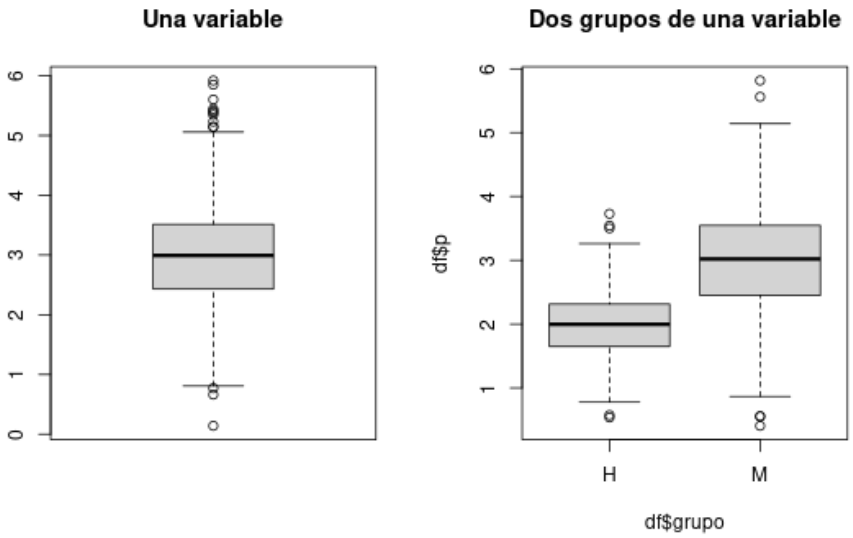


Figura 2.9: *Uso de* `boxplot` *para una variable sin y con la especificación de grupos definidos por otra variable adicional.*

```
par( mfrow = c( 1, 2 ), mar=c(5.1,4.1,4.1,2.1), oma=c(0,0,0,0) )
boxplot( pesos )
```

```
title( "Una variable" )
boxplot( df$p ~ df$grupo )
title( "Dos grupos de una variable" )
```

- Proporcionan una visión general de la simetría de la distribución de los datos, si la media no está en el centro del rectángulo, la distribución no es simétrica.
- Son útiles para ver la presencia de valores atípicos (*outliers*).
- Muy útiles para comparar distribuciones.

2.3.1. Histograma

Un histograma es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. Representar histogramas en R es tan sencillo como crear un objeto histograma, con la función `hist`.

Prueba el siguiente código y observa qué ocurre.

```
# histogramas

head( df )
```

```
     p grupo
1 3.49     M
2 2.68     M
3 3.84     M
4 3.48     M
5 3.81     M
6 3.49     M
```

```
str( df )
```

```
'data.frame':   2000 obs. of  2 variables:
 $ p    : num  3.49 2.68 3.84 3.48 3.81 ...
 $ grupo: chr  "M" "M" "M" "M" ...
```

```
par( mfrow = c( 3, 2 ), mar=c(5.1,4.1,4.1,2.1), oma=c(0,0,0,0) )
```

```
hist( df$p, main = "Por defecto" )

hist( df$p, breaks = 20, main = "Fijando el número de clases (20)" )

hist( df$p, breaks = 20, probability = TRUE,
      main = "Probabilidad en lugar de frecuencia" )
rug( jitter( p ) )  # aspecto: añadimos marcas en el
                    # eje OX de obs, con un poco de ruido

 # Personalizando el histograma
hist( df$p, breaks = 20, probability = TRUE,
      main = "Histograma de los pesos " )

hist( df$p, breaks = 20, probability = TRUE,
      main = "Histograma de los pesos (coloreado)",
      col = c("red", "yellow", "purple") )


colores <- c( "red", "yellow", "purple" )
hist( df$p,
      breaks = 20,
      probability = TRUE,
      xlab = "peso en kg",
      ylab = "frecuencia",
      col = colores,
      main = "Un histograma muy personal")

# le añadimos su curva de densidad
lines( density( df$p ), lwd = 2 )

# añadimos más cosas a la curva
lines( density(df$p),
       col = "blue",
       lty = 2,
       ps = 20 )
```

Nota: Los histogramas fueron inventados por Florence Nightingale

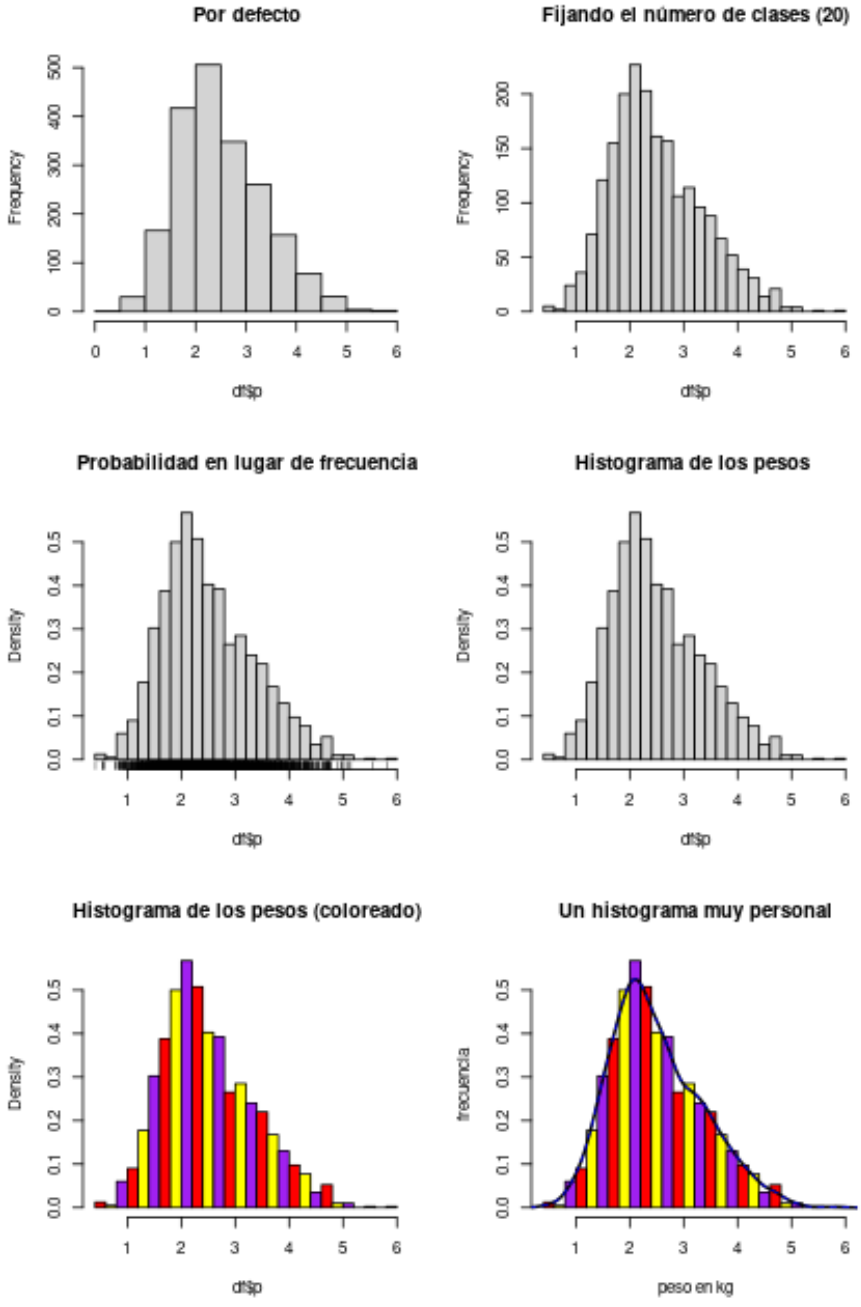


Figura 2.10: *Distintas formas de preparar un histograma.*

(1820-1910), Orden del Mérito del Reino Unido, británica, una de las pioneras en la práctica de la enfermería. Se la considera la madre de la enfermería moderna y creadora del primer modelo conceptual de enfermería. Destacó desde muy joven en matemática, aplicando después sus conocimientos de estadística a la epidemiología y a la estadística sanitaria. Inventó los gráficos de sectores o histogramas para exponer los resultados de sus reformas. En 1858, fue la primera mujer miembro de la *Statistical Society*. También fue miembro honorario de la *American Statistical Association*.
Durante la guerra de Secesión en 1861 fue llamada por el gobierno de la Unión para que organizara sus hospitales de campaña durante la guerra, la cual redujo del 44% al 2.2% los heridos que fallecían en los hospitales [11].

Un paquete muy interesante que nos permite combinar histogramas y *boxplots* es `sfsmisc` con la función `histBxp()`.

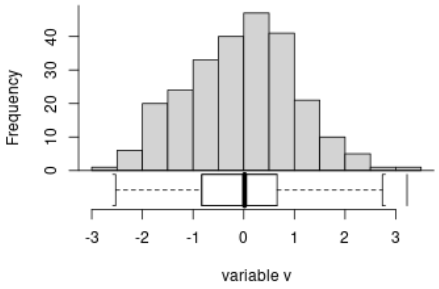


Figura 2.11: *Histograma combinado con boxplot.*

```
library( sfsmisc )
v <- rnorm( 250 )
histBxp( v,
         main = "",
         xlab = "variable v",
         probability = F,
         boxcol = 0,
         medcol = 1 )
```

2.4. Funciones útiles para obtener estadísticos descriptivos

2.4.0. summary()

Hemos presentado a lo largo del texto muchas funciones para obtener descriptivos, la más recurrida es `summary()`, pero hay muchas más y algunas no están en los paquetes por defecto. Veamos algunos ejemplos.

```
p1 <- rnorm( 1000, 3, 0.8 )
p2 <- rnorm( 1000, 2, 0.5 )
p <- c( p1, p2 )
altura <- c( rnorm( 1000, 87, 7 ), rnorm( 1000, 97, 6 ) )
# length(p); hist(p)
grupo <- c( rep( "M", 1000 ), rep( "H", 1000 ) )
df <- data.frame( p, altura, grupo )
head( df )
```

```
     p altura grupo
1 2.24   87.5     M
2 2.31   90.1     M
3 3.07  102.0     M
4 2.67   86.0     M
5 1.82   86.7     M
6 4.19   79.8     M
```

```
str( df )
```

```
'data.frame':   2000 obs. of  3 variables:
 $ p     : num  2.24 2.31 3.07 2.67 1.82 ...
 $ altura: num  87.5 90.1 102 86 86.7 ...
 $ grupo : chr  "M" "M" "M" "M" ...
```

```
summary( df )
```

```
       p            altura         grupo
 Min.   :0.50   Min.   : 67.0   Length:2000
 1st Qu.:1.88   1st Qu.: 86.1   Class :character
```

```
 Median :2.36   Median : 92.3   Mode  :character
 Mean   :2.49   Mean   : 92.0
 3rd Qu.:3.05   3rd Qu.: 97.9
 Max.   :5.30   Max.   :120.3
```

```
summary( df[ which( df$grupo == "M" ), ] )
```

```
       p            altura         grupo
 Min.   :0.55   Min.   : 67.0   Length:1000
 1st Qu.:2.45   1st Qu.: 82.2   Class :character
 Median :3.00   Median : 86.9   Mode  :character
 Mean   :2.99   Mean   : 86.9
 3rd Qu.:3.58   3rd Qu.: 91.5
 Max.   :5.30   Max.   :110.9
```

```
# también podemos asignar dataframes a cada grupo para simplificar la
# sintaxis
df.M <- df[ which( df$grupo == "M" ),  ]
summary( df.M )
```

```
       p            altura         grupo
 Min.   :0.55   Min.   : 67.0   Length:1000
 1st Qu.:2.45   1st Qu.: 82.2   Class :character
 Median :3.00   Median : 86.9   Mode  :character
 Mean   :2.99   Mean   : 86.9
 3rd Qu.:3.58   3rd Qu.: 91.5
 Max.   :5.30   Max.   :110.9
```

```
df.H <- df[ which( df$grupo == "H" ), ]
summary( df.H )
```

```
       p            altura         grupo
 Min.   :0.50   Min.   : 75.2   Length:1000
 1st Qu.:1.65   1st Qu.: 93.3   Class :character
 Median :1.99   Median : 97.1   Mode  :character
 Mean   :1.98   Mean   : 97.2
 3rd Qu.:2.31   3rd Qu.:101.3
 Max.   :3.45   Max.   :120.3
```

2.4.0. stat.desc() del paquete pastecs

La función `stat.desc()` del paquete `pastecs`, tiene varias opciones muy interesantes:

stat.desc(x, basic = TRUE, desc = TRUE, norm = FALSE, p=0.95)

- `basic`: nº de valores, nº de nulls, nº de missing, min, max, rango, suma...
- `desc`: mediana, media, SEM, IC(95%), var, sd, CV,
- `norm`: (OJO: no fijado en TRUE por defecto), curtosis, asimetría, Shapiro-Wilk

```
# del paquete pastecs, install.packages('pastecs')
# la función stat.desc()
library( "pastecs" )
```

```
Attaching package: 'pastecs'
```

```
The following object is masked from 'package:sfsmisc':

    last
```

```
stat.desc( df )
```

```
                     p      altura grupo
nbr.val       2000.0000   2000.0000    NA
nbr.null         0.0000      0.0000    NA
nbr.na           0.0000      0.0000    NA
min              0.5013     67.0228    NA
max              5.3044    120.3445    NA
range            4.8031     53.3217    NA
sum           4975.0423 184051.0781    NA
median           2.3650     92.3359    NA
mean             2.4875     92.0255    NA
SE.mean          0.0188      0.1876    NA
CI.mean.0.95     0.0368      0.3678    NA
var              0.7048     70.3520    NA
std.dev          0.8395      8.3876    NA
```

```
coef.var        0.3375      0.0911     NA
```

Nota: Sin incluir la variable *'grupo'* y con la opción `norm = T`.

```
stat.desc( df[ -3 ],
           norm = TRUE )
```

```
                     p       altura
nbr.val       2.00e+03   2000.00000
nbr.null      0.00e+00      0.00000
nbr.na        0.00e+00      0.00000
min           5.01e-01     67.02279
max           5.30e+00    120.34447
range         4.80e+00     53.32169
sum           4.98e+03 184051.07808
median        2.36e+00     92.33595
mean          2.49e+00     92.02554
SE.mean       1.88e-02      0.18755
CI.mean.0.95  3.68e-02      0.36782
var           7.05e-01     70.35205
std.dev       8.40e-01      8.38761
coef.var      3.37e-01      0.09114
skewness      4.86e-01     -0.11405
skew.2SE      4.44e+00     -1.04191
kurtosis     -1.83e-01     -0.27808
kurt.2SE     -8.37e-01     -1.27082
normtest.W    9.80e-01      0.99744
normtest.p    3.04e-16      0.00239
```

```
stat.desc( df.M[ -3 ],
           basic = FALSE,
           norm = TRUE )
```

```
                   p  altura
median        3.0002 86.8914
mean          2.9923 86.8600
SE.mean       0.0255  0.2228
CI.mean.0.95  0.0500  0.4372
var           0.6503 49.6467
```

```
std.dev       0.8064  7.0460
coef.var      0.2695  0.0811
skewness     -0.0480  0.1121
skew.2SE     -0.3103  0.7245
kurtosis     -0.2857  0.1051
kurt.2SE     -0.9244  0.3399
normtest.W    0.9979  0.9980
normtest.p    0.2320  0.2762
```

2.4.0. describe() de Hmisc

Hay más funciones que ofrecen descriptivos, por ejemplo `Hmisc` (y muchas más).

```
describe( df$p )
```

```
df$p
       n  missing distinct     Info     Mean      Gmd      .05
    2000        0     2000        1    2.488   0.9457    1.315
     .10      .25      .50      .75      .90      .95
   1.491    1.883    2.365    3.046    3.693    4.002

lowest : 0.501 0.504 0.548 0.626 0.628, highest: 5.020 5.063 5.149 5.205 5.304
```

```
head( df$p )
```

```
[1] 2.24 2.31 3.07 2.67 1.82 4.19
```

```
describe( df$altura )
```

```
df$altura
       n  missing distinct     Info     Mean      Gmd      .05
    2000        0     2000        1    92.03    9.522    77.76
     .10      .25      .50      .75      .90      .95
   80.98    86.11    92.34    97.89   102.95   105.20

lowest :  67.0  67.6  67.8  68.6  69.3, highest: 114.4 114.6 115.3 115.8 120.3
```

```
head( df$altura, 25 )
```

```
 [1]  87.5  90.1 102.0  86.0  86.7  79.8  98.5  78.9  85.5  85.4  85.1
```

```
[12]  96.5  91.0  95.5  88.1  87.0  81.0  96.1  98.7  90.4  70.6  90.3
[23]  91.1  85.1  86.6
```

2.4.1. Función tapply()

Con la función `tapply()` nos podemos construir fácilmente nuestras tablas de descriptivos de una forma muy elegante.

```
# tapply
tapply( df$p, df$g, mean )
```

```
   H    M
1.98 2.99
```

```
m  <- tapply( df$p, df$g, mean )
s  <- tapply( df$p, df$g, sd )
m2 <- tapply( df$p, df$g, median )
n  <- tapply( df$p, df$g, length )
cbind( media = m,
       sd = s,
       mediana = m2,
       n )
```

```
  media    sd mediana    n
H  1.98 0.500    1.99 1000
M  2.99 0.806    3.00 1000
```

No es difícil construir una función para estadísticos descriptivos *ad hoc* haciendo uso de la función `tapply()`.

```
f.descriptivos <- function ( variable, factor ) {
    df <- na.omit( data.frame(variable, factor ) )
    m  <- tapply( df$variable , df$factor, mean )
    s  <- tapply( df$variable , df$factor, sd )
    me <- tapply( df$variable, df$factor, median )
    n  <- tapply( df$variable , df$factor, length )
    cbind( media = m, sd = s, mediana = me, n )
}
```

2.4.2. Tablas de frecuencias y probabilidades

Esto lo veremos más extensamente cuando hablemos de contrastes no paramétricos

```
pais <- c( "ES", "ES", "ES", "US", "US" )
sexo <- c( "F", "F", "M", "F", "M" )

t <- table( pais, sexo )  # tabla de frecuencias absolutas
t
```

```
    sexo
pais F M
  ES 2 1
  US 1 1
```

```
# frec relativas
prop.table( t )  # porcentajes totales
```

```
    sexo
pais   F   M
  ES 0.4 0.2
  US 0.2 0.2
```

```
prop.table( t ) * 100
```

```
    sexo
pais  F  M
  ES 40 20
  US 20 20
```

```
# porcentajes por filas
prop.table( t, 1 )
```

```
    sexo
pais     F     M
  ES 0.667 0.333
  US 0.500 0.500
```

```
# porcentajes por columnas
```

```
prop.table( t, 2 )
```

```
    sexo
pais     F     M
  ES 0.667 0.500
  US 0.333 0.500
```

3 Introducción a los contrastes

Antonio Maurandi López

3.1. Población y Muestra

La meta más simple y general que se plantea a la hora de analizar datos es *"sacar las conclusiones más fuertes con la cantidad limitada de datos de los que disponemos"*[12].

Se nos planetan dos problemas a la hora de abordar esta tarea:

- Diferencias importantes pueden ser oscurecidas por *variabilidad biológica* e *imprecisión experimental*. Se hace complicado distinguir entre *diferencias reales y variabilidad aleatoria.*
- El ser humano es capaz de encontrar modelos. Nuestra inclinación natural (especialmente con los datos observados mediante nuestra experiencia propia) es concluir que las diferencias observadas son ***REALES***: *tendemos a minimizar los efectos de la variabilidad aleatoria*. Como se dicen Martin Krzywinski y Naomi Altman en la columna de Nature Methods *"Points of*

Cómo citar este capítulo: Maurandi-López, A. (2022). Introducción a los contrastes. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (91-132). Editum. Ediciones de la Universidad de Murcia.

significance": *somos de forma natural buscadores de patrones y debemos reconocer los límites de nuestra intuición* [13].

- El rigor estadístico nos previene de cometer estos errores. (*"si lo empleamos bien"*).

Un ejemplo: Los estudiantes de Bioestadística reciben diferentes calificaciones en la asignatura (variabilidad). *¿A qué puede deberse?*

Podría deberse a *diferencias individuales en el conocimiento de la materia*, ¿no?, ¿podría haber otras razones? (*otras fuentes de variabilidad*).

Por ejemplo supongamos que todos los alumnos poseen el mismo nivel de conocimiento. ¿Las notas serían las mismas en todos?: Seguramente No.

- Dormir poco el día del examen.
  - Diferencias individuales en la habilidad para hacer un examen.
- El examen no es una medida perfecta del conocimiento.
  - Variabilidad por error de medida.
- En alguna pregunta difícil, se duda entre varias opciones, y al azar se elige la mala.
  - Variabilidad por azar, aleatoriedad.

```
La idea básica de las estadística es extrapolar,
desde los datos recogidos, para llegar a conclusiones
más generales sobre la población de la que se han
recogido los datos.
```

Los estadísticos han desarrollado métodos basados en un modelo simple:

- Si razonablemente asumimos que los datos han sido obtenidos mediante un muestreo aleatorio de una población infinita. Analizamos estos datos y hacemos inferencias sobre la población.

Pero no siempre es *tan ideal.* En un experimento típico no siempre tomamos una muestra de una población, pero queremos extrapolar desde nuestra muestra a una situación más general. En esta situación aún podemos usar el concepto de población y muestra si definimos la **muestra** como los *"datos recogidos"* y la **población** como *"los datos que habríamos recogido si repitiéramos el experimento un número infinito de veces"* [14].

No sólo es necesario que los datos provengan de una población. También es necesario que cada sujeto, (cada observación) sea 'escogido' independientemente del resto.

Ejemplos:

- Si realizas un experimento biomédico 3 veces, y cada vez por triplicado, no tenemos 9 valores independientes. Si promediamos los triplicados, entonces tenemos 3 valores medios independientes.
- Si en un estudio clínico muestreamos 10 pacientes de una clínica y otros 10 de un Hospital: no hemos muestreado 20 individuos independientes de la población. Probablemente hemos muestreado dos poblaciones distintas.

Hay tres enfoques básicos.

- El primer método consiste en asumir que la población sigue un distribución especial conocida como por ejemplo la Normal o Gaussiana (campana de Gauss).
  - Los tests te permiten hacer inferencias sobre la media (y también sobre otras propiedades).
  - Los tests más conocidos pertenecen a este enfoque.
  - También se conoce como enfoque **paramétrico**.
- El segundo enfoque consiste en ordenar los valores de mayor a menor (rangos) y comparar distribuciones de rangos.
  - Es el principio básico de los **tests no-paramétricos.**
- El tercer enfoque es conocido como **'Resampling'** (se escapa de los objetivos de este libro, aquí también incluiríamos los métodos conocidos como *bootstraping*).
- Incluso hay un cuarto enfoque: **Métodos Bayesianos** (que también se escapan de los objetivos de este libro).
  Ya hemos dicho que la idea básica de las estadística es extrapolar desde los datos recogidos para llegar a conclusiones más generales sobre la población de la que proceden los datos. El problema es que sólo podemos aplicar las inferencias a la población de la que hemos obtenido las muestras y *'generalmente'* queremos ir más lejos. Desafortunadamente la estadística no nos

puede ayudar con estas extrapolaciones. En este caso se necesita **juicio científico y sentido común** para hacer inferencias más allá de las limitaciones de la estadística. Así *el análisis estadístico es sólo parte de la interpretación de nuestros datos* [14].

3.2. La distribución Normal

La distribución normal tiene características matemáticas especiales que la hacen la base de la mayoría de los test estadísticos. La razón: el Teorema del Límite Central (TCL) que veremos más adelante [15].

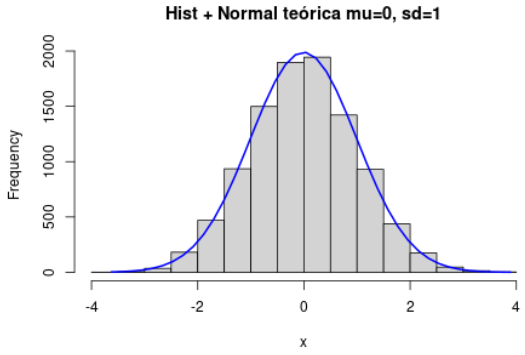


Figura 3.1: *Histograma y curva teórica normal.*

$$\int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}} dt$$

¿Qué cosas observamos en esta distribución?

- La distribución de probabilidad normal es simétrica alrededor de su media. **(Coef simetría=0)**
- **media=mediana=moda** (todas las medidas de tendencia central coinciden).
- La curva normal desciende suavemente en ambas direcciones a partir del valor central.
- Es **asintótica**, lo que quiere decir que la curva se acerca cada vez más al eje X pero jamás llega a tocarlo. Es decir, las "colas" de la curva se extienden de manera indefinida en ambas direcciones.
- **Coef. Kurtosis = 0**
- El área total encerrada por f(x) vale 1: $\int_{-\infty}^{+\infty} f(x)dx = P(-\infty < X < +\infty) = 1$
- Si X es una v.a. que se distribuye según una normal de media μ y sd σ, $Z = \frac{x-\mu}{\sigma}$ se distribuye como una normal tipificada, es decir, de media 0 y sd 1.

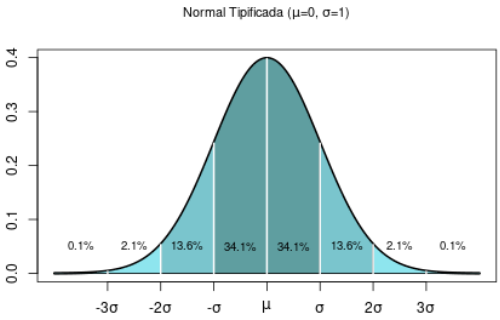


Figura 3.2: *Curva normal tipificada.*

3.2.1. TCL: Teorema del Límite Central

El Teorema del limite central se puede enunciar de la siguiente manera [16]:

`Si tus muestras son suficientemente grandes, la distribución de las medias seguirá una distribución Normal con la misma media que la distribución original.`

y varianza σ^2/n

De una forma más sencilla...

1. Dada una población con una distribución cualquiera.
2. Aleatoriamente obtenemos varias muestras de esa población. Calculamos sus medias.
3. Construimos un histograma de la distribución de frecuencias de las medias (¡ojo! *de las medias*)
4. Esta distribución de medias sigue una distribución 'Normal'.

¿Qué significa suficientemente grandes?: Depende de cuán distinta sea la distribución de una normal.

Así pues, el teorema del límite central *garantiza* una distribución normal cuando n es suficientemente grande.

Existen diferentes versiones del teorema, en función de las condiciones utilizadas para asegurar la convergencia. Una de las más simples establece que es suficiente que las variables que se suman sean independientes, idénticamente distribuidas, con valor esperado y varianza finitas.

Nota: La aproximación entre las dos distribuciones es, en general, mayor en el centro de las mismas que en sus extremos o colas, motivo por el cual es preferible el nombre "*teorema del límite central*", a la también usual denominacion como "*teorema central del límite*", ya que "*central*" califica al límite, más que al teorema.

```
# simulación del TCL (aml!)
# b<-rbinom(1000,10,0.25);hist(b)
# b<-rnorm(1000)
# b<-runif(100,0,1);hist(b)
# b<-rexp(100,1/10);hist(b)
b<-rchisq(100,5);hist(b)
plot(density(b))

N1<-1000        # el número de repeticiones debe ser alto
N2<-15          #tamaño muestras. Probar con 2, 5, 15, 30,50
vectordemedias<-NA
for (i in 1:N1){
    vectordemedias[i]=mean(sample(b,N2, replace=T))
  }

#vectordemedias
hist(vectordemedias,breaks=20) # obvs si parece una normal
SEM<-sd(vectordemedias);SEM#ind calidad estimación media
mean(vectordemedias)
mean(b)  #observad que difieren poco
```

Ver el vídeo del profesor F. J. Barón López de la Universidad de Málaga: [http://www.youtube.com/watch?v=FcDcJnw00hk]

El TCL nos da una idea de lo importante que es en estadística la distribución normal o gaussiana, ya que muestreando, con muestras suficientemente grandes, podemos estimar la media *bastante bien* (dependerá del error típico de la media, es

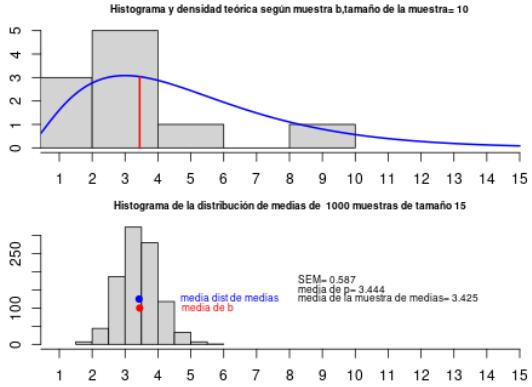


Figura 3.3: *Simulación TCL.*

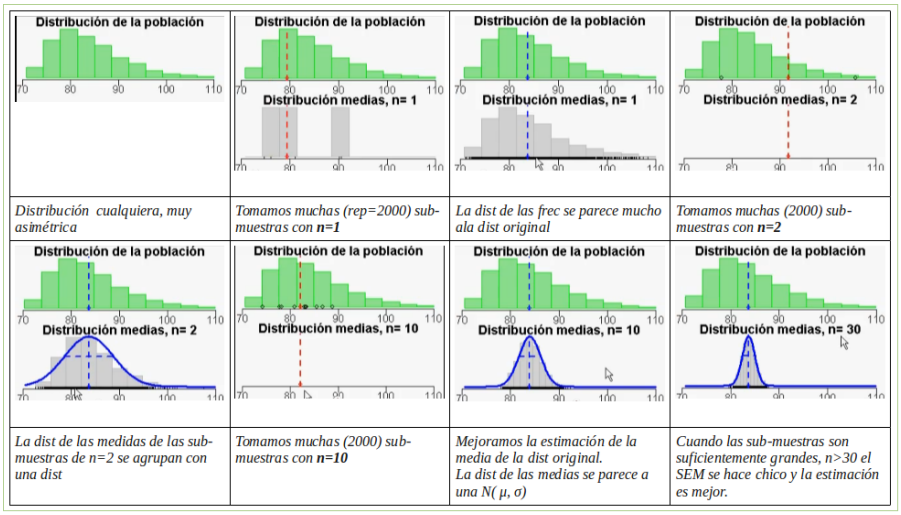


Figura 3.4: *Simulación TCL y tamaño muestral (JF.J Braón-López, 2011).*

decir, la precisión de la estimación de la media, hecha con una muestra, depende del tamaño de la muestra y de la desviación típica de la misma, a mayor muestras menor error típico de la media, *mayor precisión*), como sabemos que la distribución de las medias sigue una distribución normal, esto nos puede dar una idea de lo importante que es esta distribución [17].

¿Cómo de grande ha de ser la muestra?, pues tan grande como quiera que sea de pequeño el error típico de la media, es decir, va a depender de la desviación típica de los valores de la muestra. Así que no hay una *receta mágica*, esta pregunta enlaza con el concepto de potencia estadística que trataremos mas adelante.

Veremos como el TCL es el que nos permite calcular intervalos de confianza, p-valores (contrastes) y tamaños muestrales.

¿Cómo evaluamos la normalidad de una distribución?: Hay muchas formas de abordar este problema, pero básicamente hay dos estrategias:

- **Gráficamente:** mediante histogramas, Q-Q plots...
- **Analíticamente:** con contrastes de hipótesis de normalidad: test de Shapiro-Wilk, de Kolmogorov–Smirnov, de Lilliefors, de Jaques-Bera...

Lo veremos en detalle más adelante.

3.3. Intervalos de confianza

Ya hemos visto que la media que calculamos de una muestra probablemente no sea igual a la media de la población. El tamaño de la diferencia dependerá del tamaño y de la variabilidad de la muestra (n, σ). Basándonos en el TCL, combinamos estos dos factores: **tamaño de la muestra** (n) y **variabilidad** (σ), para calcular intervalos de confianza a un determinado nivel de confianza; generalmente *95 %*.

IC(95 %) ~ n, σ

Si asumimos que tenemos una muestra aleatoria de una población, podemos estar seguros al 95 % de que el IC (95 %) contiene a la media poblacional. Dicho de otra manera: si generamos 100 intervalos de confianza al 95 % de una población, se espera que contengan la media poblacional en 95 casos y no lo hagan en 5. No sabemos cuál es la media poblacional, así que no sabemos cuándo ocurre esto.

Cómo los calculamos:

$$(\overline{x} - z_{\alpha/2}\frac{S}{\sqrt{n}}, \overline{x} + z_{\alpha/2}\frac{S}{\sqrt{n}})$$

$z_{\sigma/2}$ para la normal es muy próximo a 1.96, en R lo calculamos con la función: `qnorm(0.975)`.

```
# Intervalos de confianza para una Gaussiana
media <- 5
sd    <- 2
n     <- 20

inf<-NA; sup<-NA
ic    <- data.frame(inf,sup)

error <- qnorm(0.975) * sd / sqrt(n)

ic[1,1] <- media - error
ic[1,2] <- media + error
ic
```

```
inf  sup
1 4.12 5.88
```

$$\left.\begin{matrix}(x_1, x_2, \cdots, x_n) \\ X \equiv D(\mu, \sigma)\end{matrix}\right\} \quad \Rightarrow \begin{cases} n \gg;\ \bar{X} \equiv N(\mu, \sigma^2/n) \\ n \ll;\ \bar{X} \equiv T_{n-1} \end{cases} \tag{3.1}$$

$$\frac{\bar{X} - \mu}{S/\sqrt{n}} \equiv t_{n-1}$$

```
# Intervalos de confianza para una T
media <- 5
sd    <- 2
n     <- 20

inf<-NA; sup<-NA
ic    <- data.frame(inf,sup)
```

```
error <- qt(0.975, df=n-1) * sd / sqrt(n)

ic[1,1] <- media - error
ic[1,2] <- media + error
ic
```

```
inf  sup
1 4.06 5.94
```

Podemos también hacer uso de la función `t.test()` para calcular IC.

```
# IC para una media de una muestra
 x <- rnorm( n=20 , mean=5 , sd=2 )
t.test( x )
```

```
One Sample t-test

data:  x
t = 17, df = 19, p-value = 6e-13
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
4.61 5.90
sample estimates:
mean of x
5.25
```

En R para crear intervalos de confianza para proporciones empleamos la función `prop.test()`.

```
# IC para una proporción
# 42 de 100 dijeron que sí,
prop.test( 42, 100 )
```

```
1-sample proportions test with continuity correction

data:  42 out of 100, null probability 0.5
X-squared = 2, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
```

```
0.323 0.523
sample estimates:
p
0.42
```

Para la mediana podemos calcularlo empleando la función `wilcox.test()`.

```
x <- c( 110, 12, 2.5, 98, 1017, 540, 54, 4.2, 150, 432)
mean( x )
```

```
[1] 242
```

```
median( x )
```

```
[1] 104
```

```
wilcox.test( x, conf.int=T )
```

```
Wilcoxon signed rank exact test

data:  x
V = 55, p-value = 0.002
alternative hypothesis: true location is not equal to 0
95 percent confidence interval:
33 514
sample estimates:
(pseudo)median
150
```

¿Qué asumimos cuando interpretamos un IC para una media?:

- Muestreo aleatorio de una población.
- La población se distribuye, aproximadamente, como una Normal.
  - Si n es grande no es tan importante está condición.
- Existe independencia de las observaciones.

Nota: Si se viola alguna de estas condiciones, el IC real es más 'ancho' que el calculado.

Así pues un intervalo de confianza es un rango de valores (calculado en una muestra) en el cual se encuentra el verdadero valor del parámetro, con una

probabilidad determinada.

- Nivel de confianza $1-\alpha$: probabilidad de que el verdadero valor del parámetro se encuentre en el intervalo.
- Nivel de significación α: probabilidad de equivocarnos.
- Normalmente $1-\alpha = 95\,\%$ ($\alpha = 5\,\%$)

```
set.seed(30)
n  <- 30
mu <- 5
s  <- 2
Ns <- 100               # tomamos Ns samples
muestras = matrix( rnorm(Ns * n, mu, s),  n )

fci <- function(x) {
t.test(x)$conf.int
}
conf.int = apply(muestras, 2, fci)

plot( range(conf.int)
      , c(0, 1 + Ns)
      , type = "n"
      , xlab = "ICs"
      , ylab = "Número de muestra"
      )
for (i in 1:Ns) {
if ((conf.int[1, i]<=mu && conf.int[2, i]<=mu)|(conf.int[1, i]>=mu && conf.int[2, i]>=mu)) {
lines(conf.int[, i], rep(i, 2), lwd = 2, col="red")
    } else lines(conf.int[, i], rep(i, 2), lwd = 2)

}
abline(v = mu, lwd = 2, lty = 2, col="blue")
cumplen <- sum( conf.int[1, ] <= mu & conf.int[2, ] >= mu)
text(mu-0.7*s, 40, paste(100-cumplen,"/100"), col="red")
```

3.4. P-valor. Contrastes de hipótesis

Observar medias muestrales diferentes no es suficiente evidencia de que sean diferentes las medias poblacionales. Es posible que sean iguales y que la diferencia observada se deba a una coincidencia, nunca se puede estar seguro. Lo único que se puede hacer es **calcular las probabilidades de equivocarnos**.

"Statistics does not tell us wheter we are rigth. It tells us the chances of beeing wrong"[13]

Si las poblaciones tienen la misma media realmente: ¿cuál es la probabilidad

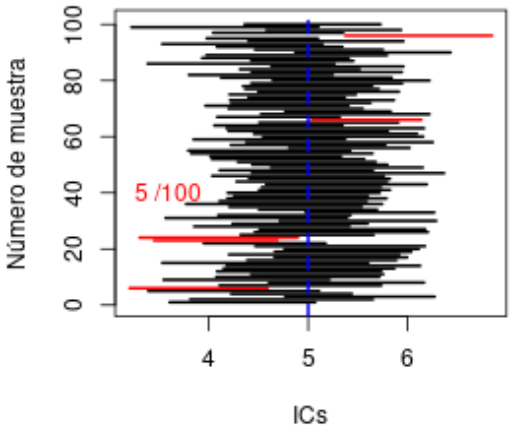


Figura 3.5: *Intervalos de Confianza que no contienen a la media.*

de observar una diferencia tan grande o mayor entre las medias muestrales?

- El p-valor es una probabilidad de 0 a 1.
- Si p es pequeño, podemos concluir que la diferencia entre muestras ('probablemente') no es debida al azar.
- Concluiríamos que tiene distintas medias.

Otra forma de ver lo mismo.

- Los estadísticos hablan de hipótesis nula (H_0).
- La hipótesis nula dice que "no hay diferencia entre las medias".
- La hipótesis nula es lo contrario que la hipótesis experimental.
- Así podemos definir **p-valor** como la ***probabilidad de observar una diferencia tan grande o mayor que la observada si la hipótesis nula fuera cierta***.

Así pues:

- Se dice que rechazamos la hipótesis nula si $p < 0.05$ y la diferencia es estadísticamente significativa (ss).
- Si $p > 0.05$, **no rechazamos la hipótesis nula** y decimos que la diferencia no es estadísticamente significativa (ns).

- No podemos decir que la H_0 sea verdad, simplemente "no la rechazamos", es decir, no tenemos suficiente evidencia para rechazar la hipótesis de igualdad (la H_0).

¿En qué se basa el cálculo del p-valor? Pues en el TCL y en las propiedades de la distribución normal, la cual tenemos tabulada.

Se puede también abordar el tema de los contrastes de hipótesis y cálculo del p-valor desde una perspectiva integradora con el concepto de intervalo de confianza [18].

Supongamos que deseamos hacer un contraste acerca de *un parámetro q, de la población*. Para llevarlo a cabo consideraremos la distribución de algún estadístico muestral que de alguna manera se corresponda y se relacione con el parámetro q. Designemos en general a este estadístico como T. *Si con los datos muestrales obtenemos un valor concreto para T tal que pertenezca a una determinada región del campo de variación de T optaremos por no rechazar la hipótesis y en caso contrario por rechazarla*. Obviamente la clave del problema será delimitar la región del campo de variación de T que consideraremos como zona de aceptación de la hipótesis. *Esto se resolverá por un criterio probabílistico partiendo de la distribución muestral de ese estadístico T.* Un esquema de decisión se puede ver en la figura 3.7

Región crítica: Región del campo de variación del estadístico tal que si contiene al valor evaluado del mismo con los datos muestrales nos llevará a rechazar la hipótesis. La designaremos por R_1.

Región de aceptación: Es la región complementaria a la región crítica. Si el valor evaluado del estadístico pertenece a ella: no rechazamos la hipótesis. La designaremos por R_0. Ambos conjuntos/regiones son disjuntos.

NOTA: La región de aceptación no es más que un intervalo de confianza al 95% (si la significación es 0.05) del estadístico T.

Un ejemplo intuitivo: *Supongamos que un fabricante de coches dice que un determinado vehículo recorre 25m por galón de gasolina, 25mpg. El consumidor pregunta a 10 usuarios su consumo y en media le responden que gasta 22mpg con una sd de 1,5. ¿Nos engaña el fabricante?*

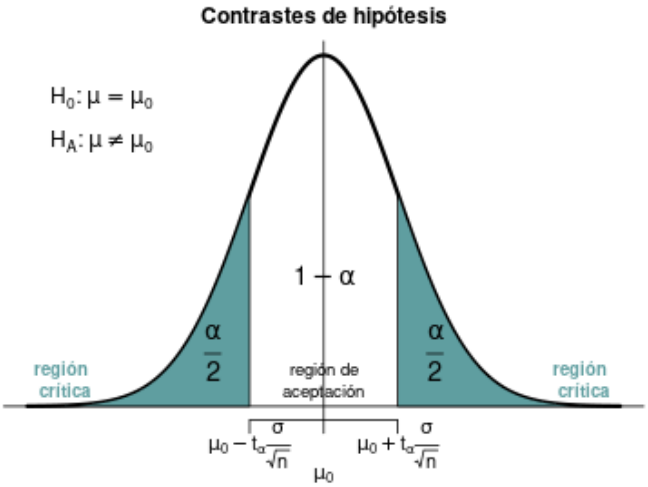


Figura 3.6: *Regiones de aceptación.*

$$\begin{cases} H_0 : \mu = 25 \\ H_a : \mu < 25 \end{cases}$$

(la función `t.test()` no funciona porque los datos vienen resumidos, ¡no tenemos los datos crudos!, así que construimos nosotros el estimador y el p-valor) [19].

```
# t test a mano (aml!)
# asumimos mu=25 (H0)
 mua<-22
 sda<-1.5
 n<-10
 t<-(mua-25)/(sda/(sqrt(n)))
 t
```

```
[1] -6.32
```

```
# ahora con la función 'pt', porque
# asumimos que el estadístico 't' sigue una dist t (TCL)
```

```
#calculamos la probabilidad de que ese valor se dé
pt(t,df=n-1)
```

```
[1] 0.0000685
```

```
# ¡Ojo!¡Hay que preocuparse...!
```

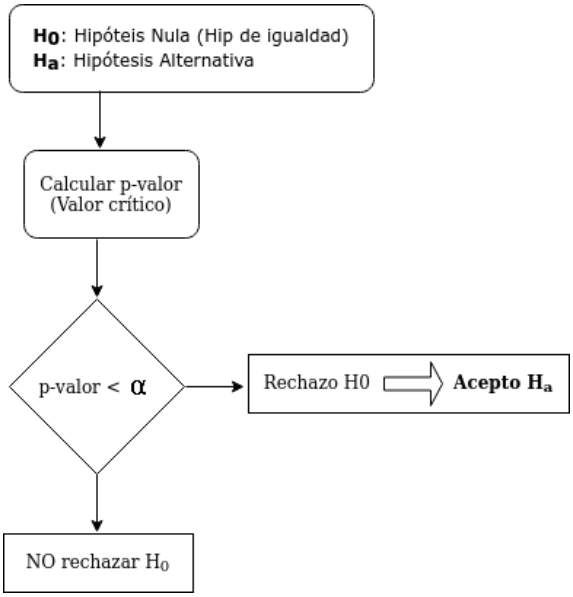


Figura 3.7: *Esquema de decisión contraste de hipótesis.*

3.5. Potencia Estadística

Cuando un experimento concluye diciendo que no se ha encontrado una *'diferencia significativa'*, no implica que no haya diferencia; simplemente no la hemos encontrado. Decimos entonces que no tenemos *suficiente evidencia para rechazar la hipótesis nula (la igualdad), así que la asumimos*.

Posibles razones para un p-valor no significativo ($p > 0.05$):

- En verdad no hay diferencia, es decir, es correcto nuestro test (nuestro p-valor) y no hay diferencias reales poblacionales.

- Sí hay diferencias poblacionales (no lo podemos saber con seguridad porque no conocemos el parámetro, sólo el estadístico/estimador).
  Algunas causas posibles:
  - Tamaño de la muestra.
  - Alta variabilidad.

  Si éstas son las razones estamos cometiendo **error de tipo II**.

3.5.1. Error de tipo I y error de tipo II

- Cometemos error de tipo II o β, cuando 'afirmamos' que no hay diferencias ($p > 0.05$) y en verdad sí las hay.
- Cometemos error de tipo I o α, cuando 'afirmamos' que sí hay diferencias ($p < 0.05$) y en verdad no las hay.

Llamamos **potencia estadística** de un test a la capacidad de un test para revelar diferencias que realmente existen.

¿Cuánta potencia tiene nuestro análisis para encontrar hipotéticas diferencias en el caso de existir éstas? La potencia depende del tamaño (n) y de la cantidad de variación de la muestra (d, desviación estándar o típica). También influye el tamaño de lo que es para nosotros una diferencia, o ¿cuánto han de diferir para considerarlos distintos?

$$n = (\frac{\sigma \times 1.96}{d})^2 \ , \ z_{1-\alpha/2} = 1.96 \ , \ \alpha = 0.05$$

Tabla 3.1: *Condición de la Población*

Acción	H_0 Verdadera	H_A Falsa
Aceptar	Conclusión Correcta	Error de Tipo II
Rechazar	Error de Tipo I	Conclusión Correcta

3.5.2. Múltiples comparaciones

Interpretar un p-valor es sencillo, *si la hipótesis nula (igualdad) es cierta hay un 5% de posibilidades de que una selección aleatoria de sujetos muestre una diferencia tan grande (o mayor) como la que muestran"*. Pero interpretar muchos p-valores a la vez puede no ser tan sencillo. Si testeamos diferentes hipótesis nulas (independientes) a la vez, con un nivel de significación del 0.05 hay más de un 5% de probabilidades obtener un resultado significativo por azar [12].

Ejemplo: Si hacemos 3 tests con $\alpha = 0.05$. La probabilidad de no cometer error de tipo I es **0.95 (95%) para cada test**. Suponemos que los tests son independientes.

- la probabilidad general de **NO** cometer error de tipo I es: $(0.95)^3 = 0.875$.
- Así la probabilidad general de cometer error de tipo I es: $1 - (0.95)^3 = 1 - 0.875 = 0.143$, es decir, un **14,3%**. ¡¡Se ha incrementado de 5% a 14,3%!! con sólo 3 comparaciones.

Cuantas más comparaciones hagamos más crece el error de tipo I.

Tabla 3.2: *Comparaciónes múltiples*

Nº hip nulas	Prob de al menos un p< 0.05	α para mantener Error de tipo I=0.05
1	5%	0.05
2	10%	0.0253
3	14%	0.0170
4	19%	0.0127
...	...	...
100	99%	0.0005
N	$100(1 - 0.95^N)$	$(1 - 0.95^N)$

Veremos más sobre este problema y de cómo resolverlo (correcciones de la significación y comparaciones planificadas) más adelante.

3.6. Contrastes de normalidad

Cuando contrastamos lo primero que debemos tener en mente es cuál es la hipótesis nula. En los contrastes de normalidad la hipótesis nula es también conocida como la *hipótesis de normalidad*, es decir, *"no hay diferencias entre nuestra distribución y una distribución normal con esa media y esa sd"*.

Recordemos que hay muchas maneras de evaluar la normalidad:

- **Gráficamente:** histogramas, Q-Q plots...
- **Analíticamente:** test de Shapiro-Wilk, de Kolmogorov–Smirnov, de Lilliefors, de Jarque-Bera... (los veremos luego)

3.6.1. Métodos Gráficos

3.6.1. Histogramas y curvas de densidad

La primera aproximación al problema de evaluar la normalidad nos la dan los histogramas cruzados con la curva de densidad teórica. De esta forma se representan los datos *empíricos* (*reales*) con la distribución teórica y se comparan *visualmente*. Es decir, la distribución que deberían tener los datos empíricos de seguir una distribución normal con esa media y desviación típica y nuestros datos)

```
x<- mtcars$mpg
h<-hist(x,breaks=10,xlab="Histograma empírico" ,main= "Hist + Normal teórica")

xfit <- seq( min(x), max(x), length=40)
yfit <- dnorm( xfit,mean=mean(x), sd=sd(x))
yfit <- yfit * diff( h$mids[1:2]) * length(x)

lines( xfit, yfit, col="blue", lwd=2)
```

3.6.1. Gráficos Q-Q

La manera de evaluarla mediante histogramas es muy *'rustica'* pues simplemente lo que se hace es comparar el histograma frente a la densidad teórica de una normal con esos parámetros. Los Q-Q plots son más informativos, *"Q"*

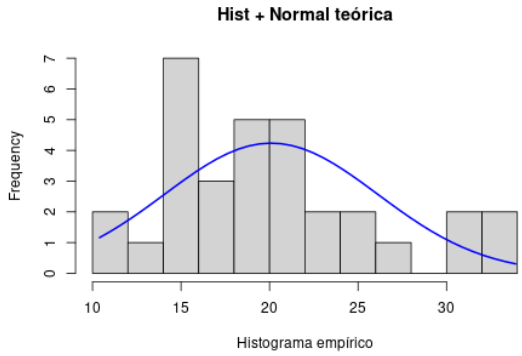


Figura 3.8: *Histograma y curva de densidad.*

viene de cuantil (*quantile* en inglés). El método se emplea para el diagnóstico de diferencias entre la distribución de probabilidad de una población de la que se ha extraído una muestra aleatoria y una distribución usada para la comparación, es decir, no sólo para contrastar normalidad.

Cuando queremos contrastar normalidad para una muestra de tamaño n, se dibujan n puntos con los $(n+1)-cuantiles$ de la distribución de comparación, la distribución normal, en el eje horizontal y el estadístico de k-ésimo orden (para k = 1,2,...n) de la muestra en el eje vertical. Si la distribución de la variable es la misma que la distribución de comparación se obtendrá, aproximadamente, una línea recta, especialmente cerca de su centro.

En R los podemos crear con la función `qqnorm()`.

```
qqnorm( mtcars$mpg)
qqline( mtcars$mpg)    # añadimos laa línea diagonal
```

Ejercicio: Compara los siguientes Q-Q plots.

```
par( mfrow = c(2,2) )
par( mar = c(2,2,2,2) )

x<-rnorm( 100,0,1)  ;qqnorm( x, main="normal(0,1)")       ; qqline(x)
x<-rnorm( 100,10,15);qqnorm( x, main="normal(10,15)")     ; qqline(x)
x<-rexp(  100,1/10) ;qqnorm( x, main="exponencial mu=10") ; qqline(x)
```

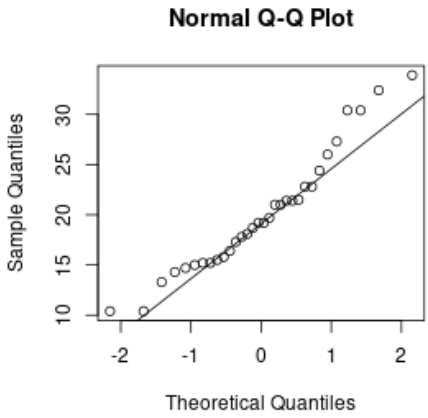


Figura 3.9: *Q-Q plot.*

```
x<-runif( 100,0,1)  ;qqnorm( x, main="unif(0,1)")          ; qqline(x)
```

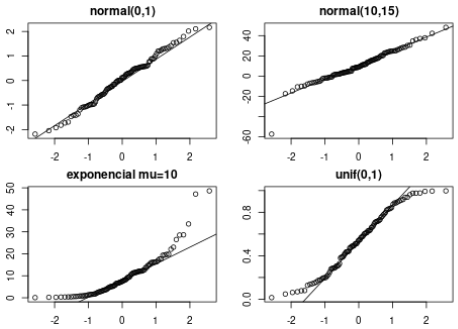


Figura 3.10: *Comparación de Q-Q plots.*

```
par(mfrow=c(1,1))
```

3.6.2. Métodos analíticos

3.6.2. Regla aproximativa

Regla aproximativa de evaluación de normalidad según la asimetría y la curtosis.

Curtosis y/o coeficiente de asimetría entre -1 y 1, es generalmente considerada una muy ligera desviación de la normalidad ([20], [21]).

Entre -2 y 2 tampoco es malo del todo, según el caso.

```
# cálculo de la curtosis y el coef. de asimetría en R

g3 <- mean( (x-mean(x))^3) / sqrt(sd(x)^3 )
g4 <- mean( (x-mean(x))^4) / sqrt(sd(x)^4 )
```

3.6.2. Contraste sobre asimetría y curtosis

Se trata siemplemente de ver si son atípicos. Primero normalizamos los coeficientes:

$$Z_s = \frac{S}{S \times E_s}$$
$$Z_k = \frac{K}{S \times E_k}$$

¡¡Ojo, para ambas la media es 0!!

Si alguno de los valores es mayor en valor absoluto que 1.96 se considera significativo ($|Z_s| > 1.96$ = significativo). Y consideramos que hay demasiada asimetría (en su caso curtosis).

Nota: En muestras grandes es más interesante mirar a la distribución de forma gráfica (histograma).

3.6.2. Contraste de normalidad de Shapiro-Wilk

El test de Shapiro-Wilk funciona bien con muestras pequeñas (menores de 50), para muestras grandes es equivalente al test de kolmogorov-Smirnov que veremos a continuación. Se ejecuta con la función `shapiro.test()`.

```
shapiro.test(x)
```

```
Shapiro-Wilk normality test
```

```
data:  x
W = 1, p-value = 0.007
```

3.6.2. Contraste de normalidad de Kolmogorov-Smirnov

El test de Kolmogorov-Smirnov puede testar si nuestra muestra proviene de una población con distribución cualquiera, no sólo de la normal. Se ejecuta con la función `ks.test()`. Es preferible a Shapiro-Wilk si las muestras son grandes (mayores a 100).

```
ks.test(x, y, ...,
alternative = c("two.sided", "less", "greater"),
exact = NULL)
```

```
# testamos si 'x' sigue una normal (0,1) (pnorm)
ks.test(x,"pnorm", 0, 1)
```

```
One-sample Kolmogorov-Smirnov test

data:  x
D = 0.5, p-value <2e-16
alternative hypothesis: two-sided
```

En este ejemplo rechazariamos la hipótesis nula (*rechazariamos la normalidad*) pues $p < 0.05$,

```
# testamos si 'x' sigue una normal de los parametros de x (pnorm)
ks.test(x,"pnorm", mean(x), sd(x) )
```

```
One-sample Kolmogorov-Smirnov test

data:  x
D = 0.07, p-value = 0.7
alternative hypothesis: two-sided
```

En este ejemplo no rechazaríamos la hipótesis nula ($p > 0.05$), así pues aceptamos que la muestra pueda provenir de una distribución normal con esos parámetros (media y sd).

También puede ser usado para contrastar si dos muestras siguen la misma

ley. En el siguiente ejemplo contrastaríamos si podemos aceptar que una muestra que sigue una normal sigue la misma ley que una muestra que sigue una exponencial.

```
ks.test(rnorm(100), rexp(80))
```

```
Two-sample Kolmogorov-Smirnov test

data:  rnorm(100) and rexp(80)
D = 0.6, p-value = 2e-13
alternative hypothesis: two-sided
```

El resultado es que no, el test resulta significativo y rechazamos la hipótesis de igualdad, es decir, que ambas variables tienen la misma distribución.

3.6.2. Test de normalidad de Jarque-Bera

El test de Jarque-Bera no requiere estimaciones de los parámetros que caracterizan la normal.

```
#install.packages("tseries")
library(tseries)
x <- mtcars$mpg
jarque.bera.test(x)
```

```
Jarque Bera Test

data:  x
X-squared = 2, df = 2, p-value = 0.3
```

Cuando no podemos asumir la normalidad esto influye en los modelo en:

- Los estimadores mínimo-cuadráticos no son eficientes (de mínima varianza).
- Los intervalos de confianza de los parámetros del modelo y los contrastes de significación son solamente aproximados y no exactos.

En otras palabras: Se pierde precisión.
EL TCL, nos permite *'saltarnos'* esta hipótesis para muestras suficientemente grandes.

A menudo es interesante localizar las causas de la falta de normalidad de nuestras muestras.

Causas comunes que originan falta de normalidad:

- Existen observaciones heterogéneas.
  - Errores en la recogida de datos.
  - El modelo especificado no es correcto (por ejemplo, no se ha tenido en cuenta una variable de clasificación cuando las observaciones proceden de diferentes poblaciones: *existen factores que no se han tenido en cuenta*).
- Hay observaciones atípicas (Se puede recurrir a estimadores robustos).
- Existe asimetría en la distribución (Se puede recurrir a transformaciones de los datos: familia de transformaciones de Box-Cox).

3.7. Contrastes de homogeneidad de varianza

El supuesto de homogeneidad de varianzas también se conoce como *supuesto de homocedasticidad*, nosotros habitualmente lo abreviamos como HOV, y dice que: *"la varianza es constante (no varía) en los diferentes niveles del factor"*. La falta de homocedasticidad se denomina heterocedasticidad (HEV).

Nota: Si el diseño es balanceado ($n_i = n_j$, para todo $i, j \in \{1,..,I\}$), la heterocedasticidad no afecta tanto a la calidad de los contrastes, a no ser que la varianza de la respuesta para algún grupo particular sea considerablemente mayor que para otros.

Regla: Balanceadas y HEV: ok si $\frac{\widehat{S}^2_{Max}}{\widehat{S}^2_{Min}} < 3$ *(Si no hay balanceo esta regla puede usarse con un 2)*

Si los tamaños muestrales son muy distintos se verifica que: si los grupos **con tamaños muestrales pequeños tienen mayor varianza** la probabilidad de cometer un error de tipo I en las pruebas de hipótesis será menor de lo que se obtiene y los niveles de confianza de los intervalos serán inferiores a lo que se cree (**conservador**).

Si por el contrario son los tratamientos con **tamaños muestrales grandes tienen mayor varianza** entonces se tendrá el efecto contrario y las pruebas serán más **liberales**.

Así pues es deseable que se verifique el supuesto de homocedasticidad.

3.7.0. Test de Levene

El test de Levene del paquete `car`, función `leveneTest()` (`levene.test()`). El test de Levene se resuelve con un ANOVA (ya veremos lo que es) de los valores absolutos de las desviaciones de los valores muestrales respecto a un estadístico de centralidad (media, mediana o media truncada) para cada grupo. La elección del estadístico de centralidad de los grupos determina la robustez y la potencia del test. Por *robustez* se entiende la habilidad del test para no detectar erróneamente varianzas distintas, cuando la distribución no es normal y las varianzas son realmente iguales. La potencia significa la habilidad del test para señalar varianzas distintas, cuando efectivamente lo son. El test de Levene permite comparar más de dos grupos a la vez.

```
#install.packages("car");
library(car)

levene.test(df$q1,df$gender, center="mean")
  Levene's Test for Homogeneity of Variance (center = "mean")
               Df F value  Pr(>F)
         group  1      15 0.03047 *
               3
```

Con la mediana:

```
leveneTest(df$q1,df$gender, center="median")
  Levene's Test for Homogeneity of Variance (center = "median")
               Df F value Pr(>F)
         group  1     2.4 0.2191
               3
```

3.7.0. F test para la evaluación de homocedasticidad

También podemos emplear la función `var.test()` para hacer un *F test* y comparar las varianzas de dos poblaciones.

Ejemplo para dos variables:

```
var.test(df$q1,df$q2)
```

```
            F test to compare two variances

    data:  df$q1 and df$q2
    F = 0.7059, num df = 4, denom df = 4, p-value = 0.7439
  alternative hypothesis: true ratio of variances is not equal to 1
        95 percent confidence interval:
        0.07349473 6.77966815
    sample estimates:
    ratio of variances
        0.7058824
```

Ejemplo cuando tenemos un factor que me determina dos grupos en una variabe:

```
var.test(df$q1~df$gender)
```

```
            F test to compare two variances

    data:  df$q1 by df$gender
        F = 0.1667, num df = 2, denom df = 1, p-value = 0.2679
    alternative hypothesis: true ratio of variances is not equal to 1
        95 percent confidence interval:
        0.0002084636 6.4177215190
    sample estimates:
    ratio of variances
        0.1666667
```

3.7.0. Test de Bartlett para testar más de dos grupos

Empleamos la función `bartlett.test()` para testar la homocedasticidad de más de dos muestras. El test de Levene es menos sensible a la falta de normalidad que el de Bartlett. Sin embargo, si estamos seguros de que los datos provienen de una distribución normal, entonces el test de Bartlett es el mejor.

```
require(graphics)
str(InsectSprays)
```

```
'data.frame':   72 obs. of  2 variables:
$ count: num  10 7 20 14 14 12 10 23 17 20 ...
$ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
head(InsectSprays)
```

```
count spray
1    10     A
2     7     A
3    20     A
4    14     A
5    14     A
6    12     A
```

```
plot(count ~ spray, data = InsectSprays)
```

```
fit<-bartlett.test(InsectSprays$count, InsectSprays$spray)
fit
```

```
Bartlett test of homogeneity of variances


data:  InsectSprays$count and InsectSprays$spray
Bartlett's K-squared = 26, df = 5, p-value = 0.00009
```

y efectivamente se comprueba que no debemos aceptar la hipótesis de homogeneidad de varianzas ya que $p < 0.05$

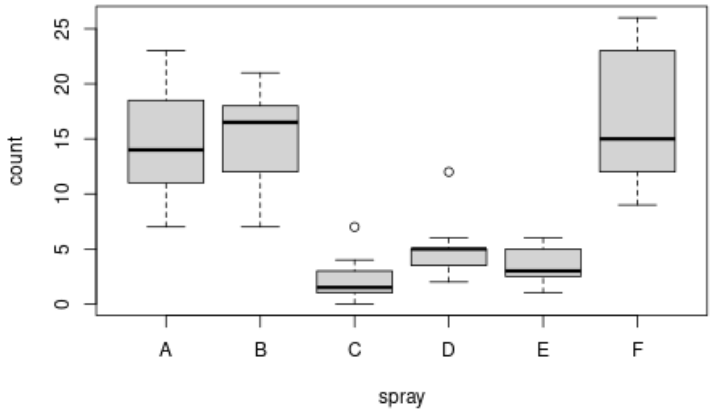


Figura 3.11: *Boxplolt comparando diferentes niveles.*

3.7.0. Test de Brown-Forsyth

La función `plot.hov()` (paquete `HH`), nos ofrece un test gráfico de HOV basado en Brown-Forsyth.

```
#install.packages("HH")
gender <- c(rep("M",10), rep("F",10))
q1 <- c(rnorm(10,0,0.5),rnorm(10,0,2))
df <- data.frame(gender, q1,  stringsAsFactors = TRUE)
library(HH)

hov(df$q1~df$gender)

hov: Brown-Forsyth

data:  df$q1
F = 11, df:df$gender = 1, df:Residuals = 18, p-value = 0.003
alternative hypothesis: variances are not identical

hovPlot(df$q1~df$gender)
```

Ver figura 3.12

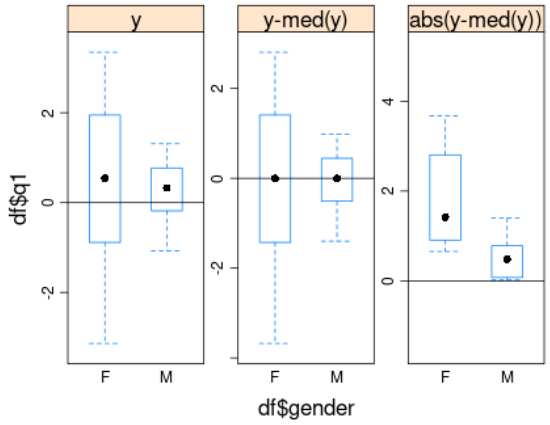


Figura 3.12: *Gráfico de HOV para el test de Brown-Forsyth.*

3.7.0. Test de Fligner-Killeen

Podemos también comparar varianzas empleando un test no paramétrico con la función `Fligner.test()`que se basa en la mediana.

```
> fligner.test(InsectSprays$count, InsectSprays$spray)

        Fligner-Killeen test of homogeneity of variances

data:  InsectSprays$count and InsectSprays$spray
  Fligner-Killeen:med chi-squared = 14.4828, df = 5, p-value = 0.01282
```

Notas generales sobre la elección del test: El artículo original de Levene proponía la media como estadístico de centralidad. Brown y Forsythe [22] extendieron este test al utilizar la mediana e incluso la media truncada al 10%. Sus estudios de Monte Carlo mostraron que la utilización de la media truncada mejoraba el test cuando los datos seguían una distribución de Cauchy (colas grandes) y la mediana conseguía mejorarlo cuando los datos seguían una (distribución asimétrica). Con la media se consigue el mejor test para distribuciones simétricas y con colas moderadas. Así pues, aunque la elección óptima depende de la distribución de los datos, la definición del test basada en la mediana es la recomendación general, ya que, proporciona una buena robustez para la mayoría

de distribuciones no normales y, al mismo tiempo, una aceptable potencia. Si conocemos la distribución de los datos, podemos optar por alguna otra de las opciones.

3.8. Transformación de datos

Vamos a comparar dos conjuntos de datos, analizar su normalidad y aplicarles transformaciones.

Generamos datos aleatorios con dos leyes diferentes.

```
set.seed( 111 )
d_normal <- rnorm( 20 )
d_exp    <- rexp(  20 )
```

Representamos las gráficas y vemos que son muy diferentes. La función `layout()` permite decidir cómo van a ser representadas las gráficas (en este caso, una al lado de la otra).

```
layout( matrix( c( 1, 2 ), nrow = 1) )
hist( d_normal, breaks = "FD" )
hist( d_exp,    breaks = "FD")
```

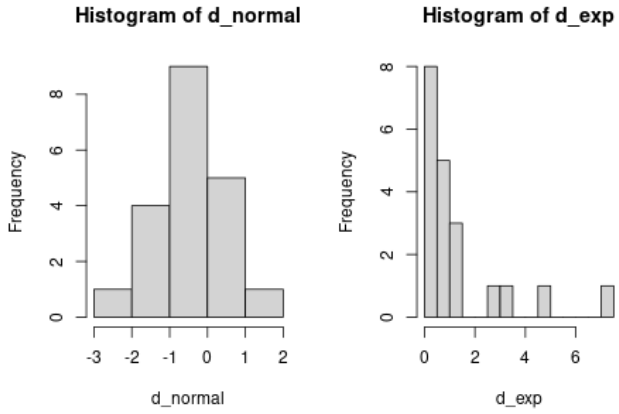


Figura 3.13: *Histogramas para comparar distribuciones.*

La función `hist()` selecciona de forma automática la anchura de cada marca/caja del histograma, podemos seleccionar diferentes algoritmos para determinar el número de bins y su anchura con la opción "`breaks`". Generalmente el algoritmo "FD" es el mejor en casi todos los casos `breaks="Sturges"/"Scott"/"FD"`.

Si ahora analizamos los Q-Q plots:

```
layout( matrix( c( 1, 2 ), nrow = 1) )
qqnorm( d_normal )
qqline( d_normal )
qqnorm( d_exp )
qqline( d_exp )
```

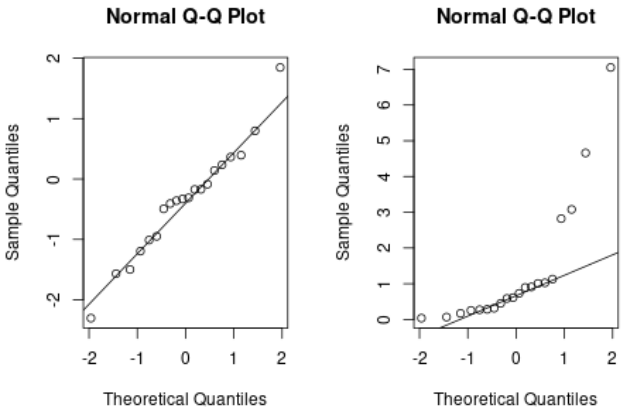


Figura 3.14: *Q-Q Plots para comparar distribuciones.*

Podemos aplicar un **test de normalidad** para evaluarla.

```
shapiro.test( d_normal )
```

```
Shapiro-Wilk normality test

data:  d_normal
W = 1, p-value = 0.8
```

El p-valor > 0.05, es decir, se acepta la hipótesis nula de normalidad.

```
shapiro.test( d_exp )
```

```
Shapiro-Wilk normality test

data:  d_exp
W = 0.7, p-value = 0.00002
```

El p-valor < 0.05, es decir, se rechaza la hipótesis nula de normalidad.

Más generalmente podemos aplicar el test de Kolmogorov-Smirnov obteniendo de nuevo los mismos resultados.

```
ks.test( d_normal, "pnorm", mean = mean( d_normal ), sd = sd( d_normal ) )
```

```
One-sample Kolmogorov-Smirnov test

data:  d_normal
D = 0.1, p-value = 0.8
alternative hypothesis: two-sided
```

```
ks.test( d_exp, "pnorm", mean = mean( d_exp ), sd = sd( d_exp ) )
```

```
One-sample Kolmogorov-Smirnov test

data:  d_exp
D = 0.3, p-value = 0.01
alternative hypothesis: two-sided
```

Este test `ks.test()` nos permite comprobar que nuestra distribución d_exp viene de una exponencial (ya lo sabíamos por cómo la hemos generado).

```
ks.test( d_exp, "pexp" )
```

```
One-sample Kolmogorov-Smirnov test

data:  d_exp
D = 0.1, p-value = 0.8
alternative hypothesis: two-sided
```

También nos permite testar si ambas siguen la misma distribución.

```
ks.test( d_normal, d_exp )
```

```
Two-sample Kolmogorov-Smirnov test

data:  d_normal and d_exp
D = 0.7, p-value = 0.00006
alternative hypothesis: two-sided
```

Vemos que efectivamente **NO siguen la misma**, $p < 0.05$.

Imaginemos ahora que queremos contrastar esas poblaciones, ¿qué test empleamos?. Como para una de ellas no se cumple la hipótesis de normalidad, una opción es aplicar transformaciones a ver si logramos obtenerla. Hay un conjunto de transformaciones que son las usuales y las que mejores resultados proporcionan.

En todo caso después de la transformación podemos aplicar tests paramétricos si hemos alcanzado los supuestos, y cuando comuniquemos el análisis descriptivo lo hacemos de los datos sin transformar, aunque los test hayan sido llevados a cabo con los datos transformados.

Las transformaciones más usuales son el logaritmo y la raíz cuadrada. **Ojo** cuando transformemos si tenemos ceros o valores negativos, en ese caso habrá que trasladar las muestras.

Transformamos la primera aplicándole la función `log()`:

```
ld_exp <- log( d_exp )
shapiro.test( ld_exp )

Shapiro-Wilk normality test

data:  ld_exp
W = 1, p-value = 0.9
```

Y hacemos lo mismo con la segunda pero, además, le sumaremos uno para evitar hacer el logaritmo de cero:

```
k          <- abs( min( d_normal ) ) + 1
ld_normal <- log( d_normal + k )
shapiro.test( ld_normal )

Shapiro-Wilk normality test
```

```
data:  ld_normal
W = 0.9, p-value = 0.08
```

Una vez que se han transformado ambas se pueden comparar de nuevo:

```
layout( matrix( c( 1, 2 ), nrow = 1 ) )
hist( ld_normal, breaks = "FD" )
hist( ld_exp,    breaks = "FD" )
```

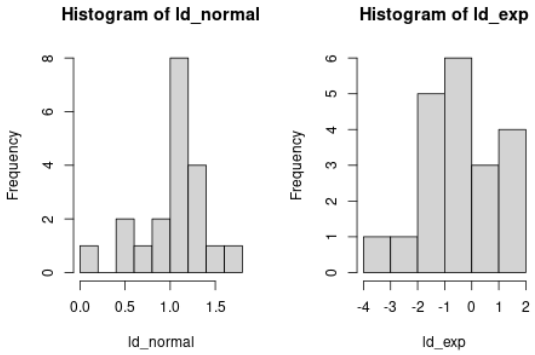


Figura 3.15: *Comparación por histogramas de las variables transformadas.*

```
layout( matrix(c( 1, 2 ), nrow = 1 ) )
qqnorm( ld_normal )
qqline( ld_normal )
qqnorm( ld_exp )
qqline( ld_exp )
```

Ver figura 3.16

3.9. Distribución t de Student

3.9.1. Representación de la Normal

Si queremos representar la función de densidad de la distribución normal podemos utilizar la función `plot()`. Una forma de utilizar esta función es tomar

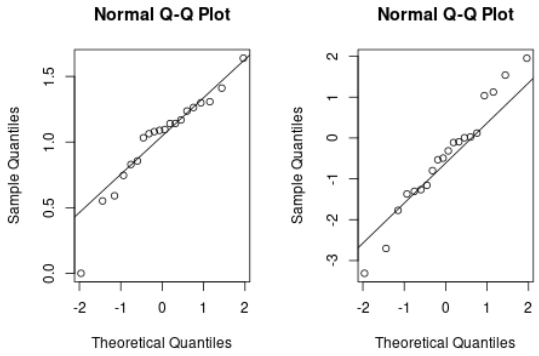


Figura 3.16: *Comparación por Q-Q plots de las variables transformadas.*

un montón de puntos x y para cada uno el valor de la función de densidad.

```
x <- seq( -5, 5, by = 0.1)
y <- dnorm( x )
plot( x, y, ylim = c( 0, 0.4 ), col = "blue", type = "l", lwd = 2,
        main = "Función densidad N(0,1)" )
```

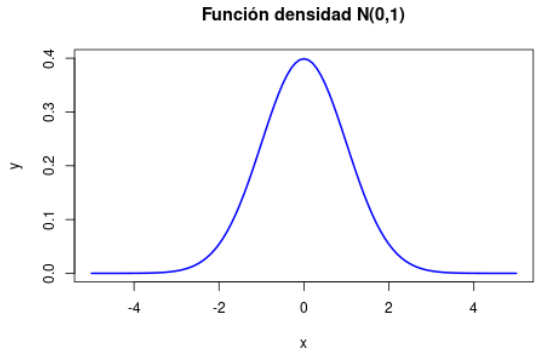


Figura 3.17: *Función de densidad N(01).*

En este caso, como es una función continua también podemos pasarle como argumento la propia función y `plot()` la dibujará eligiendo él los puntos entre los valores que fijemos.

```
plot( function(x) dnorm( x ), -5, 5, ylim = c( 0, 0.4 ),
```

```
        col = "blue", type = "l", lwd = 2,
        main = "Función densidad N(0,1)" )

plot( dnorm, -5, 5, ylim = c( 0, 0.4 ),
        col = "blue", type = "l", lwd = 2,
        main = "Función densidad N(0,1)" )
```

3.9.2. Representación de la t de Student

En este caso se hace igual que con la normal (también es continua). Si queremos representar la t de Student con 2 grados de libertad.

```
plot( function(x) dt( x, df = 2 ), -5, 5, ylim = c( 0, 0.4 ),
        col = "red", type = "l", lwd = 2,
        main = "Función densidad t de Student df = 2" )
```

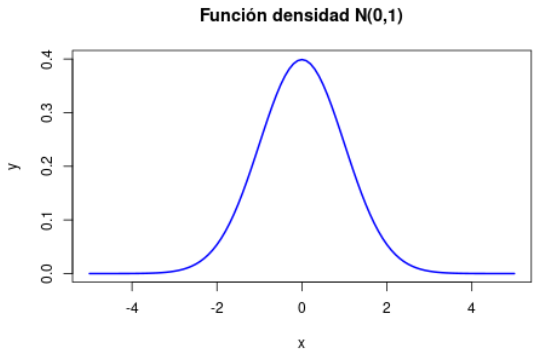


Figura 3.18: *Función de densidad t de Student.*

Si queremos añadir un gráfico al que teníamos basta con escribir el argumento `add = TRUE`.

```
plot( function(x) dt( x, df = 2 ), -5, 5, ylim = c( 0, 0.4 ),
        col = "red", type = "l", lwd = 2,
        main = "Función densidad t de Student df = 2 y 5" )
plot( function(x) dt( x, df = 5 ), -5, 5, col = "green",
        type = "l", lwd = 2, add = TRUE )
```

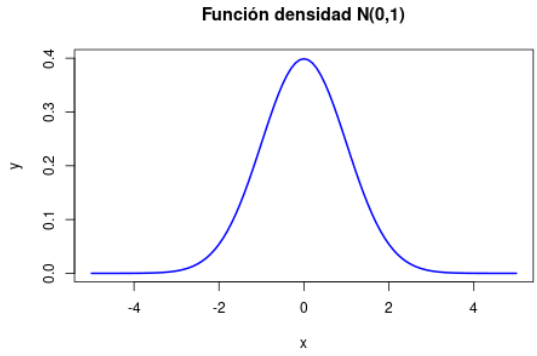


Figura 3.19: *Función de densidad t de Student, 2 y 5 gl.*

3.9.3. Normal y t de Student

Haciendo un bucle `for` y con el argumento `add = TRUE` podemos generar un gráfico con el número de distribuciones t de Student que queramos fijando el grado de libertad donde parar.

También podemos añadir texto en el punto `(x,y)` con `text( x, y )`. Con el argumento `cex` fijamos el tamaño y con un vector fijamos los sucesivos tamaños (en cada paso del bucle).

Con `col` especificamos el color. En este caso, con `gray` decimos que el color sea gris, y con un vector lo que hacemos es representar más negro cuanto mayor es el grado de libertad.

```
# Normal
plot( dnorm, -5, 5, ylim = c( 0, 0.4 ), col="blue", lwd=3 )

# t de Student
dfmax <- 9 # hasta cuál queremos representar
for(i in 1:dfmax){
plot( function(x) dt( x, df = i ), -5, 5, add = TRUE,
        lwd = 1, col = gray( 1 - i/dfmax ) )
text( 0, dt( 0, df = i ), i, cex = 0.5/i+0.5 )
}
```

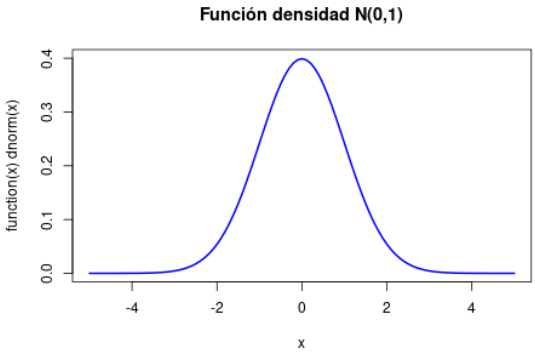


Figura 3.20: *Aproximación de t-studnet a la Normal.*

También podemos representar la función de densidad de la t de Student con un grado de libertad infinito (el límite). Como esperamos que se solape con la normal dibujamos puntos vacíos con `type = 'b'`.

```
# Normal
plot( dnorm, -5, 5, ylim = c( 0, 0.4 ), col="blue", lwd=3 )

# t de Student
dfmax <- 9 # hasta cuál queremos representar
for(i in 1:dfmax){
plot( function(x) dt( x, df = i ), -5, 5, add = TRUE,
        lwd = 1, col = gray( 1 - i/dfmax ) )
text( 0, dt( 0, df = i ), i, cex = 0.5/i+0.5 )
}

# t de Student grados de libertad Inf
plot( function(x) dt( x, df = Inf ), -5, 5, add = TRUE,
        type = "b", lwd = 2, col = "green" )
```

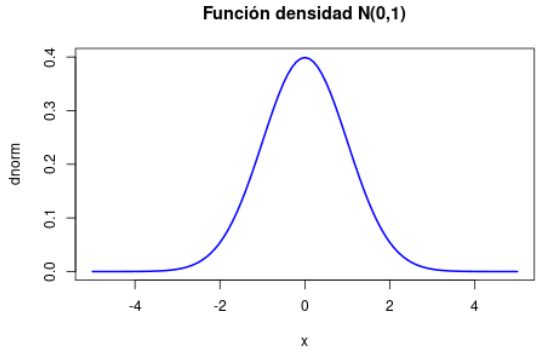


Figura 3.21: *t de Student con un grado de libertad infinito.*

4 Comparación paramétrica de medias

Aurora Gónzalez Vidal
Álvaro Hernández Vicente

4.1. Comparaciones entre dos grupos

4.1.1. Introducción

En los análisis estadísticos de datos procedentes de investigación es muy común que nos encontremos con la necesidad de comparar dos o más grupos.

Un investigador se puede encontrar con la necesidad de comparar la longitud del rabo de 3 tipos de ratones de laboratorio cuando son alimentados con dos dietas diferentes.

Otro, sin embargo, puede estar interesado en comparar la aceptación de la homosexualidad dependiendo de la edad: niños, adultos y ancianos.

Son muy diversas las áreas que necesitan analizar datos clasificados por grupos y los modelos que se pueden aplicar para resolver estas cuestiones serán el

Cómo citar este capítulo: Gónzalez-Vidal, A. y Hernández-Vicente A. (2022). Comparación paramétrica de medias. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (133-184). Editum. Ediciones de la Universidad de Murcia.

objeto de estudio.

Test paramétricos frente a tests no paramétricos

Una de las las categorizaciones más importantes en estadística consiste en clasificar los problemas en paremétricos y no paramétricos. Paramétrico significa que con unos pocos parámetros podemos describir bien la distribución de la variable. Normalmente, para poder hacer esto hemos de suponer ciertas propiedades de la distribución: media, desviación típica...; y, con estos parámetros, debemos ser capaces de asumir que la distribución de la variable es de una determinada manera, generalmente una disribución **normal**. Hay otras condiciones iniciales para los test paramétricos (por ejemplo, homocedasticidad) que iremos discutiendo más adelante.

Esta condición de normalidad, asumir que la población de la que provienen los datos es normal, no implica necesariamente que los datos, la muestra, deba parecerse a una normal: sólo que podamos asumir que provienen de una población normal. Emplearemos tests como el test de Shapiro-Wilk, Kolmogorov-Smirnov..., analizaremos histogramas, Q-Q plots etc.

¿Cuándo emplear tests no paramétricos en lugar de los paramétricos?

- Con datos ordinales en general empleamos test no paramétricos: por ejemplo datos procedentes de escalas Likert (grado de satisfacción del 1 al 5,etc...)
- Cuando la normalidad de la población no pueda ser asumida.

Afortunadamente, muchos de los test paramétricos son bastante robustos a la falta de normalidad. También podemos optar por emplear transformaciones de datos para "alcanzar" la normalidad.

4.1.2. t-test

El más común de los test para comparar medias de dos grupos es el **t-test**.

Para poder emplear un t-test hemos de comprobar ciertos supuestos primero. Lo más importante es que las distribuciones de las poblaciones de las cuales muestreamos han de ser normales. Decimos distribución de las poblaciones, no de las muestras, que siempre hemos de suponerlas procedentes de un muestreo

aleatorio simple. Si las muestras a comparar son suficientemente grandes podemos invocar el Teorema Central del Límite (TCL) y asumir la normalidad, pero en caso contrario, sí es importante comprobar la normalidad con muestras pequeñas donde el TCL no puede ser aplicado.

Hay tres tipos de t-test: t-test dependiente, t-test independiente y t-test para una muestra.

A la hora de decidir cuál usar hay que tener en cuenta el diseño experimental que tengamos, ya que no podemos emplear uno u otro a nuestro antojo.

Vamos a tomar estos datos simulados para todos los experimentos:

```
medida <- c( 3, 1, 2, 3, 1, 6, 2, 4, 1, 2, 6, 5, 1, 3, 5, 4, 2, 3, 4, 5 )
grupo  <- c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 )
df     <- data.frame( grupo, medida )
head( df )
```

```
  grupo medida
1     0      3
2     0      1
3     0      2
4     0      3
5     0      1
6     0      6
```

4.1.2. t-test dependiente (paired)

4.1.2.1.1. Introducción Se utiliza cuando tratamos de evaluar diferencias de medias entre dos grupos que están relacionados. Es decir, los mismos sujetos (personas, animales...) son medidos en dos condiciones diferentes, como por ejemplo de tiempo: antes y después. En este caso, primero calculamos la diferencia entre un grupo y otro (antes y después) y hacemos un **t-test para una muestra** (one-sample t-test). Estos test se denominan también **t-test para datos pareados**, diseño con factores intra sujetos (within-subjet design) y diseños con medidas repetidas.

Un diseño apareado (paired) considera las diferencias individuales de cada observación, cuando hace la diferencia entre los dos grupos experimentales.

Así que tiene ciertas ventajas sobre el test independiente que a continuación estudiaremos.

Si entendemos que la variación sistemática es la producida por el experimento (es decir, las diferencias entre grupos producidas por el hecho de ser sometidos a experimentos diferentes) y la variación no-sistemática es la debida a factores aleatorios que pueden existir entre las distintas condiciones experimentales (por ejemplo: personas que sin saber por qué tienen diferentes tolerancias a ciertas drogas). Así pues, con los diseños pareados :

- Se controla mejor la variación no sistemática porque se mide varias veces el mismo sujeto.
- Es más fácil descubrir los efectos de nuestra manipulación experimental.
- En diseños de medidas repetidas el ruido se mantiene en el mínimo.
- Los diseños de medidas repetidas tienen más potencia estadística, detectan diferencias que existen realmente más fácilmente que los diseños independientes.

4.1.2.1.2. Supuestos El t-test dependiente no nos exige homocedasticidad pero sí normalidad. Lo comprobamos para cada grupo. Para seleccionar únicamente las medidas del grupo 0 hacemos `df[df$grupo==0,2]`.

```
df[ df$grupo == 0, 2 ]
```

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

El 2 es porque cogemos la segunda columna del conjunto de datos `df`. Al escribir el número 1 se selecciona la primera, que no es otra cosa que ceros

```
df[ df$grupo == 0, 1 ]
```

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

El análisis:

```
shapiro.test( df[ df$grupo == 0, 2 ] )
```

```
    Shapiro-Wilk normality test
```

```
data:  df[df$grupo == 0, 2]
W = 0.9, p-value = 0.08
```

```
shapiro.test( df[ df$grupo == 1, 2 ] )
```

```
    Shapiro-Wilk normality test

data:  df[df$grupo == 1, 2]
W = 0.9, p-value = 0.7
```

Ambos p-valores 0.083 y 0.668 son mayores que 0.05 luego podemos aceptar la hipótesis nula, que es la de normalidad.

4.1.2.1.3. Hipótesis a contrastar La hipótesis nula (H_0) es "no hay diferencia entre las medias de dos poblaciones de las cuales tenemos dos muestras" y la alternativa "sí la hay". Así, si fijamos la significación en 0.05, rechazaremos esta hipótesis nula si $p < 0.05$ y concluiremos que aceptamos la hipótesis alternativa de que existen diferencias.

$$H_0 : \mu_1 = \mu_2$$
$$H_1 : \mu_1 \neq \mu_2$$

Ejecutamos el t-test en R con la función `t.test()`.
Los dos primeros argumentos son los grupos a comparar y escribimos `paired=TRUE` porque son dependientes.

```
fitDep <- t.test( df[ df$grupo == 0, 2 ], df[ df$grupo == 1, 2 ], alternative= "two.sided",
        conf.level = 0.95, paired = TRUE )
```

El p-valor es $0.109 > 0.05$ luego se acepta la hipótesis nula. El intervalo de confianza nos indica que la diferencia de medias está en el intervalo $(-2.95, 0.35)$ con una confianza del 95%.

Nota: la opción `alternative= "two.sided"` es la que viene por defecto e igualmente por defecto se calculan los intervalos de confianza de nivel 0.95 así que escribir

```
t.test( df[ df$grupo == 0, 2 ], df[ df$grupo == 1, 2 ], paired = TRUE )
```

será equivalente en este caso particular.

Cuando se desea contrastar si la media de un grupo es mayor que la media del otro, la hipótesis alternativa (H_1) es: "la media del primer grupo es mayor que la del segundo" y la hipótesis nula es: "la media del primero es menor o igual que la del segundo". Es decir,

$$H_0 : \mu_1 \leq \mu_2$$
$$H_1 : \mu_1 > \mu_2$$

En R se hará `t.test( x, y, alternative="greater", paired=TRUE )`.

Cuando, por el contrario, se desea contrastar si la media de un grupo es menor que la media del otro, la hipótesis alternativa (H_1) es: "la media del primer grupo es menor que la del segundo" y la hipótesis nula es: "la media del primero es mayor o igual que la del segundo". Es decir,

$$H_0 : \mu_1 \geq \mu_2$$
$$H_1 : \mu_1 < \mu_2$$

En R se hará `t.test( x,y,alternative="less",paired=TRUE )`.

4.1.2. t-test independiente (unpaired)

4.1.2.2.1. Introducción Se utiliza cuando los sujetos que pertenecen a cada grupo son distintos, es decir, a cada nivel del experimento son sometidos diferentes sujetos. En este caso se construye un modelo para cada grupo, se calcula su media y varianza y se ve cuándo hay o no hay diferencias entre los dos modelos mediante un t-test para muestra independientes.

4.1.2.2.2. Supuestos Este test, ademas de la hipótesis de normalidad, nos exige homocedasticidad (homogeneidad de las varianzas de las dos muestras a comparar).

Para comprobar si el modelo es homocedástico podemos utilizar un test de varianza por ejemplo `var.test()`.

```
var.test( df[ df$grupo == 0, 2 ], df[ df$grupo == 1, 2 ] )
```

```
    F test to compare two variances
```

```
data:  df[df$grupo == 0, 2] and df[df$grupo == 1, 2]
F = 1, num df = 9, denom df = 9, p-value = 1
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.259 4.194
sample estimates:
ratio of variances
              1.04
```

El p-valor es 0.95 > 0.05 luego se acepta la hipótesis nula, esto es, la igualdad de varianzas.

4.1.2.2.3. Hipótesis a contrastar Como antes, pueden ser

$$H_0 : \mu_1 = \mu_2$$
$$H_1 : \mu_1 \neq \mu_2.$$

Así, si fijamos la significación en 0.05, rechazaremos esta hipótesis nula si p < 0.05 y concluiremos que aceptamos la hipótesis alternativa de que existen diferencias.

Ejecutamos t-test en R con la función `t.test()` y elegiremos `var.equal=FALSE` o `var.equal= TRUE` dependiendo del resultado del test de homocedasticidad.

```
fitIndep <- t.test(df[df$grupo ==0,2], df[df$grupo == 1,2],var.equal = TRUE)
```

Como p = 0.079 > 0.05 no hay una diferencia significativa.

De nuevo, las otras hipótesis pueden ser

$$H_0 : \mu_1 \leq \mu_2$$
$$H_1 : \mu_1 > \mu_2$$

se hará `t.test( x, y, alternative="greater" )` o

$$H_0 : \mu_1 \geq \mu_2$$
$$H_1 : \mu_1 < \mu_2$$

se hará `t.test( x, y, alternative="less" )`.

4.1.2. t-test para una muestra

Este es el caso en que vamos a contrastar la media de un conjunto de observaciones que pertenecen a un mismo grupo.

4.1.2.3.1. Supuestos: normalidad

```
shapiro.test( df[ , 2 ] )
```

```
    Shapiro-Wilk normality test

data:  df[, 2]
W = 0.9, p-value = 0.1
```

Aceptamos la hipótesis de normalidad (p=0.09553 > 0.05).

4.1.2.3.2. Hipótesis a contrastar

$$H_0 : \mu = \mu_0 \qquad H_0 : \mu \leq \mu_0 \qquad H_0 : \mu \geq \mu_0$$
$$H_1 : \mu \neq \mu_0 \qquad H_1 : \mu > \mu_0 \qquad H_1 : \mu < \mu_0$$

```
t.test( df[ , 2 ] )
```

```
    One Sample t-test

data:  df[, 2]
t = 8, df = 19, p-value = 7e-08
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.37 3.93
sample estimates:
mean of x
     3.15
```

Es equivalente a

```
t.test( df[ , 2 ], mu = 0, conf.level = 0.95 )
```

4.1.3. Tamaño del efecto:

La idea básica que hemos seguido para obtener resultados se trata de generar una hipótesis nula y una hipótesis alternativa, encontrar un modelo estadístico para nuestros datos y evaluar este modelo con un test estadístico. Si la probabilidad de obtener nuestro test estadístico por azar es menor que 0.05 entonces aceptamos la hipótesis alternativa (rechazamos la hipótesis nula) y decimos normalmente que hay un efecto significativo, es decir, que los grupos reaccionan de forma diferente con respecto a los diferentes factores del experimento. **El experimento produce un efecto.**

Pero no nos engañemos; el hecho de que la probabilidad de que nuestro efecto sea azaroso sea pequeña (menor que 0.05) no implica que el efecto es importante.

Es decir, hemos encontrado que hay menos de un 5% (el p valor) de probabilidad de que el efecto que estamos observando provenga del azar y hemos establecido que eso significa que no proviene del azar. Lo que pasa es que el p-valor depende del tamaño muestral y efectos muy pequeños e insignificantes pueden llegar a ser estadísticamente significantes porque se usan muchos sujetos en el experimento. Por esto, se necesita medir de una forma estandarizada e independiente del tamaño muestral el tamaño del efecto (effect size o "es"). Es estandarizada para poder comparar los efectos de diferentes estudios que miden diferentes variables y que usan diversas escalas.

El tamaño del efecto puede ser medido según diferentes métricas. Las más usuales son la d de Cohen, la r de Pearson o coeficiente de correlación de Pearson, y el coeficiente eta-cuadrado.

Además deberíamos de calcular intervalos de confianza para esos tamaños de efectos, ya que su cálculo es simplemente un estimador del tamaño del efecto.

Una diferencia con respecto a los IC de medias es que estos IC pueden ser asimétricos (así el effect size no es la media de los extremos como ocurre con los IC de la media). Si este IC contiene al "0" concluiremos que el tratamiento no tiene ningún efecto, es decir, que no había diferencia significativa y la hipótesis nula no se rechaza, sino que se acepta.

[Lectura recomendada: [23]]

El IC no contiene el 0:

abs(es)	tamaño del IC	Lectura
pequeño	pequeño	El efecto aparentemente existe, pero estamos seguros de que es pequeño.
pequeño	grande	Sumamente improbable que ocurra esto.
grande	pequeño	El efecto aparentemente existe, y estamos seguros de que es grande.
grande	grande	El efecto aparentemente existe y puede ser grande, pero no estamos seguros de su tamaño.

El IC contiene el 0:

abs(es)	tamaño del IC	Lectura
pequeño	pequeño	No estamos seguros de que exista un efecto, en caso de existir es pequeño.
pequeño	grande	No estamos seguros de que exista un efecto. Necesitamos más datos (muestra).
grande	pequeño	Sumamente improbable que ocurra esto.
grande	grande	No estamos seguros de que existan un efecto. Necesitamos más datos (muestra).

Un detalle es que el intervalo de confianza (IC) depende del tamaño muestral, así que aumentando el tamaño muestral podemos hacer mas estrecho el IC y llegar a conclusiones más fuertes. Pero eso sí, incrementar la muestra no contribuye a aumentar el tamaño del efecto (al contrario que ocurre con los p-valores). [24]

Tamaño del efecto para un t-test

Hay dos métricas disponibles para el tamaño del efecto ("es" de *effect size* para un t-test: la d de Cohen (*Cohen's d*), y la r de Pearson. Ambas son equivalentes y puedes pasar de d a r y viceversa. Aunque no podemos emplear la r de Pearson para los test dependientes, en este caso solo Cohen.

Dependiendo del área de conocimiento se consideran ciertos valores de d y r como efectos pequeños, medianos o grandes pero la sugerencia general suele ser la siguiente:

Efecto	Pequeño	Mediano	Grande
d de Cohen	0.2	0.5	0.8
r de Pearson	0.1	0.3	0.5

Efecto	d de Cohen	r de Pearson
t- test dependiente (paired)	$d = \frac{\lvert M \rvert}{SD}$	No se puede emplear
t- test independiente (*unpaired*)	$d = \frac{\lvert \mu_1 - \mu_2 \rvert}{\sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}}}$	$r = \sqrt{\frac{t^2}{t^2+gl}}$

Nota: M es la media de las diferencias y SD la desviación estándar de las diferencias.

Cálculo del tamaño del efecto con R

Debemos cerciorarnos de que en los ejemplos de t-test que hemos llevado a cabo hemos tomado la decisión correcta.

En el ejemplo del t-test dependiente decíamos que como $p = 0.109 > 0.05$ se acepta la hipótesis nula, es decir, no hay efecto significativo. Como no hay efecto, carece de sentido calcular su tamaño. De todas formas, lo haremos para comprobar que efectivamente el intervalo de confianza del mismo va a contener al 0.

Para el t-test dependiente lo hacemos con la d de Cohen.

```
s <- sd(    df[ df$grupo == 1, 2 ] - df[ df$grupo == 0, 2 ] )
s
```

```
[1] 2.31
```

```
M <- mean( df[ df$grupo == 1, 2 ] - df[ df$grupo == 0, 2 ] )
M
```

```
[1] 1.3
```

```
d <- abs( M ) / s
d
```

```
[1] 0.562
```

d = 0.56, es decir, el tamaño del efecto es mediano.

Para calcular el IC del tamaño del efecto podemos emplear la función `ci.sm()` del paquete MBESS.

```
#install.packages( "MBESS" )
library( "MBESS" )
ci.sm( Mean = M, SD = s, N = 10, conf.level = 0.95 )
```

```
[1] "The 0.95 confidence limits for the standardized mean are given as:"

$Lower.Conf.Limit.Standardized.Mean
[1] -0.121

$Standardized.Mean
[1] 0.562

$Upper.Conf.Limit.Standardized.Mean
```

```
[1] 1.22
```

Esta función no solo nos proporciona el intervalo de confianza IC = (-0.12,1.22) sino que también devuelve la d de Cohen.

Efectivamente el intervalo de confianza contiene al "0" y la hipótesis nula no se rechaza tal y como habíamos concluido con el t-test.

Para la continuación del ejemplo del t-test independiente que arrojaba un p-valor = 0.079 > 0.05 empleamos la función `tes()` del paquete compute.es para **calcular el tamaño del efecto.**

El modelo `fitIndep` devuelve una lista de valores. El primero de ellos es el **estadístico t** y su valor numérico se llama escribiendo `fitIndep[[1]]`. Los grados de libertad son el segundo valor y se llama escribiendo `fitIndep[[2]]`.

```
library( compute.es )
tes( fitIndep[[1]], 10, 10 )
```

```
Mean Differences ES:

 d [ 95 %CI] = -0.83 [ -1.74 , 0.08 ]
  var(d) = 0.22
  p-value(d) = 0.09
  U3(d) = 20.3 %
  CLES(d) = 27.9 %
  Cliff's Delta = -0.44

 g [ 95 %CI] = -0.8 [ -1.67 , 0.08 ]
  var(g) = 0.2
  p-value(g) = 0.09
  U3(g) = 21.3 %
  CLES(g) = 28.7 %

 Correlation ES:

 r [ 95 %CI] = 0.4 [ -0.05 , 0.72 ]
  var(r) = 0.04
  p-value(r) = 0.1
```

```
 z [ 95 %CI] = 0.42 [ -0.05 , 0.9 ]
  var(z) = 0.06
  p-value(z) = 0.1

 Odds Ratio ES:

 OR [ 95 %CI] = 0.22 [ 0.04 , 1.16 ]
  p-value(OR) = 0.09

 Log OR [ 95 %CI] = -1.51 [ -3.16 , 0.15 ]
  var(lOR) = 0.71
  p-value(Log OR) = 0.09

 Other:

 NNT = -6.55
 Total N = 20
```

Donde podemos encontrar, entre otras medidas del tamaño del efecto, la r de Pearson = 0.4 y su intervalo de confianza (-0.08,0.73). Además de la r de Pearson esta función nos proporciona la d de Cohen y su intervalo de confianza para un t-test independiente.

Igualmente, podemos utilizar su fórmula para calcular la r de Pearson:

```
r<-function( t, gl ){
  sqrt( t ^ 2 / ( t ^ 2 + gl ) )
}
r( fitIndep[[1]], fitIndep[[2]] )
```

```
    t
0.401
```

O también la d de Cohen para los t test independientes.

```
dCohen<-function( x, y ){
lx <- length(x)- 1
ly <- length(y)- 1
```

```
md  <- abs(mean(x) - mean(y))
csd <- lx * var(x) + ly * var(y)
csd <- csd/(lx + ly)
csd <- sqrt(csd)
cd  <- md/csd
  }
d <- dCohen( df$medida[ df$grupo == 1 ], df$medida[ df$grupo == 0 ] )
```

O también la función `cohensD()` del paquete `lsr`.

```
library(lsr)
cohensD( df$medida[ df$grupo == 1 ] , df$medida[ df$grupo == 0 ] )
```

```
[1] 0.831
```

4.2. Comparaciones entre dos o más grupos

Vamos a generalizar el concepto de comparación entre dos grupos que hemos visto anteriormente a más grupos. Para ello consideraremos dos tipos de técnicas:

- Paramétricas: cuando la distribución de los datos de la muestra tiene cierta distribución particular usaremos el modelo ANOVA.
- No paramétricas: no será necesario suponer nada sobre la distribución de los datos. Estas pruebas se estudiarán en el apartado "Contrastes no paramétricos: datos ordinales" del próximo tema.

Borramos del entorno de trabajo el anterior conjunto de datos porque usaremos esta variable para un ejemplo general en el siguiente apartado.

4.2.1. ANOVA

El análisis de varianza, ANOVA, es una generalización de los t-test para dos muestras al caso de diseños con más de dos muestras.

A la variable categórica que define los grupos que deseamos comparar la llamamos variable independiente o factor y a la variable cuantitativa en la que

deseamos comparar los grupos la llamamos variable dependiente.

Los factores o variables independientes pueden ser a su vez de dos tipos: que varían entre sujetos (*between subjects*) o que varían dentro de los sujetos, conocidos también como intra sujetos (*within subjects*).

Esta diferenciación será muy importante a la hora de elegir el modelo ANOVA a llevar a cabo, ya que hay varios.

- Los factores que varían entre sujetos (*between*) son los que no se miden dos (o más) veces para un mismo sujeto: la edad (un sujeto no puede tener dos edades diferentes), el género, la raza... o simplemente los que, según el experimento, solo afectan una vez a cada sujeto. A cada sujeto se le asigna un único valor para ese factor.

- Los factores que varían dentro de los sujetos (*within*) son los que se miden varias veces para el mismo sujeto. Por ejemplo: seguimiento del mismo tratamiento tomando muestras en varios momentos. Los momentos en los que se toman las muestras son niveles y el tiempo es el factor intra sujetos ya que para un mismo sujeto se repiten las mediciones. A cada sujeto se le asignan tantos valores de ese factor como niveles tenga éste.

¿Por qué usar un nuevo modelo diferente de los t-test?

Ya se ha comentado con anterioridad el desorbitado crecimiento del error de Tipo I al crecer las comparaciones[1]. Para solventar esto podemos aplicar el modelo ANOVA.

El conjunto de datos que utilizaremos para ilustrar todos los modelos ANOVA que vamos a estudiar es

```
df <- read.table( "ejemplo.csv", sep = ";", head = TRUE )
head( df )
```

```
  id genero raza m0 m1 m3
1  1      1    3 35 25 16
2  2      2    1 37 23 12
```

[1] Si cada test tiene un nivel de significación $\alpha = 0.05$ la probabilidad de no cometer error de Tipo I al hacer N comparaciones es 0.95^N. Así, para 3 grupos hay un $100(1-(0.95)^3) = 14.3\,\%$ de probabilidad de cometer error de tipo I, no aceptar H_0 cuando es verdadera, y conforme crece N lo hace la probabilidad de cometer este tipo de error.

```
3  3       2     1 36 22 14
4  4       1     2 34 21 13
5  5       2     3 60 43 22
6  6       1     3 54 46 26
```

donde se mide la efectividad de un tratamiento a un conjunto de enfermos de una enfermedad rara. Se realiza una prueba de falta de capacidad de raciocinio antes de ser expuestos a este tratamiento (mes cero), cuando se lleva un mes de tratamiento y cuando se llevan tres meses. A simple vista se aprecia una gran mejora. Además, se diferencian los sujetos por género y por raza. Podemos identificar los factores:

- género: varía entre sujetos (*between*).
- raza: varía entre sujetos (*between*).
- mes: varía dentro de los sujetos (*within*) y a su vez tiene 3 niveles: m0 (mes 0), m1 (mes 1) y m3 (mes 3).

En nuestro conjunto de datos, los factores vienen dados como variables numéricas donde por ejemplo, la edad 1 puede significar adulto y la edad 2 anciano. Sin embargo, los factores en R deben ser tratados como variables categóricas lo cual se consigue mediante la función `factor()`.

```
df$id     <- factor( df$id )
df$raza   <- factor( df$raza )
df$genero <- factor( df$genero )
```

Si se desea comprobar que una variable es un factor se utiliza la función `is.factor()`.

```
is.factor( df$genero )
```

```
[1] TRUE
```

4.2.1. ANOVA de una vía (entre sujetos)

El ANOVA de una vía (*one way ANOVA*), también conocido como ANOVA de un factor, examina la igualdad de las medias de la población para un resultado cuantitativo y una única variable categórica con dos o más niveles de tratamiento.

Cada sujeto estará expuesto a un único nivel de este tratamiento. La hipótesis nula H_0 es que no hay diferencia entre las medias y la alternativa, H_1 que al menos una de las medias difiere del resto.

El hecho de verificar la hipótesis nula de que hay igualdad de medias entre grupos se puede interpretar como que las observaciones proceden de un único grupo cuya media y variabilidad es la misma que la de cualquiera de los grupos por separado. ANOVA es la forma más simple de mediar la variabilidad.

La cuestión es que si alguno de los grupos presenta unos valores que en media se alejan del resto, esto se apreciará en el contraste como una fuente extra de variabilidad no explicable por el azar. Nosotros nos dedicaremos a identificar cuándo se rechaza la hipótesis nula (cuando el p-valor es menor que 0.05) y cuándo se acepta (cuando es mayor) [25].

Vamos a tomar para este ANOVA de un factor la variable independiente raza y la variable dependiente, los datos, m0 para inferir si el género influye en la prueba que hemos llevado a cabo.

4.2.1.1.1. Supuestos y robustez de ANOVA Hay una serie de premisas que se deben cumplir, ordenadas de mayor exigencia a menor, para poder llevar a cabo el modelo ANOVA de una vía:

Independencia de las observaciones Dos observaciones son independientes cuando una no depende de la otra. Es decir, el resultado de la primera observación no influye en el resultado de la siguiente.

Este supuesto es el maś importante. Su violación va a invalidar las conclusiones del análisis porque produce errores en el cálculo de las varianzas y, por tanto, en los intervalos de confianza y las pruebas de hipótesis deducidas.

La falta de independencia se produce por ejemplo cuando se trabaja con variables aleatorias que se observan a lo largo del tiempo, por ello, si sucede hay que cerciorarse de que el análisis es correcto aplicar este análisis o por el contrario la variable varía dentro de los sujetos [26].

En otro caso, es debido a errores en el muestreo: efecto aprendizaje, descuidos, falta de aleatorización... o a la existencia de otros factores que también

influyen en la variable respuesta, y no han sido tomados en consideración.

Homocedasticidad: Homogeneidad de varianzas. Tenemos que comprobar que las varianzas de los grupos son iguales. Para ello podemos emplear el test de Bartlett o el de Levene. Como ya se comentó en el capítulo 2 Introducción a los contrastes en la sección "Contrastes de homogeneidad de varianza", el test de Levene es menos sensible a la falta de normalidad que el de Bartlett. Sin embargo, si estamos seguros de que los datos provienen de una distribución normal, entonces el test de Bartlett es el mejor. Para ambos la hipótesis nula es la igualdad de varianzas entre grupos y la hipótesis alternativa es la no igualdad.

Para realizar el test de Bartlett la función que vamos a utilizar es `bartlett.test()` y para el test de Levene la función `LeveneTest()`.

Si falla la homocedasticidad, siempre que no haya grandes diferencias entre el número de observaciones en los distintos grupos, es decir, siempre que nuestro modelo sea balanceado el ANOVA sigue siendo fiable. Con otras palabras, ANOVA es robusto con respecto a esta condición.

Para nuestro conjunto:

```
fit2 <- bartlett.test( df$m0 ~ df$raza )
fit2
```

```
    Bartlett test of homogeneity of variances

data:  df$m0 by df$raza
Bartlett's K-squared = 3, df = 2, p-value = 0.2
```

El p-valor = 0.182 > 0.05 luego aceptamos la hipótesis nula; los grupos son homocedásticos.

Veamos ahora cómo actuar cuando no se da igualdad de varianzas:

```
dfIS <- InsectSprays
head( dfIS )
```

```
  count spray
1    10     A
2     7     A
```

```
3    20     A
4    14     A
5    14     A
6    12     A
```

```
fit3 <- bartlett.test( dfIS$count ~ dfIS$spray )
fit3
```

```
    Bartlett test of homogeneity of variances

data:  dfIS$count by dfIS$spray
Bartlett's K-squared = 26, df = 5, p-value = 0.00009
```

El p-valor < 0.05 luego los grupos no son homocedásticos.

La alternativa que se suele emplear cuando el diseño no es homocedástico para seguir adelante con el análisis es la función `oneway.test()` que lleva a cabo el test de Welch (conocido también como ANOVA heterocedástica).

```
oneway.test( dfIS$count ~ dfIS$spray )
```

```
    One-way analysis of means (not assuming equal variances)

data:  dfIS$count and dfIS$spray
F = 36, num df = 5, denom df = 30, p-value = 8e-12
```

se obtiene que se rechaza la hipótesis nula, esto eso, hay diferencia entre grupos.

Otra opción es considerar las pruebas no paramétricas que se estudiarán a continuación.

Normalidad La distribución de los datos debe aproximarse a la de una normal. Ya hemos visto anteriormente en la subsección "Contrastes de normalidad" las herramientas para comprobarlo.

El contraste de ANOVA es robusto frente a la violación del supuesto de normalidad, es decir, si los datos no son normales todavía da buenos resultados.

Para muchos problemas de la biología la falta de normalidad se soluciona

haciendo una transformación de los datos (por ejemplo, tomando logaritmos, en cuyo caso los datos seguían una distribución lognormal).

```
fit4 <- shapiro.test( df$m0[df$raza == 1] )
fit5 <- shapiro.test( df$m0[df$raza == 2] )
fit55 <- shapiro.test( df$m0[df$raza == 3] )
```

En este caso, el p-valor = 0.1433 > 0.05 luego la distribución de los datos es normal.

4.2.1.1.2. ANOVA de una vía con R Una vez comprobados los supuestos, queremos averiguar si la falta de raciocinio antes del tratamiento tiene que ver con la raza del paciente. Para ello llevaremos a cabo el análisis ANOVA de un factor.

Es muy importante el orden de los argumentos. El primero es siempre la variable dependiente (antes). Le sigue el símbolo tilde (~) y la variable independiente (raza). El último argumento de la función aov() es el nombre del conjunto de datos donde se encuentran los datos que queremos analizar. Este formato será igual para todos los análisis ANOVA que se hagan con la función aov.

```
#ANOVA de una vía (entre sujetos)
# VI: m0
# VD: grupo
fit1E <- aov( m0 ~ raza, data=df )
summary( fit1E )
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
raza         2    845     422    7.28 0.0062 **
Residuals   15    870      58
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos que el p-valor es 0.00178 < 0.05 luego se aprecia una diferencia de resultados para las distintas razas antes del tratamiento.

4.2.1.1.3. Tamaño del efecto ANOVA de 1 vía Ya hemos comentado la importancia del tamaño del efecto. Las medida más común para el ANOVA de una vía es `eta cuadrado` (η^2) que es la razón entre la suma de cuadrados del efecto y la

suma de cuadrados total (de todos los efectos y los residuos).

$$\eta^2 = \frac{SC_{efecto}}{SC_{total}}$$

Primero la realizamos a mano ya que mediante la función `summary()` se obtienen los datos necesarios:

$$\eta^2 = \frac{844.8}{844.8 + 870.3} = 0.492$$

y teniendo en cuenta que el tamaño para el ANOVA de una vía es

Efecto	Pequeño	Mediano	Grande
η^2	0.01	0.06	0.14

Luego en nuestro caso el efecto es grande.

Además, podemos calcular el intervalo de confianza para η^2

```
ci.pvaf(F.value=7.28, df.1=2, df.2=15, N=18, conf.level=.95)
```

```
$Lower.Limit.Proportion.of.Variance.Accounted.for
[1] 0.0641

$Probability.Less.Lower.Limit
[1] 0.025

$Upper.Limit.Proportion.of.Variance.Accounted.for
[1] 0.671

$Probability.Greater.Upper.Limit
[1] 0.025

$Actual.Coverage
[1] 0.95
```

Es decir, hay un 95% de probabilidad de que el verdadero valor de η^2 esté

en (0.064, 0.6) [27] y [28].

El valor de η^2 puede ser obtenido también mediante la función `etaSquared()` del paquete `lsr` cuyos argumentos son el modelo tipo ANOVA y el tipo de sumas de cuadrados. Además, devuelve también el valor de η_p^2 (eta cuadrado parcial), donde

$$\eta_p^2 = \frac{SC_{efecto}}{SC_{efecto} + SC_{residuos}}$$

```
library(lsr)
etaSquared( fit1E, type = 1, anova = FALSE )
```

```
     eta.sq eta.sq.part
raza  0.493       0.493
```

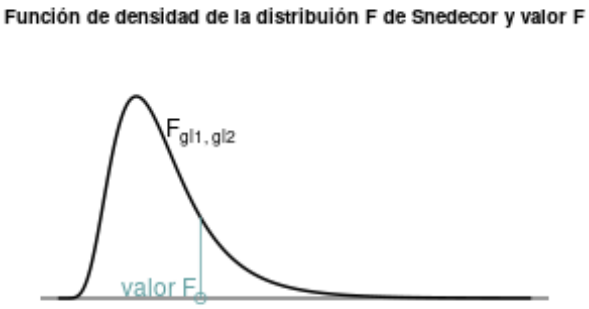
4.2.1. El estadístico F. Valor de F (F value)

Además del p-valor, que es en lo que nos hemos estado fijando para aceptar o rechazar la hipótesis nula, vemos que la función `summary()` nos proporciona el valor de F. Este valor es a partir del cual se obtiene el p-valor.

El valor F es la división de la varianza entre grupos (en nuestro caso tenemos dos grupos: femenino y masculino) con la varianza dentro de los grupos, es decir, la división entre la varianza explicada entre la varianza inexplicada.

Se considera que la varianza entre grupos es la explicada porque es la que se debe a la variable independiente, es decir, se debe a que se supone que entre los grupos habrá ciertas diferencias. Sin embargo, la varianza dentro de los grupos es lo que se llama error de varianza o varianza inexplicada porque se supone que dentro del mismo grupo los resultados serían los mismos [29].

Si en la población de la que proceden las muestras no hay diferencias reales entre los grupos, la varianza entre grupos será similar a la varianza dentro de los grupos y por tanto el cociente entre ambas estará cerca de 1. En el caso de que existan diferencias reales entre los grupos la varianza entre grupos será mayor que la varianza dentro de los grupos (el cociente entre ambas será mayor de 1). El estadístico que nos dice si las desviaciones respecto a ese valor de 1 son significativas es el estadístico F de Snedecor.



Como vemos, a mayor valor de F menor p valor porque el área bajo la curva que queda a la derecha es menor conforme crece el valor de F.

En nuestro caso la probabilidad de que el estadístico F sea mayor que 5.802 (el p-valor) es menor que 0.05 luego se rechaza la hipótesis nula y concluimos que las medias no son iguales.

4.2.1. Cómo comunicar los resultados de un ANOVA

Cuando nos disponemos a hacer públicos o a comunicar los resultados de nuestro análisis tenemos que dar detalles del valor de F y de los grados de libertad con los que el estadístico F ha sido calculado. Así, los grados de libertad que se usan para estimar el valor de F son los grados de libertad para el efecto del modelo (glM = 2) y para los residuos (glR = 15). Por lo tanto, la forma correcta de expresar los resultados del modelo sería

Hay un efecto significativo de la raza en el test, F(2,15) = 7.28, $p < 0.05$, $\eta^2 = 0.492$.

[24].

4.2.1. ANOVA de dos factores (de dos vías)

Éste análisis sirve para estudiar la relación entre una variable dependiente y dos variables independientes (dos factores). Cada factor puede tener a su vez varios niveles. El modelo ANOVA de dos vías no solo evalúa los efectos de las variables independientes sino que también puede evaluar los efectos de la interacción entre ellas

4.2.1.4.1. Supuestos Son los mismos que en el de una vía. Para la **raza** ya los habíamos comprobado, nos falta para el **género**:

```
fit6 <- bartlett.test( df$m0 ~ df$genero )
fit6
```

```
    Bartlett test of homogeneity of variances

data:  df$m0 by df$genero
Bartlett's K-squared = 0.05, df = 1, p-value = 0.8
```

Los p valores son > 0.05 así que podemos seguir con el análisis ANOVA.

4.2.1.4.2. ANOVA de dos vías con R La estructura del modelo es muy similar. Ahora aparecen los factores independientes multiplicados entre sí, lo que indica que su interacción será evaluada.

```
#ANOVA de dos vías (entre sujetos)
# VI: género (entre sujetos)
# VI: raza (entre sujetos)
# VD: joven
fit2E <- aov( m0 ~ raza * genero, data = df )
summary( fit2E )
```

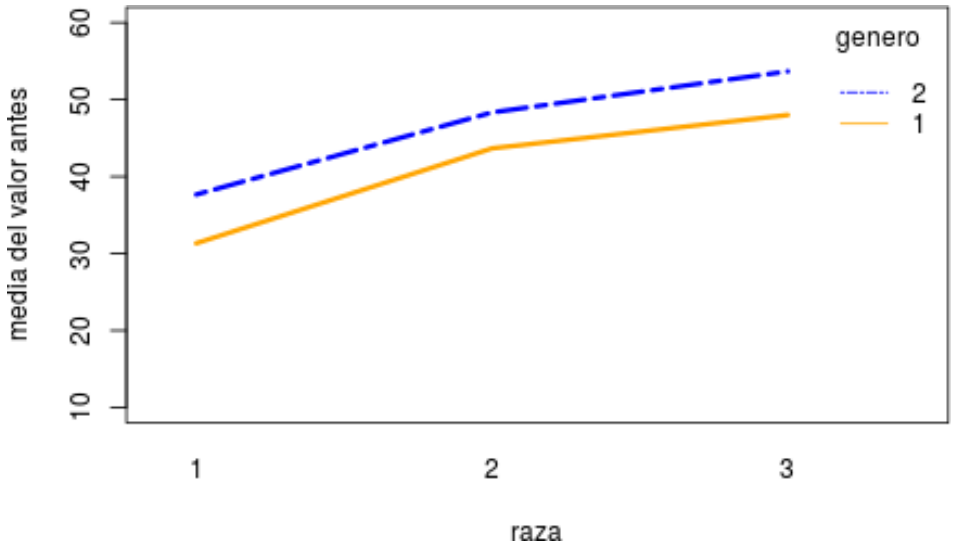
```
             Df Sum Sq Mean Sq F value Pr(>F)
raza          2    845     422    6.95 0.0099 **
genero        1    139     139    2.29 0.1565
raza:genero   2      2       1    0.02 0.9828
Residuals    12    729      61
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Así, el p-valor para la variable **raza** es menor que 0.05 por lo que ésta tiene un efecto significativo, el **género** no lo es y no hay interacción entre ambas porque el p-valor para la interacción es $0.9828 > 0.05$, esto es, se acepta la hipótesis nula.

Vamos a hacer una interpretación visual entre los efectos de los factores mediante un interaction plot.

```
interaction.plot( df$raza, df$genero, df$m0, ylim = c( 10, 60 ),
                  col = c( "orange", "blue" ), lty = c( 1, 12 ),
                  lwd = 3, ylab = "media del valor antes",
                  xlab = "raza", trace.label = "genero" )
```



El análisis entre **géneros** indica que la variable no tiene un efecto significativo, cosa que se aprecia en el interaction plot porque las líneas de ambos géneros están relativamente cerca. Sin embargo, la **raza** sí tiene un efecto significativo, ya que claramente las líneas crecen o decrecen conforme nos movemos entre las razas. Asimismo, no hay interacción entre **género** y **raza** porque las líneas del interaction plot van a la par, es decir, tienen la misma pendiente. Si hubiera interacción sería porque las líneas se cruzan o acabarían cruzándose al extenderlas.

```
fit2E2 <- aov( m0 ~ genero * raza, data = df )
summary( fit2E2 )
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
genero       1    139     139    2.29 0.1565
raza         2    845     422    6.95 0.0099 **
genero:raza  2      2       1    0.02 0.9828
Residuals   12    729      61
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nota: el orden en el que multiplicamos los factores independientes SI altera el producto cuando los datos no son balanceados. Es decir, en nuestro caso el modelo `fit2E2` arroja los mismos resultados. Sin embargo, si los datos no son balanceados no es así. Vamos a modificar nuestro conjunto de datos usando solamente las 9 primeras filas de manera que ya no hay el mismo de personas de cada **raza** ni el mismo número de personas de cada **género**.

```
dfMOD <- df[ 1:11, ]
dfMOD
```

```
   id genero raza m0 m1 m3
1   1      1    3 35 25 16
2   2      2    1 37 23 12
3   3      2    1 36 22 14
4   4      1    2 34 21 13
5   5      2    3 60 43 22
6   6      1    3 54 46 26
7   7      1    2 50 46 23
8   8      2    2 60 47 25
9   9      2    3 48 35 20
10 10      2    2 47 32 19
11 11      1    1 30 20 11
```

Ahora volvemos a hacer el ANOVA de 2 vías:

```
fit2E3 <- aov( m0 ~ raza * genero, data = dfMOD )
summary( fit2E3 )
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
raza         2    442   221.2    2.38   0.19
genero       1    241   240.7    2.59   0.17
```

```
raza:genero  2     10     5.0    0.05   0.95
Residuals    5    466    93.1
```

y cambiando el orden de los factores

```
fit2E4 <- aov( m0 ~ genero * raza, data = dfMOD )
summary( fit2E4 )
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
genero        1    149   149.3    1.60   0.26
raza          2    534   266.8    2.87   0.15
genero:raza   2     10     5.0    0.05   0.95
Residuals     5    466    93.1
```

¡Los resultados son diferentes! ¿Está mal? NO. Solamente tenemos que hacer una buena interpretación de los resultados.

Lo que sucede es que la función `aov()` usa por defecto las sumas de cuadrados de tipo I. Esto significa que en `fit2E3` el valor de F = 2.376 nos indica la variación explicada por la **raza** en la variable *m*0 (medición antes de aplicar el tratamiento) sin tener en cuenta nada más y de ahí se obtiene el p-valor y, a continuación, la variable **género** tiene un valor F = 2.585 que se interpreta de una forma algo diferente en el sentido de que como es la segunda variable incluida en el modelo mide la variación **adicional** que **género** explica pero que **raza** no explicaba.

En `fit2E4` vemos que los resultados son diferentes porque la variación en los resultados explicada únicamente por el **género** es 1.60 y la variación explicada por la **raza** tras tener en cuenta el **género** es 2.87. Si hacemos la interpretación correcta, todo será correcto. Sin embargo, la alternativa en el caso en que nos encontremos con un diseño no balanceado es utilizar la suma de cuadrados del Tipo III. Para ello, usaremos la función Anova() del paquete car. Esta función tiene como primer argumento un modelo lineal que se obtendrá mediante la función `lm()`.

```
options(contrasts = c("contr.sum","contr.poly"))
fit2E5 <- lm( m0 ~ genero * raza, data = dfMOD )
library( "car" )
Anova( fit2E5, type = "III" )
```

```
Anova Table (Type III tests)

Response: m0
            Sum Sq Df F value   Pr(>F)
(Intercept)  19389  1  208.26 0.000029 ***
genero         216  1    2.32     0.19
raza           470  2    2.52     0.17
genero:raza     10  2    0.05     0.95
Residuals      466  5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ahora sí se obtienen los mismos resultados independientemente del orden [30]

```
fit2E6 <- lm( m0 ~ raza * genero, data = dfMOD)
library( "car" )
Anova( fit2E2, type = "III" )
```

```
Anova Table (Type III tests)

Response: m0
            Sum Sq Df F value   Pr(>F)
(Intercept)   2945  1   48.46 0.000015 ***
genero          60  1    0.99    0.339
raza           449  2    3.69    0.056 .
genero:raza      2  2    0.02    0.983
Residuals      729 12
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.1. Tipos de suma de cuadrados

La suma de cuadrados se utiliza para descomponer la variabilidad. Si tenemos la fórmula

```
mod <- aov( y ~ A * B, data = datos )
```

- Tipo I: Se conoce como suma de cuadrados secuencial porque tiene en cuenta el orden de los factores a la hora del cálculo de la suma de cuadrados. Los efectos se ajustan para aquellos que aparecen antes en la fórmula. A no tiene en cuenta nada. B es ajustada teniendo en cuenta A y la interacción A*B es ajustada teniendo en cuenta A y B.
- Tipo II: Se conoce como suma de cuadrados jerárquica porque los efectos se ajustan entre los que están al mismo nivel o en niveles inferiores, es decir, A se ajusta para B, B para A y A*B para A y B.
- Tipo III: Se conoce como suma de cuadrados marginal porque tiene en cuenta todos los factores en todo momento. A se ajusta para B y para $A*B$, B se ajusta para A y para $A*B$ y, por último, $A*B$ se ajusta para A y B.

Para datos balanceados, es decir, cada nivel tiene el mismo número de observaciones, serán equivalentes. Sin embargo, cuando no tengamos datos balanceados el adecuado es el tipo III como hemos explicado anteriormente [7].

4.2.1. Efectos fijos vs efectos aleatorios

Hasta ahora los factores han sido de efectos fijos. Esto ocurre cuando los niveles del factor son seleccionados específicamente por el experimentador ya que el interés del experimento se centra en conocer los efectos sobre la respuesta de estos niveles particulares. No es que en nuestra población haya ocurrido que hemos encontrado miembros de la raza 1, 2 y 3 sino que hemos buscado sujetos de dichas razas. Cuando ocurre lo contrario se dice que el factor es de efectos aleatorios.

Para llevar a cabo este análisis se utiliza la función `lme()` del paquete `nlme`. Esta función posee un argumento para la fórmula de factores fijos y otro para la de factores aleatorios. Veamos ejemplos que ilustran el modelo:

En 8 lugares de una isla (DBAN, LFAN, NSAN...) se eligen 4 parcelas cultivadas de maíz a las que se les aplica el mismo tratamiento.

```
#install.packages("DAAG")
library("DAAG")
dfA<-ant111b
head(dfA)
```

```
  site parcel code island  id plot trt ears harvwt
1 DBAN      I   58      1   3  3.0 111 43.5   5.16
2 LFAN      I   58      1  40  4.0 111 40.5   2.93
3 NSAN      I   58      1 186  5.5 111 20.0   1.73
4 ORAN      I   58      1 256  4.5 111 42.5   6.79
5 OVAN      I   58      1 220  3.5 111 31.5   3.25
6 TEAN      I   58      1  77  5.0 111 32.5   2.65
```

Hay dos niveles de variación aleatoria: lugares y parcelas dentro de los lugares. La variación entre lugares puede ser, por ejemplo, debido a variaciones en la elevación del terreno o proximidad a fuentes de agua. Entre parcelas se espera que estos factores no afecten pero sí lo hagan otros como la fertilidad del terreno o el drenaje.

- factores fijos: no hay
- factores aleatorios: lugar y parcela.

Si no solo incluimos en el modelo el lugar sino también las parcelas el modelo se satura porque contiene todo, es decir, hay tantos efectos aleatorios como puntos en el conjunto de datos, así que nos libramos del nivel inferior:

```
#install.packages("nlme")
library(nlme)
modA <- lme( fixed = harvwt ~ 1, random = ~1 |site, data= dfA)
summary(modA)
```

```
Linear mixed-effects model fit by REML
  Data: dfA
  AIC BIC logLik
  100 105  -47.2

Random effects:
 Formula: ~1 | site
        (Intercept) Residual
StdDev:        1.54     0.76

Fixed effects:  harvwt ~ 1
            Value Std.Error DF t-value p-value
(Intercept)  4.29      0.56 24    7.66       0
```

```
Standardized Within-Group Residuals:
    Min      Q1     Med      Q3     Max
-2.0349 -0.5607  0.0706  0.7437  1.8864

Number of Observations: 32
Number of Groups: 8
```

2)

```
library("DAAG")
dfB<-science
head(dfB)
```

```
  State PrivPub school class sex like Class
1   ACT  public      1     1   f    8   1.1
2   ACT  public      1     1   f    6   1.1
3   ACT  public      1     1   f    5   1.1
4   ACT  public      1     1   f    2   1.1
6   ACT  public      1     1   f    5   1.1
7   ACT  public      1     1   f    6   1.1
```

factores fijos: sexo y tipo (de escuela: pública y privada), factores aleatorios: escuela (dentro de las cuales se examinan ciertas clases). Variable dependiente: puntuación. modelo:

```
modB<- lme(fixed=like~sex + PrivPub, data=dfB,random=  ~ 1 | school/class,
                                                  na.action=na.omit)
summary(modB)
```

```
Linear mixed-effects model fit by REML
  Data: dfB
   AIC  BIC logLik
  5561 5593  -2775

Random effects:
 Formula: ~1 | school
        (Intercept)
StdDev:    0.000309
```

```
 Formula: ~1 | class %in% school
        (Intercept) Residual
StdDev:       0.566     1.75

Fixed effects:  like ~ sex + PrivPub
             Value Std.Error   DF t-value p-value
(Intercept)  5.02    0.0929 1316    54.1  0.0000
sex1        -0.09    0.0491 1316    -1.9  0.0636
PrivPub1    -0.21    0.0929   39    -2.2  0.0325
 Correlation:
         (Intr) sex1
sex1     -0.001
PrivPub1  0.384  0.012

Standardized Within-Group Residuals:
   Min     Q1    Med     Q3    Max
-2.693 -0.689  0.154  0.679  2.558

Number of Observations: 1383
Number of Groups:
           school class %in% school
               41                66
```

Otro más:

```
mod <- lme(fixed = yield ~ shade, random = ~ 1 |block/plot, data=kiwishade)
summary(mod)
```

```
Linear mixed-effects model fit by REML
  Data: kiwishade
  AIC BIC logLik
  269 281   -127

Random effects:
 Formula: ~1 | block
        (Intercept)
StdDev:        2.02
```

```
 Formula: ~1 | plot %in% block
        (Intercept) Residual
StdDev:        1.48     3.49

Fixed effects:  yield ~ shade
            Value Std.Error DF t-value p-value
(Intercept)  96.5      1.34 36    72.0  0.0000
shade1        3.7      1.14  6     3.2  0.0184
shade2        6.7      1.14  6     5.9  0.0011
shade3       -6.6      1.14  6    -5.8  0.0012
 Correlation:
       (Intr) shade1 shade2
shade1  0.000
shade2  0.000 -0.333
shade3  0.000 -0.333 -0.333

Standardized Within-Group Residuals:
   Min     Q1    Med     Q3    Max
-2.415 -0.598 -0.069  0.780  1.589

Number of Observations: 48
Number of Groups:
          block plot %in% block
              3              12
```

4.2.1. ANOVA con medidas repetidas (within factors)

Este modelo se utiliza cuando se toman medidas para el mismo sujeto en condiciones diferentes (es el equivalente al t-test dependiente o paired). Es decir, de una lista de sujetos a cada uno le corresponden varias medidas.

No podemos usar el ANOVA estándar porque se viola la hipótesis de independencia.

Una ventaja de este experimento es que tiene menor coste porque se necesitan menos sujetos ya que todos están expuestos a todas las condiciones del

experimento. Además, se controla mucho mejor la variabilidad interna.

En nuestro caso de estudio, la variable que varía entre sujetos es el mes y tiene tres niveles: m0, m1 y m3. [Más info en: [31]].

4.2.1.7.1. Supuestos El único supuesto necesario es el de esfericidad, que quiere decir que las varianzas de las diferencias entre todos los pares de medidas repetidas sean iguales. Para ello usamos la prueba de esfericidad de Mauchly. Si hay 3 niveles denominados A, B y C, entonces se tendrá esferidad si la varianza de las diferencias entre A y B es igual a la varianza de las diferencias entre A y C, igual a la de entre B y C. (4) Para evaluar la esfericidad vamos a usar el test de Mauchly cuyas hipótesis son:

$$H_0 : \text{hay esfericidad}$$
$$H_1 : \text{no hay esfericidad}$$

Si los datos violan la condición de esfericidad, cosa que es bastante común, hay algunas correcciones que se pueden hacer para conseguir un valor de F adecuado.

Las que vamos a utilizar son la **corrección de Greenhouse–Geisser** y la **corrección de Huynh–Feldt**. Ambos nos conducen a un factor de corrección (que no es otra cosa que un estadístico) que se aplica a los grados de libertad usados para evaluar el valor de F obtenido. No entraremos en detalles sobre cómo se calculan pero diremos que la de **HF** se calcula a partir de la de **GG** con la intención de ser menos conservativa. [Para los curiosos: [32]]

La corrección de Greenhouse-Geisser arroja un estadístico (número) ϵ. Si k es el número de grupos que se testean, esta corrección está entre $\frac{1}{k-1}$ y 1. Si ϵ se sitúa más cerca de la cota inferior $\frac{1}{k-1}$ entonces esta corrección no ha conseguido hacer las modificaciones necesarias para alcanzar la esfericidad y, por lo tanto, no podremos usar el p-valor obtenido. Por otra parte, si se excede de 0.75 debemos utilizar la corrección de Huynh–Feld porque cuando ϵ es grande, Greenhouse-Geisser tiende a realizar un análisis muy estricto. Cuando ϵ es menor que 0.4 ambas correcciones indican que la violación de la esfericidad está afectando a los p valores.

4.2.1.7.2. Estructura de los datos Si consideramos cuatro sujetos $S1,...,S4$ y cuatro niveles $n1,...,n4$ normalmente se nos proporcionarán los datos en una tabla como esta:

sujeto	n1	n2	n3	n4
S1	7	5	6	2
S2	1	0	3	0
S3	8	8	6	1
S4	4	3	1	2

Sin embargo, no es de esta forma como podremos realizar el análisis con R sino que hay que definir la variable entre sujetos y poner los niveles como una medida más, es decir:

sujeto	nivel	medida
S1	n1	7
S1	n2	1
S1	n3	8
S1	n4	4
S2	n1	5
S2	n2	0
S2	n3	8
...	...	...
S4	n4	2

Para hacer esta transformación del conjunto de datos utilizaremos el la función `melt()` del paquete `reshape2` donde seleccionaremos las columnas que queremos mantener y las que queremos escribir como una sola, además del nombre de la nueva variable que definirán y el nombre de la variable dependiente.

```
library( reshape2 )
df2 <- melt( df, id = c( "id", "genero", "raza" ),
                 measure = c( "m0", "m1", "m3" ),
                 variable.name = "mes", value.name = "valor")
head( df2 )
```

```
  id genero raza mes valor
1  1      1    3  m0    35
2  2      2    1  m0    37
3  3      2    1  m0    36
4  4      1    2  m0    34
5  5      2    3  m0    60
6  6      1    3  m0    54
```

4.2.1.7.3. ANOVA de medidas repetidas (un factor intra sujetos) con R Hay al menos dos formas de realizar este análisis con R. Con aov el modelo es:

```
mod <- aov( variable dependiente ~ factor + Error( sujetos / factor ) )
```

```
# IV (dentroDe): mes
# DV:            valor
fit1D <- aov(valor ~ mes + Error(id / mes ), data = df2 )
summary( fit1D )
```

```
Error: id
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 17   3247     191

Error: id:mes
          Df Sum Sq Mean Sq F value Pr(>F)
mes        2   5684    2842     142 <2e-16 ***
Residuals 34    680      20
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Como el p-valor es menor que 0.05 se rechaza la hipótesis nula, es decir, el mes tiene un efecto sobre la variable dependiente.

El problema es que aquí nada nos asegura la esfericidad. Podríamos llevar a cabo el test de Mauchly, sin embargo, si ocurre que este es significativo nos veríamos obligados a calcular las correcciones, proceso que puede resultar tedioso.

Por esta razón, vamos a utilizar la función `ezANOVA()` del paquete `ez` que lleva a cabo, además del modelo ANOVA el análisis de supuestos previo:

```
library( ez )
options( contrasts = c( "contr.sum", "contr.poly" ) )
fit12 <-ezANOVA(data = df2, dv = .( valor ), wid = .( id ),
                within = .( mes ), type = 3)
fit12
```

```
$ANOVA
  Effect DFn DFd   F        p p<.05   ges
2    mes   2  34 142 3.09e-17     * 0.591

$`Mauchly's Test for Sphericity`
  Effect     W       p p<.05
2    mes 0.523 0.00563     *

$`Sphericity Corrections`
  Effect   GGe    p[GG] p[GG]<.05   HFe    p[HF] p[HF]<.05
2    mes 0.677 2.3e-12         * 0.715 6.12e-13         *
```

El test de Mauchly nos da un p-valor = 0.005 < 0.05, es decir, rechaza la hipótesis nula y no podemos asumir la esfericidad. Como $\epsilon = 0.67 < 0.75$ usamos la corrección de Greenhouse–Geisser que nos dice que el p-valor = 2.298E10-12 < 0.05, esto es, que el mes tiene un efecto significativo.

Un modelo para resolver este tipo de problema en el que hay algún factor intra sujetos y que no precisa de esfericidad es el MANOVA [24].

4.2.1. Modelos mixtos ANOVA

Hasta ahora teníamos ANOVA sin medidas repetidas donde las medidas deben ser independientes y hay un sujeto para cada grupo y ANOVA con medidas repetidas donde cada sujeto se mide varias veces bajo diferentes condiciones. ¿Y si no se da una condición o la otra sino ambas? Utilizaremos ANOVA mixto (*mixed ANOVA*).

4.2.1.8.1. ANOVA mixto con R El modelo es:

```
mod <- aov( variable dep ~ factores de interes multiplicados entre sí +
          Error(Sujetos/factoresIntraSujetosMultiplicadosEntreSí)
)
```

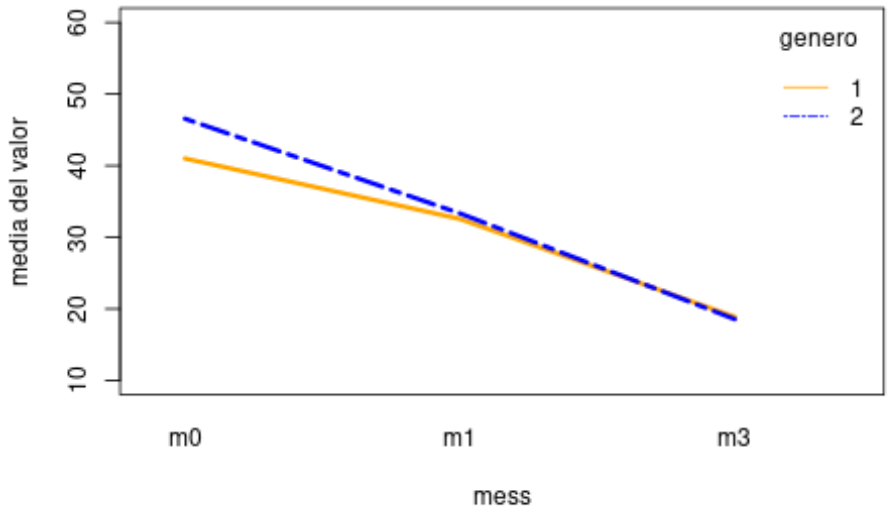
Si tenemos un factor entre sujetos y un factor intra sujetos:

```
# 2x2 mixed:
# IV between: género
# IV within:  mes
# DV:         valor
fitM <- aov( valor ~ mes * genero + Error( id / mes ), data = df2 )
summary( fitM )
```

```
Error: id
          Df Sum Sq Mean Sq F value Pr(>F)
genero     1     54      54    0.27   0.61
Residuals 16   3193     200

Error: id:mes
           Df Sum Sq Mean Sq F value Pr(>F)
mes         2   5684    2842  153.62 <2e-16 ***
mes:genero  2     88      44    2.38   0.11
Residuals  32    592      19
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Para interpetrar los resultados recurriremos de nuevo a un *interation.plot*

```
interaction.plot( df2$mes, df2$genero, df2$valor, ylim = c( 10, 60 ),
         col = c( "orange", "blue" ), lty = c( 1, 12 ), lwd = 3,
         ylab = "media del valor", xlab = "mess",  trace.label = "genero")
```



El análisis entre **géneros** indica que el género no tiene un efecto significativo, cosa que se aprecia en el interaction plot porque las líneas de ambos géneros están muy juntas. El **mes** sí tiene un efecto significativo, es decir, los grupos cambian a través del tiempo (el valor va decreciendo). Por último no hay interacción entre **género** y **mes** porque los grupos cambian a través del tiempo pero de forma muy similar, casi paralelamente llegando a tocarse pero no a cruzarse.

Con `ezANOVA()`:

```
fit13 <-ezANOVA(data=df2,dv= .( valor ), wid =.(id),between= .(genero),
                within = .( mes ),type = 3)
fit13

$ANOVA
      Effect DFn DFd       F        p p<.05     ges
2     genero   1  16   0.271 6.10e-01       0.0141
3        mes   2  32 153.615 3.93e-17     * 0.6003
4 genero:mes   2  32   2.381 1.09e-01       0.0227

$`Mauchly's Test for Sphericity`
```

```
      Effect     W       p p<.05
3        mes 0.402 0.00107     *
4 genero:mes 0.402 0.00107     *

$`Sphericity Corrections`
      Effect   GGe    p[GG] p[GG]<.05   HFe    p[HF] p[HF]<.05
3        mes 0.626 1.63e-11         * 0.653 6.24e-12         *
4 genero:mes 0.626 1.34e-01           0.653 1.32e-01
```

Si ahora consideramos dos factores entre sujetos y un factor intra sujetos:

```
# 2x2 mixed:
# IV between: raza
# IV between: género
# IV within:  mes
# DV: valor
fit2M <- aov( valor ~ mes * genero * raza +
                    Error( id / mes ) , data = df2 )
summary( fit2M )
```

```
Error: id
            Df Sum Sq Mean Sq F value Pr(>F)
genero       1     54      54    0.33  0.575
raza         2   1217     608    3.74  0.055 .
genero:raza  2     26      13    0.08  0.923
Residuals   12   1950     163
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: id:mes
               Df Sum Sq Mean Sq F value  Pr(>F)
mes             2   5684    2842  256.23 < 2e-16 ***
mes:genero      2     88      44    3.97 0.03236 *
mes:raza        4    297      74    6.69 0.00091 ***
mes:genero:raza 4     29       7    0.65 0.63169
Residuals      24    266      11
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.1.8.2. Otros ANOVA mixtos:

- 2 factores intra sujetos (w1, w2)

```
fit2E <- aov(y ~ w1 * w2 + Error( sujetos/( w1 * w2) ), data = df )
```

- 1 factor entre sujetos (b1) y 2 intra sujetos (w1 y w2)

```
fit1D2E <- aov(y ~ b1 * w1 * w2 + Error( sujetos / ( w1 * w2 ) ), data = df )
```

- 2 factores entre sujetos (b1 y b2) y 1 intra sujetos (w1)

```
fit2D1E <- aov( y ~ b1 * b2 * w1 + Error( sujetos / ( w1 ) ) )
```

[33]

4.2.1. ANOVA y regresión lineal

Vamos a ver con un ejemplo que no es que ANOVA sea un caso particular de regresión lineal múltiple sino que son lo mismo. Primero hacemos el análisis de varianza

```
fit1E <- aov( m0 ~ raza, data = df )
summary( fit1E )
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
raza         2    845     422    7.28 0.0062 **
Residuals   15    870      58
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

y mostramos las medias para cada raza

```
print( model.tables( fit1E, "means" ) )
```

```
Tables of means
Grand mean

43.8

 raza
```

```
raza
   1    2    3
34.5 46.0 50.8
```

Si ahora hacemos regresión lineal

```
summary( lm( m0 ~ factor( raza ), data = df ) )
```

```
Call:
lm(formula = m0 ~ factor(raza), data = df)

Residuals:
   Min     1Q Median     3Q    Max
-15.83  -3.33   1.25   3.79  14.00

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)      43.78       1.80   24.38  1.8e-13 ***
factor(raza)1    -9.28       2.54   -3.65   0.0024 **
factor(raza)2     2.22       2.54    0.88   0.3953
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.62 on 15 degrees of freedom
Multiple R-squared:  0.493, Adjusted R-squared:  0.425
F-statistic: 7.28 on 2 and 15 DF,  p-value: 0.00617
```

podemos observar que el estadístico F (y en consecuencia el p-valor) es igual para ambos análisis.

En el modelo de regresión lineal lo que se está haciendo es representar los grupos mediante variables binarias. Lo hacemos con **raza**: como tiene 3 niveles se necesitan 2 variables binarias X_1 y X_2 definidas tal que así:

$$X_1 = \begin{Bmatrix} 0 & \text{si no es de raza 2} \\ 1 & \text{si es de raza 2} \end{Bmatrix}$$

$$X_2 = \begin{cases} 0 & \text{si no es de raza 3} \\ 1 & \text{si es de raza 3} \end{cases}$$

De esta forma hemos podido representar los 3 niveles del factor de la siguiente manera:

Grupo	X_1	X_2
1	0	0
2	1	0
3	0	1

El modelo de regresión lineal es $y = b_0 + b_1 X_1 + b_2 X_2$ y, por los resultados de la regresión

$$y = 34.5 + 11.5 X_1 + 16.33 X_2.$$

Vamos a repasar los resutlados para cada grupo:

raza 1: $y = 34.5 + 11.5(0) + 16.33(0) = 34.5$

raza 2: $y = 34.5 + 11.5(1) + 16.33(0) = 46$

raza 3: $y = 34.5 + 11.5(0) + 16.33(1) = 50.83$

Que coinciden con las medias calculadas.

Así, ANOVA nos da cada media y el p-valor que dice si al menos dos son diferentes (de forma significativa) mientras que la regresión nos da solo una media (intercept) y las diferencias entre esta y las otras, pero los p-valores evalúan estas diferencias. Se obtiene la misma información con una estructura diferente [34].

4.2.2. Depués del ANOVA

Si al realizar el ANOVA se obtiene que el experimento es significativo, es decir, el p-valor es inferior a 0.05 se rechaza la hipótesis nula de igualdad de medias nos va a interesar, naturalmente, saber entre qué grupos se han producido diferencias. Para abordar la cuestión se tienen:

- Contrastes *post hoc* o no planificados: cuando no tenemos una idea previa de en qué grupos eran de esperar las mayores diferencias.
- Comparaciones planificadas: cuando tenemos alguna sospecha de dónde se pueden encontrar las diferencias.

4.2.2. Contrastes *post hoc* o no planificados

Los tests *post hoc* consisten en comparar los grupos de dos en dos. Es como hacer varios t-test consecutivos pero controlando el error de tipo I. Muchos de estos modelos son conservadores, es decir, reducen el error de tipo I a costa de aumentar la posibilidad de errores del tipo II, luego puede que hayan diferencias que no sean detectadas. Dependiendo de las características de nuestro modelo es aconsejable utilizar las una u otra.

El método de **Bonferroni** simplemente divide el nivel de confianza ($\alpha = 0.05$) que queremos obtener por el número de test que hay que hacer. Este método se considera bastante conservador.

Si k es el número de grupos a comparar, Bonferroni impone a cada comparación un nivel de significación igual a $\frac{\alpha}{k}$, es decir, si hay 5 grupos este será $\frac{0.05}{5} = 0.01$.

Este ajuste se lleva a cabo usando el argumento `p.adj = "bonferroni"`en la función `pairwise.t.test()`.

```
fit1D <- aov(valor ~ mes + Error( id / mes), data = df2)
summary(fit1D)
```

```
Error: id
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 17   3247     191

Error: id:mes
          Df Sum Sq Mean Sq F value Pr(>F)
mes        2   5684    2842     142 <2e-16 ***
Residuals 34    680      20
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pairwise.t.test(df2$valor, df2$mes, p.adj = "bonferroni", paired = TRUE)
```

```
    Pairwise comparisons using paired t tests

data:  df2$valor and df2$mes

   m0    m1
m1 3e-09 -
m3 5e-10 2e-07

P value adjustment method: bonferroni
```

Todos los p-valores son menores que 0.05, es decir, todas las razas difieren entre ellas.

El ajuste de **Holm** compara secuencialmente el p-valor con una significación que se reduce para cada test. Es decir, en nuestro caso tenemos 3 grupos, lo que quiere decir que el primer p-valor se compara con el nivel de significación $\alpha = \frac{0.05}{3} = 0.017$, el segundo con $\alpha = \frac{0.05}{2} = 0.025$, y el último con $\alpha = \frac{0.05}{1} = 0.05$.

En general, este método se considera superior al ajuste de Bonferroni y se utiliza escribiendo el argumento `p.adj = "holm"` en la función `pairwise.t.test()`.

```
pairwise.t.test( df2$valor, df2$mes, p.adj = "holm", paired = TRUE )
```

```
    Pairwise comparisons using paired t tests

data:  df2$valor and df2$mes

   m0    m1
m1 2e-09 -
m3 5e-10 6e-08

P value adjustment method: holm
```

Cuando haya más de 6 grupos evitaremos usar estos tests [35].

Otro método que no es generalmente la opción más recomendada cuando tenemos otras a nuestra disposición es **Mínima diferencia significativa de**

Fisher (*Fishers Least Significant Difference (LSD)*) pero en caso de que sea necesario aplicarlo utilizaremos la usando el argumento `p.adj = "none"`en la función `pairwise.t.test()`.

```
pairwise.t.test( df$m0, df$raza, p.adj = "none" )
```

```
    Pairwise comparisons using t tests with pooled SD

data:  df$m0 and df$raza

  1     2
2 0.020 -
3 0.002 0.289

P value adjustment method: none
```

Como vemos, las diferencias residen entre las razas 1 y 2 porque el p-valor es $0.019 < 0.05$ y entre las razas 1 y 3 porque es también $0.021 < 0.05$.

Por último, cuando tenemos más de 6 grupos, el diseño es balanceado, es decir, todos los grupos tienen el mismo número de individuos y se da homocedasticidad usaremos **Tukey HSD** que compara cada grupo con los demás.

Recordamos que `fit1E` nos hacía ver que las medias entre todas las razas no son iguales.

Veamos dónde se encuentran esas diferencias:

```
fit1E <- aov( m0 ~ raza, data = df )
TukeyHSD( fit1E )
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = m0 ~ raza, data = df)

$raza
     diff     lwr  upr p adj
2-1 11.50  0.0768 22.9 0.048
3-1 16.33  4.9102 27.8 0.006
```

```
3-2  4.83 -6.5898 16.3 0.529
```

De nuevo, las diferencias residen entre las razas 1 y 2 y entre las razas 1 y 3 porque sus p valores son inferiores a 0.05.

(muy buen video Tukey HSD y Fisher LSD: https://www.youtube.com/watch?v=9iL8v6A-gi0)

4.2.2. Comparaciones planificadas

Supongamos que hemos rechazado la hipótesis nula del análisis de varianza y queremos ver entre qué grupos hay diferencias pero tenemos una sospecha honesta de dónde se encuentran.

Esto ocurre por ejemplo cuando a un grupo de control se le aplica un placebo, una cierta cantidad de un medicamento y una cantidad superior pero que no esperamos que mejore mucho los resultados del segundo grupo. Posiblemente sospechemos que al que se le administra el placebo va a obtener resultados muy diferentes al segundo y al tercero pero no pensamos encontrar diferencias estos últimos.

Supongamos que nosotros sospechamos que la raza 1 va a diferir mucho de las razas 2 y 3 pero entre ellas no difieren significativamente.

Los contrastes se hacen de 2 en 2. En nuestro caso habrá 2 contrastes porque tenemos 3 grupos (siempre uno menos).

El primer contraste comparará la raza 1 con el grupo constituido por la raza 2 y 3 y el segundo contraste comparará las razas 2 y 3. Lo primero que vamos a hacer es ver si el modelo es balanceado. Podemos usar la función `summary()`

```
summary( df2 )
```

```
      id     genero raza   mes         valor
 1     : 3   1:27   1:18   m0:18   Min.   :11.0
 2     : 3   2:27   2:18   m1:18   1st Qu.:21.2
 3     : 3          3:18   m3:18   Median :30.5
 4     : 3                         Mean   :31.8
 5     : 3                         3rd Qu.:42.2
 6     : 3                         Max.   :60.0
```

```
(Other):36
```

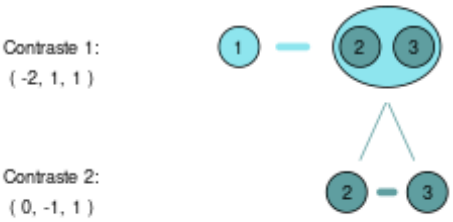
Lo que muestra que tenemos un diseño balanceado, con 6 replicaciones por **raza**. Para cada comparación necesitamos crear un vector de coeficientes y luego unirlos en una matriz mediante la función `cbind()`.

```
c1  <- c( -2,  1, 1 )
c2  <- c(  0, -1, 1 )
mat <- cbind( c1, c2 )
```

Los contrastes se realizan de dos en dos, es decir, uno de nuestros grupos será comparado a un conjunto comprendido por el resto de los grupos. Una vez que un grupo ha sido aisladamente comparado con un conjunto de grupos, el primero no vuelve a aparecer en los contrastes posteriores. En cada contraste se le asigna a cada grupo un peso que describe exactamente como será este contraste. Para asignar los factores se siguen las siguientes reglas:

- Si el grupo no interviene en el contraste se le asigna valor 0
- A los grupos que pertenecen a un conjunto se les dará un peso negativo y a los que pertenecen al otro se le darán pesos positivos.
- Para un contraste, los pesos asignados a cada grupo en cada conjunto debe ser igual al número de grupos en el conjunto opuesto. Es decir, en nuestro ejemplo primero comparamos el conjunto: raza 1 con el conjunto: raza 2 y raza 3. De forma que el grupo “raza 1” tendrá como valor -2 porque hay 2 grupos en el conjunto opuesto y los grupos “raza 2” y “raza 3” tendrán como valor 1 y 1 porque solo hay uno en el opuesto.
- Cada vector del contraste puede ser simplificado, es decir, si simplificamos dividiendo por algún divisor común el contraste seguirá siendo el mismo (esto es, el vector (2,2,-2) será equivalente al vector (1,1,-1)). [24]

Lo podemos ver con un esquema:



A la variable independiente le tenemos que asignar la matriz del constraste mediante la función `contrasts()`.

```
contrasts( df$raza ) <- mat
```

Observamos en qué se ha transformado:

```
df$raza
```

```
 [1] 3 1 1 2 3 3 2 2 3 2 1 3 1 1 1 2 3 2
attr(,"contrasts")
  c1 c2
1 -2  0
2  1 -1
3  1  1
Levels: 1 2 3
```

Ahora el factor tiene dos contrastes atribuidos.

Por último, realizamos el análisis mediante la función `aov()` como habitualmente pero usamos la opción `split` cuando vamos a mostrar el `summary()`. Esta

opción muestra la lista donde los resultados de los contrastes están guardados.

```
fit11<-aov( m0 ~ raza, data = df)
summary( fit11, split=list( raza = list( "C1" = 1, "C2" = 2 ) ) )
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
raza         2    845     422    7.28 0.0062 **
  raza: C1   1    775     775   13.35 0.0024 **
  raza: C2   1     70      70    1.21 0.2891
Residuals   15    870      58
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

que sugiere que hay diferencias significativas en el segundo contraste pero no en el primero.

También podemos usar la función

```
summary.lm( fit11 )
```

```
Call:
aov(formula = m0 ~ raza, data = df)

Residuals:
   Min     1Q Median     3Q    Max
-15.83  -3.33   1.25   3.79  14.00

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    43.78       1.80   24.38  1.8e-13 ***
razac1          4.64       1.27    3.65   0.0024 **
razac2          2.42       2.20    1.10   0.2891
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.62 on 15 degrees of freedom
Multiple R-squared:  0.493, Adjusted R-squared:  0.425
F-statistic: 7.28 on 2 and 15 DF,  p-value: 0.00617
```

5 Contrastes no paramétricos

Álvaro Hernández Vicente

5.1. Contrastes paramétricos frente a no paramétricos

Una de las primeras dificultades con las que uno se encuentra al hacer estadística es decidir correctamente qué prueba utilizar en un caso concreto. Esta decisión depende del tipo de datos que estemos manejando (y las suposiciones que podemos hacer) y de lo que queramos contrastar.

Debido a la gran importancia que tienen las suposiciones sobre la distribución (normalidad, simetría, etc.) los contrastes se pueden dividir entre **paramétricos** y **no paramétricos**. La estadística **paramétrica** es aquella en la que se tiene (o se asume) cierta información sobre la distribución de probabilidad de la población y se quiere decidir sobre los parámetros que la definen. Por ejemplo, la prueba t supone que la población es normal y se decide sobre la media.

Los casos en los que no se tenga información sobre la distribución (o no se

Cómo citar este capítulo: Hernández-Vicente A. (2022). Contrastes no paramétricos. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (185-224). Editum. Ediciones de la Universidad de Murcia.

pueda asumir) son los que trata la estadística **no paramétrica**. También se suele denominar *libre de distribución* (debido a que algunos autores consideran una definición más amplia de estadística no paramétrica).

Se deben emplear pruebas no paramétricas en lugar de paramétricas cuando:

- los datos se puedan ordenar de alguna manera (ordinales) pero no haya normalidad. Por ejemplo, datos provenientes de escalas Likert donde *"1"* significa *"Muy en desacuerdo"* y *"10" "Muy de acuerdo"*;
- los datos son categóricos o nominales, esto es, se pueden distinguir en categorías. Por ejemplo, datos de personas según sexo o raza;
- los datos son numéricos pero no provienen de una normal. En este caso se podría intentar emplear transformaciones de datos para lograr normalidad.

Muchos de los métodos paramétricos, por suerte, funcionan bien cuando la normalidad solo se puede suponer aproximadamente. Por otro lado, los no paramétricos se ha visto que son casi tan capaces de detectar diferencias entre poblaciones como los paramétricos cuando se cumplen todos los supuestos, y cuando no se cumplen a menudo son más potentes en hacerlo. Por esa razón, algunos estadísticos prefieren utilizar contrastes no paramétricos frente a sus análogos paramétricos ver [36].

Un resumen de las pruebas que veremos en este tema junto con sus alternativas paramétricas se recogen en la siguiente tabla:

Tabla 5.1: *Pruebas paramétricas y no paramétricas*

Comparar \ Tipo de datos	Paramétricos	Ordinales	Categóricos
Dos grupos independientes	t independiente	Mann-Whitney	Exacto de Fisher
Dos grupos dependientes	t dependiente	Wilcoxon	McNemar
Dos o más grupos independientes	ANOVA de una vía	Kruskal-Wallis	Chi-cuadrado
Dos o más grupos dependientes	ANOVA medidas repetidas	Friedman	Q de Cochran

Nota: En la url: https://www.graphpad.com/support/faqid/1790/ [37] se puede encontrar más información y una tabla más amplia que la anterior.

5.2. Datos ordinales

En este apartado nos centramos en las diferentes pruebas que podemos utilizar cuando los datos de los que disponemos se pueden ordenar de alguna manera. Por ejemplo, las ya comentadas escalas Likert.

Las pruebas que veremos en este apartado se basan en el cálculo de los números de orden o rangos (del inglés *ranking*). Los números de orden de un vector son los valores que resultan de ordenarlos de menor a mayor y asignar a cada elemento su posición. Por ejemplo, si tenemos un vector con cinco valores el vector de números de orden asigna al menor de ellos un "1" y al mayor un "5". Veamos un ejemplo sencillo de cómo se calcula el vector de números de orden.

```
vector       <- c( 14, 10, 13, 11, 12, 17, 19 )
rangosVector <- rank( vector ) # números de orden
rbind( vector, rangosVector ) # mostrar los dos vectores por filas
```

```
             [,1] [,2] [,3] [,4] [,5] [,6] [,7]
vector         14   10   13   11   12   17   19
rangosVector    5    1    4    2    3    6    7
```

Como podemos ver, el primer valor del vector es "14". El primer número de orden (correspondiente al "14") indica la posición que le correspondería a ese valor si se ordenara el vector de menor a mayor. El "14" es el quinto valor más pequeño del vector (más pequeños son 10, 11, 12 y 13), por lo que si se ordenara quedaría en la posición "5". Así, el primer número de orden es "5".

¿Y si hubieran dos valores "14"? Al ordenar nos quedarían en las posiciones "5" y "6", ¿qué números de orden se les asignan? La función `rank()` por defecto les asigna a cada uno el valor medio "5.5", pero con el argumento `ties.method` se puede cambiar.

```
vector2       <- c( 14, 10, 13, 11, 12, 14, 19 )
rangosVector2 <- rank( vector2, ties.method = "average" )
rbind( vector2, rangosVector2 )
```

```
              [,1] [,2] [,3] [,4] [,5] [,6] [,7]
vector2       14.0   10   13   11   12 14.0   19
rangosVector2  5.5    1    4    2    3  5.5    7
```

5.2.1. Comparación entre dos grupos

Pasamos a ver algunas pruebas no paramétricas para comparar las distribuciones de dos grupos.

5.2.1. Prueba U de Mann-Whitney (suma de rangos de Wilcoxon)

La **prueba U de Mann-Whitney**, conocida simplemente como *U-test* y equivalente a la *prueba de los rangos sumados de Wilcoxon*, es la prueba no paramétrica alternativa al t-test para muestras independientes (*unpaired* en inglés)

Se utiliza cuando los datos son ordinales (que tienen un orden) pero no se puede asumir normalidad. Así, cuando tenemos muestras pequeñas se prefiere el U-test antes que el t-test. Al contrario de lo que mucha gente cree, y se encuentra erróneamente en bastantes artículos, el U-test requiere de homogeneidad de varianzas (homocedasticidad). Y otro detalle es que aunque el U-test se considera el equivalente no paramétrico al t-test, este compara medianas y no medias.

Este test, al utilizar números de orden o rangos, también es preferible cuando existen valores atípicos.

¿Cómo funciona la prueba U de Mann-Whitney?. Supongamos que tenemos un par de muestras y queremos contrastar si hay diferencias entre las poblaciones de las que provienen.

```
muestraA <- c( 1.1, 3.4, 4.3, 2.1, 7.0 , 2.5 )
muestraB <- c( 7.0, 8.0, 3.0, 5.0, 6.2 , 4.4 )
```

La idea que hay detrás del U-test es juntar las dos muestras y asignar a cada valor un número de orden, esto es, al valor más pequeño se le asigna un "1", al segundo más pequeño un "2", y así hasta el más grande (en este caso, el valor más grande es "8.0" y, por tanto, se le asigna un "12"). Si hubiera valores repetidos, llamadas ligaduras (o *ties* en inglés), se les asigna la media (en este caso, el valor

"7.0" está repetido y en lugar de asignarles "10" y "11" se les asigna "10.5" a cada uno).

```
muestraTotal  <- c( muestraA, muestraB ) # unión de las muestras
rangosMuestra <- rank( muestraTotal )
muestraTotal  <- cbind( muestraTotal, rangosMuestra )
muestraTotal
```

```
      muestraTotal rangosMuestra
 [1,]          1.1           1.0
 [2,]          3.4           5.0
 [3,]          4.3           6.0
 [4,]          2.1           2.0
 [5,]          7.0          10.5
 [6,]          2.5           3.0
 [7,]          7.0          10.5
 [8,]          8.0          12.0
 [9,]          3.0           4.0
[10,]          5.0           8.0
[11,]          6.2           9.0
[12,]          4.4           7.0
```

Después se suman los números de orden para cada muestra (el mínimo de estos dos valores, que llamamos R_1 y R_2, es lo que se conoce como **suma de rangos de Wilcoxon**). El estadístico U estará definido como

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2}$$
$$U_2 = R_2 - \frac{n_2(n_2+1)}{2}$$
$$U = \text{mín}(U_1, U_2)$$

donde n_1 y n_2 son los tamaños de cada muestra.

```
sumaRangosA <- sum( rangosMuestra[ 1:6 ] )
sumaRangosB <- sum( rangosMuestra[ 6:12 ] )
n1 <- length( muestraA )
n2 <- length( muestraB )
U1 <- sumaRangosA - n1 * ( n1 + 1 ) / 2
```

```
U2 <- sumaRangosB - n2 * ( n2 + 1 ) / 2
c( U1, U2 )
```

```
[1]  6.5 32.5
```

En la práctica, cuando aparecen ligaduras (valores repetidos) no se calcula el valor exacto del estadístico U; se calcula una aproximación.

En R. Para hacer todos estos cálculos con R podemos utilizar la función `wilcox.test()`.

```
wilcox.test( muestraA, muestraB )
```

```
    Wilcoxon rank sum test with continuity correction

data:  muestraA and muestraB
W = 6, p-value = 0.08
alternative hypothesis: true location shift is not equal to 0
```

Nota: El estadístico U en R se denomina W. Por defecto en la función se tiene `paired = FALSE`. Nos aparece un mensaje de aviso (que no de error) debido a que las ligaduras no permiten calcular el p-valor exacto (se usa una aproximación a la normal).

El estadístico U se aproxima a una distribución normal (cuando los tamaños muestrales son mayores a 20 se considera muy buena) y, por tanto, se puede tipificar a una normal Z (hay fórmulas con las que se pueden calcular la media y la desviación típica). En R podemos emplear la función `wilcox_test()` del paquete `coin` (no preinstalado por defecto) que requerirá un fórmula como argumento (no la pareja de muestras). Es la opción recomendada si hay ligaduras en los datos.

```
## install.packages( "coin" )
library( "coin" )
grupo <- factor( c( rep( "A", length(muestraA) ),
                    rep( "B", length(muestraB) ) ) )
muestraTotal <- c( muestraA, muestraB )
wilcox_test( muestraTotal ~ grupo, distribution = "exact" )
```

```
    Exact Wilcoxon-Mann-Whitney Test
```

```
data:  muestraTotal by grupo (A, B)
Z = -2, p-value = 0.07
alternative hypothesis: true mu is not equal to 0
```

Tamaño del efecto. El tamaño del efecto para un U-test se calcula a partir de la Z que nos devuelve la función `wilcox_test()` con la fórmula

$$r = \frac{Z}{\sqrt{N}}$$

donde N es el tamaño total de la muestra ($N = n_1 + n_2$). El signo no importa, así que se comunica el valor absoluto de r.

Tabla 5.2: *Tamaño del efecto*

	Pequeño	Mediano	Grande
abs(r)	0.1	0.3	0.5

En nuestro caso anterior

```
N <- length( muestraA ) + length ( muestraB )
# tamaño muestral
wilcoxTest <- wilcox_test( muestraTotal ~ grupo, distribution = "exact" )
tamañoEfecto <- statistic( wilcoxTest ) / sqrt( N )
tamañoEfecto
```

```
[1] -0.533
```

Ejemplo. Supongamos que queremos saber las diferencias de opinión entre jóvenes y adultos sobre una cierta ley. Para ello se ha pasado una encuesta en la que se ha puntuado de 1 a 10 el grado de acuerdo (siendo "1" *"Muy en desacuerdo"* y *"10" "Muy de acuerdo"*). ¿La respuesta de los jóvenes es más negativa que la de los adultos?

```
dfEncuesta <- read.table( "feir50A-encuesta.csv" , header = TRUE, sep = ";" )
head( dfEncuesta )
```

```
  grupo opinion
1 joven       3
```

```
2 joven       4
3 joven       5
4 joven       4
5 joven       8
6 joven       4
```

```
wilcox.test( dfEncuesta$opinion ~ dfEncuesta$grupo, alternative = "greater" )
```

```
    Wilcoxon rank sum test with continuity correction

data:  dfEncuesta$opinion by dfEncuesta$grupo
W = 78, p-value = 0.02
alternative hypothesis: true location shift is greater than 0
```

Como el p-valor es menor a 0.05 rechazamos la hipótesis nula a favor de la alternativa de que los adultos están más de acuerdo con la ley.

```
wilcoxTest<- wilcox_test(dfEncuesta$opinion ~dfEncuesta$grupo,
          alternative = "greater",distribution = "exact" )
N          <- nrow( dfEncuesta ) # tamaño muestral
statistic( wilcoxTest ) / sqrt( N ) # tamaño del efecto
```

En valor absoluto obtenemos un tamaño del efecto de 0.4776, cercano a considerarse *grande*.

Nota: En este ejemplo solo hemos visto cómo aplicar la prueba U. Para que fuese completo se debería contrastar primero si cumple el requisito de homogeneidad de varianzas con, por ejemplo, el test de Levene.

5.2.1. Prueba de los rangos con signo de Wilcoxon

La **prueba de los rangos con signo de Wilcoxon** (*Wilcoxon Signed-rank test* en inglés), también conocida como test de Wilcoxon, es la versión no paramétrica del t-test para muestras dependientes. Igual que la prueba U de Mann-Whitney este test compara medianas en lugar de medias.

¿Cómo funciona el test de los rangos con signo de Wilcoxon?. Cuando tenemos muestras dependientes lo primero que se hace habitualmente es calcular sus diferencias. La idea de esta prueba es guardar los signos de esas diferencias

y, con los valores absolutos, asignar números de orden como en la prueba de Mann-Whitney (quitando los de diferencia nula).

```
grupoAntes       <- c( 2, 4, 6, 1, 3 )
grupoDespues     <- c( 5, 2, 7, 1, 6 )
grupoDiferencia  <- grupoAntes - grupoDespues
rangosDiferencia <- rank( abs( grupoDiferencia[ grupoDiferencia != 0 ] ) )
## el resultado de los números de orden es c( 3.5, 2.0, 1.0, 3.5 )
## añadimos un 0 a mano para cuadrar la tabla
rangosDiferencia <- c( 3.5, 2.0, 1.0, 0, 3.5 )
dfGrupo <- data.frame( grupoAntes, grupoDespues, sign( grupoDiferencia ),
                       abs( grupoDiferencia ), rangosDiferencia )
names(dfGrupo ) <- c( "antes", "despues", "signo", "diferencia", "rangos" )
dfGrupo
```

```
  antes despues signo diferencia rangos
1     2       5    -1          3    3.5
2     4       2     1          2    2.0
3     6       7    -1          1    1.0
4     1       1     0          0    0.0
5     3       6    -1          3    3.5
```

Una vez hecho esto hay que sumar, por un lado, los números de orden que provienen de diferencias positivas y, por otro lado, los de diferencias negativas (los de diferencias nulas no se suman). El mínimo de estos dos valores es el valor del estadístico `W` de Wilcoxon.

$$W_+ = \sum_{\text{signo}>0} \text{rango}(i)$$

$$W_- = \sum_{\text{signo}<0} \text{rango}(i)$$

$$W = \text{mín}(W_+, W_-)$$

donde rango(i) es el número de orden que le corresponde a la observación i.

```
sumaPositivos <- sum( dfGrupo[ dfGrupo$signo ==  1, ]$rangos )
sumaNegativos <- sum( dfGrupo[ dfGrupo$signo == -1, ]$rangos )
c( sumaPositivos, sumaNegativos )
```

```
[1] 2 8
```

En R. Para hacer estos cálculos con R podemos utilizar la función `wilcox.test()` con el argumento `paired = TRUE`.

```
wilcox.test( grupoAntes, grupoDespues, paired = TRUE )
```

```
    Wilcoxon signed rank test with continuity correction

data:  grupoAntes and grupoDespues
V = 2, p-value = 0.4
alternative hypothesis: true location shift is not equal to 0
```

Nota: El estadístico se denomina `V` en lugar de `W`. Cuando hay ceros o ligaduras se calcula un p-valor aproximado.

Este estadístico `W` (denominado `V` en R) también se puede tipificar a un valor Z (hay fórmulas para la media y la varianza) que será útil para calcular posteriormente el tamaño del efecto. En R se puede utilizar la función `wilcoxsign_test()` del paquete `coin`. Es el recomendado cuando hay ligaduras o ceros.

```
## install.packages( "coin" )
library( "coin" )
wilcoxsign_test( grupoAntes ~ grupoDespues, distribution = "exact" )
```

```
    Exact Wilcoxon-Pratt Signed-Rank Test

data:  y by x (pos, neg)
     stratified by block
Z = -1, p-value = 0.4
alternative hypothesis: true mu is not equal to 0
```

Tamaño del efecto. El tamaño del efecto para un test de Wilcoxon se calcula y considera de manera similar al del test de Mann-Whitney

$$r = \frac{Z}{\sqrt{N}}$$

donde N es el tamaño total de la muestra (aunque hayan sido observaciones sobre los mismos sujetos).

En nuestro caso anterior

```
N <- length( grupoAntes ) + length ( grupoDespues ) # tamaño muestral
wilcoxSignTest <- wilcoxsign_test( grupoAntes ~ grupoDespues,
                  distribution = "exact" )
statistic( wilcoxSignTest ) / sqrt( N ) # tamaño del efecto
```

```
[1] -0.346
```

5.2.1. Otros contrastes

Test de Kolmogorov-Smirnov para dos muestras. Sirve también para comparar dos muestras independientes, es decir, es una alternativa al U-test. En este caso, sin embargo, se comparan distribuciones de probabilidad en general (posición, forma, etc.); por tanto, se podría dar el caso que diera resultados distintos a la prueba U. Ver `?ks.test()`.

Algunos test del siguiente apartado también se pueden utilizar para comparar dos grupos.

5.2.2. Comparación entre más de dos grupos

En este apartado pasamos a ver las pruebas no paramétricas para contrastar si hay diferencias entre más de dos grupos. Además, para los casos en los que se obtengan resultados significativos veremos cómo realizar análisis post-hoc controlando la significación.

5.2.2. Prueba H de Kruskal-Wallis

La **prueba H de Kruskal-Wallis** es la alternativa no paramétrica al modelo ANOVA de una vía para comparar más de dos grupos independientes. Cuando

este test nos devuelve resultados significativos quiere decir que, en al menos dos grupos, hay diferencias pero no sabemos en cuáles de ellos (ni cuántas hay). Para saber qué grupos difieren entre sí se utilizan pruebas post-hoc (comparaciones dos a dos controlando la significación).

Al igual que la prueba U, bajo la hipótesis nula se asume que los datos provienen de la misma distribución. Esto quiere decir que es necesaria la homogeneidad de varianzas.

¿Cómo funciona la prueba H de Kruskal-Wallis?. Esta prueba, al igual que las dos vistas en el apartado anterior (para comparar dos grupos), utiliza la suma de rangos de Wilcoxon. En este caso, como comparamos más de dos grupos (digamos k grupos) llamaremos R_k a la suma de números de orden de cada grupo.

Este test lo que hace es calcular el estadístico H a partir de la fórmula

$$H = \frac{12}{N(N-1)} \sum_{i=1}^{k} \left(\frac{R_i^2}{n_i} \right) - 3(N+1)$$

donde N es el tamaño total de la muestra y n_i el tamaño de la muestra de cada grupo. O bien, en el caso de que hubieran ligaduras (*ties*, valores repetidos) la fórmula se corrige dividiendo la anterior

$$H = \frac{\frac{12}{N(N-1)} \sum_{i=1}^{k} \left(\frac{R_i^2}{n_i} \right) - 3(N+1)}{1 - \frac{\sum_{i=1}^{g} (t_i^3 - t_i)}{N^3 - N}}$$

donde g denota el número de grupos de ligaduras y t_i el total de números de orden ligados en el i-ésimo grupo.

Si el tamaño muestral es mayor a 5, el estadístico H sigue una distribución χ^2 con $k-1$ grados de libertad.

En R. Podemos utilizar la función `kruskal.test()`. Veamos un ejemplo de cómo se usa.

Ejemplo. Supongamos que queremos ver si hay diferencias en el número de piropos que recibe el profesorado de matemáticas que da clases en la facultad

de educación (E), con el de matemáticas (M) y con el de biología (B). (Datos simulados).

```
piropos   <- c( 6, 1, 7, 4, 8, 10, 5, 0, 2, 1, 2, 4, 4, 1, 4, 2, 3, 1, 5, 9)
facultad  <- factor( c( rep( "E", 7 ), rep( "M", 7 ), rep( "B", 6 ) ) )
dfPiropos <- data.frame( piropos, facultad )
head( dfPiropos )
```

```
  piropos facultad
1       6        E
2       1        E
3       7        E
4       4        E
5       8        E
6      10        E
```

```
kruskal.test( piropos ~ facultad, data = dfPiropos )
```

```
    Kruskal-Wallis rank sum test

data:  piropos by facultad
Kruskal-Wallis chi-squared = 6, df = 2, p-value = 0.04
```

Nota: El estadístico `H` en R se denomina *Kruskal-Wallis chi-squared*.

Al hacer la prueba con R obtenemos un p-valor de 0.0403 (menor a 0.05) luego rechazamos la hipótesis nula de que no hay diferencias en el número de piropos recibidos según la facultad.

Post-hoc. En la prueba de Kruskal-Wallis la hipótesis alternativa es que no todos los grupos tienen la misma distribución, esto es, que en al menos dos grupos hay diferencias. Para saber entre qué par de grupos se han encontrado diferencias realizamos un análisis post-hoc.

Se puede abordar de dos formas:

Primera opción: Hacer la prueba de Mann-Whitney sobre cada par de grupos. Como ya sabemos que esto aumenta el error de tipo I, después hacemos alguna corrección de la significación (Bonferroni u otra). Se puede hacer utilizando la función `pairwise.wilcox.test()`.

Recordemos que la corrección de Bonferroni es bastante estricta cuando el número de grupos es grande (más de 6). Para ver las posibles correcciones que permite esta función (y cuándo utilizarlas) podemos ver la ayuda de R con `?p.adjust`.

```
pairwise.wilcox.test(piropos,facultad,p.adjust= "bonferroni",
                exact = FALSE )
```

```
    Pairwise comparisons using Wilcoxon rank sum test with
    continuity correction


data:  piropos and facultad

  B    E
E 0.75 -
M 0.50 0.06

P value adjustment method: bonferroni
```

Hacer la prueba de Mann-Whitney sobre todas las parejas dispara el error de tipo I bastante rápido, por eso algunos autores recomiendan seleccionar primero los pares de grupos que más nos interese comparar y, de esta forma, hacer menos comparaciones.

Segunda opción: También podemos comparar cada pareja con Mann-Whitney, pero esta vez utilizando una desigualdad que se obtiene aplicando el test de Tukey (*Tukey's range test* en inglés). La fórmula que se utiliza es

$$|\bar{R}_u - \bar{R}_v| \geq z_{\alpha/k(k-1)} \sqrt{\frac{N(N+1)}{12}\left(\frac{1}{n_u} + \frac{1}{n_v}\right)}$$

donde $\bar{R}_u$ es la media de números de orden del grupo u, n_u el tamaño muestral del grupo u y en lo demás se repite notación. Esta es la alternativa que aparece explicada en [24] y [38]. Podemos utilizar la función `kruskalmc()` del paquete `pgirmess`.

```
## install.packages( "pgirmess" )
library( "pgirmess" )
kruskalmc( piropos ~ facultad, data = dfPiropos )
```

```
Multiple comparison test after Kruskal-Wallis
p.value: 0.05
Comparisons
    obs.dif critical.dif difference
B-E    3.61         7.88      FALSE
B-M    4.32         7.88      FALSE
E-M    7.93         7.57       TRUE
```

Esta función nos devuelve los dos valores de la desigualdad para cada par de grupos y una tercera columna indicando si se cumple (cuando lo hace es cuando se han encontrado diferencias).

Nota: La instalación del paquete `pgirmess` en GNU/Linux puede dar algún problema. En este caso habría que instalar las librerías `proj-dev` y `libgdal1`. Ver detalles en [39].

Tamaño del efecto. Cuando se hace una prueba de Kruskal-Wallis no hay una manera general para calcular el tamaño del efecto. Lo que se suele hacer es calcular el tamaño del efecto sobre las parejas con diferencias significativas del análisis post-hoc. Es decir, se calculan como ya vimos en el apartado de la prueba de Mann-Whitney sobre cada pareja.

En este caso, solo se han encontrado diferencias entre las facultades de educación (E) y matemáticas (M).

```
library( "coin" )
wilcoxTestEM <- wilcox_test( piropos ~ factor( facultad ),
distribution = "exact", data =
 dfPiropos[ facultad == "E" | facultad=="M", ] )
N <- nrow( dfPiropos[ facultad == "E" | facultad == "M", ] ) # tamaño muestral
statistic( wilcoxTestEM ) / sqrt( N ) # tamaño del efecto
```

```
[1] 0.638
```

5.2.2. Prueba de Friedman

La **prueba de Friedman** (también conocida como **ANOVA de Friedman**) es la alternativa no paramétrica al modelo ANOVA de un vía con medidas repetidas para comparar más de dos grupos dependientes. Igual que en la prueba de Kruskal-

Wallis para saber qué grupos difieren (en el caso de resultados significativos del test) también se utilizan pruebas post-hoc.

¿Cómo funciona la prueba de Friedman?. Se vuelven a utilizar los rangos de Wilcoxon, pero de una manera un poco diferente. Supongamos que en lugar de tener un par de muestras "antes" y "después" (como en la prueba de rangos con signo de Wilcoxon) tenemos más de dos (por ejemplo, algún seguimiento más prolongado sobre el tiempo) y queremos saber si hay diferencias entre algunas de ellas.

Lo que se hace es calcular los números de orden sobre cada individuo, por ejemplo, si hubieran tres grupos (`ayer`, `hoy` y `mañana`) se les asignarían los valores del uno al tres. Así, después se sumarían estos números de orden sobre cada grupo obteniéndose los R_i (se estarían sumando los unos, doses y treses de cada individuo).

Una vez que la suma de números de orden está calculada sobre cada grupo se calcula el estadístico F_r con la fórmula

$$F_r = \left(\frac{12}{Nk(k+1)} \sum_{i=1}^{k} R_i^2 \right) - 3N(k+1)$$

donde N es el número de individuos (no de tamaño de la muestra, ya que, los valores son apareados) y k es el número de grupos. Cuando hay ligaduras se utiliza la fórmula con corrección

$$F_r = \frac{12 \sum_{i=1}^{k} R_i^2 - 3N^2 k(k+1)^2}{Nk(k^2-1) - \sum \sum t(t^2-1)}$$

donde la doble suma indica que se sumas las ligaduras repetidas t sobre cada uno de los individuos.

Cuando el número de individuos (N) es mayor a 10, igual que en la prueba `H`, el estadístico F_r sigue una distribución χ^2 con $k-1$ grados de libertad.

En R. Podemos utilizar la función `friedman.test()`. Veamos un ejemplo de cómo se usa.

Ejemplo: Ahora se desea probar si hay diferencias en el número de piropos recibidos por el profesorado de didáctica de las matemáticas en la facultad de educación según el día en que se recogen los datos: el de la presentación (*t1*), el anterior al examen (*t2*) y después de la evaluación (*t3*). (Datos simulados).

```
piroposEd   <- c( 9, 5, 2, 6, 3, 1, 5, 5, 7, 11, 5, 1, 8, 4, 3, 10, 4,
1, 7, 3, 4 )
dia         <- factor( rep( c( "t1", "t2", "t3" ), 7 ) )
profesor    <- factor( rep( 1:7, each = 3 ) )
dfPiroposEd <- data.frame( piroposEd, dia, profesor )
head( dfPiroposEd )
```

```
  piroposEd dia profesor
1         9  t1        1
2         5  t2        1
3         2  t3        1
4         6  t1        2
5         3  t2        2
6         1  t3        2
```

```
## friedman.test ( piroposEd ~ dia | profesor )
friedman.test( piroposEd, dia, profesor )
```

```
    Friedman rank sum test

data:  piroposEd, dia and profesor
Friedman chi-squared = 7, df = 2, p-value = 0.03
```

Nota: El estadístico F_r en R se denomina *Friedman chi-squared*.

Obtenemos un p-valor de 0.02753 (menor a 0.05) luego rechazamos la hipótesis nula de que no hay diferencias en el número de piropos recibidos según el día.

Post-hoc. Igual que con la prueba de Kruskal-Wallis para saber sobre qué grupos se han encontrado diferencias hacemos un análisis post-hoc. Y de nuevo, se puede abordar de dos formas (las análogas a Kruskal-Wallis):

Primera opción: Hacer la prueba de los rangos con signo de Wilcoxon sobre cada pareja. Se puede utilizar la función `pairwise.wilcox.test()` con el

argumento `paired = TRUE` y la corrección que se quiera.

```
pairwise.wilcox.test(piroposEd, dia, p.adjust = "bonferroni",
                         exact = FALSE,paired = TRUE )
```

```
Pairwise comparisons using Wilcoxon signed rank test with continuity correctio

data:  piroposEd and dia

   t1  t2
t2 0.1 -
t3 0.1 0.4

P value adjustment method: bonferroni
```

Segunda opción: Igual que con Kruskal-Wallis se utiliza una desigualdad que se obtiene a partir del test de Tukey. En este caso

$$|\bar{R}_u - \bar{R}_v| \geq z_{\alpha/k(k-1)} \sqrt{\frac{Nk(k+1)}{6}}$$

donde se usa la misma notación que anteriormente. Una función para hacer esto con R es `friedmanmc()` del paquete `pgirmess`.

```
library( "pgirmess" )
friedmanmc( piroposEd, dia, profesor )
```

```
Multiple comparisons between groups after Friedman test
p.value: 0.05
Comparisons
      obs.dif critical.dif difference
t1-t2     7.0         8.96      FALSE
t1-t3     9.5         8.96       TRUE
t2-t3     2.5         8.96      FALSE
```

Tamaño del efecto. Ocurre lo mismo que con el tamaño del efecto para la prueba de Kruskal-Wallis. En este caso, se calculan de igual manera que para la prueba de rangos con signo de Wilcoxon sobre cada pareja.

En nuestro ejemplo solo se han encontrado diferencias entre el día de la presentación (*t1*) y el de la evaluación (*t3*).

```
dfPiroposEd13  <- dfPiroposEd[ dia == "t1" | dia == "t3", ]
wSignTest <- wilcoxsign_test( piroposEd ~ factor( dia ) | profesor,
                              distribution = "exact",
                              data = dfPiroposEd13 )
N     <- nrow( dfPiroposEd13 ) # tamaño muestral de t1 y t3
statistic( wSignTest ) / sqrt( N )
```

```
[1] 0.588
```

5.2.2. Otros

Prueba de Jonckheere-Terpstra. Se utiliza en lugar de la de Kruskal-Wallis si existe una ordenación natural del factor (un orden ascendente o descendente que tenga más sentido). En ese caso, esta prueba es más potente que la de Kruskal-Wallis.

Por ejemplo, supongamos que queremos comparar el número de piropos recibidos por los profesores según la cantidad de asignaturas que imparten (de 1 a 3). En este caso, una hipótesis alternativa puede ser que cuantas más asignaturas dan los profesores más piropos reciben.

En R se puede utilizar la función `jonckheere.test()` del paquete `clinfun`. Con el argumento `alternative =` se puede modificar la hipótesis alternativa.

```
## install.packages( "clinfun" )
library( "clinfun" )
piropos2    <- c( 3, 1, 1, 4, 8, 10, 5, 0, 2, 4, 1, 2, 4, 4, 10, 12,
14, 7, 11, 9)
asignaturas <- c( rep( 1, 7 ), rep( 2, 7 ), rep( 3, 6 ) )
head( cbind( piropos2, asignaturas ) ) # mostramos los datos
```

```
     piropos2 asignaturas
[1,]        3           1
[2,]        1           1
[3,]        1           1
[4,]        4           1
```

```
[5,]        8           1
[6,]       10           1
```

```
jonckheere.test( piropos2, asignaturas, alternative = "increasing" )
```

```
    Jonckheere-Terpstra test

data:
JT = 96, p-value = 0.02
alternative hypothesis: increasing
```

Un texto recomendable sobre esta prueba se puede encontrar en [24].

Prueba Q de Cochran. Es una alternativa a la prueba de Friedman cuando las variables son dicotómicas. Se estudiará en el apartado siguiente *Datos categóricos*.

Nota: Las fórmulas que se utilizan en los contrastes anteriores tienen su interpretación y su significado. Si alguien tiene interés en ver cómo se obtienen puede ver [38].

5.3. Datos nominales o categóricos

En el apartado anterior hemos visto métodos para comparar grupos cuando nuestros datos son de tipo ordinal (que tienen un orden) como, por ejemplo, las escalas Likert. En este apartado vamos a ver qué pruebas se pueden hacer cuando los datos son nominales (también conocidos como categóricos).

5.3.1. Introducción: tablas de contingencia

Supongamos que tenemos una muestra de personas en la que se indica para cada una si tiene problemas del corazón y si hace deporte.

```
deporte   <- c( "Sí", "No", "No", "Sí", "No" )
enfermo   <- c( "No", "No", "Sí", "No", "No" )
dfCorazon <- data.frame( deporte, enfermo )
dfCorazon
```

```
  deporte enfermo
1      Sí      No
2      No      No
3      No      Sí
4      Sí      No
5      No      No
```

En estos casos, en lugar de trabajar con un `data.frame` con un par de columnas de "Sí" o "No" lo que se hace es trabajar con tablas de contingencia (que significa posibilidad de que algo suceda) o frecuencias donde se indica el número de casos según ambas categorías. Se pueden utilizar las funciones `xtabs()` o `table()`.

```
tCorazon <- table( deporte, enfermo )
## tCorazon <- xtabs( ~ deporte + enfermo, data = dfCorazon)
tCorazon
```

```
       enfermo
deporte No Sí
     No  2  1
     Sí  2  0
```

Como vemos, con esta tabla podemos observar, por ejemplo, que hay dos personas de la muestra que hacen deporte y no están enfermos del corazón. Con la función `addmargins()` podemos completar la tabla con las sumas por filas y columnas.

```
addmargins( tCorazon )
```

```
       enfermo
deporte No Sí Sum
    No   2  1   3
    Sí   2  0   2
    Sum  4  1   5
```

Ahora podemos identificar fácilmente, por ejemplo, que de las cuatro personas que no están enfermas del corazón dos hacen deporte y dos no lo hacen.

Si queremos ver (en lugar del número de casos absolutos) la tabla de proporciones utilizamos la función `prop.table()`

```
tCorazonFrec <- prop.table( tCorazon )
tCorazonFrec
```

```
       enfermo
deporte  No  Sí
     No 0.4 0.2
     Sí 0.4 0.0
```

Y otra vez se pueden añadir las sumas por filas y columnas

```
addmargins( prop.table ( tCorazonFrec ) )
```

```
       enfermo
deporte  No  Sí Sum
    No  0.4 0.2 0.6
    Sí  0.4 0.0 0.4
    Sum 0.8 0.2 1.0
```

Para que nos aparezcan porcentajes (si es que nos gustan más) bastaría con multiplicar por cien

```
addmargins( prop.table ( tCorazonFrec ) ) * 100
```

```
       enfermo
deporte  No  Sí Sum
    No   40  20  60
    Sí   40   0  40
    Sum  80  20 100
```

Para ver más detalles sobre estas (y otras) funciones se puede consultar el libro de texto [7].

Si hubieran más de dos categorías (por ejemplo: hacer deporte, enfermo del corazón y fumar) se utiliza la función `ftable()`.

```
fuma      <- c( "Sí", "Sí", "Sí", "No", "Sí")
dfCorazon <- data.frame ( deporte, enfermo, fuma )
ftable ( table( fuma, deporte, enfermo ) )
```

```
             enfermo No Sí
fuma deporte
```

```
No    No          0  0
      Sí          1  0
Sí    No          2  1
      Sí          1  0
```

Nota: La función `table()` ignora los valores perdidos (NAs) por defecto. Para incluir NA como una categoría válida en el conteo de frecuencias hay que incluir el argumento `useNA = ''ifany''`.

5.3.2. Comparación entre dos o más proporciones/-frecuencias

En este apartado vamos a ver las diferentes pruebas que podemos utilizar para comparar dos (o más) variables categóricas. Si no obtuviéramos resultados significativos diríamos que las variables no están relacionadas entre sí (hipótesis nula).

5.3.2. Prueba χ^2 de Pearson

La **prueba χ^2 de Pearson** (pronunciada **ji-cuadrado** y a veces como **chi-cuadrado**) nos permite contrastar si existe relación entre dos variables categóricas mediante la tabla de contingencia cuando tenemos datos no pareados.

¿Cómo funciona la prueba \chi^2 de Pearson?. La idea se basa en medir la desviación de los datos observados en la tabla de frecuencias con respecto a los datos esperados por azar. Esto se hace aplicando la fórmula

$$\chi^2 = \sum_{i,j} \frac{(\text{observado}_{ij} - \text{esperado}_{ij})^2}{\text{esperado}_{ij}}$$

donde i, j recorren las filas y las columnas de la tabla de contingencia. Los valores observados son los que aparecen en las tablas de contingencia y los valores esperados se calculan a partir de las sumas de columnas y filas como pasamos a ver.

Supongamos, por ejemplo, que tenemos una muestra de 260 personas de un determinado país en la que se muestra su género y si la persona va en bicicleta al trabajo. Queremos ver si están relacionadas las variables `genero` con la de `bicicleta`.

```
personas  <- c( 71, 48, 65, 76 )
genero    <- c( "hombre", "hombre", "mujer", "mujer" )
bicicleta <- c( "Sí", "No", "Sí", "No" )
dfBici    <- data.frame( personas, bicicleta, genero )
addmargins( xtabs( dfBici ) )
```

```
          genero
bicicleta hombre mujer Sum
      No      48    76 124
      Sí      71    65 136
      Sum    119   141 260
```

Fijémonos en los hombres que sí se desplazan en bicicleta ($i = 2$ y $j = 1$). El valor que aparece en la tabla es 71, luego observado$_{21} = 71$. El valor esperado se calcula multiplicando el número de hombres que hay (119) por el número de personas que van en bicicleta (136) entre el número total de personas, esto es

$$\text{esperado}_{21} = \frac{\text{totalFila2} \times \text{totalColumna1}}{\text{totalMuestra}} = \frac{136 \times 119}{260} \approx 62.246$$

Por tanto, para la entrada de la fila 2 columna 1 el valor que se suma al estadístico es

$$\frac{(71 - 62.246)^2}{62.246} \approx 1.23$$

y, repitiendo el proceso en toda la tabla de contingencia y sumando se calcula el estadístico χ^2.

Como es obvio por el nombre, el estadístico sigue aproximadamente una distribución χ^2 con $(f-1)(c-1)$ grados de libertad, donde f indica el número total de filas y c el de columnas.

El problema de esta prueba surge de que utiliza una aproximación a la χ^2, es decir, surge cuando tenemos tablas de contingencia pequeñas. Así, cuando

tenemos tablas 2×2 Yates propuso una corrección a la fórmula, ver [24] para más información] que es la que utilizan algunas funciones de R.

Las únicas suposiciones que deben cumplirse para utilizar la prueba χ^2 de Pearson son:

- que los datos no estén apareados (dependientes). En esa situación se utilizaría la prueba de McNemar que veremos más adelante;
- y que las frecuencias esperadas no sean menores a 5. En tablas grandes es suficiente que el 80 % lo cumplan (nunca siendo menores a 1). Si no se da esta situación también se opta por usar el test exacto de Fisher. Estas condiciones también se conocen como *"condiciones de Cochran"*.

En R. Utilizamos la función `chisq.test()` indicando la tabla de contingencia

```
chisq.test( xtabs( dfBici ) )
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  xtabs(dfBici)
X-squared = 4, df = 1, p-value = 0.04
```

En este caso, como el p-valor es menor a 0.05 se rechaza la hipótesis nula de que las variables no guardan relación entre sí. Al ser una tabla 2×2 vemos que utiliza la corrección de Yates; para no hacerla basta poner el argumento `correct = FALSE`.

Nota: En R el estadístico se denomina `X-squared`.

Fuerza de asociación (tamaño del efecto). Como en esta prueba estamos contrastando si las variables están relacionadas (esto es, asociadas) se suele hablar de **fuerza de asociación** en lugar de tamaño del efecto.

Para medir la fuerza de asociación con R podemos utilizar la función `assocstats()` del paquete `vcd`. Esta función nos devuelve el valor de las tres medidas de asociación más frecuentes:

- El **coeficiente `phi`**: cuando tenemos una tabla 2×2 varía entre 0 y 1 y tiene una interpretación similar a la r de Pearson (más correlación cuanto más cercano a 1). En tablas mayores el límite superior sobrepasa el 1 y es más

difícil de interpretar.

- El **coeficiente de contingencia**: siempre varía entre 0 y 1 y se interpreta como la `phi`. El problema es que pocas veces alcanza el valor 1. Por esa razón Cramer propuso posteriormente la `V`.
- La **`V` de Cramer**: para tablas 2×2 coincide con la `phi`. Siempre varía entre 0 y 1, y en tablas mayores de 2×2 no le pasa lo que al coeficiente de contingencia, es decir, puede alcanzar su valor máximo 1.

```
## install.packages( "vcd" )
library( "vcd" )
assocstats( xtabs( dfBici ) )
```

```
                    X^2 df P(> X^2)
Likelihood Ratio 4.7785  1 0.028818
Pearson          4.7598  1 0.029131

Phi-Coefficient   : 0.135
Contingency Coeff.: 0.134
Cramer's V        : 0.135
```

Al ejecutar la función con el ejemplo anterior podemos observar que al tratarse de una tabla 2×2 las tres medidas ofrecen valores muy similares.

Otra medida que se suele utilizar además de estas es la **razón de oportunidades**, más conocida por el inglés *odds ratio*.

Post-hoc. Cuando hay más de de tres categorías en una variable (por ejemplo, una tabla 2×3) y se han obtenido resultados significativos nos puede interesar saber en qué niveles de la variable se han encontrado las relaciones.

Primera opción: ver los valores residuales de Pearson. Los valores residuales son las diferencias entre los valores observados y esperados, y si estos se dividen por la raíz de los esperados se obtienen los residuales de Pearson. Estos valores se estandarizan y se pueden comparar con la normal (por ejemplo, si supera 1.96 es significativo a un nivel de 0.05)

$$\text{residual Pearson}_{i,j} = \frac{\text{observado}_{ij} - \text{esperado}_{ij}}{\sqrt{\text{esperado}_{ij}}}$$

En R se calculan cuando hacemos nuestro test con `chisq.test()` pero no se muestran. Para verlos hacemos `chisq.test()$residuals` o `chisq.test()$stdres` para los estandarizados.

```
testChi <- chisq.test( xtabs( dfBici ) )
testChi$residuals
```

```
         genero
bicicleta hombre mujer
       No  -1.16  1.07
       Sí   1.11 -1.02
```

```
testChi$stdres
```

```
         genero
bicicleta hombre mujer
       No  -2.18  2.18
       Sí   2.18 -2.18
```

Nota: Aquí solo vemos cómo utilizar la función, por eso utilizamos la anterior tabla 2×2.

Si uno se fija en la fórmula puede ver que el estadístico χ^2 es la suma de los cuadrados de los residuales de Pearson. Así, lo bueno (y lo malo) de esta opción es que es una manera visual (y no muy exacta) de ver qué valores tienen más peso en el estadístico.

Segunda opción: partir la tabla en varias tablas 2×2 y a cada una hacerle una prueba χ^2 de Pearson corrigiendo la significación. En el ejemplo siguiente vamos a ver cómo partir una tabla 2×4 escogiendo cada pareja de columnas sin más.

Para corregir la significación podemos utilizar Bonferroni dividiendo el nivel de significación α entre el número k de comparaciones que realicemos. Si hacemos todas las comparaciones posibles, k se calcula como

$$k = \frac{r!}{2!(r-2)!} \cdot \frac{c!}{2!(c-2)!} = \frac{r(r-1)c(c-1)}{4}.$$

En muchos libros y apuntes [40, 41, 42] se recomienda una partición di-

ferente que mantiene el valor del estadístico (los estadísticos de las particiones suman el de la tabla original). En este caso, se utiliza un valor de k menos restrictivo, esto es, un valor que no reduzca tanto el nivel de significación ver [43]

$$k = \frac{r!}{2!(r-1)!} \cdot \frac{c!}{2!(c-1)!} = \frac{r \cdot c}{4}$$

Ejemplo: El conjunto de datos `HairEyeColor` de R contiene la distribución del color del pelo, el color los ojos y el género de 592 estudiantes de la Universidad de Delaware en 1974. ¿Están relacionados el género con el color del pelo?

Primero nos quedamos solo con el color del pelo y el género. Después le aplicamos la prueba χ^2

```
## trabajamos solo con el género y el color del pelo
tColorPelo <- margin.table( HairEyeColor, c( 3, 1 ) )
tColorPelo
```

```
        Hair
Sex      Black Brown Red Blond
  Male      56   143  34    46
  Female    52   143  37    81
```

```
chisq.test( tColorPelo )
```

```
    Pearson's Chi-squared test

data:  tColorPelo
X-squared = 8, df = 3, p-value = 0.05
```

Obtenemos un p-valor de 0.04613 (menor a 0.05) luego rechazamos la hipótesis nula de que no existe relación. El siguiente paso es ir escogiendo parejas de columnas y repitiendo la prueba χ^2 a cada una. En este caso, al haber cuatro columnas (los colores del pelo: moreno , castaño, rojo y rubio) todas las posibles parejas serían: 1-2, 1-3, 1-4, 2-3, 2-4 y 3-4. Si quisiéramos hacer comparaciones dos a dos sobre todas las parejas utilizaríamos $\alpha = \frac{0.05}{6} = 0.0083$.

Hagamos, por ejemplo, los casos 1-2 (moreno-castaño) y el 2-4 (castaño-rubio) para ver la dinámica. En este caso utilizamos $\alpha = \frac{0.05}{2} = 0.025$ por hacer solo dos comparaciones.

```
## seleccionamos las columnas 1 = moreno y 2 = castaño
t12ColorPelo <- tColorPelo[ , c( 1, 2 ) ]
t12ColorPelo
```

```
        Hair
Sex      Black Brown
  Male      56   143
  Female    52   143
```

```
chisq.test( t12ColorPelo )
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  t12ColorPelo
X-squared = 0.05, df = 1, p-value = 0.8
```

En este caso, como el p-valor sale mayor a 0.025 aceptamos la hipótesis nula de que no existe relación entre el género y el color del pelo moreno o castaño.

```
## seleccionamos las columnas 2 = castaño y 4 = rubio
t24ColorPelo <- tColorPelo[ , c( 2, 4 ) ]
t24ColorPelo
```

```
        Hair
Sex      Brown Blond
  Male     143    46
  Female   143    81
```

```
chisq.test( t24ColorPelo )
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  t24ColorPelo
X-squared = 6, df = 1, p-value = 0.01
```

Ahora el p-valor sale significativo (menor a 0.025), luego rechazamos la hipótesis nula de que no existe relación entre el género y el color del pelo castaño y rubio.

5.3.2. Test exacto de Fisher

El **test exacto de Fisher** aunque se denomina test es más bien un método de cálculo de las probabilidades de la χ^2 cuando el tamaño muestral es pequeño.

Uno de los problemas con la prueba χ^2 de Pearson es que utiliza una aproximación a la distribución χ^2 (de ahí el comentario de los valores mayores a cinco) y cuanto mayor es el tamaño muestral mejor se aproxima. En estos casos en los que se tiene poca muestra es donde se utiliza el método ideado por Fisher para calcular de manera exacta la probabilidad del estadístico de la χ^2.

Generalmente se utiliza con tablas de contingencia 2×2 y con poca muestra, aunque también se puede utilizar con tablas mayores (eso sí, tardaría más tiempo en calcular y en esos casos se podría usar la prueba χ^2 de Pearson). También suele utilizarse cuando no se cumplen las *condiciones de Cochran* y no podemos aplicar la prueba χ^2 de Pearson.

¿Cómo funciona el test exacto de Fisher?. Dada una tabla de contingencia con los valores marginales (los de los márgenes, las sumas de filas y columnas) se calcula la probabilidad de que, con esos valores marginales, se hayan dado por azar los valores de la tabla o más raros. Veamos un ejemplo.

Supongamos que tenemos una tabla de contingencia cualquiera

Tabla 5.3: *Tabla de contingencia. Prueba exacta de Fisher.*

	Categoría1	Categoría2	Total
Categoría3	a	b	a+b
Categoría4	c	d	c+d
Total	a+c	b+d	n=a+b+c+d

Suponiendo la hipótesis nula de independencia los valores marginales están fijos y solo hay un grado de libertad (esto es, sabiendo el valor de una celda de la tabla se pueden calcular los demás). La probabilidad de que en la posición de "a" aparezca el valor "a" por azar (y por tanto, que salgan "b", "c" y "d" en las demás) se calcula con la fórmula de la distribución hipergeométrica

$$p = \frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)!\,(c+d)!\,(a+c)!\,(b+d)!}{a!\,b!\,c!\,d!\,n!}$$

Si se calculan todas las probabilidades de todas las posibles tablas, y después se suman las que tengan probabilidad menor o igual que la nuestra (esto depende de la hipótesis alternativa) obtenemos el p-valor.

En R. Podemos utilizar la función `fisher.test()` indicando la tabla de contingencia. Repitiendo el ejemplo de la prueba χ^2 de Pearson (aunque en este caso no haría falta utilizar la de Fisher)

```
fisher.test( xtabs( dfBici ) )
```

```
    Fisher's Exact Test for Count Data

data:  xtabs(dfBici)
p-value = 0.03
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.342 0.975
sample estimates:
odds ratio
     0.579
```

En este caso, volvemos a obtener un p-valor menor a 0.05 (muy cercano al que devuelve la prueba χ^2 de Pearson) y se rechaza la hipótesis nula de que las variables no están relacionadas.

Fuerza de asociación (tamaño del efecto). Se calcula de la misma manera que para la prueba χ^2 de Pearson.

```
library( "vcd" )
assocstats( xtabs( dfBici ) )
```

```
                    X^2 df P(> X^2)
Likelihood Ratio 4.7785  1 0.028818
Pearson          4.7598  1 0.029131

Phi-Coefficient   : 0.135
```

```
Contingency Coeff.: 0.134
Cramer's V        : 0.135
```

5.3.2. Prueba de McNemar

La **prueba de McNemar** es la alternativa a la prueba χ^2 de Pearson para muestras dependientes o pareadas. Se aplica en tablas 2×2 cuando tenemos variables dicotómicas (dos valores).

¿Cómo funciona la prueba de McNemar?. Supongamos que tenemos dos variables dependientes, por ejemplo, un test que se pasa antes y después de aplicar cierto tratamiento y que toma valores dicotómicos (puede dar positivo o negativo). El objetivo de la prueba de McNemar es contrastar si el tratamiento es efectivo, esto es, si hace cambiar los test de positivo a negativo (o viceversa).

Tabla 5.4: *Tabla de contingencia. Prueba de McNemar.*

	Después Positivo	Después Negativo	Total
Antes Positivo	a	b	a+b
Antes Negativo	c	d	c+d
Total	a+c	b+d	n=a+b+c+d

Si el tratamiento no tuviera efecto se esperaría que las proporciones de mejorar (pasar de negativo a positivo) y de empeorar (pasar de positivo a negativo) fueran iguales respectivamente, esto es, que $p_a + p_b = p_a + p_c$ y $p_c + p_d = p_b + p_d$; lo que queda como $p_b = p_c$ (hipótesis nula). El estadístico que se utiliza en este caso es

$$\chi^2 = \frac{(b-(b+c)/2)^2}{(b+c)/2} + \frac{(c-(b+c)/2)^2}{(b+c)/2} = \frac{(b-c)^2}{b+c}$$

que se aproxima a una distribución χ^2 con 1 grado de libertad.

Al usar una aproximación, hay que tener cuidado cuando b o c son demasiado pequeños ($b + c < 25$). En este caso se utiliza la distribución binomial para calcular el valor exacto del estadístico (similar a lo que se hace con la prueba χ^2 de Pearson y el test exacto de Fisher).

En R. Se puede realizar con la función `mcnemar.test()`. Cuando la suma $b+c$ es pequeña se puede utilizar la función `binom.test()`.

Fuerza de asociación (tamaño del efecto). De nuevo, se calcula de la misma manera que para la prueba χ^2 de Pearson y el test exacto de Fisher. Veámoslo es el siguiente ejemplo.

Ejemplo: Supongamos que le preguntamos a un grupo de turistas chinos si se comprarían en España el iPhone6, ya que, sale unos 150 euros más barato por culpa del cambio entre el yuan y el euro. Después les explicamos que en Apple China no valen las garantías internacionales y ni siquiera gestionan las reparaciones de smartphones no comprados en China. Finalmente le volvemos a formular la pregunta de si comprarían un iPhone6 en España. ¿Ha supuesto la información dada un cambio en la intención de compra de los turistas? (Datos simulados).

```
dfMovil <- read.table( "feir50A-mcnemar.csv" , header = TRUE, sep = ";" )
dfMovil <- dfMovil[ , -1 ] # limpiamos los datos
head( dfMovil )
```

```
  antes despues
1    Sí      No
2    Sí      No
3    Sí      Sí
4    No      No
5    No      No
6    Sí      Sí
```

```
tMovil <- table( dfMovil )
tMovil
```

```
     despues
antes No Sí
   No  7  1
   Sí 10  2
```

```
mcnemar.test( tMovil )
```

```
    McNemar's Chi-squared test with continuity correction
```

```
data:  tMovil
McNemar's chi-squared = 6, df = 1, p-value = 0.02
```

El test nos devuelve un p-valor de 0.01586 (menor a 0.05), luego rechazamos la hipótesis nula de que no hay un cambio en la intención de compra de los turistas.

Para calcular la fuerza de asociación recordemos que podemos utilizar la función `assocstats()` del paquete `vcd`.

```
library( "vcd" )
assocstats( tMovil )
```

```
                    X^2 df P(> X^2)
Likelihood Ratio 0.066572  1  0.79640
Pearson          0.065359  1  0.79822

Phi-Coefficient   : 0.057
Contingency Coeff.: 0.057
Cramer's V        : 0.057
```

Nota: Aunque la función `mcnemar.test()` por defecto hace corrección por continuidad (se puede modificar con `correct = FALSE`) la suma $b+c$ es pequeña. En este caso se suele utilizar el test binomial exacto con `binom.test()` al que se le pasa la suma de los casos que han cambiado de opinión ($b+c$) y uno de esos dos valores (b o c).

```
## en el ejemplo anterior: b + c = 11
binom.test( x = 10, n = 11, p = 0.5  )
```

```
    Exact binomial test

data:  10 and 11
number of successes = 10, number of trials = 11, p-value =
0.01
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.587 0.998
sample estimates:
probability of success
```

```
0.909
```

Observamos que sale un p-valor de 0.01172, menor a 0.05 como antes.

5.3.2. Prueba Q de Cochran

La **prueba Q de Cochran** es equivalente a la prueba de McNemar para más de dos grupos, esto es, para contrastar la independencia entre varias muestras apareadas. Se utiliza como alternativa a la prueba de Friedman cuando se tienen variables dicotómicas.

¿Cómo funciona la prueba Q?. Supongamos que preguntamos a un grupo de 17 personas si compraría ropa de una cierta marca (`inicial`). Luego le ponemos publicidad de esa marca y le volvemos a hacer la misma pregunta (`publicidad`). Finalmente, le enseñamos los comentarios y opiniones de gente en internet sobre esa marca para repetirles la misma pregunta por última vez (`internet`). Nos preguntamos si la publicidad y los comentarios y opiniones en internet cambia la intención de la gente de comprarla (para bien o para mal).

Tabla 5.5: *Intención de compra según información dada.*

	inicial	publicidad	internet
1	1	1	1
2	0	1	1
3	1	1	1
⋮	⋮	⋮	⋮
16	0	0	1
17	0	1	1

Para saber si hay diferencias significativas en la intención de compra según la información que se le proporcione al cliente se calcula un estadístico Q a partir de la tabla anterior

$$Q = (k-1)\frac{k\sum_{j=1}^{k} G_j^2 - \left(\sum_{j=1}^{k} G_j\right)^2}{k\sum_{i=1}^{b} L_i - \sum_{i=1}^{b} L_i^2}$$

donde k es el número de columnas, b el de filas, G_j la suma de la columna j y L_i la suma de la fila i.

Este estadístico se aproxima a una χ^2 con $k-1$ grados de libertad. Cuando mayor sea el tamaño muestral (b en la fórmula) mejor será la aproximación.

En este ejemplo hemos visto cómo la prueba Q nos puede servir para poder contrastar diferencias de una misma variable (la intención de compra) en diferentes momentos (antes y después de ofrecer cierta información). Otro caso que puede parecer distinto y en el que se puede utilizar también la prueba Q es cuando queremos comparar varias variables (dicotómicas) generalmente en un mismo tiempo. Por ejemplo, si queremos ver si a la hora de comprar un smartphone en un primer momento la gente consideraría hacerse con un iPhone, con un Samsung y con otro modelo. La hipótesis nula es que las variables tienen a coincidir para cada persona, que no hay una preferencia antes de comparar los móviles.

Un texto muy recomendable es [44].

En R. Podemos utilizar la función `symmetry_test()` del paquete `coin` ver [45]

Otra alternativa es utilizar la función `cochran.qtest()` del paquete `RVAideMemoire` (puede tardar un poco en instalarse).

Ejemplo: Vamos a desarrollar el ejemplo anterior con R. Recordemos que tenemos un grupo de 17 personas a los que se les ha preguntado si comprarían una cierta marca de ropa.

Para llevar a cabo una prueba Q de Cochran en R primero debemos pasar la tabla a vectores como se muestra a continuación.

```
dfCompra <- read.table( "feir50A-cochranQ.csv", header = TRUE, sep = ";" )
## preparamos los datos
informacion <- rep( colnames( dfCompra )[], 17 )
head( informacion )
```

```
[1] "inicial"    "publicidad" "internet"   "inicial"    "publicidad"
[6] "internet"
```

```
persona <- rep( 1:17, each = 3 )
head( persona, 10 )
```

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

```
intencion <- as.vector( t( dfCompra ) )
head( intencion, 15 )
```

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

```
library( "coin" )
symmetry_test( intencion ~ factor( informacion ) | factor( persona ),
                        ,teststat = "quad" )
```

```
    Asymptotic General Symmetry Test

data:  intencion by
     factor(informacion) (inicial, internet, publicidad)
     stratified by factor(persona)
chi-squared = 15, df = 2, p-value = 0.0005
```

Obtenemos un p-valor menor de 0.05 luego rechazamos la hipótesis nula de que la información no provoca ningún cambio en la intención de compra.

Post-hoc. Para saber qué información ha hecho cambiar a la gente de opinión sobre la marca hacemos un análisis post-hoc. Lo más habitual es hacer comparaciones dos a dos con la prueba de McNemar y corregir la significación (Bonferroni u otra).

En el ejemplo anterior podemos hacer la prueba de McNemar sobre las parejas: `inicial-publicidad`, `inicial-internet` y `publicidad-internet`. Si las vamos guardando para luego poder obtener el p-valor añadiendo `$p.value` al final podemos corregir la significación con la función `p.adjust()`.

Hacemos la prueba de McNemar entre las categorías `inicial` y `publicidad`

```
mcnemarTest12 <- mcnemar.test( table( dfCompra[ , c( 1, 2 ) ] ) )
mcnemarTest12
```

```
    McNemar's Chi-squared test with continuity correction

data:  table(dfCompra[, c(1, 2)])
McNemar's chi-squared = 2, df = 1, p-value = 0.1
```

ahora entre `inicial` e `internet`

```
mcnemarTest13 <- mcnemar.test( table( dfCompra[ , c( 1, 3 ) ] ) )
mcnemarTest13
```

```
    McNemar's Chi-squared test with continuity correction

data:  table(dfCompra[, c(1, 3)])
McNemar's chi-squared = 8, df = 1, p-value = 0.004
```

y, por último, entre `publicidad` e `internet`

```
mcnemarTest23 <- mcnemar.test( table( dfCompra[ , c( 2, 3 ) ] ) )
mcnemarTest23
```

```
    McNemar's Chi-squared test with continuity correction

data:  table(dfCompra[, c(2, 3)])
McNemar's chi-squared = 4, df = 1, p-value = 0.04
```

Como hemos dicho, al guardarlas en variables `mcnemarTestXY` podemos obtener el p-valor y corregir la significación con la función `p.adjust()`

```
p.adjust( c( mcnemarTest12$p.value, mcnemarTest13$p.value,
mcnemarTest23$p.value ),
            method = "bonferroni" )
```

```
[1] 0.4008 0.0133 0.1237
```

Obtenemos un p-valor menor a 0.05 en el segundo caso, correspondiente a las categorías `inicial` e `internet`. Por tanto, hemos encontrado diferencias significativas en la intención de compra del grupo de personas entre el momento inicial y después de enseñarle los comentarios y opiniones de internet.

Fuerza de asociación (tamaño del efecto). Igual que ocurre con la prueba de Kruskal-Wallis no hay una manera general para calcular la fuerza de asociación (tamaño del efecto). Podemos calcular la fuerza de asociación sobre las parejas con resultados significativos del análisis post-hoc. Esto es, con lo visto en la prueba de McNemar.

En el ejemplo anterior hemos encontrado diferencias entre las categorías `inicial` e `internet`, por tanto, podemos calcular la fuerza de asociación entre ellas

```
library( "vcd" )
assocstats( table( dfCompra[ , c( 1, 3 ) ] ) )
```

```
                    X^2 df P(> X^2)
Likelihood Ratio 0.90442  1  0.34160
Pearson          0.57955  1  0.44649

Phi-Coefficient   : 0.185
Contingency Coeff.: 0.182
Cramer's V        : 0.185
```

5.3.2. Otros

Prueba de la razón de verosimilitud. Es una alternativa a la prueba χ^2 de Pearson. También se conoce como **prueba G** o, en inglés, *likelihood-ratio test*. Un texto recomendable sobre esta prueba es [24] o [46].

Prueba de Cochran-Mantel-Haenszel. Se utiliza normalmente cuando queremos comparar tablas de frecuencias 2×2 en diferentes tiempos o situaciones. La hipótesis nula es que no existen diferencias entre las frecuencias de las tablas de contingencia.

Por ejemplo, supongamos que queremos comparar si cierto tratamiento provoca igual mejoría para hombres y mujeres. En este caso, para cada género (masculino y femenino) tendríamos una tabla de frecuencias de tratamiento frente a mejoría.

En R podemos utilizar la función `mantelhaen.test()`. Más ejemplos en los que se puede aplicar esta prueba se pueden encontrar en la ayuda de esta función `?mantelhaen.test()`. Un texto recomendable de esta prueba es [47].

6 Modelos de Regresión Lineal

Antonio Maurandi López
María Elvira Ferre Jaén

6.1. Introducción

Como referencia bibliográfica básica para el desarrollo de este capítulo hemos utilizado el libro de Andy Field, Jeremy Miles, y Zoe Field: "Discovering Statistics Using R." [24], aunque también nos hemos servido de numerosos documentos que iremos referenciando a lo largo del texto.

6.1.1. Aproximación no formal al modelo de regresión lineal

El análisis de regresión lineal es una técnica estadística utilizada para estudiar la relación entre variables. A menudo resulta de interés conocer el efecto que una o varias variables pueden causar sobre otra, e incluso predecir en mayor

Cómo citar este capítulo: Maurandi-López, A. y Ferre Jaén M. A. (2022). Modelos de Regresión Lineal. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (225-308). Editum. Ediciones de la Universidad de Murcia.

o menor grado valores de una variable a partir de otra. Por ejemplo, supongamos que queremos estudiar si la altura de los padres influye significativamente en la de los hijos.

Regresión es el conjunto de técnicas usadas para explorar y cuantificar la relación de dependencia entre una variable cuantitativa llamada *variable dependiente o respuesta* y una o más variables independientes llamadas *variables predictoras*.

El primer paso para determinar si puede existir o no dependencia/relación entre variables es representando gráficamente los pares de valores observados mediante una nube de puntos, lo que se conoce como diagrama de dispersión de la figura 6.1.

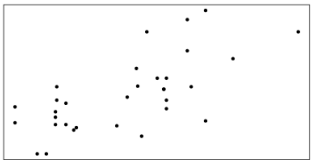


Figura 6.1: *Diagrama de Dispersión.*

Una vez representados los datos y tras detectar que entre dos o más variables existe una relación el siguiente paso sería intentar modelizar dicha relación.

La modelización estadística más sencilla para expresar la variable dependiente a través de sus variables predictoras es mediante una ecuación lineal de la forma $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_k$.

El caso más simple para una única variable sería una recta $Y = mx + n$ y recibirá el nombre de *regresión lineal simple*. Cuando $k > 1$ la llamaremos *regresión múltiple*.

Así, el proceso consistiría en ajustar la recta a nuestro conjunto de datos y crear una expresión matemática que permita predecir, de forma aproximada, el valor de la variable dependiente en un individuo cuando se conoce el valor de una variable predictora (regresión simple) o varias variables predictoras (regresión múltiple) en ese mismo individuo. A la ecuación que representa esta relación se le llama **modelo de regresión** [48].

Podemos considerar varias formas de estimar los parámetros de la ecuación del modelo de regresión. Sin embargo, nos centraremos en *el método de mínimos cuadrados* por ser el de más amplia aceptación, aunque existan también otros métodos como el de máxima verosimilitud.

Una vez creado el modelo de regresión, lo primero que debemos *analizar es su utilidad explicando los datos* que queremos relacionar. Así por ejemplo, la recta del siguiente gráfico, figura 6.2, describe, aproximadamente, la relación lineal entre las variables [49].

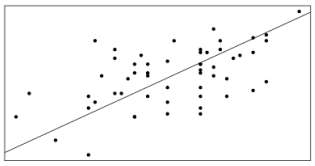


Figura 6.2: *Relación lineal.*

En cambio, los datos de la figura 6.3, no se explican bien mediante una la ecuación lineal.

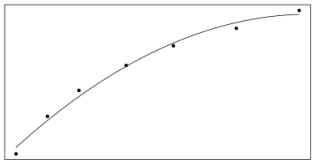


Figura 6.3: *Relación no lineal.*

Aunque sirve para hacernos una idea, no es suficiente con ver gráficamente que se trata de un modelo útil, sino que debemos *comprobar que el modelo de regresión cumple unos ciertos supuestos "matemáticos"*, que nos hablan de la bondad y calidad del modelo para nuestros fines.

Que la recta se ajuste a los datos no significa que el modelo sea correcto, depende del uso que queramos darle. Si sólo pretendemos hallar la relación entre

dos variables, con calcular la recta de mínimos cuadrados es suficiente, esa recta describe la relación entre las variables, otra cosa es que los datos tenga una buena relación lineal. Podría ser que los datos tuvieran muy mala relación lineal y la recta seguiría existiendo. En cambio si pretendemos describir la estructura general de los datos, o inferir/predecir con la recta de regresión debemos comprobar que se verifican unas reglas ya establecidas y aceptadas que aseguran que nuestro modelo es bueno.

Existen una serie de procedimientos de diagnostico que nos informaran sobre la estabilidad e idoneidad del modelo de regresión. Los *supuestos* que tendremos que comprobar son

- En el modelo de regresión: linealidad
- En los residuos:
  - normalidad
  - varianza constante
  - valores atípicos

Por otro lado, para cada conjunto de datos existen varias rectas con las que podríamos resumir la tendencia general de los mismos. Necesitamos encontrar la recta del mejor ajuste, aquella que da lugar a la menor diferencia entre los datos originales y los estimados por la recta.

Para buscar esta recta utilizaremos el *criterio de mínimos cuadrados*, método con el que calculamos la recta que minimiza la suma de *los residuos*, esto es, las distancias verticales entre cada punto y la recta, ver figura 6.4.

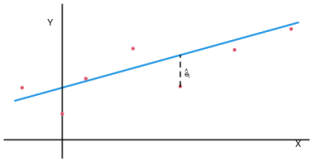


Figura 6.4: *Criterio de mínimos cuadrados.*

El objetivo que hay tras este método es que los residuos sean pequeños, lo que matemáticamente se traduce en que tengan media cero y en que bailen lo menos posible, es decir, en una σ^2 pequeña. De aquí es de donde surgen todos los supuestos que se le exigen al modelo de regresión lineal.

Uno de los resultados que obtenemos al aplicar el método de los mínimos cuadrados es que el coeficiente m, que cuantifica la relación entre la x y la y en nuestra ecuación, es en realidad el coeficiente de correlación de Pearson. Por ello, antes de crear el modelo de regresión tenemos que analizar si este coeficiente es significativamente distinto de cero y en caso de serlo plantearemos el modelo de regresión lineal.

6.1.2. Correlación lineal

Un análisis de correlación nos permite cuantificar el grado de asociación lineal entre variables continuas, indica la fuerza y dirección de la relación lineal entre dos o más variables. Cuando exista dicha relación se podrá proceder a la obtención del modelo de regresión (simple o múltiple) que veremos posteriormente [48].

Existen *diferentes tipos de correlación*, la correlación simple, la correlación múltiple y la correlación parcial. Utilizaremos la correlación simple cuando contemos con una sola variable predictora para explicar una respuesta, y los coeficientes de correlación parcial y múltiple cuando tengamos varios predictores.

6.1.2. Correlación lineal simple

Utilizamos la correlación lineal simple para estudiar el grado de variación conjunta entre dos o más variables. Queremos detectar si la variación de una de las variables tiene conexión con la variación de la otra, esperamos que si una variable de desvía de la media, la otra variable se desvíe de la media de manera similar.

Una *relación lineal positiva* entre dos variables indica que los valores de las dos variables varían de forma parecida: los sujetos que puntúan alto en una variable tienden a puntuar alto en la otra y los que puntúan bajo en la primera

tienden a puntuar bajo en la segunda, existe una relación directa entre ambas variables.

Una *relación lineal negativa* significa que los valores de las dos variables tienen una relación inversa: valores pequeños de una variable van asociados ahora a valores grandes de la otra y, equivalentemente, valores grandes de una se asocian a valores pequeños de la otra.

La forma más directa e intuitiva de formarnos una primera impresión sobre el tipo de relación existente entre dos variables es a través de un *diagrama de dispersión*. Se trata de un gráfico en el que una de las variables, X, se coloca en el eje de abscisas, la otra, Y, en el de ordenadas y los pares (x_i, y_i) se representan como una *nube de puntos*. La forma de la nube de puntos nos informa sobre el tipo de relación existente entre las variables.

Una regla fundamental es que cuanta mayor correlación haya entre dos variables en la representación bidimensional, más próximos a la recta estarán los valores.

Veamos un ejemplo: en el siguiente gráfico mostramos cuatro diagramas de dispersión que reflejan cuatro tipos de relación diferentes [50].

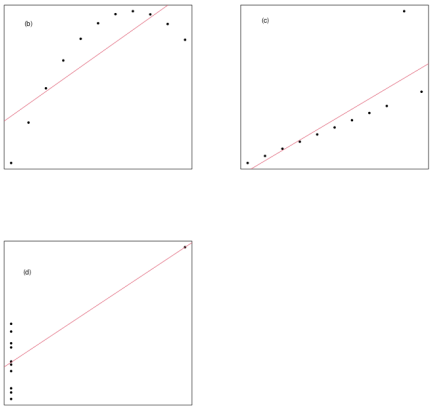


Figura 6.5: *Cuatro tipos de relación diferentes.*

Para todos estos conjuntos de datos la recta de regresión es la misma

$$\hat{y} = 3 + 0.5 \times x$$

con los coeficientes significativos con un nivel de significación < 0.01, y además todos tienen la misma $R^2 = 0.67$ y $\hat{\sigma} = 1.24$.

Sin embargo, solamente podemos escribir mediante un modelo lineal los datos del primer gráfico de dispersión de la figura 6.5. El segundo gráfico de dispersión muestra un conjunto de datos es claramente no lineal y sería mejor ajustarlo mediante una función cuadrática.

El tercero de los gráficos de la figura 6.5 muestra un conjunto de datos que tiene un punto que distorsiona los coeficientes de la recta ajustada. Por último, el gráfico muestra un conjunto de datos totalmente inapropiado para un ajuste lineal, la recta ajustada está determinada esencialmente por la observación extrema [51].

Tras haber realizado una representación de los datos, una buena manera de cuantificar la relación a entre dos variables es mediante la **covarianza**

$$r = Cov(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{N - 1},$$

donde N es el número de observaciones.

Sin embargo, la covarianza no es una medida útil para comparar rectas de regresión de variables distintas, o comparar el grado de asociación lineal entre distintos pares de variables, ya que depende de las escalas de medida de las variables. La solución está en estandarizarla y es de aquí de donde surgen llamados *coeficientes de correlación*.

6.1.2.1.1. Coeficientes de correlación El más importante de los coeficientes de correlación es el *Coeficiente de Pearson*, que explicaremos en mayor profundidad, pero también están la *Rho de Spearman* y la *Tau de Kendall*. Veamos sus **propiedades generales**:

- Todos los coeficientes varían entre -1 y 1.
- Si el coeficiente de correlación es -1 existe correlación negativa, es decir, a medida que una variable aumenta, la otra disminuye. Cuando el coefi-

ciente es 1 hay correlación positiva, cuando aumenta una variable, también aumenta la otra.

- Un valor cercano o igual a cero indica poca o nula relación lineal entre las variables.
- Se utilizan como una medida de la fuerza de asociación: valores ±0.1 representan pequeñas asociación, ±0.3 asociación mediana, ±0.5 asociación moderada, ±0.7 gran asociación y ±0.9 asociación muy alta.

Las principales **diferencias entre los coeficientes** son:

- La correlación de *Pearson* funciona bien con variables cuantitativas y que sigan bien la distribución normal.
- La correlación de *Spearman* se utiliza para datos ordinales o de intervalo que **no** satisfacen la condición de normalidad. (usualmente tiene valores muy parecidos a la de Pearson).
- La correlación de *Kendall* es una medida no paramétrica para el estudio de la correlación. Debemos utilizar este coeficiente en vez de la de Spearman cuando tengamos un conjunto de datos pequeño y muchas puntuaciones estén en el mismo nivel.

6.1.2.1.2. Coeficiente de Pearson El coeficiente de correlación lineal de *Pearson (r)* viene definido como

$$r = \frac{Cov(X,Y)}{Sd(X)Sd(Y)} = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2\sum_{i=1}^{n}(y_i-\overline{y})^2}}$$

y se trata de la covarianza de entre las variables X e Y estandarizada.

Para que el coeficientes de correlación de Pearson sea una medida precisa de la relación lineal entre dos variables exige que las variables sean cuantitativas y que las dos variables se distribuyan normalmente, aunque podemos hacer una excepción si sólo una de las variables es normal y la otra es categórica con dos categorías. Si los datos no son normales o cuantitativos entonces se debe usar otro tipo de coeficientes como el de Spearman o el de Kendall.

Las *principales características* de este coeficiente son:

1. Medida de asociación lineal libre de escala
2. Valores comprendidos entre -1 y 1
3. Invariante a transformaciones lineales de las variables.

Su *interpretación* es la siguiente:

- Si r = 0 (asociación lineal nula) no existe relación entre las variables.
- Si r = 1 o -1 (asociación lineal perfecta).
- Cuando r > 0 (correlación positiva) existe una relación directa entre las variables
- Cuando r < 0 (correlación negativa) existe una relación inversa entre las variables.

El coeficiente hay que interpretarlo en magnitud, es decir, tomar su valor absoluto. Esto significa que cuanto más cerca estemos de los extremos (± 1) más relación existe entre las variables. Por eso, una correlación con valor $r = -0.9$ es más fuerte que una con $r = 0.7$, pues 0.9 es más grande que 0.7 aunque sea negativa.

Por último queda ver que la correlación entre las variables es *significativa*, es un valor fiable que no cambiaría mucho en otra muestra tomada en las mismas condiciones.

Una *correlación* será *significativa* si su p-valor es inferior a 0.05, de lo contrario supondremos que $r = 0$.

Según esto podemos decir que una $r = 0.8$ con un p-valor de 0.26 es en realidad una correlación más baja que una $r = 0.4$ con $p = 0.001$, ya que al no ser significativa la $r = 0.8$ no es una medida fiable, puede ser un efecto del azar del muestreo. De la misma forma que en esta muestra hemos calculado una $r = 0.8$ en otra muestra tomada en las mismas condiciones podríamos obtener $r = -0.8$. Debido a ello, y ante la duda, es mejor afirmar que no hay relación, que r es igual a 0. Para el caso de la correlación $r = 0.4$, aunque no se trata de una gran correlación, sí que es fiable [48].

6.1.2.1.3. Coeficiente de Spearman El coeficiente de correlación de Spearman es el mismo que el coeficiente de Pearson pero tras transformar las puntuaciones originales a rangos.

El coeficiente de Sperman puede utilizarse como una alternativa a Pearson cuando las variables son ordinales y/o no se incumple el supuesto de normalidad.

$$r_s = 1 - \frac{6\sum d^2}{n(n^2-1)},$$

donde d es la distancia entre los rangos (X menos Y) y n es el número de datos.

6.1.2.1.4. Tau de Kendall Es un coeficiente de correlación no paramétrico que se basa en el concepto de inversión, no-inversión y empate. Se calcula a partir de los desórdenes entre los rangos, su fórmula es la siguiente

$$\tau = \frac{C-D}{\frac{1}{2}n(n-1)},$$

donde C es el número de pares concordantes, aquellos en los que el rango de la segunda variable es mayor que el rango de la primera variable, y D el número de pares discordantes, cuando el rango de la segunda es igual o menor que el rango de la variable primera.

Podemos utilizarlo, al igual que en el caso de Spearman, cuando las variables no alcanzan el nivel de medida de intervalo y no podemos suponer que la distribución poblacional conjunta de las variables sea normal.

6.1.2. La correlación simple en R

Para el cálculo del coeficiente de correlación vamos a utilizar la función `cor( )`, que tiene la forma general

```
cor( x,y use = "string", method = "tipo de correlación" ),
```

donde:

- x: variable numérica o un dataframe.
- y: otra variable numérica (si x es un dataframe no hay que especificarla).
- use: especifica el tratamiento para los datos perdidos.
  - use = all.obs: se asume que no existen valores perdidos, si existiera alguno produciría un error
  - use = everything: cualquier correlación que envuelva una variable con valores perdidos se tratará como *missing*
  - use = complete.obs: sólo se ejecutan los casos que están completos para todas las variables
  - use = pairwise.complete.obs: correlación entre pares de variables que se ejecuta para los casos que estén completos para estas dos variables.
- method: especifica el tipo de correlación. Podemos elegir entre "pearson" (por defecto), "kendall", o "spearman").

Ejemplo:_Calculamos la correlación entre las variables "Horsepower" y "Weight" del archivo Cars93_

```
library( MASS )
data( Cars93 )
df <- data.frame( Cars93 )
cor( df$Horsepower, df$Weight, method = "pearson" )
```

```
[1] 0.739
```

6.1.2.2.1. Correlación significativa No resulta suficiente la estimación puntual del coeficiente de correlación. Para asegurar la existencia de relación entre las variables dependiente y predictora debemos realizar un **test para estudiar la significación estadística**.

Enfrentaremos la hipótesis nula ($H_0 : r = 0$, no relación) frente a la hipótesis alternativa ($H_1 : r \neq 0$ existe relación) mediante la función cor.test() que toma la siguiente forma:

`cor.test( x, y, alternative = " ", method = " " )` donde

- `x` e `y` son las variables a estudiar
- `alternative` será "two.side", "less"" o "greater"
- `method` especificaremos el tipo de correlación (pearson, spearman o kendall).

```
cor.test( df$Horsepower, df$Weight, method = "pearson" )
```

```
    Pearson's product-moment correlation

data:  df$Horsepower and df$Weight
t = 10, df = 91, p-value <2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.630 0.819
sample estimates:
  cor
0.739
```

Por defecto selecciona el método de Pearson. Fijándonos en el p-valor podemos asegurar la existencia de correlación entre las variables. Además este test estima el valor de la correlación y nos da un intervalo de confianza para dicho valor.

En el caso de querer calcular el coeficiente de correlación simple entre **varias variables de un archivo** no tenemos porque hacerlo dos a dos, podemos crear una matriz de correlaciones:

```
newdf <- data.frame( df$Price, df$Weight, df$RPM, df$Horsepower )
cor( newdf, use = "everything", method = "pearson" )
```

```
              df.Price df.Weight   df.RPM df.Horsepower
df.Price       1.00000     0.647 -0.00495        0.7882
df.Weight      0.64718     1.000 -0.42793        0.7388
df.RPM        -0.00495    -0.428  1.00000        0.0367
df.Horsepower  0.78822     0.739  0.03669        1.0000
```

Además de las correlaciones queremos también los p-valores pero la función `cor.test` no funciona con matrices así que utilizamos una *nueva función*:

```
library( "psych" )
corr.test( newdf, use = "complete", method = "pearson" )
```

```
Call:corr.test(x = newdf, use = "complete", method = "pearson")
Correlation matrix
              df.Price df.Weight df.RPM df.Horsepower
df.Price          1.00      0.65   0.00          0.79
df.Weight         0.65      1.00  -0.43          0.74
df.RPM            0.00     -0.43   1.00          0.04
df.Horsepower     0.79      0.74   0.04          1.00
Sample Size
[1] 93
Probability values
          (Entries above the diagonal are adjusted for multiple tests.)
              df.Price df.Weight df.RPM df.Horsepower
df.Price          0.00         0   1.00             0
df.Weight         0.00         0   0.00             0
df.RPM            0.96         0   0.00             1
df.Horsepower     0.00         0   0.73             0

 To see confidence intervals of the correlations,
     print with the short=FALSE option
```

Analizando la salida vemos que se obtienen las mismas correlaciones que con la función `cor( )`, aunque aproximadas, y que los p-valores muy bajos ($p < 0.05$) han sido aproximados a 0, así que todas las correlaciones son significativas.

Observación: El procedimiento para hacer una correlación de Spearman o Kendall es el mismo que para una correlación de Pearson excepto que tenemos que especificar que queremos otra correlación, que se realiza mediante el `method = "spearman"` o `method = "kendall"` para `cor( )`, `cor.test( )`y `corr.test( )`.

6.1.2. Correlación parcial

La *correlación parcial* es una correlación entre dos variables en la que el efecto de otras variables auxiliares se mantiene contante, se busca la relación entre dos variables mientras 'se controla' el efecto de una o más variables adicionales.

Esta medida surge ya que en ocasiones las variables continuas con las que pretendemos predecir una respuesta no son totalmente independientes entre sí lo provoca que las variables compartan y solapen información a la hora de explicar la respuesta.

Por ejemplo, si queremos estudiar la relación entre las variables "inteligencia"" y "rendimiento escolar" tendremos que tener en cuenta terceras variables como el "número de horas de estudio", el "nivel educativo de los padres".

La correlación parcial se trata, por tanto, de un coeficiente de correlación que nos da una idea sobre la relación lineal existente entre dos variables pero ajustada a los efectos lineales que sobre las mismas puedan tener otra o más variables que intervengan. Utilizaremos la función `pcor( )` incluida en el paquete `ppcor`. Su forma general es:

```
pcor( var1 , var2 , control1 , control2,..., method = " " )
```

- `var1` y `var2` son las variables a ser correladas.
- `control1` , `control2` y las siguientes posibles son las variables con las que controlamos la correlación.
- `method = c( "pearson", "kendall", "spearman" )`, que por defecto empleará `spearman`.

Vamos a **calcular la correlación parcial** entre `Price` y `Weight` controlando el efecto de la variable `Length`.

```
library( "ppcor" )
pcor.test( df$Price, df$Weight, df$Length )
```

```
  estimate    p.value statistic  n gp  Method
1    0.472 0.00000206      5.08 93  1 pearson
```

tenemos que

- `estimate` es el coeficiente de correlación parcial entre las dos variables.
- `p.value` es el p-valor del test.
- `statistic` es el valor del estadístico del test.

- n es el número de muestras.
- gn es el número de variables.
- `method` es el método de correlación empleado (spearman, pearson o kendall).

Si calculamos la correlación simple entre las variables `Price` y `Weight`:

```
cor( df$Price, df$Weight )
```

```
[1] 0.647
```

observamos que tiene un valor diferente a la correlación parcial controlada por `df$Length`. Por tanto, las variables `Price` y `Weight` están influenciadas por `Length` ya que al controlar su efecto la correlación se reduce de 0.647 a 0.47.

6.1.2. Otras consideraciones

6.1.2.4.1. Causalidad Debemos tener precaución a la hora de interpretar los coeficientes de correlación ya que estos no nos indican la dirección de *causalidad* de las variables, no nos dicen nada sobre qué variable causa que la otra varíe.

Aunque es intuitivo pensar que ver anuncios nos provoque comprar más paquetes de galletas, no hay razón estadística por la que comprar paquetes de galletas no nos pueda provocar ver más anuncios. Pese a que la última conclusión tiene menos sentido, el coeficiente de correlación no nos dice que no puede ser cierta, para un matemático la dirección no importa.

Por otro lado existe *el problema de la tercera variable*. Este nos dice que no podemos asumir causalidad entre dos variables porque podría haber otras variables afectando a los resultados.

6.1.2.4.2. Tamaño del efecto Recordemos que $(\hat{Y}_i - \bar{Y}) = \hat{\beta}_1(X_i - \bar{X})$ y que $\hat{\beta}_1 = r = \frac{Cov(X,Y)Sd(Y)}{Sd(X)}$ así que

$$R^2 = \frac{\sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2} = \hat{\beta_1}^2 \frac{\sum_{i=1}^n (\hat{X}_i - \bar{X})}{\sum_{i=1}^n (Y_i - \bar{Y})^2} = Cor(Y,X)^2 = r^2.$$

Entonces, aunque no podemos hacer conclusiones directas sobre la causalidad de una correlación, para dos variables sí podemos elevar el coeficiente de correlación al cuadrado y utilizarlo como una medida de la cantidad de variabilidad que una variable comparte con la otra. Es lo que se conoce como **coeficiente de determinación,** R^2, y es una medida tremendamente útil de la importancia de un efecto.

Para calcular este coeficiente, R^2, podemos elevar al cuadrado tanto el coeficiente de Pearson, r, como el coeficiente de Spearman r_s, ya que este usa la misma ecuación que Pearson. Lo único que debemos tener en cuenta es que el resultante R_S^2 hay que interpretarlo como la proporción de varianza en las categorías que las dos variables comparten.

El coeficiente de Kendall, sin embargo, no es numéricamente similar a r o r_s por lo que τ^2 no nos dice nada sobre la proporción de varianza compartida por las dos variables.

Calculamos el coeficiente de determinación para el conjunto de datos `newdf` anterior:

```
cor( newdf, use = "everything" ) ^ 2
```

```
                df.Price df.Weight    df.RPM df.Horsepower
df.Price       1.0000000     0.419 0.0000246       0.62129
df.Weight      0.4188407     1.000 0.1831253       0.54582
df.RPM         0.0000246     0.183 1.0000000       0.00135
df.Horsepower 0.6212870     0.546 0.0013460       1.00000
```

Se observa que el tamaño del efecto de `EngineSize` sobre `Weightes` muy elevado, así como para `Lenght` y `Weight`, siendo sin embargo muy bajo el efecto de `Lenght` sobre `Price`. Si queremos expresar estos valores en porcentajes basta multiplicar por 100.

6.1.2.4.3. Comunicar los coeficientes de correlación Sólo hay que decir cómo de grande es y qué valor de significación tiene. La forma de reportar los coeficientes sería

- Existe una relación significativa entre `var1` y `var2`, $r = 0.78$, $p < 0.05$.

- Var1 está significativamente correlacionada con var2, $r_s = 0.57$, y con var3, $r_s = 0.50$; la var2 está también correlacionada con var3, $r_s = 0.83$ (todas $p < 0.01$).
- Var2 está significativamente relacionada con var1, $\tau = -0.45$, $p < 0.01$.

6.1.2. Ejemplo de los tractores

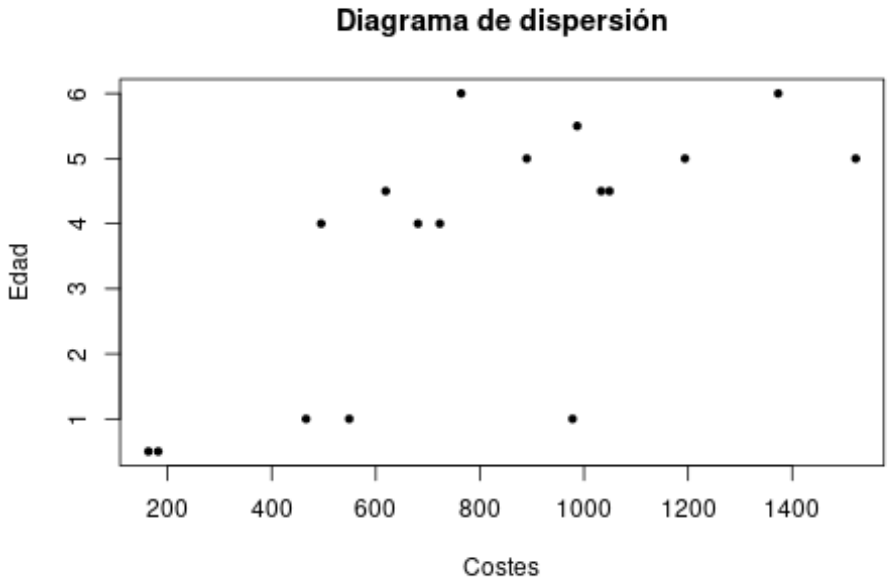
Supongamos que una empresa de tractores que pretende saber qué le es más conveniente, si renovar su flota de tractores, seguir manteniendo la que tienen o cambiar solo una parte. Utilizamos el conjunto de datos tractores.rda para intentar relacionar los costes de manutención de tractores con la edad de éstos.

Comenzamos calculando la correlación entre edad y costes, y realizamos el correspondiente gráfico de dispersión

```
load( "files/40A-tractores.rda" )
cor.test( tractores$costes, tractores$edad )
```

```
    Pearson's product-moment correlation

data:  tractores$costes and tractores$edad
t = 4, df = 15, p-value = 0.002
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.314 0.879
sample estimates:
  cor
0.691
```

```
plot( tractores$costes, tractores$edad, pch = 20, xlab = "Costes",
      ylab = "Edad", main = "Diagrama de dispersión" )
```



Como existe mucha diferencia en las escalas de medida aplicamos la función logaritmo, `log( )`, a los datos ya que es la que más puede reducir estos valores. Creamos una nueva variable que sea el logaritmo de los costes y realizamos de nuevo el análisis de correlación

```
tractores$logcostes <- log( tractores$costes )
cor.test(  tractores$logcostes, tractores$edad )
```

```
    Pearson's product-moment correlation

data:  tractores$logcostes and tractores$edad
t = 4, df = 15, p-value = 0.0008
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.394 0.898
sample estimates:
  cor
0.735
```

```
plot(tractores$logcostes,tractores$edad ,pch = 20, xlab="log(Costes)",
```

```
      ylab = "Edad", main = "Diagrama de dispersión" )
```

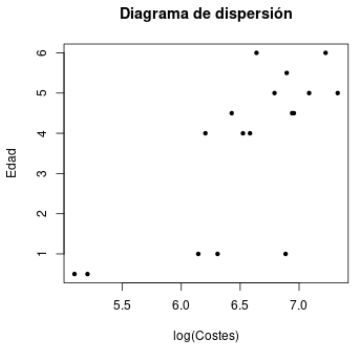


Figura 6.6: *Diagrama de dispesrión, ejemplo de los tractores.*

Como vemos la correlación ahora es más elevada y los puntos están menos dispersos en el plano.

Una vez detectada una relación significativa entre dos o más variables, el siguiente paso es intentar crear una fórmula matemática que formalice esa relación y que permita calcular pronósticos de una variable a partir de una o varias variables evaluadas en un individuo concreto. Este proceso se conoce como *regresión* y es el que estudiaremos en los siguientes apartados.

6.2. Regresión lineal simple

Para el desarrollo de los siguientes tres apartados nos hemos servido esencialmente de [49].

6.2.1. Introducción

El caso de modelo de regresión más sencillo es la construcción de una recta que modelice la relación que hay entre la variable respuesta, Y, y la variable predictora X. El modelo tiene la forma

$$Y = \beta_0 + \beta_1 X + e,$$

donde β_0 y β_1 se conocen como *coeficientes de regresión* y son, respectivamente, la ordenada en el origen (punto de corte con el eje Y) y la pendiente de la recta del modelo de regresión.

En la ecuación e es el error aleatorio, representa la diferencia entre el valor ajustado por la recta y el valor real. Refleja la ausencia de dependencia perfecta entre las variables, la relación está sujeta a incertidumbre.

Por ejemplo, en el consumo de gasolina de un vehículo, Y, influyen la velocidad X y una serie de factores como el efecto conductor, el tipo de carretera, las condiciones ambientales, etc. Todos estos elementos quedarían englobados en el error e.

Los coeficientes de regresión se pueden interpretar como:

- β_0 el valor medio de la variable dependiente cuando la predictora es cero.

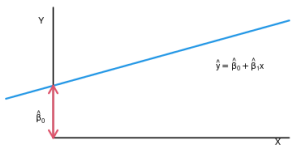


Figura 6.7: *Diagrama de dispersión, ejemplo de los tractores, variale transformada.*

- β_1 el efecto medio (positivo o negativo) sobre la variable dependiente al aumentar en una unidad el valor de la predictora X.

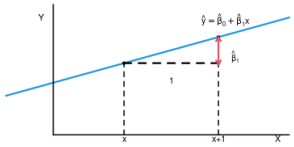


Figura 6.8: *Modelo de regresion lineal.*

Una recta que tiene una pendiente con valor positivo describe una relación positiva, mientras que una recta con una pendiente negativa describe una relación negativa. Entonces tenemos básicamente que la pendiente (β_1) nos da la apariencia del modelo (su forma) y la ordenada en el origen (β_0) nos dice dónde se sitúa el modelo en el plano.

6.2.2. Estructura del modelo de regresión simple

El modelo de regresión lineal simple tiene la siguiente estructura $y_i = \beta_0 + \beta_1 x_i + e_i$ para $i = 1, ..., n$. Vamos a estudiarlo más detenidamente.

Supongamos que hemos ajustado una recta de regresión a un conjunto de datos, y sea (x_i, y_i) un punto cualquiera de la nube. Entonces y_i se puede descomponer como $y_i = f(y_i) + e_i = \hat{y}_i + e_i$

donde $\hat{y}_i$ es el valor ajustado a la recta del valore observado y_i, y e_i es el error que cometemos y al que llamaremos **residuo**.

Una vez calculado el modelo, el valor de $\hat{y}$ queda determinado para cada x_i, pero el valor $e_i = y_i - \hat{y}_i$ no queda determinado, puede haber dos observaciones con el mismo x_i y distinto e_i. En este razonamiento se basará la hipótesis de independencia de los residuos.

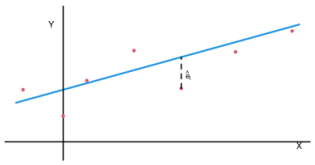


Figura 6.9: *Residuos en un modelo de Reg lineal.*

6.2.3. Supuestos del modelo

Para cada x_i, valor fijo de X, se cumple la ecuación $y_i = \beta_0 + \beta_1 x_i + e_i$, donde β_0 y β_1 son constantes desconocidas. Las hipótesis básicas del modelo son:

1. Incorrelación de los residuos $Corr(e_i, e_j) = 0$. Cualquier par de errores e_i y e_j son independientes.
2. Media cero de los residuos $E(e_i) = 0$.
3. Varianza constante de los residuos $Var(e_i) = \sigma^2$.
4. Normalidad de los residuos $e_i \sim N(0, \sigma^2)$.

Como consecuencia:

- Cada valor x_i de la variable aleatoria X tiene distribución

$$(Y \mid X = x_i) \approx N(\beta_0 + \beta_1 x_i, \sigma^2).$$

- Las observaciones y_i de la variable Y son independientes.

Gráficamente, si las hipótesis del modelo son ciertas tenemos (figura 6.10).

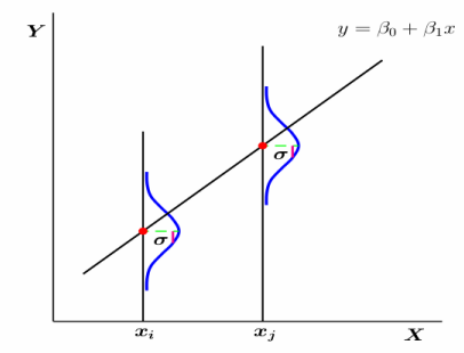


Figura 6.10: *Supuestos del modelo*

6.2.3. Estimación de la recta de regresión. Método de mínimos cuadrados

Si nos encontrásemos en la situación ideal de que todos los puntos del diagrama de dispersión se encontraran en una línea recta no tendríamos que preocuparnos por encontrar la recta que mejor resume los puntos del diagrama, simplemente uniendo los puntos entre sí la obtendríamos.

Sin embargo si nos situamos en una situación más realista, en una nube de puntos es posible trazar muchas rectas diferentes, aunque obviamente, no todas ellas se ajustarán igualmente bien a la nube [52]. Se trata entonces de estimar la recta que el mejor represente el conjunto total de puntos.

El procedimiento va a consistir en estimar los coeficientes de regresión β_0 y β_1 para obtener la recta

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

donde $\hat{y}$ denota el valor ajustado por la recta para el valor observado x.

Para estimar la ecuación de la recta de regresión podemos a utilizar el **criterio de mínimos cuadrados**, pues es el más empleado usualmente. Vamos a estudiarlo.

Siempre que ajustamos cualquier recta a un conjunto de datos existen pequeñas diferencias entre los valores estimados por la recta y los valores reales observados, así cada valor del modelo ajustado lleva asociado su error aleatorio $e_i = y_i - \hat{y}_i$, ver figura 6.9.

Se nos podría ocurrir sumar todos los residuos para obtener así una estimación del error total, sin embargo, al sumar diferencias positivas y negativas estas tienden a cancelarse unas con otras. Para solucionar este problema decidimos elevar al cuadrado las diferencias antes de sumarlas [50].

Por tanto, con el criterio de mínimos cuadrados estimamos los coeficientes de regresión, β_0 y β_1, haciendo mínima la suma de los cuadrados de los residuos, $SS_E = \sum_{i=1}^{n} e_i^2$.

$$SS_E = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - b_1 x_i)^2$$

Esto significa que, de todas las rectas posibles, existe una y sólo una que consigue que las distancias verticales entre cada punto y la recta sean mínimas [52].

Las diferencias al cuadrado resultantes son un indicador de la capacidad de la recta ajustándose a los datos; si las diferencias al cuadrado son grandes la recta no es representativa de los datos, mientras que si son pequeñas la recta sí es representativa.

6.2.3.1.1. Consecuencias del criterio de mínimos cuadrados

- $\hat{\beta_1} = r = \frac{Cov(X,Y)Sd(Y)}{Sd(X)}$.
- $\hat{\beta_0} = \bar{Y} - \hat{\beta_1}\bar{X}$.
- La suma de los residuos es cero
- La media de los valores observados Y_i coincide con la media de los valores ajustados $\overline{Y_i}$.
- La recta de regresión pasa por el punto $(\bar{x}, \bar{y})$.
- Los valores $\hat{\beta_0}$ y $\hat{\beta_1}$ son estimadores de β_0 y β_1.
- Las estimaciones de la respuesta para un valor $X = x$ se obtiene como

$$\overline{y} = \hat{\beta_0} + \hat{\beta_1} x$$

6.2.4. Ejemplo. Ajuste del modelo y proceso inferencial

Vamos a desarrollar esta sección mediante un ejemplo aplicado:

El presidente de personal de una multinacional está buscando si existe relación entre el salario de un trabajador y su porcentaje de absentismo. Éste dividió el intervalo de salarios en categorías y muestreó aleatoriamente a un grupo de trabajadores para determinar número de días que habían faltado en los últimos 3 años. ¿Es posible establecer un modelo que relacione la categoría y las ausencias?

6.2.4. Ajuste del modelo en R

Vamos a establecer el modelo que relaciona Ausencias con Categoría, pero antes de esto estudiaremos la normalidad de los datos y calcularemos la correlación entre categoría y ausencias, realizando además el correspondiente gráfico de dispersión.

```
datos <- read.table( "40A-william.csv", sep = ";", head = TRUE )
```

Empezamos estudiando la normalidad de la variable explicativa

```
shapiro.test( datos$Categoria )
```

```
Shapiro-Wilk normality test

data:  datos$Categoria
W = 0.9, p-value = 0.3
```

visto que los datos son normales, realizamos el análisis de correlación

```
cor.test( datos$Categoria, datos$Ausencias )
```

```
Pearson's product-moment correlation

data:  datos$Categoria and datos$Ausencias
t = -5, df = 14, p-value = 0.0003
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
-0.922 -0.474
sample estimates:
cor
-0.785
```

y representamos los puntos

```
plot( datos$Categoria, datos$Ausencias, pch = 20,
xlab = "Categoría",  ylab = "Ausencias",
main = "Diagrama de dispersión",cex.main = 0.95)
```

La correlación entre ambas variables es significativa con un p-valor menor a 0.05 y se trata de una relación inversa y alta (-0.7851), según crece la categoría

disminuyen las ausencias.

Una vez visto que existe relación entre las variables pasamos a realizar el **ajuste del modelo**. Para ello usamos la función `lm( )` que toma la forma

`lm( dependiente ~ predictora(s), data = dataFrame, na.action = "acción" )`

donde `na.action` es opcional, puede ser útil si tenemos valores perdidos.

Creamos el objeto `modelAu` que contiene todos los resultados del ajuste.

```
modelAu <- lm( Ausencias ~ Categoria, data = datos )
summary( modelAu )
```

```
Call:
lm(formula = Ausencias ~ Categoria, data = datos)

Residuals:
Min      1Q Median      3Q     Max
-13.30  -2.60   1.80   3.69   6.45

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)   41.596      3.579   11.62  1.4e-08 ***
Categoria     -2.292      0.483   -4.74  0.00031 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.9 on 14 degrees of freedom
Multiple R-squared:  0.616, Adjusted R-squared:  0.589
F-statistic: 22.5 on 1 and 14 DF,  p-value: 0.000314
```

La parte 'Residuals' nos da la diferencia entre los valores experimentales y ajustados por el modelo. Las estimaciones de los coeficientes del modelo se proporcionan junto con el sus desviaciones estándar ('error estándar'), un t-valor y la probabilidad de la hipótesis nula de que los coeficientes tengan valor de cero. En este caso, por ejemplo, hay evidencia de que ambos coeficientes son significativamente diferentes de cero.

En la parte inferior de la tabla se encuentra la desviación sobre la recta regresión (error estándar s_r o residual), el coeficiente de correlación y el resultado del *test F* sobre la hipótesis nula de que los $\frac{MS_{reg}}{MS_{res}}$ es 1.

```
plot( datos$Categoria, datos$Ausencias, pch = 20,
xlab = "Categoria", ylab = "Ausencias" )
abline( modelAu )
```

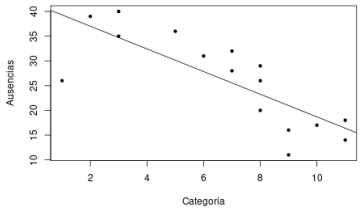


Figura 6.11: *Reg. Lineal. Primera aproximación.*

En primer lugar deseamos obtener los estimadores puntuales, errores estándar y p-valores asociados con cada coeficiente

```
summary( modelAu )$coefficients
```

```
Estimate Std. Error t value Pr(>|t|)
(Intercept)    41.60      3.579   11.62 1.41e-08
Categoria      -2.29      0.483   -4.74 3.14e-04
```

El resultado del ajuste es

```
(3.5795) (0.4832) Ausencias = 41.5956-2.2919 * Categoria
```

donde los valores entre paréntesis indican los errores estándar de cada coeficiente. Además, puesto que los p-valores asociados son inferiores a `0.05`, podemos concluir que:

1. En este caso no tiene sentido analizar el valor de la constante para `Categoría = 0`, ya que no pertenecería a la empresa, de ahí que el valor de Ausencias para `Categoría = 0` sea de `41.5956`, mayor que cualquiera de los datos de nuestro conjunto.

2. Existen evidencias estadísticas suficientes para considerar que hay una rela-

ción lineal entre Categoría y Ausencias. Dicha relación es negativa cuando aumenta la categoría laboral del individuo disminuyen las ausencias. Además vemos que por cada grado que aumenta la categoría del trabajador, disminuyen las ausencias en 2,29 días por año.

3. El error estándar residual estimado (s) es de 5.898. Este valor es muy importante, es un medidor de la calidad (precisión) del modelo. Además nos vamos a basar en él para calcular los intervalos de confianza para los coeficientes del modelo. Se calcula haciendo la raíz cuadrada de la media de la suma de cuadrados de los residuos (MS_R).

6.2.4.1.1. Intervalos de Confianza Los intervalos de confianza (IC) complementan la información que proporcionan los contraste de hipótesis a la hora de expresar el grado de incertidumbre en nuestras estimaciones.

Obtenemos los correspondientes intervalos de confianza para cada parámetro del modelo con nivel significación al 95%

```
confint( modelAu, level = 0.95 )
```

```
2.5 % 97.5 %
(Intercept) 33.92  49.27
Categoria   -3.33  -1.26
```

como el intervalo no contiene al cero, podemos rechazar la hipótesis nula de que $H_0 : \beta_0 = \beta_1 = 0$.

Interpretamos los intervalos: con una probabilidad del 95%, la ordenada en el origen del modelo, β_0, se encuentra en el intervalo `(33.92, 49.27)`, mientras que el efecto asociado con la *Categoria* se encuentra en el intervalo `(-3.32, -1.26)`.

6.2.5. Bondad de ajuste

Una vez realizado el ajuste, debemos verificar la eficiencia del modelo a la hora de explicar la variable dependiente, ya que aunque la recta sea la mejor disponible, ésta puede seguir siendo un ajuste terrible de los datos.

Las medidas fundamentales son el *error residual estimado*, el *test F* para la

bondad de ajuste de la tabla ANOVA y el *coeficiente de determinación* R^2. Iremos explicándolas una a una pero antes vamos a hablar de la variabilidad del modelo de regresión.

La variabilidad del ajuste se puede descomponer como

`Variación total= variación explicada modelo + variación residual`, es decir,

$SS_T = SS_M + SS_R$, donde

- $SS_T = \sum(y - \bar{y})^2$ es la cantidad total de variabilidad existente al aplicar el modelo más básico, el modelo nulo (la media).
- $SS_R = \sum(y - \hat{y})^2$ representa el grado de imprecisión cuando se ha ajustado el mejor modelo a los datos.
- $SS_M = SS_T - SS_R$ muestra cómo mejora la predicción al usar el modelo de regresión en vez predecir con la media. Es la reducción de la imprecisión al ajustar el modelo de regresión a los datos.

Si SS_M es grande entonces el modelo de regresión es muy diferente de la media, lo que significa que se ha hecho una gran mejora a la hora de predecir la variable dependiente.

6.2.5. Coeficiente de determinación, R^2

El *coeficiente de determinación* que representa la proporción de mejora causada por el modelo, es decir, la proporción de variabilidad de la variable dependiente (Y) explicada por el modelo (SS_M), relativa a toda la variabilidad existente en el modelo (SS_T). Se puede escribir como

$$R^2 = \frac{SS_M}{SS_T}.$$

Para la regresión lineal simple, R^2 se corresponde con el cuadrado de la correlación entre Y y X.

Una variante de esta medida es la R^2 *ajustada* que se utiliza para la regresión múltiple, pues tiene en cuenta el número de grados de libertad. Vemos cómo se define.

Utilizando la fórmula de la variación total tenemos la siguiente igualdad

$$R^2 = \frac{SS_M}{SS_T} = 1 - \frac{SS_R}{SS_T}$$

y a partir de ella se define la R_a^2 dividiendo por los grados de libertad la introducción de variables innecesarias en el modelo

$$R_a^2 = 1 - \frac{SS_R/df_R}{SS_T/df_T}$$

Al añadir al modelo una variable que no aporte nada el df_R disminuye, por lo que el cociente $\frac{SS_R}{df_R}$ crecerá, haciendolo también $\frac{SS_R/df_R}{SS_T/df_T}$. Esto implica por tanto que el valor de la R_a^2 sea cada vez más pequeño.

Mientras que R^2 nos dice cuánta varianza de Y representa el modelo de regresión, la R_a^2 cuantifica la varianza de Y que representaría el modelo si este hubiera sido obtenido de la población donde hemos tomado la muestra. Si los valores de R^2 y R_a^2 están próximos significa que el modelo de regresión es bueno.

Estas medidas toman valores entre 0 y 1, y cuanto más se aproximen a 1 mejor será el ajuste, y por lo tanto, mayor la fiabilidad de las predicciones que con él realicemos.

Observación: ni R^2 ni R_a^2 son una indicación directa de la eficacia del modelo en la predicción de nuevas observaciones.

6.2.5. Test F

La última medida de ajuste que vamos a estudiar es el `test F`, una medida de cuánto ha mejorado el modelo prediciendo la variable dependiente con respecto al nivel de inexactitud del modelo. Se define como

$$F = \frac{MS_M}{MS_R},$$

donde MS son las medias de las sumas de cuadrados. Se definen como las sumas de cuadrados entre sus grados de libertad. Así tenemos

$$MS_M = \frac{SS_M}{df_M}$$

$$MS_R = \frac{SS_R}{df_R}$$

Un buen modelo debe tener un valor F grande (mayor que 1) ya que el numerador, la mejora en la predicción del modelo, será mayor que denominador, la diferencia entre el modelo y los datos observados.

Otra medida importante que se obtiene a partir de la suma de cuadrados de los residuos es el error estándar que se define como

$$SE_R = \sqrt{MS_R}.$$

Vamos a aplicar todo esto en R continuando con el ejemplo anterior.

6.2.5. Tabla ANOVA

Volvemos al ejemplo de las categorías y las ausencias. Obtenemos la correspondiente tabla ANOVA donde vemos la descomposición de la variabilidad del modelo

```
anova( modelAu )
```

```
Analysis of Variance Table

Response: Ausencias
Df Sum Sq Mean Sq F value  Pr(>F)
Categoria  1    783     783    22.5 0.00031 ***
Residuals 14    487      35
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Observamos que la variabilidad explicada por el modelo, SSM=782.70, es superior a la que queda por explicar (residuos), SSR=487.05 y el estadístico F=22.5, mayor que 1. Además, volviendo a ver el resumen del modelo

```
### F-statistic:  22.5 on 1 and 14 DF,  p-value: 0.0003144
```

tenemos que el p-valor asociado con el estadístico F es inferior a 0.05.

La conclusión es que hay evidencias suficientes para poder rechazar la hipótesis nula, $F = 1$ y por tanto, resulta posible establecer un modelo de regresión lineal para explicar el comportamiento de las ausencias en función de la categoría del empleado.

6.2.5.3.1. Coeficiente de determinación En el `modelAu` el valor de R^2 es `Multiple R-squared: 0.6164`, alrededor del 62 % de la variabilidad de *Ausencias* es explicada por la recta ajustada.

6.2.6. Análisis de los parámetros del modelo

El test ANOVA significativo nos dice si el modelo tiene, en general, un grado de predicción significativamente bueno para la variable resultado, pero no nos dice nada sobre la contribución individual del modelo. Para encontrar los parámetros del modelo y su significación tenemos que volver a la parte `Coefficients` en el resumen del modelo.

```
summary( modelAu )$coefficients
```

```
Estimate Std. Error t value Pr(>|t|)
(Intercept)    41.60      3.579   11.62 1.41e-08
Categoria      -2.29      0.483   -4.74 3.14e-04
```

Observando la tabla vemos que $\beta_0 = 41.6$ (`intercept`) que podemos interpretar como que si no hubiera categorías ($X = 0$) el modelo predice que en la empresa habría un 41.6 % de ausencias, aunque en este caso no tiene sentido.

Por otro lado, β_1 es la pendiente de la recta y representa el cambio en la variable dependiente (ausencias) asociado al cambio de una unidad en la variable predictora. Si nuestra variable predictora incrementa una unidad, nuestro modelo predice que las ausencias se reducirán en 2.3, pues en este caso $\beta_1 = -2.2919$. Por tanto, la ecuación del modelo queda $Y = 41.6 - 2.3X$.

6.2.7. Diagnóstico del modelo

En este apartado hemos hecho uso tanto de [53] como de @palomo para el desarrollo del mismo.

Una vez que tenemos el modelo ajustado procedemos con su diagnóstico, que se realiza a través del análisis de los residuos, e_i.

- Las hipótesis de linealidad, homocedasticidad e independencia se contrastan a través de un análisis gráfico que enfrenta los valores de los residuos, e_i, con los valores ajustados $\hat{x}_i$.
- Las hipótesis de media cero, varianza constante, incorrelación y normalidad la comprobamos analíticamente.

Comenzaremos con **el análisis gráfico**. Los residuos deberían formar una nube de puntos sin estructura y con, aproximadamente, la misma variabilidad por todas las zonas como se muestra en el gráfico.

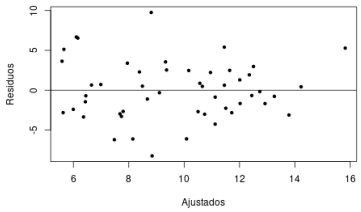


Figura 6.12: *Análisis gráfico de residuos con tendencia lineal.*

En los siguientes gráficos no se cumplen las hipótesis. Los residuos de esta primera gráfica muestran una estructura que sugiere una relación no lineal entre las variables

y los de la siguiente sugieren la *ausencia de homocedasticidad* (figura 6.14).

Continuamos ahora realizando el **diagnóstico analítico**. El primer paso es obtener los residuos, valores ajustados y estadísticos del modelo analizado para poder así estudiar si se cumplen los supuestos del mismo.

Obtención de residuos, valores ajustados y estadísticos necesarios

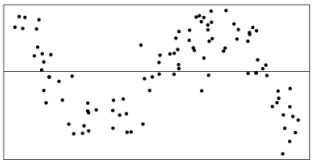


Figura 6.13: *Análisis gráfico de residuos con tendencia.*

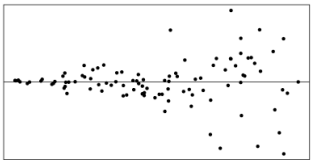


Figura 6.14: *Análisis gráfico del supuesto de homocedasticidad.*

Para ello, añadimos los correspondientes resultados a nuestros datos a través del siguiente código:

```
datos$fitted.modelAu     <- fitted( modelAu )
datos$residuals.modelAu <- residuals( modelAu )
datos$rstudent.modelAu  <- rstudent( modelAu )
```

El resultado es la creación de las siguientes variables:

- `fitted.modelAu`: valores ajustados (valores de la variable respuesta) para las observaciones originales de la predictora.
- `residuals.modelAu`: residuos del modelo, esto es, diferencia entre valor observado de la respuesta y valor ajustado por el modelo.
- `rstudent.modelAu`: residuos estudentizados del modelo ajustado.
- `obsNumber`: número de la observación en el orden en que has sido recogidas.

Vamos a utilizar todas estas variables para estudiar si nuestro modelo

cumple las hipótesis.

6.2.7. Test de normalidad (test de Shapiro)

Empezamos el análisis con un gráfico `qqplot`, que enfrenta los valores reales a los valores que obtendríamos si la distribución fuera normal. Si los datos reales se distribuyen normalmente, estos tendrán la misma distribución que los valores esperados y en el gráfico `qqplot` obtendremos una linea recta en la diagonal

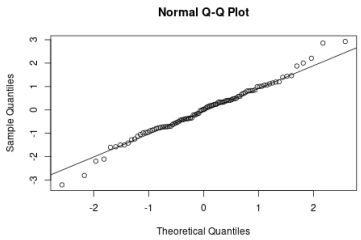


Figura 6.15: *Q-Q Plot.*

Analizamos nuestros residuos

```
shapiro.test( datos$rstudent.modelAu )
```

```
Shapiro-Wilk normality test

data:  datos$rstudent.modelAu
W = 0.8, p-value = 0.006
```

```
qqnorm( datos$rstudent.modelAu, main = "Normal(0,1)" )
qqline( datos$rstudent.modelAu )
```

Tenemos problemas con la condición de normalidad de los errores ya que obtenemos un p-valor para el contraste de `0.0063`, inferior a `0.05`. Como en el gráfico `qqplot` los puntos no se sitúan en la diagonal, efectivamente vemos que los datos no son normales.

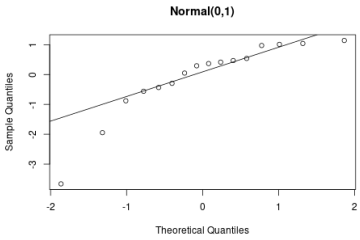


Figura 6.16: *Kolmogorov-Smirnov.*

6.2.7. Homogeneidad de varianzas

```
library( lmtest )
bptest( modelAu )
```

```
studentized Breusch-Pagan test


data:  modelAu
BP = 2, df = 1, p-value = 0.1
```

Existe homogeneidad pues la significación es mayor de 0.05, la varianza es constante a lo largo de la muestra.

6.2.7. Autocorrelación (test de Durbin-Watson)

Hemos asumido que los residuos son incorrelados, vamos a comprobarlo.

```
plot( datos$residuals.modelAu, pch = 20, ylab = "Residuos", xlab= "Índices")
abline( h = cor( datos$Ausencias, datos$Categoria ) )
```

Si hubiera una correlación seria, veríamos picos más largos de residuos por encima y por debajo de la línea de correlación. A menos que estos efectos sean fuertes, puede ser difícil de detectar la autocorrelación, por ello realizamos el *contraste de Durbin-Watson*.

```
dwtest( Ausencias ~ Categoria, alternative = "two.sided", data = datos )
```

```
Durbin-Watson test
```

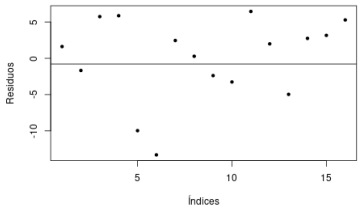


Figura 6.17: *Autocorrelacion.*

```
data:  Ausencias ~ Categoria
DW = 2, p-value = 0.5
alternative hypothesis: true autocorrelation is not 0
```

En el contraste de autocorrelación también aceptamos la hipótesis nula de que no existe correlación entre los residuos con un p-valor superior a 0.05.

Una vez comprobado el resto de supuestos del modelo, vamos a intentar solucionar el problema de normalidad. Lo primero que hacemos es representar de nuevo los datos en un QQ-plot y un diagrama de dispersión para detectar posibles perturbaciones.

```
par( mfrow = c( 1, 2 ) )
qqnorm( datos$rstudent.modelAu, main = "normal(0,1)" )
qqline( datos$rstudent.modelAu )
plot(   datos$rstudent.modelAu,
pch = 20, ylab = "Residuaos", xlab = "Índices"  )
```

Si observamos de nuevo el gráfico vemos que hay un punto que está totalmente fuera de lugar, lo que parece en principio un valor atípico. Vamos a realizar un test de valores atípicos (Bonferroni).

6.2.7. Valores atípicos

Un valor atípico es aquel que difiere sustancialmente de la tendencia general de los datos. Estos valores atípicos pueden perjudicar el modelo ya que afectan a los coeficientes de regresión estimados. Veamos gráficamente cómo pueden influir a la recta de regresión [49].

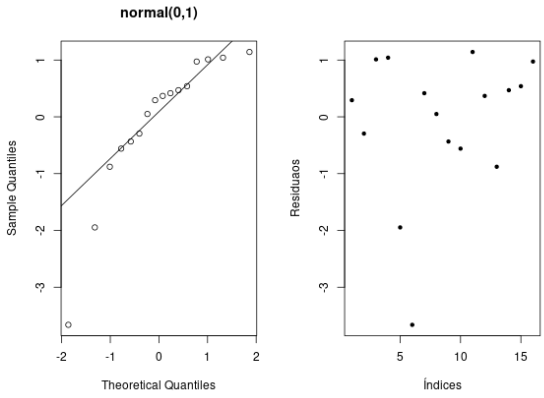


Figura 6.18: *Q-Q plot y diag de dispersión.*

En los gráficos la linea discontinua representa la recta de regresión calculada sin considerar el punto P.

En la figura 6.19 primero tenemos que el punto P sí es influyente pues modifica sustancialmente la recta de regresión. Mientras que en el segundo el punto P apenas influye en el modelo.

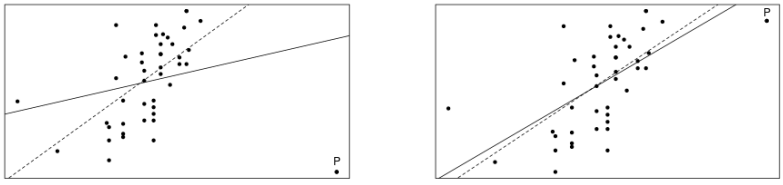


Figura 6.19: *Valores atípicos, Influencia.*

En el caso de observar valores atípicos los pasos a seguir son:

1. Descartar que sea un error.
2. Analizar si es un caso influyente.
3. En caso de ser influyente calcular las rectas de regresión incluyéndolo y ex-

cluyéndolo, y elegir la que mejor se adapte al problema y a las observaciones futuras.

Para el estudio de los valores atípico vamos a usar los **residuos estandarizados**, los residuos divididos por una estimación de su error estándar. Existen unas reglas generales:

1) Residuos estandarizados con un valor absoluto mayor de 3.29 (redondearemos a 3) son causa de preocupación ya que es improbable que en una muestra media un valor tan grande ocurra por azar.

2) Si más del 1% de los valores muéstrales tienen residuos estandarizados con un valor absoluto mayor de 2.58 (podemos decir 2.5) hay evidencias de que el nivel de error en nuestro modelo es inaceptable (ajuste pobre del modelo a los datos).

3) Si más del 5% de los casos tienen residuos estandarizados con un valor absoluto mayor de 1.96 (usamos 2 por conveniencia) entonces vuelven a haber indicios de que el modelo es una pobre representación de los datos reales.

Vamos a hacer un **estudio de valores atípicos de nuestro modelo**. Empezamos con un gráfico en el que representamos el diagrama de puntos y un *boxplot* para cada una de las variables.

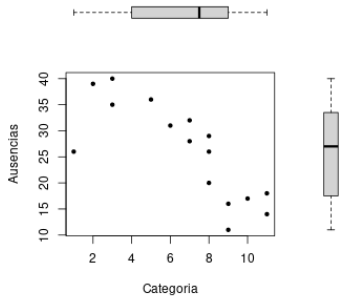


Figura 6.20: *Dispersión y boxplots.*

En la variable `Categoria` vemos que la mediana no está centrada en la media, los datos no son uniformes. Con la variable `Ausencias` ocurre lo mismo.

Lo bueno es que en ninguno de los dos casos se aprecian valores atípicos [54].

Continuamos con un análisis más analítico:

```
library( car )
outlierTest( modelAu, cutoff = 0.05, n.max = 10, order = TRUE )
```

```
rstudent unadjusted p-value Bonferroni p
6    -3.66           0.00287       0.0459
```

```
influencePlot( modelAu, id.n = 2 )
```

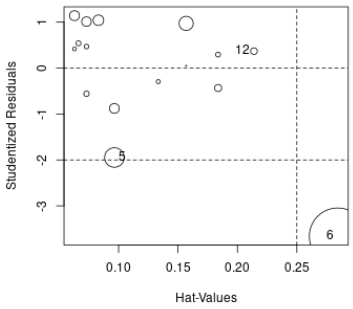


Figura 6.21: *Reg lineal. Residuos estandarizados.*

```
StudRes    Hat  CookD
5    -1.947 0.0965 0.1688
6    -3.662 0.2844 1.4127
12    0.368 0.2139 0.0197
```

El test y el gráfico nos indican que la observación número 6 es un valor atípico. Las observaciones 5 y 12 que vemos en el gráfico son medidas influyentes para ver si llegan a ser atípicos dibujamos el gráfico de las distancias de *Cook* [53].

```
cook <- cooks.distance( modelAu )
labels <- rownames( datos )
library( faraway )
halfnorm( cook, 3, labs = labels, ylab = "Distancia de Cook" )
```

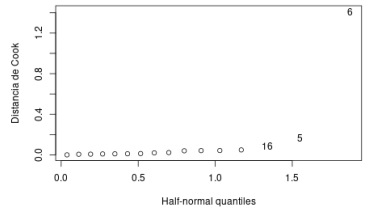


Figura 6.22: *Reg lineal. Distancia de Cook.*

Se confirma que el valor 6 es un atípico, mientras que los puntos 16y 5 no lo son por ser su distancia de Cook menor que 1.

Aunque nunca es recomendable suprimir datos salvo estar seguros de que ha sido una mala medición o cualquier otro tipo de error, en este caso y en vista de lo obtenido, decidimos eliminar dicho dato.

```
datos <- datos[ -c( 6 ), ]
head( datos )
```

```
Categoria Ausencias fitted.modelAu residuals.modelAu
1        11         18          16.4              1.62
2        10         17          18.7             -1.68
3         8         29          23.3              5.74
4         5         36          30.1              5.86
5         9         11          21.0             -9.97
7         7         28          25.6              2.45
rstudent.modelAu
1            0.293
2           -0.295
3            1.012
4            1.041
5           -1.947
7            0.416
```

NOTA: Cuidado con la eliminación de datos. ¡El diagnóstico del modelo es para fines predictivos! Para obtener un buen modelo aunque sin fines predictivos, únicamente debemos evitar el problema de la multicolinealidad.

Tras eliminar el valor atípico de la base de datos volvemos a realizar el test de *Shapiro-Wilk* para comprobar si se cumple ahora la condición de normalidad

```
shapiro.test( datos$rstudent.modelAu )
```

```
Shapiro-Wilk normality test

data:  datos$rstudent.modelAu
W = 0.9, p-value = 0.2
```

Como este nuevo p-valor es mayor que 0.05 ahora sí existe normalidad en los datos. Una vez solucionados los problemas de diagnóstico pasamos a la fase de predicción.

6.2.8. Predicción

Tenemos un modelo de regresión con la capacidad de relacionar la variable predictora y la variable dependiente. Podemos utilizarlo ahora para predecir eventos futuros de la variable dependiente a través de nuevos valores de la variable predictora.

Para ello debe verificarse alguna de las siguientes condiciones

- el valor de la predictora está dentro del rango de la variable original.
- si el valor de la predictora está fuera del rango de la original, debemos asegurar que los valores futuros mantendrán el modelo lineal propuesto.

6.2.8. Predicción de nuevas observaciones

```
x0      <- seq( min( datos$Categoria ), max( datos$Categoria ), length = 15 )
dfp     <- data.frame( Categoria = x0 )
pred.ip <- predict( modelAu, dfp, interval = "prediction",
se.fit = TRUE, data = datos )
head( pred.ip$fit )
```

```
fit  lwr  upr
1 37.0 23.1 50.9
2 35.5 21.8 49.3
```

```
3 34.1 20.5 47.6
4 32.6 19.2 46.0
5 31.1 17.9 44.4
6 29.6 16.5 42.8
```

Dibujamos las bandas de predicción, que reflejan la incertidumbre sobre futuras observaciones:

```
matplot( x0, pred.ip$fit, type = "l", xlab = "Categoria", ylab = "Ausencias" )
```

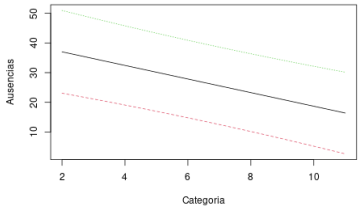


Figura 6.23: *Predicción de nuevas observaciones.*

Supongamos que no tuviéramos los datos en la escala original de la variable dependiente, sino que los hemos transformado mediante alguna función. En ese caso, para obtener las predicciones originales basta con deshacer la correspondiente transformación. Si hubiésemos transformado, por ejemplo, los datos originales mediante `log( )`, el código para obtener las predicciones sería

```
newpred <- exp( pred.ip$fit )
head( newpred )
```

6.2.8. Intervalos de confianza para los predictores

Dado un nuevo conjunto de predictores, x_0, debemos evaluar la incertidumbre en esta predicción. Para tomar decisiones racionales necesitamos algo más que puntos estimados. Si la predicción tiene intervalo de confianza ancho entonces entonces los resultados estarán lejos de la estimación puntual.

Las bandas de confianza reflejan la incertidumbre en la línea de regresión (lo bien que la línea está calculada).

```
pred.ic <- predict( modelAu, dfp, interval = "confidence",
se.fit = TRUE, data = datos )
head( pred.ic$fit )
```

```
fit  lwr  upr
1 37.0 31.2 42.9
2 35.5 30.2 40.8
3 34.1 29.3 38.8
4 32.6 28.3 36.9
5 31.1 27.2 35.0
6 29.6 26.1 33.2
```

Dibujamos las bandas de confianza, que además reflejan la incertidumbre sobre futuras observaciones:

```
library( graphics )
matplot( x0, pred.ic$fit, type = "l", xlab = "Categoria", ylab = "Ausencias" )
```

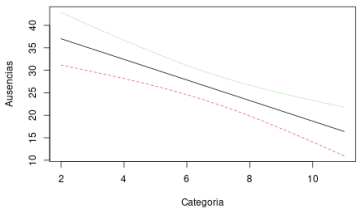


Figura 6.24: *Intervalos de confianza para predicciones.*

Por último podemos hacer un gráfico con la nube de puntos y los dos bandas, la de confianza y la de predicción [50].

```
plot( datos$Categoria, datos$Ausencias, pch = 20,
ylim = range ( datos$Categoria, pred.ip, na.rm = TRUE ),
xlab = "Categoria", ylab = "Ausencias" )

# Añadimos las bandas
matlines(dfp$Categoria, pred.ic$fit, lty = c( 1 ,2 ,2), lwd = 1.5, col = 1)
matlines(dfp$Categoria, pred.ip$fit, lty = c( 1 ,3 ,3), lwd = 1.5, col = 1)
```

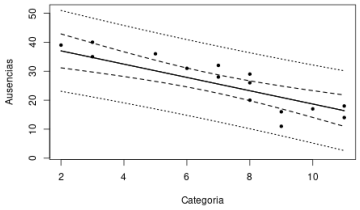


Figura 6.25: *Bandas e intervalos de confianza.*

6.2.9. Resumen de código en R

```
#Leer los datos de un fichero .csv
df <- read.table( "40A-file.csv", sep = ";", head = TRUE )

# Correlación
### Gráfico de dispersión (nube de puntos)
plot( df$var1, df$var2 )

### Normalidad de las variables explicativas
shapiro.test( df$var2 )

### Calculamos la correlación entre las varibales a estudiar
cor( df$var1, df$var2 )

#### Además de calcularla vemos su significación con un test
cor.test( df$var1, df$var2, method = "pearson" )

### Calculamos la correlación de una matriz de variables
ndf <- data.frame( df$var1, df$var2, df$var3, df$var4 )
cor( ndf, use = "everything", method = "pearson" )

### Coeficiente de determinación (R^2)
cor( ndf, use = "everything" ) ^ 2

### Hacemos el test de correlaciones para la matriz (reg. multiple)
library( "psych" )
```

```r
corr.test( ndf, use = "complete", method = "pearson" )


# Modelo de regresión simple

### Creamos el modelo de regresión
model <- lm( var1 ~ var2, data = df )

### Representamos gráficamente el ajuste
plot( df$var1, df$var2, xlab = "var1", ylab = "var2" )
abline( model )

### Resumen del modelo
summary( model )

### Estudiamos los coeficientes del modelo
summary( model )$coefficients

#### Intervalos de confianza para los coeficientes
confint( model, level = 0.95 )

### tabla ANOVA (ajuste del modelo)
anova( model )


# Diagnóstico del modelo (comprobar supuestos)

### Obtención de los residuos

df$fitted.model   <- fitted( model )
dfresiduals.model <- residuals( model )
df$rstudent.model <- rstudent(  model )

#### Normalidad
shapiro.test( df$rstudent.model )
qqnorm( df$rstudent.model, main = "Normal(0,1)" )
qqline( df$rstudent.model )
```

```r
#### Homogeneidad de varianzas
library( lmtest )
bptest( model )

#### Autocorrelación
plot( df$residuals.model, ylab = "Residuaos", xlab = "Índices" )
abline( h = cor( df$var1, df$var2 ) )
dwtest( var1 ~ var2, alternative = "two.sided", data = df )

#### Valores atípicos
library( car )
outlierTest( model, cutoff = 0.05, n.max = 10, order = TRUE )

### Predicción
x0   <- seq( min( df$var2 ), max( df$var2 ), length = 15 )
pred <- predict( model, data.frame( var2 = x0 ), interval = "prediction",
se.fit = TRUE, data = df )
head( pred )


#### Intervalo de confianzar para los predictores
ic <- predict( model, data.frame( var2 = x0 ), interval = "confidence",
se.fit = TRUE, data = df )
head( ic )

##### banda de confianza
#library( graphics )
matplot( x0, ic$fit, type = "l", xlab = "var2", ylab = "var1" )
```

6.3. Regresión lineal múltiple

6.3.1. Introducción

En la regresión lineal simple predecíamos la variable resultado Y a partir de los valores de X, usando la ecuación de una linea recta. Con los valores que habíamos ido obteniendo de X e Y calculábamos los parámetros de la ecuación ajustando el modelo a los datos mediante el método de mínimos cuadrados. La regresión múltiple es una extensión lógica de esto a situaciones en las que hay más de una variable predictora. La nueva ecuación será

$$Y_i = (\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_n X_{ni}) + e_i.$$

Básicamente se trata de la misma ecuación que para la regresión simple excepto porque hemos incluido predictores extra. Cada predictor tienen asociado su propio coeficiente y predecimos la variable dependiente a partir de una combinación de todas las variables más un residuo, e_i, la diferencia entre el valor ajustado y observado de Y en la i-ésima observación.

Los coeficientes de regresión se pueden interpretar como:

- β_i el efecto medio (positivo o negativo) sobre la variable dependiente al aumentar en una unidad el valor de la predictora $X_i, i = 1, \cdots, k$.
- β_0 el valor medio de la variable dependiente cuando las predictoras son cero.

6.3.2. Ejemplo de un modelo de regresión lineal múltiple

Para entender el modelo de regresión lineal múltiple vamos a usar un ejemplo de una empresa de fabricación y reparto de pizzas. Utilizaremos la base de datos pizza.rda.

Planteamos el modelo $sales \sim ads + cost$ que tendrá ecuación es

$$sales = \beta_0 + \beta_1 ads + \beta_2 cost + e.$$

6.3.2. Análisis de correlación

Comenzamos representando los datos en una nube de puntos múltiple, fig 6.26, donde vemos la relación entre cada par de variables.

```
load( "files/40A-pizza.rda" )
pairs( pizza, panel = panel.smooth )
```

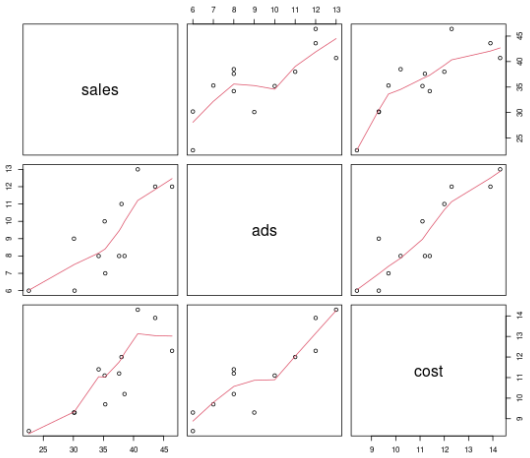


Figura 6.26: *Análisis de correlación.*

```
cor( pizza, use = "everything", method = "pearson" )
```

```
sales   ads  cost
sales 1.000 0.781 0.820
ads   0.781 1.000 0.895
cost  0.820 0.895 1.000
```

vemos que todas las variables tiene una correlación elevada.

6.3.2. Ajuste del modelo

```
modelPizza1 <- lm( sales ~ ads + cost, data = pizza )
summary( modelPizza1 )
```

```
Call:
```

```
lm(formula = sales ~ ads + cost, data = pizza)

Residuals:
Min      1Q Median      3Q     Max
-5.698 -1.822 -0.666  2.447  6.012

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.584      8.542    0.77     0.46
ads            0.625      1.120    0.56     0.59
cost           2.139      1.470    1.45     0.18

Residual standard error: 3.99 on 9 degrees of freedom
Multiple R-squared:  0.684, Adjusted R-squared:  0.614
F-statistic: 9.74 on 2 and 9 DF,  p-value: 0.0056
```

El error típico residual es 3.99, la $R^2 = 0.684$, aunque para el modelo múltiple es mejor fijarnos en su valor ajustado $R_a^2 = 0.614$. Esto que significa que la recta de regresión explica el 61 % de la variabilidad del modelo. Además, $F = 9.74$ con una significación $p < 0.05$, lo que nos dice que nuestro modelo de regresión resulta significativamente mejor que el modelo básico.

6.3.3. Comparación de modelos

Pretendemos seleccionar el "mejor" subconjunto de predictores por varias razones

1. Explicar los datos de la manera más simple. Debemos eliminar predictores redundantes.

2. Predictores innecesarios añade ruido a las estimaciones.

3. La causa de la multicolinealidad es tener demasiadas variables tratando de hacer el mismo trabajo. Eliminar el exceso de predictores ayuda a la interpretación del modelo.

4. Si vamos a utilizar el modelo para la predicción, podemos ahorrar tiempo y/o dinero al no medir predictores redundantes.

Puesto que tenemos dos variables explicativas disponemos de tres modelos posibles

$$modelo1 : sales \sim ads + cost$$

$$modelo2 : sales \sim ads$$

$$modelo3 : sales \sim cost$$

Vamos a ajustar cada uno de los modelos

```
modelPizza2 <- lm( sales ~ ads, data = pizza )
summary( modelPizza2 )

Call:
lm(formula = sales ~ ads, data = pizza)

Residuals:
Min      1Q Median      3Q     Max
-6.84  -2.76   0.68   3.83   4.90

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)   16.937      4.982    3.40   0.0068 **
ads            2.083      0.527    3.95   0.0027 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.21 on 10 degrees of freedom
Multiple R-squared:  0.61,  Adjusted R-squared:  0.571
F-statistic: 15.6 on 1 and 10 DF,  p-value: 0.00272

modelPizza3 <- lm( sales ~ cost, data = pizza )
summary( modelPizza3 )

Call:
lm(formula = sales ~ cost, data = pizza)

Residuals:
Min      1Q Median      3Q     Max
-5.702 -1.323 -0.665  1.758  6.896

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.173      7.109    0.59   0.5702
cost           2.872      0.633    4.54   0.0011 **
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.85 on 10 degrees of freedom
Multiple R-squared:  0.673, Adjusted R-squared:  0.64
F-statistic: 20.6 on 1 and 10 DF,  p-value: 0.00108
```

Para evitar la elección subjetiva del mejor modelo, podemos comparar todos los modelos mediante una tabla ANOVA conjunta para cada par de modelos. Hay que tener en cuenta que para poder comparar modelos estos deben estar encajados, es decir, que uno de ellos contenga al otro más otro conjunto de variables explicativas.

```
anova( modelPizza3, modelPizza1 )
```

```
Analysis of Variance Table

Model 1: sales ~ cost
Model 2: sales ~ ads + cost
Res.Df RSS Df Sum of Sq    F Pr(>F)
1     10 148
2      9 143  1      4.95 0.31   0.59
```

```
anova( modelPizza3, modelPizza2 )
```

```
Analysis of Variance Table

Model 1: sales ~ cost
Model 2: sales ~ ads
Res.Df RSS Df Sum of Sq F Pr(>F)
1     10 148
2     10 177  0     -28.7
```

Comparando ambas tablas anova deducimos que el modelo que mejor se ajusta a los datos es el `modelo3` pues reduce el error estándar.

Para este conjunto de datos, al tener sólo dos variables explicativas, aún lo podemos realizar "a mano" comparando los modelos de dos en dos. Pero cuando tenemos más variables este proceso se vuelve muy tedioso por lo que mejor hacerlo automáticamente con los *métodos paso a paso*.

6.3.4. Selección del "mejor" modelo

Existen distintos métodos a la hora de construir un modelo complejo de regresión con varios predictores

- El *método jerárquico* en el que se seleccionan los predictores basándose en un trabajo anterior y el investigador decide en qué orden introducir las variables predictoras al modelo.
- El *método de entrada forzada* en el que todas las variables entran a la fuerza en el modelo simultáneamente.
- Los *métodos paso a paso* que se basan en un criterio matemático para decidir el orden en que los predictores entran en el modelo.

Nosotros vamos a utilizar en R los métodos paso a paso, pero antes de verlos vamos a introducir una nueva medida de ajuste.

6.3.4. Criterio de información de Akaike (AIC)

El problema de utilizar R^2 para comparar modelos es que al añadir nuevas variables al modelo, esta medida siempre crece. Si estamos decidiendo cual de todos los modelos ajusta mejor a los datos, el modelo con más predictores siempre será el mejor ajustando. Para evitar esto se utiliza el AIC, una medida de ajuste que penaliza el modelo por tener más variables. Viene definido por

$$AIC = n \times \log \frac{SS_R}{n} + 2k,$$

donde n es el número de casos en el modelo, SS_R es la suma de cuadrados de los residuos del modelo y k es el número de variables predictoras.

El único problema es que no existen directrices sobre este criterio, sólo que si el AIC es mayor, el modelo es peor.

6.3.4. Metodos paso a paso

En R accedemos a estos métodos utilizando el comando `step( modelo, direction= "" )`, donde las direcciones pueden ser:

- `forward`: el modelo inicial contiene solo la constante β_0 y a partir de ahí el ordenador busca la variable predictora (dentro de las disponibles) que mejor predice la variable dependiente. Si este predictor mejora la habilidad del modelo para predecir la variable respuesta, ésta permanece en el modelo y se busca otra variable predictora. Para la segunda variable se usa como criterio de selección coger aquella que tenga la mayor correlación parcial con la respuesta. R tiene que decidir cuándo parar de añadir predictores al modelo, y para hacerlo se basa en el criterio de AIC.

- `blackward`: este método es el opuesto al anterior, R empieza con todas las variables predictoras en el modelo y estudia si el AIC disminuye cuando eliminamos del modelo alguna de las variables.

- `both`: empieza del mismo modo que el método `forward` salvo que cada vez que una variable predictora es añadida a la ecuación, se realiza un test de extracción del predictor menos útil.

El método más preferible es `backward` debido al *efecto represor* que ocurre cuando una variable predictora tiene influencia pero sólo si otra de las variables se mantiene constante. Al usar métodos *paso a paso* es aconsejable después hacer una validación cruzada, método que estudiaremos más adelante.

6.3.4.2.1. Métodos paso a paso en R Vamos a desarollar estos métodos con el ejemplo `bebidas.csv`. En él se pretende explicar las muertes por cirrosis según la bebida que consuman los pacientes [55].

```
dfbeb <- read.table( "40A-bebidas.csv", sep = ";", head = TRUE )
str( dfbeb )
```

```
'data.frame':   46 obs. of  6 variables:
$ caseid   : int  1 2 3 4 5 6 7 8 9 10 ...
$ cirrosis : num  41.2 31.7 39.4 57.5 74.8 59.8 54.3 47.9 77.2 56.6 ...
$ poblacion: int  44 43 48 52 71 44 57 34 70 54 ...
$ cerveza  : num  33.2 33.8 40.6 39.2 45.5 37.5 44.2 31.9 45.6 45.9 ...
$ vino     : int  5 4 3 7 11 9 6 3 12 7 ...
$ licorDuro: int  30 41 38 48 53 65 73 32 56 57 ...
```

```
# Eliminamos la variable "caseid" del conjunto pues no nos interesa.
dfbeb <- dfbeb[ , 2:6 ]
```

El archivo recoge los datos de muerte por cirrosis, el tamaño de la población, el consumo de cerveza, el consumo de vino y el consumo de licores duros. Echamos un primer vistazo a los datos

```
summary( dfbeb )
```

```
cirrosis       poblacion       cerveza          vino
Min.   : 28.0   Min.   :27.0   Min.   :31.2   Min.   : 2.00
1st Qu.: 48.9   1st Qu.:44.2   1st Qu.:35.6   1st Qu.: 6.25
Median : 57.6   Median :55.0   Median :42.2   Median :10.00
Mean   : 63.5   Mean   :56.3   Mean   :41.5   Mean   :11.59
3rd Qu.: 75.7   3rd Qu.:65.0   3rd Qu.:45.8   3rd Qu.:15.75
Max.   :129.9   Max.   :87.0   Max.   :56.1   Max.   :31.00
licorDuro
Min.   : 26.0
1st Qu.: 41.5
Median : 56.0
Mean   : 57.5
3rd Qu.: 68.8
Max.   :149.0
```

En todas las variables explicativas los valores de la media y la mediana son muy cercanos, lo cual es muy bueno.

Correlación

```
pairs( dfbeb, panel = panel.smooth )
```

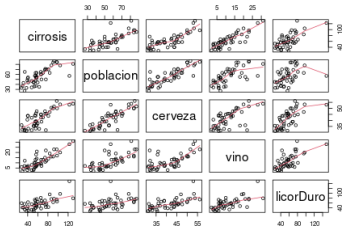


Figura 6.27: *Matriz de correlaciones por pares.*

```
cor( dfbeb, use = "everything", method = "pearson" )
```

```
cirrosis poblacion cerveza  vino licorDuro
cirrosis     1.000     0.749   0.783 0.845     0.682
poblacion    0.749     1.000   0.843 0.679     0.440
cerveza      0.783     0.843   1.000 0.640     0.686
vino         0.845     0.679   0.640 1.000     0.676
licorDuro    0.682     0.440   0.686 0.676     1.000
```

Como vemos en la tabla cirrosis está muy correlacionada con todas las variables explicativas y entre ellas también existe bastante correlación.

Pasamos a definir el __modelo general_ con todas las variables.

```
modelCir<- lm(cirrosis ~ poblacion + cerveza + vino + licorDuro, data =dfbeb)
summary( modelCir )
```

```
Call:
lm(formula = cirrosis ~ poblacion + cerveza + vino + licorDuro,
data = dfbeb)

Residuals:
Min      1Q  Median      3Q     Max
-18.872  -6.780   0.151   7.325  16.442

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -13.9631    11.4003   -1.22    0.228
poblacion     0.0983     0.2441    0.40    0.689
cerveza       1.1484     0.5830    1.97    0.056 .
vino          1.8579     0.4010    4.63 0.000036 ***
licorDuro     0.0482     0.1334    0.36    0.720
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.6 on 41 degrees of freedom
Multiple R-squared:  0.814, Adjusted R-squared:  0.795
F-statistic: 44.7 on 4 and 41 DF,  p-value: 1.95e-14
```

Analizamos el resumen de este primer modelo. Vemos que la mediana de los residuos es cercana a 0, lo cual es muy bueno pues queremos que los residuos

tengan media cero.

Observando los coeficientes vemos que según el *estadístico t* sólo son significativas las variables `vino` y `cerveza`, ahora aplicaremos el método de selección de modelos para ver si eliminamos alguna variable.

Aún así, el modelo con todas las variables tiene un error estándar de 10.46 y una $R^2 = 0.8136$, aunque para el modelo múltiple es mejor fijarnos en su valor ajustado $R_a^2 = 0.7954$. Esto quiere decir la recta de regresión explica el 79% de la variabilidad del modelo.

Por otro lado, que el estadístico F sea alto también es bueno, la variabilidad explicada por el modelo es mayor que la que se queda sin explicar. Así $F = 44.75$ con una significación $p < 0.05$ quiere decir que nuestro modelo de regresión resulta significativamente mejor que el modelo básico. Veamos ahora si podemos mejorar el ajuste.

Selección del modelo

Vamos a aplicar los tres métodos a nuestros modelos para cómo funciona cada uno de ellos. Comenzamos con el método más recomendable, la *eliminación hacia atrás (`"backward"`)*.

```
step( modelCir, direction = "backward" )
```

```
Start:  AIC=222
cirrosis ~ poblacion + cerveza + vino + licorDuro

Df Sum of Sq  RSS AIC
- licorDuro  1        15 4626 220
- poblacion  1        18 4629 220
<none>                   4611 222
- cerveza    1       436 5047 224
- vino       1      2415 7026 239

Step:  AIC=220
cirrosis ~ poblacion + cerveza + vino

Df Sum of Sq  RSS AIC
```

```
- poblacion  1         6 4632 218
<none>                  4626 220
- cerveza    1      1047 5673 228
- vino       1      4279 8905 248

Step:  AIC=218
cirrosis ~ cerveza + vino

Df Sum of Sq  RSS AIC
<none>                 4632 218
- cerveza  1      2460 7092 236
- vino     1      4951 9583 250

Call:
lm(formula = cirrosis ~ cerveza + vino, data = dfbeb)

Coefficients:
(Intercept)      cerveza         vino
-16.00         1.37         1.97
```

El proceso comienza con el modelo completo y con un AIC global de 221.95. En el primer paso se considera la eliminación de todas las variables explicativas y se calcula el AIC relativo a dicha eliminación. R selecciona la variable licorDuro (variable que quedan por encima de <none>), ya que su eliminación proporciona un AIC más pequeño. El AIC resultante tras este paso y con el que compararemos en el siguiente es 220.09.

Se considera ahora la posible eliminación de alguna de las tres variables restantes y se saca del modelo la variable población quedándonos con un AIC de 218.16.

Por último se considera la posibilidad de suprimir alguna de las dos variables restantes, sin embargo, vemos que el proceso considera que estadísticamente resulta mejor que permanezcan en el modelo ya que al eliminarlas el AIC aumenta, como mínimo, hasta 235.75.

Utilizamos ahora el método de *dos direcciones* cambiando el comando a

```
step( modelCir, direction = "both" )
```

```
Start:  AIC=222
cirrosis ~ poblacion + cerveza + vino + licorDuro

Df Sum of Sq  RSS AIC
- licorDuro  1        15 4626 220
- poblacion  1        18 4629 220
<none>                   4611 222
- cerveza    1       436 5047 224
- vino       1      2415 7026 239

Step:  AIC=220
cirrosis ~ poblacion + cerveza + vino

Df Sum of Sq  RSS AIC
- poblacion  1         6 4632 218
<none>                   4626 220
+ licorDuro  1        15 4611 222
- cerveza    1      1047 5673 228
- vino       1      4279 8905 248

Step:  AIC=218
cirrosis ~ cerveza + vino

Df Sum of Sq  RSS AIC
<none>                   4632 218
+ poblacion  1         6 4626 220
+ licorDuro  1         3 4629 220
- cerveza    1      2460 7092 236
- vino       1      4951 9583 250

Call:
lm(formula = cirrosis ~ cerveza + vino, data = dfbeb)

Coefficients:
(Intercept)      cerveza         vino
-16.00         1.37         1.97
```

Partimos de un AIC=221.95 y en el primer paso se elimina la variable licorDuro, reduciéndose a un 220.09. En el siguiente paso además de la eliminación del resto de las variables se considera la entrada de nuevo de la variable, aunque se opta por suprimir población reduciendo el AICa 218.16. En el último paso se compara entre la posibilidad de recuperar alguna de las variables eliminadas o suprimir alguna más. Se decide no hacer nada, ni eliminar más ni meter las antiguas, quedando el modelo con cerveza y vino.

Veamos por último *la selección hacia delante (forward)*. Debemos partir del modelo más sencillo, sólo con la constante, e indicar cuales son las posibles variables explicativas

```
mdlCir0 <- lm( cirrosis ~ 1 , data =dfbeb )
step(mdlCir0,direction = "forward", ~ poblacion + cerveza + vino +licorDuro)
```

```
Start:  AIC=291
cirrosis ~ 1

            Df Sum of Sq   RSS AIC
+ vino       1     17650  7092 236
+ cerveza    1     15158  9583 250
+ poblacion  1     13883 10859 255
+ licorDuro  1     11507 13235 264
<none>                   24741 291

Step:  AIC=236
cirrosis ~ vino

            Df Sum of Sq  RSS AIC
+ cerveza    1      2460 4632 218
+ poblacion  1      1419 5673 228
+ licorDuro  1       562 6530 234
<none>                   7092 236

Step:  AIC=218
cirrosis ~ vino + cerveza

            Df Sum of Sq  RSS AIC
```

```
<none>                    4632 218
+ poblacion  1      6.29 4626 220
+ licorDuro  1      2.73 4629 220

Call:
lm(formula = cirrosis ~ vino + cerveza, data = dfbeb)

Coefficients:
(Intercept)         vino      cerveza
-16.00         1.97          1.37
```

Es el mismo procedimiento que para el método hacia atrás pero aquí se parte del modelo sin variables explicativas y se considera en cada paso la posible inclusión de una nueva variable (los signos ahora son +). La primera variable que se añade al modelo es `vino` seguida de `cerveza` pues la inclusión de alguna de las otras incrementa el `AIC`.

En todos los métodos nos hemos quedamos con el mismo modelo final. La última parte del método muestra los coeficientes del modelo con el que nos quedamos finalmente, que es

```
modelCirf <- lm( cirrosis ~ cerveza + vino, data = dfbeb )
summary( modelCirf )
```

```
Call:
lm(formula = cirrosis ~ cerveza + vino, data = dfbeb)

Residuals:
Min     1Q Median     3Q    Max
-18.82  -6.85   0.06   7.22  16.37

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  -16.001     10.153   -1.58     0.12
cerveza        1.366      0.286    4.78  2.1e-05 ***
vino           1.972      0.291    6.78  2.7e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 10.4 on 43 degrees of freedom
Multiple R-squared:  0.813, Adjusted R-squared:  0.804
F-statistic: 93.3 on 2 and 43 DF,  p-value: 2.27e-16
```

En este modelo final la mediana de los residuos es prácticamente cero, lo que va a significar que los residuos van a tener una media muy cercana a 0. Vemos que las dos variables `cerveza` y `vino` son significativas. Tenemos un error estándar de 10.38, y un $R_a^2 = 0.8041$ lo que significa que el modelo explica un 80% de la variabilidad de los datos. Finalmente vemos que el *test F* es significativo ($p < 0.01$) con un valor elevado, lo cual nos indica que el modelo se ajusta significativamente a los datos.

```
anova( modelCirf )
```

```
Analysis of Variance Table

Response: cirrosis
Df Sum Sq Mean Sq F value  Pr(>F)
cerveza    1  15158   15158     141 3.8e-15 ***
vino       1   4951    4951      46 2.7e-08 ***
Residuals 43   4632     108
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La tabla anova nos confirma que las variables explicativas de nuestro modelo son significativas, y vemos que la suma de cuadrados explicada por el modelo es mucho mayor que la suma de cuadrados de los residuos, por tanto podemos afirmar que $R^2 \neq 0$.

Aplicar el modelo

Podemos utilizar la parte `Coefficients` proporciona el resumen del modelo para analizar individualmente la contribución de cada variable predictora la explicación de la dependiente.

Definimos el modelo reemplazando los b-valores en la ecuación inicial y obtenemos el modelo

$$cirrosis = -16.001 + 1.366 \times cerveza + 1.972 \times vino$$

6.3.5. Diagnóstico del modelo

Para este apartado nos hemos apoyado fundamentalmente en el libro [53].

Al haber generado el modelo basándonos en una muestra nos tenemos que preguntar si el modelo se ajusta bien a los datos observados o está influenciado por un pequeño número de casos, y por otro lado si el modelo se puede generalizar a otras muestras. Es un error pensar que porque un modelo se ajuste bien a los datos observados entonces podemos tomar conclusiones más allá de nuestra muestra.

Para poder generalizar un modelo de regresión debemos comprobar los supuestos del modelo, y una vez seguros de que se cumplen, para comprobar si el modelo se puede generalizar utilizaremos la *validación cruzada*. Empezamos analizando gráficamente los supuestos

```
plot( modelCirf, which = 1, pch = 20 )
```

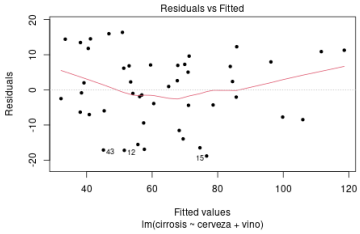


Figura 6.28: *Residuos frente a valores ajustados.*

Este primer gráfico, figura 6.28, enfrenta los errores residuales frente a sus valores ajustados. El residuos deben estar distribuidos al azar alrededor de la línea horizontal que representa un error residual de cero; es decir, no debe haber una tendencia clara en la distribución de puntos. Una tendencia en la variabilidad de los residuos sugiere que la varianza está relacionada con la media, violando el supuesto de varianza constante.

Si el gráfico tiene forma de embudo, es decir, si los puntos parecen estar más o menos extendidos a lo largo del gráfico, entonces lo más probable es que exista heterocedastididad en los datos. En este caso los datos parecen exhibir una ligera tendencia con un incremento de la varianza en los extremos.

Si hubiera algún tipo de curva en la gráfica entonces se ha violado el su-

puesto de linealidad. Y si los datos parecen seguir un patrón y además están más extendidos por en algunos puntos de la gráfica que en otros entonces probablemente se incumplan los supuestos de homogeneidad de varianza y linealidad.

En general, parace que en nuestro modelo no se violan ninguno de los supuestos.

```
plot( modelCirf, which = 2, pch = 20 )
```

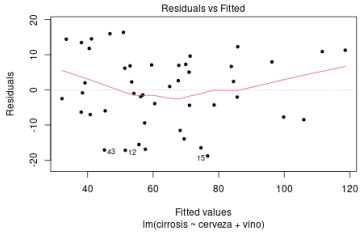


Figura 6.29: *Residuos tipificados frente a cuantiles de una distribución normal .*

En este gráfico, figura6.29, los residuos tipificados se trazan contra los cuantiles de una distribución normal estándar. Si los residuos se distribuyen normalmente los datos se deben situar a lo largo de la línea. En este caso, los datos no hacen parecen tener una distribución normal.

```
plot( modelCirf, which = 3, pch = 20 )
```

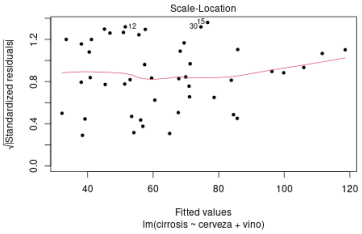


Figura 6.30: *escala-ubicación .*

El tercero es el gráfico *escala-ubicación*, figura 6.30, en el que los residuos están estandarizados por sus desviaciones estándar estimadas. Esta gráfica se utiliza para detectar si la difusión de los residuos es constante en el rango de

valores ajustados. Una vez más, se aprecia una tendencia muy leve en los datos de tal manera que los valores altos muestran una mayor variación.

```
plot( modelCirf, which = 5, pch = 20 )
```

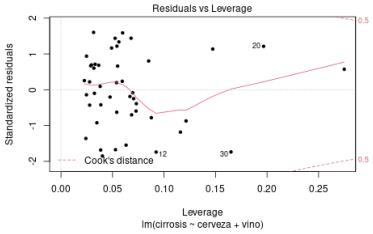


Figura 6.31: *Leverage* .

Finalmente el cuarto gráfico, figura 6.31, muestra el valor *leverage* de cada punto, la medida de su importancia en la determinación del modelo de regresión. Están representados los datos que ejercen mayor influencia.

Superponen al diagrama de puntos *leverage* las curvas de nivel para la distancia de Cook, que es otra medida de la importancia de cada observación a la regresión. Si la línea de distancia Cook abarca a algún punto de datos, significa que el análisis puede ser muy sensible a ese punto y quizá sea conveniente repetir el análisis excluyendo los datos. Distancias pequeñas significan que la eliminación de la observación tiene poco efecto sobre los resultados de la regresión y distancias mayores a 1 son sospechosas, sugieren la presencia de un posible valor atípico o de un modelo pobre.

Pasamos ahora a estudiar el modelo analíticamente, para ello obtenemos los residuos, los valores ajustados y estadísticos del modelo mediante el siguiente código:

```
dfbeb$fitted.modelCirf    <- fitted( modelCirf )
dfbeb$residuals.modelCirf <- residuals( modelCirf )
dfbeb$rstudent.modelCirf  <- rstudent( modelCirf )
```

6.3.5. Normalidad

En el gráfico *Q-Q plot* que vimos antes sugería falta de normalidad en los datos. Lo comprobamos

```
ks.test( dfbeb$rstudent.modelCirf, "pnorm" )
```

```
One-sample Kolmogorov-Smirnov test

data:  dfbeb$rstudent.modelCirf
D = 0.1, p-value = 0.6
alternative hypothesis: two-sided
```

```
hist( dfbeb$rstudent.modelCirf, xlab="residuos", main = "Histograma residuos"  )
```

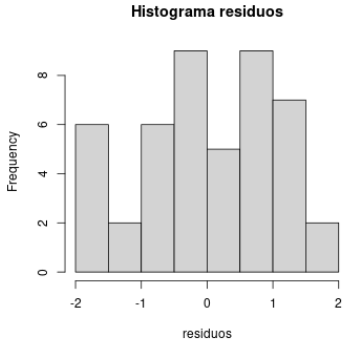


Figura 6.32: *Histograma de residuos.*

```
# densidad
```

El p-valor para el contraste de normalidad es mayor que 0.05 (p = 0.6434) y además el histograma se parece a una distribución normal (curva en forma campana) entonces no hay problemas de normalidad.

6.3.5. Homogeneidad de varianzas

```
bptest( modelCirf, studentize = FALSE, data = dfbeb )
```

```
Breusch-Pagan test
```

```
data:  modelCirf
BP = 0.7, df = 2, p-value = 0.7
```

Significación `p = 0.7166`, mayor de `0.05`, por lo que podemos decir que la varianza es constante a lo largo de la muestra.

6.3.5. Autocorrelación

```
dwtest( modelCirf, alternative = "two.sided", data = dfbeb )
```

```
Durbin-Watson test


data:  modelCirf
DW = 3, p-value = 0.07
alternative hypothesis: true autocorrelation is not 0
```

Aceptamos la hipótesis nula de que no existe correlación entre los residuos con un p-valor superior a `0.05`.

6.3.5. Casos atípicos y residuos

Podemos encontrar los valores atípicos observando grandes diferencias entre los datos muéstrales y los datos ajustados por el modelo, es decir, estudiando los **residuos**.

Si el modelo se ajusta bien a los datos muéstrales entonces todos los residuos serán pequeños, mientras que si el ajuste del modelo es pobre los residuos serán grandes. Además, si algún caso sobresale por tener un gran residuo este podría ser entonces un valor atípico.

Vamos a analizar si existen valores atípicos en nuestro ejemplo. En el primer gráfico enfrentamos `cirrosis` con `cerveza` y en el segundo `cirrosis` con `vino`.

```
Warning in par(fig = c(0, 0.8, 0, 0.8), new = TRUE): llamada
par(new=TRUE) sin gráfico
```

En la figura **??** se observan dos posibles valores atípicos par avarias variables.

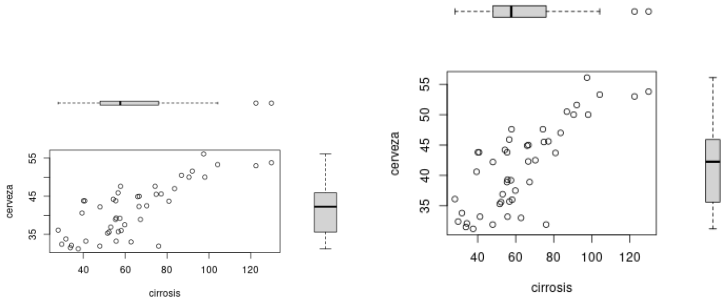


Figura 6.33: *Boxplots y dispersion.*

Se observan los mismos candidatos a valores atípicos. Hacemos el test de Bonferroni para comprobarlo.

```
outlierTest( modelCirf )
```

```
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
rstudent unadjusted p-value Bonferroni p
15    -1.91            0.0635           NA
```

Obtenemos que el valor 15 es un atípico.

6.3.6. Análisis de la influencia.

Con este análisis pretendemos ver si hay alguna observación que es demasiado influyente sobre los coeficientes del modelo, nos ayuda a determinar si el modelo de regresión es estable a lo largo de la muestra o si está perjudicado por unos pocos casos influyentes.

Utilizamos la función `influence.measures` que nos proporciona todas las medidas de influencia. Explicamos, a partir de los resultados de aplicar la función, cada una de las medidas:

```
infl <- influence.measures( modelCirf )
summary( infl )
```

```
Potentially influential observations of
lm(formula = cirrosis ~ cerveza + vino, data = dfbeb) :

dfb.1_ dfb.crvz dfb.vino dffit cov.r   cook.d hat
20 -0.10  -0.01     0.45     0.60  1.20    0.12   0.20_*
38  0.26  -0.30     0.31     0.35  1.45_*  0.04   0.27_*
```

Analizamos la tabla resumen:

- la primera columna indica el índice de las observaciones potencialmente influyentes.
- las columnas que comienzan con `dfb` proporcionan las observaciones potencialmente influyentes sobre cada uno de los coeficientes del modelo.
- la columna `dffits` identifica las observaciones influyentes según el estadístico *DFFITS*.
- la columna `cov.r` muestra las observaciones potencialmente influyentes según el estadístico *COVRATIO*.
- la columna `cook.d` proporciona la *distancia de Cook*.
- la última columna presenta las observaciones que pueden resultar influyentes según los `leverages`.

En cada columna el asterisco señala si realmente la observación puede ser influyente. En este caso tenemos que la observación 38 resulta influyente con el estadístico `cov.r`, y las 38 y la 20 para los 'leverages.

Analizamos un poco más estas medidas:

- Los **leverages (hat)** varían entre 0 (indicando que el caso no tiene influencia en absoluto) y 1 (indicando que ese caso tiene influencia completa sobre el modelo). Si ninguno de los casos ejerce excesiva influencia sobre el modelo entonces esperaremos que todos los valores 'leverage' estén entorno al valor medio (`(k+1)/n`), donde `k` es el número de predictores y `n` el número de participantes. Buscamos valores el doble o triple que (`(k+1)/n`) para considerarlos como influyentes.

- Para **la distancia de Cook** se considera que valores mayores que 1 pueden

ser causa de preocupación. Si un caso es un valor atípico pero su distancia de Cook es menor que 1, entonces no existe necesidad real de eliminar este dato ya que realmente no tiene un gran efecto sobre el modelo de regresión.

Lo estudiamos gráficamente. En el primer gráfico se muestra mediante círculos de distinto tamaño la influencia que cada punto ejerce sobre el modelo y en el segundo están representadas en orden ascendente las distancias de Cooks.

```
influencePlot( modelCirf, id.n = 2 )
```

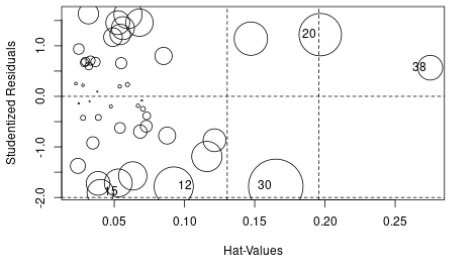


Figura 6.34: *Influencia de puntos en el modelo.*

```
StudRes    Hat  CookD
12  -1.783 0.0924 0.1027
15  -1.906 0.0402 0.0477
20   1.220 0.1967 0.1202
30  -1.781 0.1650 0.1989
38   0.566 0.2749 0.0412
```

Enla figura 6.34 vemos que las medidas más influyentes son la 30, la 20 y la 12. Vemos el gráfico de las distancias de Cook.

```
cook   <- cooks.distance( modelCirf )
labels <- rownames( dfbeb )
halfnorm( cook, 3, labs = labels, ylab = "Distancia de Cook" )
```

En la figura 6.35, volvemos a obtener que los puntos más influyentes son el 30, el 20 y el 12, pero como en ningún caso esta distancia es mayor que 1, pues para el valor más elevado es 0.2, podemos afirmar que ninguno de ellos es un caso atípico y no es necesario eliminarlos del modelo.

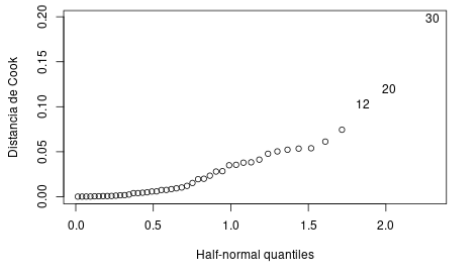


Figura 6.35: *Influencia de puntos en el modelo 2.*

La forma habitual de proceder es eliminar dichas observaciones del modelo y comenzar de nuevo todo el proceso, sin embargo como el modelo cumple todas las hipótesis, eliminar dichas observaciones podría provocar que el nuevo modelo fuera incorrecto y tuviéramos que volver al modelo anterior.

Hay que tener en cuenta que los límites marcados para identificar una observación como influyente son aproximados, y por tanto deben ser tomados como orientación, a salvo que el valor obtenido sea exageradamente llamativo.

6.3.7. Validación cruzada

Al utilizar métodos paso a paso es recomendable hacer una validación cruzada de nuestro modelo para evaluar su eficacia prediciendo la variable dependiente en una muestra diferente. Evaluar la precisión de un modelo a través de diferentes muestras es lo que se conoce como *validación cruzada*.

Para poder generalizar un modelo este debe ser capaz de predecir con precisión la misma variable dependiente del mismo conjunto de predictores en un grupo diferente de gente. Si aplicamos el modelo a una muestra diferente y su poder predictivo se reduce severamente, entonces no es generalizable.

El método usual es calcular además de la R^2 su valor ajustado, pues es un indicador de la pérdida de poder predictivo. Mientras R^2 nos dice cuánta varianza de Y representa el modelo de regresión, la R_a^2 cuantifica la varianza de Y que representaría el modelo si este hubiera sido obtenido de la población donde

hemos tomado la muestra. Si los valores de R^2 y R_a^2 están próximos significa que el modelo de regresión es bueno.

Sin embargo, esta medida ha sido criticada porque no dice nada sobre la efectividad del modelo de regresión si se aplica a un conjunto de datos totalmente distinto. Una alternativa sería *partir* los datos y cruzarlos, es decir, hacer una división aleatoria del conjunto de datos (p. ej un 80%-20%), calcular la ecuación de regresión en ambos conjuntos y comparar los modelos resultantes. Comparando los valores de R^2 y los b-valores en las dos muestras podemos saber la bondad del modelo original.

Para realizar la validación cruzada en R usamos la función `cv.lm( datos, modelo, m)`, donde `m`es el número de subconjuntos en los que asignamos los datos al azar. Cada subconjunto se retira del modelo,sucesivamente, mientras que los datos restantes se utiliza para volver a ajustar el modelo de regresión y predecir en las observaciones eliminados.

```
library( DAAG )
cv.lm( dfbeb, modelCirf, m = 2 )
```

6.3.8. Predicción

Para calcular las ecuaciones de predicción procedemos de forma similar al caso de regresión lineal simple, la única diferencia es que hay que dar valores predictivos para todas las variables que aparezcan en el modelo.

```
#Definiendo un intervalo para la vble vino.
x0<- seq( min( dfbeb$vino ), max( dfbeb$vino),length=length( dfbeb$vino))
dbp  <- data.frame( poblacion = 56, cerveza = 41, vino = x0, licorDuro = 58 )
pred<-predict(modelCirf,dbp, interval="prediction",se.fit = TRUE,data =dfbeb)
head(pred$fit )
```

```
fit  lwr  upr
1 43.9 22.1 65.8
2 45.2 23.4 67.0
3 46.5 24.8 68.1
4 47.7 26.2 69.3
5 49.0 27.5 70.5
```

```
6 50.3 28.8 71.7
```

6.3.9. Diagnósticos de colinealidad (multicolinealidad)

Si en un modelo de regresión lineal múltiple alguna variable predictora es combinación lineal de otras de las variables del modelo, entonces el modelo es irresoluble, debido a que en ese caso la matriz $X'X$ es singular, es decir, su determinante es cero y no se puede invertir.

Una variable X_1 es combinación lineal de $X_2, \ldots, X_i$ con $i > 2$, si dichas variables están relacionadas por la expresión $X_1 = \beta_1 + \beta_2 X_2 + ... + \beta_i X_i$, siendo los β_i constantes. En tal caso el coeficiente de correlación múltiple también será 1.

Por tanto, la *multicolinealidad* existe si hay una fuerte correlación entre dos o más variables predictoras del modelo, es decir, cuando alguno de los coeficientes de correlación simple o múltiple entre algunas de las variables independientes es 1. Si existe una colinealidad perfecta entre predictores es imposible obtener estimadores únicos para los coeficientes de regresión ya que hay un número infinito de coeficientes que funcionarían igual de bien.

En la práctica esta colinealidad exacta raras veces ocurre, pero sí surge con cierta frecuencia la llamada *casi-colinealidad*, cuando alguna variable es "casi" combinación lineal de otra u otras. Dicho de otro modo, algunos coeficientes de correlación simple o múltiple entre las variables independientes están cercanos a 1, aunque no llegan a dicho valor.

En ese caso la matriz $X'X$ es casi-singular, es decir, su determinante no es cero pero es muy pequeño. Como para invertir una matriz hay que dividir por su determinante surgen problemas de *precisión* en la estimación de los coeficientes, ya que los algoritmos de inversión de matrices pierden precisión al tener que dividir por un número muy pequeño, siendo además *inestables*.

Hay varias formas de **detectar este problema:**

- **Observar los estadísticos estimados**: cuando la prueba muestra que el modelo es globalmente significativo, es decir, que los coeficientes estimados son estadísticamente diferentes de cero, pero se encuentran unos valores

estimados bajos que demuestran que los coeficientes no son significativos.

- **Observar la matriz de correlación entre parejas de regresores**: si este coeficiente es mayor a 0.8 entonces la multicolinealidad es un problema grave. Sin embargo, esta condición se puede considerar suficiente pero no necesaria, la multicolinealidad puede existir a pesar de que las correlaciones sean comparativamente bajas (es decir, inferiores a 0.5).

- **Regresiones auxiliares**: dado que la multicolinealidad surge por la relación lineal entre variables explicativas, se pueden estimar regresiones entre las variables explicativas y adoptar la *regla práctica de Klien*. Este sugiere que si el modelo obtenido en la regresión auxiliar es mayor que el global obtenido con todos los regresores, hay un serio problema de multicolinealidad.

- **Estimar el Factor de Inflación de Varianza (FIV)**: indica si el predictor tiene una fuerte relación lineal con otro predictor y es el que vamos a calcular con R. Aunque no existen reglas generales se tienen los siguientes criterios:

    - Un `VIF > 10` es causa de preocupación.
    - Si `VIF` es sustancialmente mayor que `1` entonces la regresión puede verse perjudicada.
    - `Tolerancia = 1/VIF` debajo de `0.1` indica un problema serio.
    - `Tolerancia` debajo de `0.2` indica un problema potencial.

Si **identificamos multicolinealidad** no hay mucho que podamos hacer, la solución no es fácil:

- Podemos intentar eliminar la variable menos necesaria implicada en la colinealidad, a riesgo de obtener un modelo menos válido. Sin embargo, un problema común es no saber qué variable debemos omitir. Cualquiera de las variables problemáticas puede ser omitida, no hay fundamentos estadísticos para suprimir una variable en vez de otra.
- Se recomienda que si eliminamos una variable predictora, ésta se reemplace por otra igualmente importante que no tenga una colinealidad tan fuerte.
- Se puede intentar cambiar la escala de medida de la variable en conflicto (es decir, transformarla). Sin embargo estas transformaciones hacen al modelo muy dependiente de los datos actuales, invalidando su capacidad predictiva.
- También se puede recurrir a aumentar la muestra para así aumentar la

información en el modelo y ver si la multicolinealidad puede disminuir, aunque no siempre será posible.

- La última posibilidad, aunque más compleja cuando hay varios predictores, es hacer un análisis factorial y usar las puntuaciones del factor resultante como predictor.

Supongamos que estamos en el ejemplo de `dfbeb` y le realizamos un test de multicolinealidad al `modelCirf`:

```
vif( modelCirf )
```

```
cerveza    vino
1.69    1.69
```

```
sqrt( vif( modelCirf ) ) > 2
```

```
cerveza    vino
FALSE   FALSE
```

Nuestro modelo no presenta problemas de multicolinealidad.

6.3.10. Resumen de código en R

```
#Leer los datos de un fichero .csv
df <- read.table( "40A-file.csv", sep = ";", head = TRUE )

#### Primera aproximación a los datos
str( df )
summary( df )

# Correlación
# Gráfico de dispersión multivariante
pairs( df, panel = panel.smooth )

# Matriz de correlación
cor( df, use = "everything", method = "pearson" )
corr.test( df, use = "complete", method = "pearson" )
```

```
### Correlación parcial (si fuera necesario)
library( "ppcor" )
pcor.test( df$var1, df$var2, df$var3 )


# Modelo de regresión múltiple

### Creamos el modelo de regresión
modelo <- lm( var1 ~ var2 + var3 + ... , data = df )
summary(modelo) # analizamos el modelo inicial

### Comparación de modelos (encajados)
anova( model3, model1 )
anova( model3, model2 )

### Selección del modelo mediante los métodos paso a paso
#### Método hacia atrás
step( modelo, direction = "backward" )

#### Método de dos sentidos
step( modelo, direction = "both" )

#### Método hacia delante
mdlCir0 <- lm( var1 ~ 1 , data = df )
step( mdlCir0, direction = "forward",  ~ var1 + var2 + var3 + var4 )
modelo <- lm( var1 ~ var2 + var3, data = df )


# Análisis del modelo final
summary( modelo )
anova ( modelo )

### Diagnósitco del modelo


# Gráficamente
plot( modelo, which = 1 )
```

```r
plot( modelo, which = 2 )
plot( modelo, which = 3 )
plot( modelo, which = 5 )

### Contrastes
#### Obtenemos los residuos del modelo y valores ajustados
df$fitted.modelo    <- fitted( modelo )
df$residuals.modelo <- residuals( modelo )
df$rstudent.modelo  <- rstudent( modelo )

#### Normalidad
ks.test( df$rstudent.modelo, "pnorm" )
hist( df$rstudent.modelo, xlab = "residuos", main = "histograma residuos"  )

#### Homogeneidad de varianzas
library( lmtest )
bptest( modelo, studentize = FALSE, data = df )

#### Autocorrelación
dwtest( modelo, alternative = "two.sided", data = df )

#### Valores atípicos
library( car )
outlierTest( modelo )

#### Análisis de la influencia
##### Tabla con las medidas de influencia
infl <- influence.measures( modelo )
summary( infl )

##### Gráfico medidas influyentes
influencePlot( modelo, id.n = 2 )

##### Gráfico de las distancias de Cook
cook   <- cooks.distance( modelo )
labels <- rownames( df )
library( faraway )
```

```
halfnorm( cook, 3, labs = labels, ylab = "Distancia de Cook" )

### validación cruzada
library( DAAG )
cv.lm( df, modelo, m = 2 )

# Predicción.
### Valores concretos de cada vble
predict( modelo, data.frame( var1 = 39, var = 62, var3 = 18 ),
interval = "prediction", data = df )


# Poniendo un intervalo para una de las vbles.
x0   <- seq( min( df$var2 ), max( df$var2 ), length = length( df$var2 ) )
pred <- predict( modelo, data.frame( var2 = x0 ),
interval = "prediction", data = df )
head( pred )


# Multicolinealidad
library( car )
vif( modelo )
sqrt( vif( modelo ) ) > 2
```

6.3.11. Predictores categóricos. Variables dummy

Uno de los supuestos de la regresión lineal es que las variables del modelo deben ser continuas o categóricas con solo dos categorías. En el caso de variables con más de dos categorías usaremos lo que se conoce como **variables dummy**, variables ficticias, simuladas.

Esta codificación es una manera de representar varios grupos de personas pero usando sólo unos y ceros. El proceso consiste crear varias variables siguiendo estos pasos:

1. Contar el número de grupos que queremos re-codificar y restarle 1.

2. Crear tantas nuevas variables como el valor obtenido en 1. Estas serán las variables *dummy*.
3. Elegir uno de los grupos como el *grupo de referencia*, es decir, el grupo contra el que se van a comparar todos los demás grupos. Normalmente se toma el grupo control o aquel que representa a la mayoría de la población.
4. Elegido el grupo referencia fijamos el valor 0 a ese grupo en todas las variables dummy.
5. Para la primera variable dummy asignamos el valor 1 al primer grupo que queramos comparar contra el grupo referencia. Al resto de grupos le damos el valor 0.
6. En la segunda variable dummy damos el valor 1 al segundo grupo que queramos cotejar y 0 al resto de grupos.
7. Repetimos el proceso hasta acabar con todas las variables dummy.

Veamos cómo hacer una **codificación dummy en R**. Para ello utilizamos el conjunto de datos `40A-festival.csv`, archivo que contiene los niveles de higiene de los asistentes a un famoso festival murciano y una variable que mide el cambio en la higiene durante sus tres días de duración.

Los individuos están clasificados en cuatro grupos según sus estilos musicales, estos son indie, metal, pop y sin estilo predominante. Queremos estudiar los cambios de higiene para cada uno de ellos a lo largo del festival.

```
dffest <- read.table( "40A-festival.csv", sep = ";", head = TRUE )
head( dffest )
```

```
ticknumber     musica dia1 dia2 dia3 cambio
1       2111      metal 2.65 1.35 1.61  -1.04
2       2229        pop 0.97 1.41 0.29  -0.68
3       2338 sin estilo 0.84   NA   NA     NA
4       2384        pop 3.03   NA   NA     NA
5       2401 sin estilo 0.88 0.08   NA     NA
6       2405        pop 0.85   NA   NA     NA
```

Observamos que al contener texto, R ha convertido la variable `musica` en un factor de 4 niveles ordenados de 1 a 4.

```
str( dffest )
```

```
'data.frame':   810 obs. of  6 variables:
$ ticknumber: int  2111 2229 2338 2384 2401 2405 2467 2478 2490 2504 ...
$ musica    : chr  "metal" "pop" "sin estilo" "pop" ...
$ dia1      : num  2.65 0.97 0.84 3.03 0.88 0.85 1.56 3.02 2.29 1.11 ...
$ dia2      : num  1.35 1.41 NA NA 0.08 NA NA NA NA 0.44 ...
$ dia3      : num  1.61 0.29 NA NA NA NA NA NA NA 0.55 ...
$ cambio    : num  -1.04 -0.68 NA NA NA NA NA NA NA -0.56 ...
```

```
levels( dffest$musica )
```

```
NULL
```

```
dffest$musica <- as.factor(dffest$musica )
levels( dffest$musica )
```

```
[1] "indie"      "metal"      "pop"        "sin estilo"
```

Empezamos con un gráfico para hacernos una idea de cómo afecta las preferencias musicales de los asistentes a sus cambios en la higiene durante el desarrollo del festival, ver figura 6.36.

```
plot( cambio ~ musica, data = dffest )
```

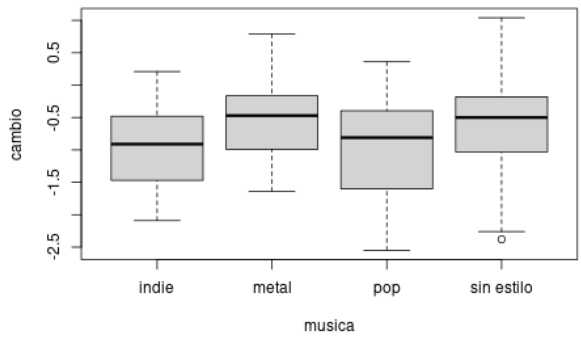


Figura 6.36: *Boxplots exploratorios.*

Creamos las variables *dummy*. Lo podemos hacer automáticamente mediante el comando `contr.treatment(numero de grupos, base = número del grupo referencia)`, donde en nuestro caso tenemos cuatro grupos y el grupo de

referencia es el último, *sin estilo*.

```
contrasts( dffest$musica ) <- contr.treatment( 4, base = 4 )

### attr(,"contrasts")
###            1 2 3
### indie      1 0 0
### metal      0 1 0
### pop        0 0 1
### sin estilo 0 0 0
### Levels: indie metal pop sin estilo
```

Es preferible hacer este proceso de forma manual ya que tenemos control sobre la codificación y podemos poner nombres significativos a las variables. Tomamos la categoría *sin estilo* como grupo referencia

```
Indie_dum <- c( 1, 0, 0, 0 )
Metal_dum <- c( 0, 1, 0, 0 )
Pop_dum <- c( 0, 0, 1, 0 )
contrasts(dffest$musica) <- cbind( Indie_dum, Metal_dum, Pop_dum )
```

Si inspeccionamos la variable `dffest$musica` vemos que se obtiene el mismo resultado

```
### attr(,"contrasts")
###            indie_dum metal_dum pop_dum
### indie              1         0       0
### metal              0         1       0
### pop                0         0       1
### sin estilo         0         0       0
### Levels: indie metal pop sin estilo
```

Una vez creadas las variables *dummy* se ejecuta el modelo de regresión de la misma manera que para cualquier otro tipo de regresión

```
modelFesti <- lm( cambio ~ musica, data = dffest )
summary( modelFesti )

Call:
lm(formula = cambio ~ musica, data = dffest)
```

```
Residuals:
Min      1Q  Median      3Q     Max
-1.8257 -0.5049  0.0559  0.4243  1.5943

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)      -0.5543     0.0904   -6.13  1.2e-08 ***
musicaIndie_dum  -0.4100     0.2049   -2.00    0.048 *
musicaMetal_dum   0.0284     0.1603    0.18    0.860
musicaPop_dum    -0.4115     0.1670   -2.46    0.015 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.688 on 119 degrees of freedom
(687 observations deleted due to missingness)
Multiple R-squared:  0.0762,    Adjusted R-squared:  0.0529
F-statistic: 3.27 on 3 and 119 DF,  p-value: 0.0237
```

El coeficiente R^2 nos dice que con las variables *dummy* podemos explicar el 7.6% de la variabilidad en el cambio de higiene del individuo según sea su afiliación musical, y el estadístico F que esta varianza es significativa. Pasamos a examinar los *coeficientes* del modelo.

Recordemos que los valores *beta* muestran el cambio en la variable respuesta provocado por el cambio de una unidad en el predictor. En este caso el cambio del predictor es de 0 a 1 y como el grupo referencia es siempre cero, los valores beta realmente nos proporcionan la diferencia relativa entre cada grupo y el grupo elegido como referencia. Así, el valor de la variable `Indide_dum` indica la diferencia en el cambio de higiene de una persona sin afiliación musical comparada con una persona a la que le gusta la música *indie*.

El estadístico t contrasta si estas diferencias son cero. Si es significante quiere decir que el grupo codificado con 1 es significativamente diferente del grupo de referencia. Para esta primera variable el *t-test* es significativo y el valor *beta* negativo por lo que podemos decir que la higiene empeora de una persona sin afiliación musical a una indie.

Para la segunda variable, `metal_dum`, obtenemos un valor positivo para beta, sin embargo no es significativo por lo que podríamos decir que el cambio en la higiene a lo largo de los tres días del festival es el mismo para una persona sin afiliación musical que para una que le gusta el metal.

7 Regresión logística

María Elvira Ferre Jaén

7.1. Introducción

Una muy buena introducción al modelo de regresión logística se puede encontrar en el texto de Andy Field [24], en el que nos basamos en este capitulo en gran medida.

La regresión logística es el conjunto de modelos estadísticos utilizados cuando se desea conocer la relación entre:

- Una *variable dependiente cualitativa*, dicotómica (regresión logística binaria o binomial) o con más de dos categorías (regresión logística multinomial).
- Una o más variables explicativas independientes, llamadas *covariables*, ya sean cualitativas o cuantitativas.

Las covariables cualitativas deben ser dicotómicas, tomando valor 0 para su ausencia y 1 para su presencia. Si la covariable tuviera más de dos categorías debemos realizar una transformación de la misma en varias covariables cualita-

Cómo citar este capítulo: Ferre Jaén M. A. (2022). Regresión logística. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (309-348). Editum. Ediciones de la Universidad de Murcia.

tivas dicotómicas ficticias (*variables dummy*). Al hacer esta transformación cada categoría de la variable entraría en el modelo de forma individual.

Los modelos de regresión logística tienen *tres finalidades:*

- Cuantificar la importancia de la relación existente entre cada una de las covariables y la variable dependiente.
- Clarificar la existencia de interacción y confusión entre covariables respecto a la variable dependiente (es decir, los odds ratio para cada covariable).
- Clasificar individuos dentro de las categorías (presente/ausente) de la variable dependiente.

Por tanto, el objetivo de la *regresión logística* no es, como en regresión lineal, predecir el valor de la variable Y a partir de una o varias variables predictoras (X_s), sino que queremos predecir la *probabilidad* de que ocurra Y conocidos los valores de las variables X_s. La ecuación general es de la forma:

$$P(Y) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n)}},$$

donde $P(Y)$ es la probabilidad de que ocurra Y, e es la función exponencial y el resto de coeficientes son análogos a los de la regresión lineal.

En su forma más sencilla, cuando tenemos sólo una variable predictora X_1, la ecuación de la regresión logística viene dada por:

$$P(Y) = \frac{1}{1 + e^{-(b_0 + b_1 X_1)}}.$$

Los valores posibles de estas ecuaciones varían entre 0 y 1. Un valor cercano a 0 significa que es muy improbable que Y haya ocurrido, y un valor cercano a 1 significa que es muy probable que tuviese lugar.

Como en la regresión lineal cada variable predictora de la ecuación logística tiene su propio coeficiente. Los valores de los parámetros se estiman utilizando el *método de máxima verosimilitud* que selecciona los coeficientes que hacen más probable que los valores observados ocurran.

7.1.1. Interpretación del modelo de regresión logística

El propósito del análisis es:

- Predecir la probabilidad de que un evento ocurra para una persona dada (notación $P(Y_i)$). Para dicha i-ésima persona, Y será 0 (la respuesta no ocurre) o 1 (la respuesta ocurre), y el valor predicho, $P(Y)$, tendrá un valor 0 (no hay probabilidad de que el resultado ocurra) o 1 (el resultado seguro que ocurre).
- Determinar qué variables pesan más para aumentar o disminuir la probabilidad de que a alguien le suceda el evento en cuestión.

Para realizar el análisis nos basamos en las características que presentan los sujetos a los que, efectivamente, les ocurren o no estos sucesos.

Por ejemplo imaginemos que queremos predecir la probabilidad de *"estar desempleado" = 1* o *"no estarlo" = 0*, la regresión logística tomará en cuenta los valores que asumen en una serie de variables (edad, sexo, nivel educativo, posición en el hogar, origen migratorio, etc.) para los sujetos que están efectivamente desocupados (= 1) y los que no lo están (= 0).

En base a ello, el modelo predecirá para cada uno de los sujetos independientemente de su estado real y actual – una determinada probabilidad de estar desocupado (es decir, de tener valor 1 en la variable dependiente). Es decir, si alguien es un joven, no amo de casa, con baja educación, de sexo masculino y origen emigrante (aunque esté ocupado) el modelo le predecirá una alta probabilidad de estar desocupado (puesto que la tasa de desempleo de el grupo así definido es alta), generando una variable con esas probabilidades estimadas. Y procederá a clasificarlo como desocupado en una nueva variable, que será el resultado de la predicción.

Además, analizará cuál es el peso de cada una de las variables independientes en el aumento o la disminución de esa probabilidad. Por ejemplo, cuando aumenta la educación disminuirá en algo la probabilidad de estar desocupado. En cambio, cuando el sexo pase de 0 ="mujer" a 1 = "varón", aumentará en algo la probabilidad de desempleo porque la tasa de desempleo de los jóvenes de sexo masculino es mayor que la de las mujeres jóvenes.

Obviamente cuanto más coincidan los estados pronosticados con los estados reales de los sujetos, mejor será el ajuste del modelo.

7.2. Regresión logística binaria

Los modelos de regresión logística binaria resultan ser los de mayor interés ya que la mayor parte de las circunstancias analizadas en las ciencias experimentales responden a este modelo (presencia o no de una cualidad, éxito o fracaso, etc.).

Como se ha visto, la variable dependiente será una variable dicotómica que se codificará como 0 ó 1 ("ausencia" y "presencia", respectivamente).

La ecuación de partida en los modelos de regresión logística es:

$$P(Y = 1 \mid X) = \frac{exp(b_0 + \sum_{i=1}^{n} b_i x_i)}{1 + exp(b_0 + \sum_{i=1}^{n} b_i x_i)},$$

donde:

- $P(Y = 1 \mid X)$ es la probabilidad de que Y tome el valor 1 (presencia de la característica estudiada).
- X es un conjunto de n covariables $x_1, \cdots, x_n$ que forman parte del modelo.
- b_0 es la constante del modelo o término independiente.
- b_i los coeficientes de las covariables.

En la figura 7.1 vemos un ejemplo de esta distribución. Está representada la probabilidad de padecer una enfermedad coronaria en función de la edad. Como puede verse, la relación entre la variable dependiente (cualitativa dicotómica), y la covariable (edad, continua en este caso), no queda definida por una recta (lo que correspondería un modelo lineal), sino que describe una forma sigmoidea (distribución logística).

Si se divide la expresión por su complementario, es decir, si se construye su odds (la probabilidad de estar enfermo entre la probabilidad de estar sano), se obtiene una expresión de manejo matemático más fácil:

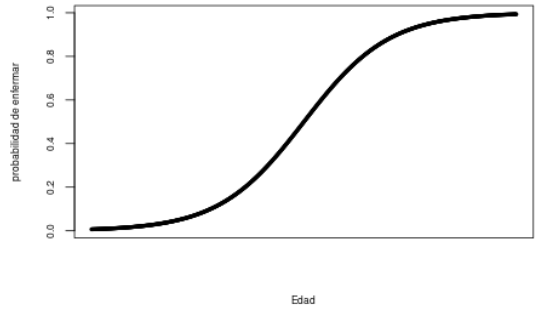


Figura 7.1: *Distribución modelos de regresión logístic.*

$$\frac{P(Y=1 \mid X)}{1-P(Y=1 \mid X)} = \exp\left(b_0 + \sum_{i=1}^{n} b_i x_i\right)$$

Pero esta expresión aún es difícil de interpretar. Su representación gráfica es la que se ve en la figura 7.2.

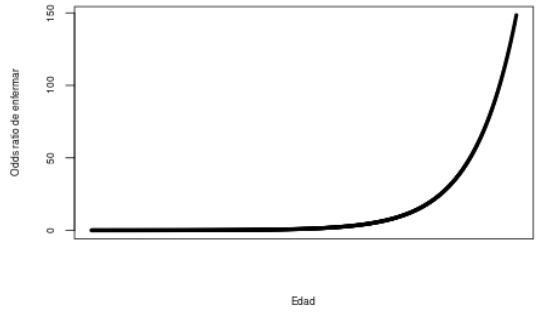


Figura 7.2: *Representación de los odds.*

Si ahora realizamos su transformación con el logaritmo natural, se obtiene una ecuación lineal:

$$\log\left(\frac{P(Y=1 \mid X)}{1-P(Y=1 \mid X)}\right) = b_0 + \sum_{i=1}^{n} b_i x_i$$

Se trata del llamado *logit*, esto es, el logaritmo natural del *odds* de la variable dependiente (p. ej, el logaritmo de la razón de proporciones de enfermar, de fallecer, de éxito, etc.). El término a la derecha de la igualdad es la expresión de una recta, idéntica a la del modelo general de regresión lineal.

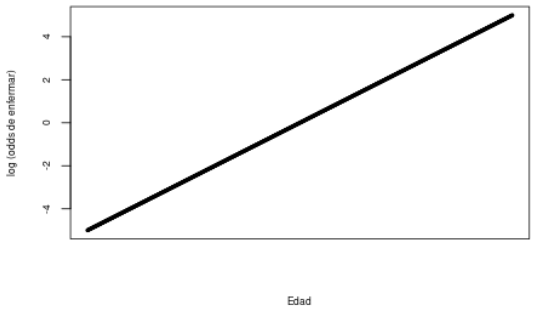


Figura 7.3: *Gráfica logit.*

No obstante la regresión logística presenta una *diferencia fundamental* respecto al modelo de regresión lineal.

En el modelo de regresión lineal se asume que los errores estándar de cada coeficiente siguen una distribución normal de media 0 y varianza constante (homocedasticidad), sin embargo, en el caso del modelo de regresión logística no pueden realizarse estas suposiciones pues la variable dependiente no es continua pues se trata de una variable dicotómica, sólo puede tomar dos valores, 0 ó 1, pero ningún valor intermedio.

7.2.1. Ajuste del modelo

El modelo debe ser aquél más reducido que explique los datos (*principio de parsimonia*), y que además sea técnicamente congruente e interpretable. Hay que tener en cuenta que un mayor número de variables en el modelo implicará mayores

errores estándar. Deben incluirse todas aquellas variables que se consideren técnicamente importantes para el modelo, no debería dejarse de incluir toda variable que en un análisis univariado previo demostrara una relación "suficiente" con la variable dependiente.

Si estamos trabajando con variables ficticias y se decide incluir (o excluir) una de estas variables, todas sus correspondientes variables ficticias deben ser incluidas (o excluidas) en bloque.

Otro aspecto de interés es la significación que pudiera tener cada variable ficticia. No siempre todas las variables ficticias de una covariable son significativas. En estos casos es recomendable contrastar el modelo completo frente al modelo sin la covariable mediante la prueba de razón de verosimilitud (es decir, se sacarían del modelo en bloque todas las variables ficticias de la covariable de interés).

Una vez se dispone de un modelo inicial debe procederse a su reducción hasta obtener el modelo más reducido que siga explicando los datos. Para ello utilizaremos los métodos de selección paso a paso.

Cuando tengamos un modelo preliminar, se podrían incluir factores de interacción, es decir, estudiar cómo la asociación de dos o más covariables puede influir en la variable dependiente. Se recomienda la inclusión en el modelo inicial de todas las covariables necesarias más las interacciones de las mismas, o por lo menos, las interacciones de primer orden.

Si decidimos incluir un factor de interacción, hay que tener en cuenta que siempre deberán estar incluidas en el modelo las covariables que componen la interacción. Por ejemplo, si la interacción es *"hipertensión-diabetes"*, hay que meter también en el modelo las covariables *hipertensión* y *diabetes*. El modelo quedaría de la siguiente forma:

$$b_0 + b_1\ hipertension + b_2\ diabetes + b_3\ hipertension : diabetes + \cdots$$

Por otra parte, y en relación con la inclusión de interacciones, hay que tener en cuenta que la inclusión de las mismas puede generar multicolinealidad, tanto más probable cuanto mayor sea el número de interacciones.

Se puede encontrar más información en [56].

7.2.1. Ajuste del modelo en R

Para el desarrollo completo de este ejemplo nos hemos servido esencialmente del libro [24].

Comenzamos leyendo el conjunto de datos:

```
enfermo <- read.table( "eel.csv", sep = ";", head = TRUE )
head( enfermo )
```

```
  Curado Tratamiento Duracion
1     No          No        7
2     No          No        7
3     No          No        6
4     Si          No        8
5     Si          Si        7
6     Si          No        6
```

```
str( enfermo )
```

```
'data.frame':   113 obs. of  3 variables:
 $ Curado     : chr  "No" "No" "No" "Si" ...
 $ Tratamiento: chr  "No" "No" "No" "No" ...
 $ Duracion   : int  7 7 6 8 7 6 7 7 8 7 ...
```

- Curado: sí o no (dependiente)
- Tratamiento: sí o no (predictora)
- Duración: días que el paciente estaba enfermo antes de comenzar el tratamiento (predictora).

Curando y Tratamiento son variables categóricas, por lo que las tenemos que convertir en factores:

```
enfermo$Curado <- as.factor(enfermo$Curado)
enfermo$Tratamiento <- as.factor(enfermo$Tratamiento)
```

En regresión logística para crear el modelo usamos el comando `glm()`, su forma general es:

`glm(resultado ~ predictor(es), data = dataFrame, family = nombre de la distribución, na.action = una acción ),`

donde `family` es el nombre de la distribución (Gausiana, binomial, poisson, gamma). En el caso de la regresión logística usamos la opción `family=binomial()`.

```
logmodel <- glm( Curado ~ Tratamiento, data = enfermo, family = binomial( ))
logmodel
```

```
Call:  glm(formula = Curado ~ Tratamiento, family = binomial(), data = enfermo)

Coefficients:
  (Intercept)  TratamientoSi
       -0.288          1.229

Degrees of Freedom: 112 Total (i.e. Null);  111 Residual
Null Deviance:      154
Residual Deviance: 144  AIC: 148
```

Ya hemos creado el modelo de regresión logística. El siguiente paso es estudiar la calidad del modelo, su bondad.

7.2.2. Bondad de ajuste

La bibliografía utilizada en este apartado ha sido fundamentalmente [24].

7.2.2. Criterio de máxima verosimilitud

Recordemos que en regresión logística lo que hacemos es predecir la probabilidad ($P(Y)$) de que un evento (Y) ocurra para una persona (i) dada, basado en las observaciones de si el evento ocurre o no para esa persona (denotamos esto como Y_i, el resultado real para la i-ésima persona). Así, para esa i-ésima persona el suceso Y toma los valores 0 (no ocurre) o 1 (ocurre), y el valor predicho, $P(Y)$, variará entre 0 (no hay probabilidad de que el evento ocurra) y 1 (el evento ocurre con seguridad).

Por tanto, al igual que en regresión múltiple, podemos usar estos valores observados y predichos para evaluar el ajuste del modelo. La medida que usamos es *log-likelihood (logaritmo de la razón de verosimilitud)*:

$$log - likelihood = \sum_{i=1}^{N} [Y_i \log P(Y_i) + (1 - Y_i) \log (1 - P(Y_i))].$$

Se basa por tanto en la suma de las probabilidades asociadas con los resultados estimados y los valores reales.

El estadístico log-likelihood es análogo a la suma de cuadrados residual en la regresión múltiple en el sentido de que es un indicador cuánta información sin explicar queda en la variable respuesta tras haber ajustado el modelo. Grandes valores del *log-likelihood* indican un pobre ajuste del modelo, cuanto mayor sea este valor, más variabilidad sin explicar queda en el modelo.

7.2.2. Devianza

Otro indicador importante para estudiar el ajuste del modelo logístico es la *devianza* que se define como el doble logaritmo del estadístico de verosimilitud, es decir, $devianza = -2\times$ *log-likelihood* y se representa como -2LL.

La devianza tiene una distribución χ^2 y compara los valores de la predicción con los valores observados en dos momentos:

- El modelo sin variables independientes, sólo con la constante (*modelo referencia*).
- El modelo con las variables predictoras introducidas.

Simplemente tomamos la devianza del nuevo modelo y le restamos la devianza del modelo referencia. Esta diferencia se lo conoce como *ratio-likelihood* y tiene una distribución χ^2 con *k-1* grados de libertad, el número de parámetros del nuevo modelo, menos el número de parámetros del modelo referencia que es siempre 1.

$$\chi^2 = 2LL(nuevo) - 2LL(referencial)$$

$$gl = k_{nuevo} - 1$$

Por lo tanto, el valor de la devianza debiera disminuir sensiblemente entre ambas instancias e, idealmente, tender a cero cuando el modelo predice bien.

7.2.2. r y R^2

Cuando hablamos de regresión lineal el coeficiente de correlación, r y el de determinación, R^2, son medidas útiles para saber cómo de bien se ajusta el modelo a los datos. En regresión logística podemos calcular una medida análoga, conocida como *R-statistic*.

Se trata de la correlación parcial entre la variable resultado y cada una de las predictoras, y puede variar entre -1 y 1. Un valor positivo significa que al crecer la variable predictora, lo hace la probabilidad de que el evento ocurra. Un valor negativo implica que si la variable predictora decrece, la probabilidad de que el resultado ocurra disminuye. Si una variable tiene un valor pequeño de R entonces esta contribuye al modelo sólo una pequeña cantidad.

Una medida análoga al R^2 en la regresión logística puede ser:

$$R_L^2 = \frac{2LL(nuevo) - 2LL(referencia)}{2LL(referencia)}$$

es la reducción proporcional en valor absoluto de *log-likelihood* y mide cuánto del error del ajuste disminuye al incluir las variables predictoras. Proporciona una medición de la significación real del modelo. Esta puede variar entre 0 (indicando que los predictores son inútiles prediciendo la variable respuesta) y 1 (indicando que el modelo predice perfectamente la respuesta).

7.2.2. Estadístico de Wadl

Como en la regresión lineal, queremos saber no sólo cómo de bueno es el modelo ajustándose a los datos, sino también la contribución individual de cada uno de las variables predictoras. Esta información la proporciona el *estadístico de Wald* (`z-statistic`) que sigue una distribución normal.

Al igual que el *estadístico t* en regresión lineal, el *estadístico z* nos dice si los coeficientes b_s para cada predictora son significativamente diferentes de cero. Si es distinto de cero asumimos que la variable predictora está haciendo una contribución significativa al modelo para predecir la respuesta (Y).

Su valor para un coeficiente concreto viene dado por el cociente entre el valor del coeficiente y su correspondiente error estándar. La obtención de

significación indica que dicho coeficiente es diferente de 0 y merece la pena su conservación en el modelo.

Sin embargo, en modelos con errores estándar grandes, el estadístico de Wald puede proporcional falsos resultados. Si el coeficiente (b) del modelo es grande, el error estándar tiende a inflarse y esto incrementa la probabilidad de rechazar un predictor cuando en realidad está haciendo una contribución al modelo.

$$z = \frac{b}{SE_b}$$

7.2.2. Odds, odds-ratio y coeficientes

El *odds* de un suceso es el cociente sus probabilidades de ocurrencia entre sus probabilidades de no ocurrencia, bajo unas determinadas condiciones C.

$$odds_c(evento) = \frac{P(evento)}{1 - P(evento)}$$

La medida más crucial para la interpretación del modelo logístico es el valor del *odds ratio*, que es la exponencial del valor B (e.d. $\exp(B)$) del modelo de regresión, y se define como el indicador del cambio en los odds resultante del cambio de una unidad en el predictor.

Cuando la variable predictora es categórica el *odds ratio* de ocurrencia de un suceso es fácil de explicar, imaginemos que queremos predecir la probabilidad de que un sujeto tenga un accidente de tráfico al haber consumido drogas.

Los *odds* de tener un accidente es la probabilidad de tener un accidente dividido por la probabilidad de no tenerlo:

$$odds_{Accidente} = \frac{P(accidente)}{P(no\ accidente)}.$$

Para calcular el cambio en el *odds* resultante del cambio de una unidad en la variable predictora, primero debemos calcular los *odds* de tener un accidente habiendo tomado drogas y después los *odds* de sufrir un accidente sin haber ingerido drogas.

Para obtener los *odds* de haber consumido drogas utilizamos las fórmulas de probabilidades:

$$P(accidente) = \frac{1}{1 + e^{-(b_0 + b_1 X)}}$$

$$P(no\ accidente) = 1 - P(accidente)$$

En la primera ecuación tenemos tres incógnitas, los coeficientes b_0 y b_1, y el valor de la propia predictora. La predictora X la codificamos como 0 = consumo, 1 = no consumo, y con ella se estimarán los coeficientes.

A continuación calculamos lo mismo pero después de que el predictor hay aumentado en una unidad, es decir, calculamos el *odds* de tener un accidente cuando no se han consumido drogas. Así que el valor de X es ahora 1 (en vez de 0).

Tenemos por tanto los *odds* antes y después de aumentar la variable predictora una unidad. Con todo esto es fácil ahora calcular el *cambio proporcional en los odds* simplemente dividiendo los *odds* después del cambio entre los *odds* antes del cambio, es lo que se conoce como *odds-ratio*:

$$oddsRatio = \frac{odds\ tras\ cambio\ en\ una\ unidad\ de\ X}{odds\ originales}$$

En el caso de que tuviéramos un modelo logístico con más de una variable, se llama *odds ratio* del predictor X_j al cociente del *odds* de ocurrencia al aumentar X_j en una unidad con respecto a no aumentarla, cuando se mantienen constantes el resto de las predictoras $X_1, \cdots, X_k$,

$$oddsRatio_(X_j) = \frac{odds\ tras\ cambio\ en\ una\ unidad\ de\ X_j}{odds\ originales}.$$

La interpretación del *odds-ratio* es que valores mayores que 1 indican que si el predictor aumenta los *odds* de la variable dependiente crecen. Inversamente, un valor menor que 1 indica que tal como el predictor aumente el *odds* del resultado decrece.

El *odds ratio* es una buena medida del tamaño del efecto. Más información en [24].

7.2.2. Bondad de ajuste en R

La referencia bibliográfica básica para el desarrollo de este apartado ha sido [24].

Devianza y χ^2

```
summary( logmodel )
```

```
Call:
glm(formula = Curado ~ Tratamiento, family = binomial(), data = enfermo)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.594  -1.058   0.812   0.812   1.302

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.288      0.270   -1.07   0.2867
TratamientoSi   1.229      0.400    3.07   0.0021 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 154.08  on 112  degrees of freedom
Residual deviance: 144.16  on 111  degrees of freedom
AIC: 148.2

Number of Fisher Scoring iterations: 4
```

- La *bondad de ajuste* global del modelo se evalúa mediante *la devianza* (-2 veces el logaritmo de la verosimilitud) y tenemos que valores grandes indican que los modelos estadísticos son pobres.

En el resumen del modelo R calcula la devianza nula (solo con la constante) y la devianza residual (todo el modelo). Para que el modelo sea bueno la devianza residual debe ser menor que la devianza nula ya que valores más bajo de -2LL indican que el modelo predice la variable respuesta con mayor precisión.

En este caso la devianza del modelo nulo es $-2LL = 154.08$, pero cuando añadimos `Tratamiento` este valor se reduce a 144.16, lo que nos dice que con esta variable el modelo mejora prediciendo si alguien está curado.

- Para saber *la eficacia* del modelo prediciendo la variable respuesta utilizamos el *estadístico chi-cuadrado*, que mide la diferencia entre el modelo en su estado actual y el modelo cuando sólo se incluyó la constante.

En este caso el valor del estadístico chi-cuadrado es igual al -2LL del modelo con Tratamiento menos el valor de -2LL cuando sólo tenemos la constante (154.08 a 144.16 = 9.92). Este valor lo podemos calcular automáticamente con R mediante la siguiente serie de comandos:

```
dev <- logmodel$deviance
nullDev <- logmodel$null.deviance
modelChi <- nullDev - dev
modelChi
```

```
[1] 9.93
```

Para calcular la probabilidad asociada al estadístico chi-cuadrado utilizamos la función `pchisq(Chi, gl)`, cuyos argumentos son el estadístico chi-cuadrado y sus grados de libertad. La probabilidad que queremos es 1 menos el valor de la función `pchisq( )`.

```
chigl <- logmodel$df.null - logmodel$df.residual
chisq.prob <- 1 - pchisq( modelChi, chigl )
chisq.prob
```

```
[1] 0.00163
```

como la probabilidad es menor que 0.05, podemos rechazar la hipótesis nula de que el modelo es mejor prediciendo la variable resultado que si elegimos por azar. Por tanto, podemos decir que, en general, el modelo tiene una aportación significativa en la perdición sobre la cura de un paciente que ha seguido un

determinado tratamiento.

Coeficientes y z-statistic. Los valores b, los coeficientes, tienen la misma función que en la regresión lineal: son los valores que tenemos que sustituir en la ecuación del modelo para establecer la probabilidad de que un caso esté comprendido en una determinada categoría.

Este coeficiente en la regresión logística se puede interpretar como el cambio en el *logit* de la variable de resultado asociado al cambio de una unidad en la variable predictora, donde el logit es simplemente el logaritmo natural de las probabilidades de Y que ocurra.

```
summary(logmodel)$coefficients
```

Un estadístico crucial es el *estadístico de Wald (z-statistic)* que tiene una distribución normal y nos dice si el coeficiente b para ese predictor es significativamente diferente de cero para poder así suponer que la variable predictora está haciendo una contribución significativa a la predicción del resultado (Y).

Así, en este caso podemos afirmar que incluir la variable `Tratamiento` produjo una mejoría significativa en el ajuste del modelo, $\chi^2(1) = 9.93$, $p = 0.002$.

R^2. Podemos calcular un valor similar a la R^2 en los modelo lineales mediante la medida de *Hosmer & Lemeshow* (R^2_L).

```
R2.hl <- modelChi/logmodel$null.deviance
R2.hl
```

```
[1] 0.0644
```

Odds ratio. Para calcular el cambio en los *odds* resultantes del cambio de una unidad en la variable predictora, primero debemos calcular los *odds* para un paciente curado que no haya sido tratado y después los de un paciente que sí hay recibido tratamiento.

Para calcularlos utilizamos la ecuación:

$$P(Y) = \frac{1}{1 + e^{-(b_0 + b_1 X_1)}}.$$

Los coeficientes b_0 y b_1 los obtenemos en la tabla resumen `summary(logmodel)`, vamos a calcular con ellos los *odds ratio*:

```
# P(Y) = -0.288+1.229 x

# Odds paciente sin intervención X=0
b0 <- -0.288 + 1.229 * 0
cura0 <- 1 / ( 1 + exp( -b0 ) )
noCura0 <- 1 - cura0
odds0 <- cura0 / noCura0
odds0
```

```
[1] 0.75
```

```
# Odds paciente con intervención X=1
b1 <- -0.288 + 1.229 * 1
cura1 <- 1 / ( 1 + exp( -b1 ) )
noCura1 <- 1 - cura1
odds1 <- cura1 / noCura1
odds1
```

```
[1] 2.56
```

```
# Odds-ratio
oddsRatio <- odds1 / odds0
oddsRatio
```

```
[1] 3.42
```

Podemos también calcular directamente los *odds ratio* haciendo la exponencial de los b-valores:

```
exp( logmodel$coefficients )
```

```
  (Intercept) TratamientoSi
         0.75          3.42
```

Vemos que se obtienen los mismos resultados, y se interpretan en relación al cambio en los odds (probabilidades).

Si el valor es mayor que 1, entonces indica que a medida que aumenta el predictor, las probabilidades de los resultados aumentan. A la inversa, un valor menor que 1 indica que a medida que aumenta el predictor, las probabilidades de los resultados disminuyen. En este ejemplo, podemos decir que las probabilidades

de un paciente se cure tras haber sido tratado son 3.42 veces superiores a las de un paciente que no tratado.

Podemos además calcular los *intervalos de confianza* de los coeficientes:

```
exp( confint( logmodel ) )
```

```
Waiting for profiling to be done...
```

```
                2.5 % 97.5 %
(Intercept)    0.437   1.27
TratamientoSi 1.582   7.63
```

Lo importante de este intervalo de confianza es que no cruza 1 (los valores en cada extremo del intervalo son mayores que 1).

Esto es fundamental porque valores mayores que 1 significan que a medida que la variable predictora aumenta, también lo hacen las probabilidades de (en este caso) curarse. Los valores inferiores a 1 significan lo contrario: si la variable predictora aumenta, las probabilidades de curarse disminuye. El hecho de que tanto los límites inferiores y superiores de nuestro intervalo de confianza estén por encima de 1 nos da la confianza de que la dirección de la relación que hemos observado es cierta en la población.

7.2.3. Valores ajustados y residuos

Para el desarrollo de este apartado hemos empleado esencialmente el libro de texto [24].

Los valores ajustados en la regresión logística son un poco diferentes a los de la regresión lineal, pues predicen las probabilidades de Y dados los valores de cada predictor para cada participante.

También podemos predecir un grupo de pertenencia, basado en el resultado más probable para cada persona en el modelo.

Además, como en la regresión lineal, podemos examinar *los residuos* para asegurarnos de que el modelo se ajusta bien a los datos observados.

El principal propósito de examinar los residuos es:

1) Aislar los puntos en los que el modelo se ajusta mal.
2) Aislar los puntos que ejercen una influencia excesiva sobre el modelo.

Para buscar los casos conflictivos tenemos que fijarnos es lo siguiente:

- Mirar los residuos estandarizados y asegurarnos de que no más de 5% de los casos tiene un valor absoluto mayor de 2, y que no más de un 1% tiene valores absolutos más allá de 2.5. Cualquier caso con valor superior a 3 podría ser un `valor atípico`.
- Calcular la media del estadístico `leverage` (número de predictores más 1, dividido por el tamaño muestral) y buscar valores mayores que dos o tres veces esa media.
- Buscar valores absolutos de `DFBeta` mayores que 1.

Aunque aislemos numerosos valores atípicos o casos influyentes, no tenemos justificación para afirmar que al eliminarlos el modelo ajuste mejor. En lugar de ello, debemos inspeccionar estos casos atentamente y tratar de encontrar una buena razón de por qué son inusuales. Podría tratarse simplemente de un error al meter los datos, o podría ser que tuviera una buena razón para ser inusual, en tal caso sí podríamos excluir ese valor del modelo.

7.2.3. Obtención de residuos, valores predichos y estadísticos necesarios

```
enfermo$probabilidades.predichas <- fitted( logmodel )
enfermo$studentized.residuals <- rstudent( logmodel )
enfermo$dfbeta <- dfbeta( logmodel )
enfermo$dffit <- dffits( logmodel )
enfermo$leverage <- hatvalues( logmodel )
```

7.2.3. Probabilidades predichas

```
head( enfermo[ , c( "Curado", "Tratamiento",
                    "Duracion", "probabilidades.predichas" ) ] )
```

```
  Curado Tratamiento Duracion probabilidades.predichas
1     No          No        7                   0.429
```

```
2      No          No          7                0.429
3      No          No          6                0.429
4      Si          No          8                0.429
5      Si          Si          7                0.719
6      Si          No          6                0.429
```

Estos valores nos dicen que cuando un paciente no recibe tratamiento (Tratamiento = 0, no), hay una probabilidad de 0.429 de que se cure - básicamente, alrededor del 43% de las personas se recuperan sin tratamiento alguno. Sin embargo, si al paciente se le aplica el Tratamiento (Tratamiento = 1, sí), hay una probabilidad de 0.719 de que mejore - alrededor del 72% de las personas tratadas mejoran.

Si consideramos que una probabilidad de 0 indica que no hay oportunidad de mejorar, y una probabilidad de 1 que el paciente conseguirá definitivamente ponerse bien, los valores obtenidos proporcionan una fuerte evidencia de que tener un Tratamiento aumenta sus posibilidades de obtener una mejora.

Asumiendo que el modelo es riguroso y que Tratamiento tiene cierta importancia, entonces podríamos concluir que nuestro Tratamiento es el mejor predictor para ponerse mejor (curarse).

7.2.3. Valores influyentes y posibles atípicos

```
head( enfermo[ ,c( "leverage", "studentized.residuals", "dfbeta" ) ] )
```

```
  leverage studentized.residuals dfbeta.(Intercept)
1   0.0179                -1.064          -3.89e-02
2   0.0179                -1.064          -3.89e-02
3   0.0179                -1.064          -3.89e-02
4   0.0179                 1.311           4.78e-02
5   0.0175                 0.816           2.28e-18
6   0.0179                 1.311           4.78e-02
  dfbeta.TratamientoSi
1             3.89e-02
2             3.89e-02
3             3.89e-02
4            -4.78e-02
```

```
5          3.23e-02
6         -4.78e-02
```

Los estadísticos residuales básicos para este ejemplo son bastante buenos: todos los casos tienen un `DfBeta` menor que 1, y los *leverage* están muy cerca del valor esperado de leverage `(k+1)/n`=0.018.

En definitiva, esto significa que no hay casos influyentes que tengan efecto sobre el modelo. Además, todos los residuos estandarizados tienen valores menores de ± 2, y así que parece que no tenemos por qué preocuparnos.

7.2.4. Resumen: etapas de la regresión logística

- Recodificar las variables independientes categóricas u ordinales en variables ficticias y la variable dependiente en 0 y 1.
- Evaluar el ajuste general del modelo final observando la *devianza* y su estadístico χ^2 asociado. Si la significación de la χ^2 es menor que 0.05, entonces el modelo se ajusta significativamente a los datos.
- Para cada variable en el modelo, mirar el valor del *estadístico de Wald* y su significación (debe ser menor que 0.05).
- Analizar la fuerza, sentido y significación de los coeficientes, sus exponenciales y estadísticos de prueba.
- Usar el *odds ratio* para interpretar el modelo. Los podemos obtener mediante el comando `exp(model$coefficients)`. Si el valor es mayor que 1 al aumentar la variable predictora el *odds* de la respuesta aumenta. Inversamente, un valor menor que 1 indica que si la variable predictora crece, el *odds* de la respuesta decrece. ¡OJO! Para que está interpretación sea cierta el intervalo de confianza del *odds ratio* no debe cruzar el 1.
- Realizar un diagnóstico del modelo para comprobar que es un modelo adecuado para predecir la variable respuesta, que se ajusta bien a los datos.

Información más detallada en [24].

7.2.5. Comparación y selección del modelo

La referencia principal aplicada en este apartado es [24].

7.2.5. Criterio de información

Como en la regresión lineal podemos usar el criterio de información de Aike (AIC) y el criterio de información de Bayes (BIC) para juzgar el ajuste del modelo. Estos criterios proporcionan una medida del ajuste de un modelo que penaliza al modelo que contiene más variables predictoras, pudiendo así comparar dos modelos.

- $AIC = -2LL + 2k$
- $BIC = -2LL + 2k \times \log(n)$, donde n es el número de casos del modelo.

7.2.5. Métodos paso a paso

Si usamos el método `hacia delante` el ordenador empieza con un modelo que incluye solo la constante y entonces añade predictores individuales al modelo basándose en la variable que mejore el `AIC` o `BIC`. El ordenador continua mientras ninguno de los predictores restantes haya disminuido el criterio.

El método `hacia atrás` usa el mismo criterio pero empieza el modelo incluyendo todas las variables predictoras. `R` comprueba si alguno de esos predictores puede ser retirado del modelo sin incrementar el criterio de información. Si se puede, esa variable se saca del modelo, y se analizan de nuevo el resto de variables.

Mejor que estos métodos es el de `ambas direcciones` que empieza como el método `hacia delante`, con solo la constante, pero cada vez que añadimos una variable, el ordenador comprueba si merece la pena eliminarla.

7.2.5. Selección del mejor modelo en R

Se tomaron datos al azar de 75 jugadores de baloncesto antes de realizar un triple en una competición para evaluar su ansiedad antes del lanzamiento. Los datos están en el archivo `triples.csv` y contiene cuatro variables:

- Marcado: está codificada como 0 = triple fallado y 1 = triple anotado
- PSWQ: grado de preocupaciones del jugador en su vida cotidiana
- Trayectoria: porcentaje de triples anotados por un jugador en particular en su carrera

- Ansiedad: estado de ansiedad antes de lanzar el triple

```
triples <- read.table( "triples.csv", sep = ";", head = TRUE )
head( triples )
```

```
  PSWQ Ansiedad Trayectoria Marcado
1   18       21          56      Si
2   17       32          35      Si
3   16       34          35      Si
4   14       40          15      Si
5    5       24          47      Si
6    1       15          67      Si
```

```
str( triples )
```

```
'data.frame':   75 obs. of  4 variables:
 $ PSWQ       : int  18 17 16 14 5 1 4 12 11 15 ...
 $ Ansiedad   : int  21 32 34 40 24 15 10 19 29 14 ...
 $ Trayectoria: int  56 35 35 15 47 67 75 53 35 65 ...
 $ Marcado    : chr  "Si" "Si" "Si" "Si" ...
```

Marcado es una variable categórica, por lo que la tenemos que convertir a factor:

```
triples$Marcado <- as.factor(triples$Marcado)
```

Modelo completo

```
modeloTriple <- glm( Marcado ~ Trayectoria + PSWQ + Ansiedad,
                     data = triples, family = binomial( ) )
summary( modeloTriple )
```

```
Call:
glm(formula = Marcado ~ Trayectoria + PSWQ + Ansiedad, family = binomial(),
    data = triples)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3137  -0.3600   0.0833   0.5386   1.6138
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -11.493     11.802   -0.97   0.3302
Trayectoria    0.203      0.129    1.57   0.1172
PSWQ          -0.251      0.084   -2.99   0.0028 **
Ansiedad       0.276      0.253    1.09   0.2748
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 103.638  on 74  degrees of freedom
Residual deviance:  47.416  on 71  degrees of freedom
AIC: 55.42

Number of Fisher Scoring iterations: 6
```

Tan solo obtenemos que la variable PSWQ es significativa, por ello vamos a aplicar el método paso a paso para ver si podemos reducir el modelo. Aún así vemos que la devianza del modelo es mucho menor que la devianza solo con la constante, lo cual significa que estas variables ayudan a predecir mejor la respuesta.

Aplicamos el *método paso a paso* hacia atrás

```
step( modeloTriple, direction = "backward" )
```

```
Start:  AIC=55.4
Marcado ~ Trayectoria + PSWQ + Ansiedad

              Df Deviance  AIC
- Ansiedad     1     48.7 54.7
<none>               47.4 55.4
- Trayectoria  1     50.1 56.1
- PSWQ         1     61.1 67.1

Step:  AIC=54.7
Marcado ~ Trayectoria + PSWQ
```

```
             Df Deviance  AIC
<none>              48.7 54.7
- Trayectoria  1     60.5 64.5
- PSWQ         1     61.2 65.2

Call:  glm(formula = Marcado ~ Trayectoria + PSWQ, family = binomial(),
    data = triples)

Coefficients:
(Intercept)  Trayectoria         PSWQ
     1.2803       0.0648      -0.2301

Degrees of Freedom: 74 Total (i.e. Null);  72 Residual
Null Deviance:      104
Residual Deviance: 48.7     AIC: 54.7
```

Comenzamos con un AIC=55.42 y la función considera la eliminación de cada una de las variables para finalmente sacar del modelo la variable ansiedad por ser la que produce un AIC más bajo (54.66). En el siguiente paso se considera la eliminación de alguna de las dos restantes variables, pero R decide quedarse con ellas ya que su eliminación supone un aumento, en el mejor de los casos, del AIC a 64.516.

Nos quedamos por tanto con el modelo $Marcado = 0.0648 \times Trayectoria - 0.2301 \times PSWQ$ que nos dice que el éxito al marcar el triple se ve influenciado positivamente por la trayectoria profesional del jugador y negativamente por el nivel de preocupación que tenga el jugador en su vida diaria.

Bondad del modelo

```
modeloTriple2 <- glm( Marcado ~ Trayectoria + PSWQ, family = binomial(),
                      data = triples )
summary( modeloTriple2 )

Call:
glm(formula = Marcado ~ Trayectoria + PSWQ, family = binomial(),
    data = triples)
```

```
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.221  -0.331   0.104   0.505   1.607

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.2803     1.6702    0.77   0.4433
Trayectoria   0.0648     0.0221    2.93   0.0033 **
PSWQ         -0.2301     0.0798   -2.88   0.0039 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 103.638  on 74  degrees of freedom
Residual deviance:  48.662  on 72  degrees of freedom
AIC: 54.66

Number of Fisher Scoring iterations: 6
```

En este caso tanto la variable `Trayectoria` como la variable `PSWQ` tienen un estadístico z significativo, tienen una aportación significativa al modelo.

```
dev <- modeloTriple2$deviance
nullDev <- modeloTriple2$null.deviance
modelChi <- nullDev - dev
modelChi
```

```
[1] 55
```

Como el valor es positivo quiere decir que la devianza del modelo es menor que la devianza nula por lo que las variables del modelo mejoran la predicción de la variable respuesta. Sin embargo su valor es elevado lo que implique que en el modelo quedan residuos sin explicar. Idealmente queríamos que este valor fuese lo más cercano posible a cero. Calculamos ahora su significación.

```
chidf <- modeloTriple2$df.null - modeloTriple2$df.residual
chisq.prob <- 1 - pchisq( modelChi, chidf )
chisq.prob
```

```
[1] 1.15e-12
```

El valor es < 0.05 lo que quiere decir que el modelo con las variables predictoras es significativamente mejor que aquel solo con la constante. Nuestro modelo en su conjunto es significativamente bueno prediciendo la ansiedad de los juradores.

Podemos decir por tanto que la variable `Trayectoria` y la variable `PSWQ` provocan una mejoría significativa en el ajuste del modelo, $Chi^2(1) = 54.97669$, $p < 0.001$.

7.2.6. Supuestos del modelo

La regresión logística comparte algunos supuestos con la regresión usual:

- **Linealidad:** en regresión lineal asumimos que la variable respuesta tiene una relación lineal con las variables predictoras. En regresión logística la respuesta es categórica y por ello este supuesto se viola. Por ello por lo que utilizamos el *logit* de los datos. Así, el supuesto de linealidad en regresión logística es que existe una relación lineal entre cada variable predictora continua y el logaritmo de la variable respuesta.
- **Independencia de los errores:** los distintos casos de los datos no deben estar relacionados, por ejemplo, no podemos medir a la misma gente en diferentes puntos del tiempo.
- **Multicolinealidad:** aunque no es un supuesto como tal, la multicolinealidad es un problema como en la regresión lineal. Las variables predictoras no deben estar altamente correlacionadas.

Vamos a utilizar el ejemplo de *triples* para estudiar más detenidamente estos supuestos y ver cómo comprobarlos en R.

7.2.6. Linealidad

Para contrastar este supuesto necesitamos ejecutar la regresión logística pero incluyendo como predictores las iteraciones entre cada predictor y el logaritmo de sí mismo. Creamos la iteración de cada término con su logaritmo mediante el comando:

```
triples$logPSWQInt <- log(triples$PSWQ)*triples$PSWQ
triples$logAnsInt <- log(triples$Ansiedad)*triples$Ansiedad
triples$logTrayeInt <- log(triples$Trayectoria)*triples$Trayectoria
```

Para realizar el contraste rehacemos el análisis exactamente de la misma manera que anteriormente excepto porque metemos todas las variables de golpe en un solo bloque y añadimos las nuevas iteraciones:

```
linealidad <- glm( Marcado ~ PSWQ + Ansiedad + Trayectoria +
                    logPSWQInt + logAnsInt
                  + logTrayeInt, data = triples, family = binomial( ) )
summary( linealidad )
```

```
Call:
glm(formula = Marcado ~ PSWQ + Ansiedad + Trayectoria + logPSWQInt +
    logAnsInt + logTrayeInt, family = binomial(), data = triples)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.068  -0.385   0.112   0.546   1.827

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.8788    14.9241   -0.26     0.79
PSWQ         -0.4223     1.1027   -0.38     0.70
Ansiedad     -2.6449     2.7970   -0.95     0.34
Trayectoria   1.6660     1.4820    1.12     0.26
logPSWQInt    0.0439     0.2967    0.15     0.88
logAnsInt     0.6808     0.6528    1.04     0.30
logTrayeInt  -0.3186     0.3173   -1.00     0.32

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 97.283  on 70  degrees of freedom
Residual deviance: 45.909  on 64  degrees of freedom
  (4 observations deleted due to missingness)
AIC: 59.91
```

```
Number of Fisher Scoring iterations: 7
```

Sólo estamos interesados en los términos de las iteraciones. Cualquier iteración que sea significativa quiere decir que el efecto principal ha violado el supuesto de linealidad del logaritmo.

En este caso las tres iteraciones tienen valores de significación (columna `Pr(>|z|)`) mayores de 0.05, indicando que el supuesto de linealidad se cumple para `PSWQ`, `Ansiedad` y `Trayectoria`.

7.2.6. Multicolinealidad

Se dice que existe multicolinealidad cuando dos o más de las covariables del modelo mantienen una relación lineal. Cuando la colinealidad es perfecta, es decir, cuando una covariable puede determinarse según una ecuación lineal de una o más de las restantes covariables, es imposible estimar un único coeficiente de todas las covariables implicadas. En estos casos debe eliminarse la covariable que actúa como dependiente.

Normalmente lo que se hallará será una multicolinealidad moderada, es decir, una mínima correlación entre covariables. Si esta correlación fuera de mayor importancia, su efecto sería el incremento exagerado de los errores estándar, y en ocasiones, del valor estimado para los coeficientes de regresión, lo que hace las estimaciones poco creíbles. Podemos verificar este supuesto con los estadísticos de VIF y tolerancia, en las matrices de correlación, etc.

Vamos a estudiar este supuesto en nuestro modelo inicial que ya sabemos que una de las variables (Ansiedad) no aporta nada al modelo.

```
library( car )
vif( modeloTriple )
```

```
Trayectoria        PSWQ    Ansiedad
      668.5        33.3       649.6
```

```
1/vif( modeloTriple )
```

```
Trayectoria        PSWQ    Ansiedad
    0.00150     0.03001     0.00154
```

Estos valores indican que hay un problema de colinealidad: un VIF más de 10 se considera problemático El resultado de este análisis es bastante tajante: existe colinealidad entre la Ansiedad y Trayectoria, y esta dependencia convierte el modelo en sesgado.

Si identificamos multicolinealidad no hay mucho que podamos hacer, la solución no es fácil:

- Podemos intentar eliminar la variable menos necesaria implicada en la colinealidad, a riesgo de obtener un modelo menos válido. Sin embargo, un problema común es no saber qué variable debemos omitir. Cualquiera de las variables problemáticas puede ser omitida, no hay fundamentos estadísticos para suprimir una variable en vez de otra.
- Se recomienda que si eliminamos una variable predictora, ésta se reemplace por otra igualmente importante que no tenga una colinealidad tan fuerte.
- Se puede intentar cambiar la escala de medida de la variable en conflicto (es decir, transformarla). Sin embargo estas transformaciones hacen al modelo muy dependiente de los datos actuales, invalidando su capacidad predictiva.
- También se puede recurrir a aumentar la muestra para así aumentar la información en el modelo y ver si la multicolinealidad puede disminuir, aunque no siempre será posible.
- La última posibilidad, aunque más compleja cuando hay varios predictores, es hacer un análisis factorial y usar las puntuaciones del factor resultante como predictor.

7.2.6. Problemas en la regresión logística

La regresión logística tiene además problemas propios, no son supuestos si no cosas que pueden ir mal:

- Si creamos una tabla con todos los posibles valores de todas las variables entonces idealmente debemos tener datos en cada celda de la tabla. Si no los tenemos debemos estar atentos a posibles errores típicos grandes.
- Si podemos predecir la variable resultado perfectamente a partir de una (o una combinación) de variables predictoras entonces tenemos el problema de *separación completa* y este crea grandes errores también. El problema es que al estar la mayoría de los datos situados en los extremos, los datos

intermedios R no sabe cómo ajustarlos y opta por tomar la curva lo más vertical posible provocando esos errores.

7.2.7. Ejemplo completo de reg. logística binaria. Interpretación

Para llevar a cabo este ejemplo hemos utilizado como referencia [57].

Un investigador está interesado en saber qué efecto tienen variables como *PAU* (nota en Selectividad), *bach* (nota media bachiller) y el *prestigio* del instituto al que asistieron en la admisión de los estudiantes en una determinada universidad. La variable respuesta, admitir/no admitirlo, es una variable binaria.

```
univ <- read.table( "universidad.csv", sep = ";", head = TRUE )
```

```
str( univ )
```

```
'data.frame':   400 obs. of  4 variables:
 $ admitido : int  0 1 1 1 0 1 1 0 1 0 ...
 $ pau      : int  380 660 800 640 520 760 560 400 540 700 ...
 $ bach     : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ prestigio: Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
```

Este conjunto de datos tiene una variable respuesta binaria llamada *admitido* y tenemos tres variables predictoras: *pau*, *bach* y *prestigio*. Vamos a tratar a las variables *pau* y *bach* como continuas y la variable *prestigio* una categórica que toma valores de 1 a 4. Las instituciones con prestigio de 1 tienen el mayor reconocimiento, mientras que aquellos con un prestigio de 4 tienen la reputación más baja.

Descriptivos básicos

```
summary( univ )
```

```
    admitido          pau           bach      prestigio
 Min.   :0.000   Min.   :220   Min.   :2.26   1: 61
 1st Qu.:0.000   1st Qu.:520   1st Qu.:3.13   2:151
 Median :0.000   Median :580   Median :3.40   3:121
 Mean   :0.318   Mean   :588   Mean   :3.39   4: 67
```

```
 3rd Qu.:1.000   3rd Qu.:660   3rd Qu.:3.67
 Max.   :1.000   Max.   :800   Max.   :4.00
```

```
sapply( univ[1:3], sd )
```

```
admitido      pau     bach
   0.466  115.517    0.381
```

Solo cogemos las 3 primeras columnas porque *prestigio* es categórica y calcular su desviación estándar no tiene sentido.

7.2.7. Creación el modelo

Como la variable *prestigio* tiene 4 categorías y en el modelo de regresión logística solo podemos meter variables dicotómicas, tendríamos que crear tres variables ficticias a partir de ella para poder usarla en el modelo logístico. Pero como R es muy listo, si no lo hacemos y metemos la variable *prestigio* directamente en el modelo, él crea por sí solo las variables ficticias.

```
modelUni <- glm( admitido ~ pau + bach + prestigio, data = univ,
                 family = "binomial" )
summary( modelUni )
```

```
Call:
glm(formula = admitido ~ pau + bach + prestigio, family = "binomial",
    data = univ)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.627  -0.866  -0.639   1.149   2.079

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.98998    1.13995   -3.50  0.00047 ***
pau          0.00226    0.00109    2.07  0.03847 *
bach         0.80404    0.33182    2.42  0.01539 *
prestigio2  -0.67544    0.31649   -2.13  0.03283 *
prestigio3  -1.34020    0.34531   -3.88  0.00010 ***
```

```
prestigio4  -1.55146    0.41783   -3.71  0.00020 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.5

Number of Fisher Scoring iterations: 4
```

Como vemos en el resumen del modelo R ha creado las variables `prestigio2`, `prestigio3` y `prestigio4`, todas ellas variables ficticias que muestran la diferencia entre cada grupo y el grupo de referencia, que en este caso es `prestigio1`. Así el valor para, por ejemplo, `prestigio3` indica la diferencia del *logit* de admisión de una persona que fue a un instituto con prestigio uno a una que fue a una institución con prestigio tres.

Nos fijamos pues en los coeficientes, sus errores estándar, el estadístico z y los p-valores asociados. Tanto la variable *pau* como *bach* son estadísticamente significativas, al igual que los tres términos para prestigio.

Los coeficientes de la regresión logística nos dan el cambio en el logaritmo de las cuotas de admisión (*logit*) como resultado de un aumento de una unidad en cada una de las variables predictoras.

Para cada cambio en una unidad en *pau*, el logaritmo de las probabilidades de admisión (en comparación con la no admisión) aumentan en 0.002. Para un aumento de una unidad en el *bach*, el logaritmo de las probabilidades de admisión en la escuela aumentan un 0.804.

La variable *prestigio* tiene una interpretación ligeramente diferente. Por ejemplo, habiendo asistido a una universidad con un *prestigio* 2, si la frente a un institución con *prestigio* 1, cambia el logaritmo de las probabilidades de admisión a −0.675, al estudiar en un instituto de peor nivel (2) disminuye la probabilidad de entrar en esa universidad.

Todos los casos se enfrentan con la variable de referencia, en este caso

contra *prestigio* 1. Para saber el cambio entre, por ejemplo, haber ido a un instituto de prestigio 2 frente a uno de prestigio 3 tenemos que restar los valores de los coeficientes. Así, $logit_{2-3} = -0.675443 - (-1.340204) = 0.665$, es más probable entrar si venimos de un instituyo con prestigio dos que si venimos de uno con prestigio tres.

Debajo de la tabla de los coeficientes están los índices de ajuste, como residuos, las devianzas y el AIC. Más adelante veremos cómo utilizar estos valores para evaluar el ajuste del modelo.

7.2.7. Bondad de ajuste

```
dev <- modelUni$deviance
nullDev <- modelUni$null.deviance
modelChi <- nullDev - dev
modelChi
```

```
[1] 41.5
```

Como el valor es positivo quiere decir que la devianza del modelo es menor que la devianza nula por lo que las variables del modelo mejoran la predicción de la variable respuesta.

```
chidf <- modelUni$df.null - modelUni$df.residual
chisq.prob <- 1 - pchisq( modelChi, chidf )
chisq.prob
```

```
[1] 7.58e-08
```

Obtenemos un estadístico χ^2 con valor 41.45 y un p-valor < 0.01, lo que indica que el efecto general del modelo es estadísticamente significativa.

En particular podemos calcular la aportación particular de la variable *prestigio* en el modelo mediante la función `wald.test(b, Sigma, Terms)`, donde `b` son los coeficientes del modelo, `Sigma` es la matriz del covarianza del modelo y en `Terms` indicamos los términos del modelo que queremos estudiar.

```
library( aod )
```

```
Attaching package: 'aod'
```

```
The following objects are masked from 'package:faraway':

    rats, salmonella

The following object is masked from 'package:survival':

    rats
```

```
wald.test( b = coef( modelUni ), Sigma = vcov( modelUni ), Terms = 4:6 )
```

```
Wald test:
----------

Chi-squared test:
X2 = 20.9, df = 3, P(> X2) = 0.00011
```

La prueba χ^2 con valor 20.9 y tres grados de libertad se asocia con un p-valor 0.00011, lo que indica que el efecto general de `prestigio` es estadísticamente significativo.

7.2.7. Odds ratio

Calculamos directamente los odds ratio y sus respectivos coeficientes y los ponemos juntos en una única matriz.

```
exp( cbind( OR = coef( modelUni ), confint( modelUni ) ) )
```

```
Waiting for profiling to be done...

                OR   2.5 % 97.5 %
(Intercept) 0.0185 0.00189  0.167
pau         1.0023 1.00014  1.004
bach        2.2345 1.17386  4.324
prestigio2  0.5089 0.27229  0.945
prestigio3  0.2618 0.13164  0.512
prestigio4  0.2119 0.09072  0.471
```

Podemos decir que, al aumentar *bach* en una unidad, las probabilidades de ingresar en la universidad (frente a no ser admitido) aumenta un factor de

2.23. Para obtener más información sobre cómo interpretar los odds ratio visitar la página web [58].

7.2.7. Predicción

Comenzamos calculando la probabilidad predicha de admisión para cada valor de *prestigio*, manteniendo *pau* y *bach* en sus valores medios.

```
univ_pre1 <- with( univ, data.frame( pau = mean( pau ), bach = mean( bach ),
                                     prestigio = factor( 1:4 ) ) )
univ_pre1$prestigioPred <- predict( modelUni, newdata = univ_pre1,
                                    type = "response" )

head(univ_pre1)
```

```
  pau bach prestigio prestigioPred
1 588 3.39         1         0.517
2 588 3.39         2         0.352
3 588 3.39         3         0.219
4 588 3.39         4         0.185
```

Vemos que la probabilidad predicha de ser aceptado en un la universidades más prestigiosas (Prestigio = 1) es de 0.52 y de 0.18 para las instituciones de prestigio más bajo (Prestigio = 4), manteniendo *pau* y *Bach* en sus valores medios.

Podemos de manera similar crear una tabla de probabilidades predichas variando el valor de *pau* y *prestigio*. Creamos 100 valores de *pau* entre para cada valor de prestigio (es decir, 1, 2, 3 y 4).

```
univ_pre2 <- with( univ,
     data.frame( pau = rep( seq( from = 200, to = 800, length= 100 ), 4),
     bach = mean( bach ), prestigio = factor( rep( 1:4, each = 100 ) )))

univ_pre2$prestigioPred2 <- predict( modelUni, newdata = univ_pre1,
     type = "response" )

head( univ_pre2 )
```

```
  pau bach prestigio prestigioPred2
```

```
1 200 3.39          1          0.517
2 206 3.39          1          0.352
3 212 3.39          1          0.219
4 218 3.39          1          0.185
5 224 3.39          1          0.517
6 230 3.39          1          0.352
```

Puede encontrar una ampliación del ejemplo en [57].

7.2.8. Resumen de código en R

```
# Leer los datos de un fichero .csv
df <- read.table("45A-file.csv", sep = ";", head = TRUE)
### Primera aproximación a los datos
str(df)
summary(df)

# Modelo de regresión logística
logmodel <- glm( var1 ~ var2, data = df, family = binomial( ) )
logmodel

## resumen del modelo
summary(logmodel)

# Bondad de ajuste del modelo
## Devianza y Chi2
dev <- logmodel$deviance
nullDev <- logmodel$null.deviance
modelChi <- nullDev - dev
modelChi

chigl <- logmodel$df.null - logmodel$df.residual
chisq.prob <- 1 - pchisq( modelChi, chigl )
chisq.prob

# R^2
```

```
R2.hl <- modelChi/logmodel$null.deviance
R2.hl

# Odds ratio
exp( logmodel$coefficients )
## intervalos de confianza
exp( confint( logmodel ) )




# Diagnóstico del modelo
df$probabilidades.predichas <- fitted( logmodel )
df$studentized.residuals <- rstudent( logmodel )
df$dfbeta <- dfbeta( logmodel )
df$dffit <- dffits( logmodel )
df$leverage <- hatvalues( logmodel )




head( df[ , c( "Curado", "Tratamiento", "Duracion",
        "probabilidades.predichas" ) ] )
head( df[ ,c( "leverage", "studentized.residuals", "dfbeta" ) ] )

# Selección del modelo
modelog <- glm(var1 ~ var2 + var3 + var4, data = df, family = binomial( ) )
summary( modelog )
step( modelog, direction = "backward" )




# Supuestos del modelo
## Linealidad
df$logvar3Int <- log(df$var3)*df$var3
```

```
df$logvar4Int <- log(df$var4)*df$var4
df$logvar2Int <- log(df$var2)*df$var2



linealidad <- glm( var1 ~ var2 + var3 + var4 + logvar2Int + logvar3Int
+ logvar4Int, data = df, family = binomial( ) )
summary( linealidad )

## multicolinealidad
library( car )
vif( modelog )
```

8 Potencia Estadística

Francisco Javier Ibáñez López

8.1. Potencia estadística

Existe una pregunta que persigue a cualquiera que haga análisis estadísticos, *"¿cuánta muestra necesito?"*, o *"¿merece la pena hacer este estudio con estos n individuos que tengo?"*. Para responder estas preguntas necesitamos hacer "análisis de potencia" y este debería de ser indispensable en cualquier diseño experimental.

El análisis de potencia nos va a permitir:

- Determinar el tamaño muestral para descubrir efectos de un determinado tamaño con un cierto grado de confianza.
- Determinar la probabilidad de encontrar un efecto de un tamaño determinado, con un cierto nivel de confianza, con la muestra de la que actualmente se dispone (dado un tamaño muestral): *si la probabilidad es muy baja quizás deberíamos dedicarnos a otra cosa en lugar de llevar ese experimento a termino, o deberíamos de aumenta la muestra,...*

Cómo citar este capítulo: Ibáñez López F. J. (2022). Potencia Estadística. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (349-364). Editum. Ediciones de la Universidad de Murcia.

El análisis de potencia hay que entenderlo desde el punto de vista del contraste de hipótesis, así que repasemos algunos conceptos.

8.1.1. Test de hipótesis (null hypothesis significance testing).

En el test de hipótesis partimos de una hipótesis sobre un parámetro de una población, identificada como hipótesis **nula** (H_0). Entonces obtenemos una muestra de esa población y con esa muestra construimos un *"estadístico"* (un *"estimador"*) que empleamos para hacer inferencias sobre ese parámetro poblacional (desconocido).

Seguidamente, asumiendo que la hipótesis nula es cierta, calculamos la probabilidad de observar el estadístico que calculamos con la muestra u otro aún mayor (diferencias como esas o mayores).

Un ejemplo: *Imaginemos que tenemos unas mediciones de tiempo que representan el tiempo que tarda un adolescente en responder a un estimulo visual con o sin usar unos cascos de música en los que suena a toda caña el "highway to hell" de los ACDC.*

La hipótesis nula será:

$$H_0 : \mu_1 = \mu_2 \qquad (\mu_1 - \mu_2 = 0)$$

(μ_1 es la media de los tiempos de respuesta de los adolescentes con cascos (tratamiento) y μ_2 es la media de respuesta en segundos de los adolescentes sin oír a los ACDC).

Si rechazamos la Hipótesis nula, nos quedamos ('aceptamos') con la Hipótesis alternativa (H_1 o H_a), de que los dos tiempos no son iguales,

$$H_a : \mu_1 \neq \mu_2$$

Tomamos dos muestras , una de cada condición experimental: jóvenes alienados con las música de ACDC y jóvenes sin oír musica. Se les plantean los estímulos visuales y se les cronometra. Basándonos en estas dos muertas calculamos el estadístico:

$$\left(\overline{X_1} - \overline{X_2}\right) / \left(\frac{S}{\sqrt{n}}\right)$$

(S es la desviación estándar de las dos muestras, y n es el número de participantes en cada experimento, suponiendo que los tamaños muestrales son iguales).

Si la hipótesis nula es cierta y podemos suponer que los tiempos de reacción se distribuyen de forma normal, el estadístico que hemos construido sigue una distribución t con $2n - 2$ grados de libertad. Con esto podemos calcular la probabilidad de que el estadístico tome ese valor o valores superiores.

Si la probabilidad es pequeña y está por debajo de un punto de corte que fijemos, por ejemplo menos que 0.05, entonces la hipótesis nula es falsa, debe ser rechazada y se debe aceptar la hipótesis alternativa. A este punto de corte se le conoce como "*nivel de significación*" (o "significancia").

Cuando usamos una muestra para hacer una inferencia sobre una población, las cuatro posibilidades que se nos pueden plantear son:

- La Hipótesis nula es **falsa** y nuestro contraste nos invita a rechazarla: hemos llegado entonces a una **conclusión correcta**.
- La Hipótesis nula es cierta y nuestro contraste **NO** nos invita a rechazarla: hemos llegado entonces a una **conclusión correcta**.
- La Hipótesis nula es cierta y nuestro contraste nos invita a rechazarla: hemos llegado entonces a una **conclusión incorrecta**. Se dice que cometemos error de tipo I o α (decimos que hay una diferencia cuando no la hay, y perjudicamos a los vendedores de cascos y a la productora de ACDC).
- La Hipótesis nula es **falsa** y nuestro contraste NO nos invita a rechazarla: hemos llegado entonces a una **conclusión incorrecta**. Se dice que cometemos error de tipo II o β (decimos que hay NO hay una diferencia cuando sí la hay y perjudicamos a los jóvenes que piensan que reacciona de igual modo).

Así, cuando planificamos un experimento, hay que prestar atención a cuatro cosas principalmente: tamaño muestral, nivel de significación, potencia y tamaño del efecto.

- Tamaño muestral: número de observaciones en cada condición/grupo (con-

	Rechazamos H_0	Aceptramos H_0
H_0 Cierto	**Error de tipo I**	Correcto
H_0 Falso	Correcto	**Error de tipo II**

Tabla 8.1: *Situación real frente a decisión.*

dición experimental) del experimento.

- Nivel de significación (α): probabilidad de cometer error de tipo I (cometemos **error de tipo I o** α, cuando *'afirmamos'* que si hay diferencias (p<0.05) y en verdad no las hay).
- Potencia (*power*): se define como 1 menos la probabilidad de cometer error de tipo II. Es "la probabilidad de encontrar un efecto que realmente existe" (es la capacidad que tiene un test para revelar diferencias que realmente existen). Cometemos **error de tipo II o** β, cuando 'afirmamos' que no hay diferencias (p>0.05) y en verdad sí las hay.
- Tamaño del efecto: es la magnitud del efecto bajo la hipótesis alternativa.

Un esquema de las posibles situaciones que se dan después de tomar una decisión dependiendo de si la hipótesis nula era cierta o falsa se puede ver en la tabla 8.1.

El tamaño muestral y la significación están bajo el control del investigador; la potencia y los tamaños de los efecto son controladas de forma más indirecta. *Nosotros quisiéramos maximizar la probabilidad de encontrar un efecto real y minimizar la probabilidad de encontrar un efecto que no es real y, a la misma vez, mantener la muestra en un tamaño "razonable"*. Las cuatro magnitudes de las que estamos hablando están íntimamente relacionadas, cumpliendose que, **fijadas tres de ellas, la cuarta puede ser determinada**. Un esquema de este compromiso puede verse en la figura 8.1

La librería pwr. Vamos a emplear la librería `pwr`[59], que tiene muchas funciones, centrándonos solo en algunas de ellas. En estas podemos especificar tres de los factores $(n, \alpha, 1-p, ES)$ y determinar el cuarto.

Generalmente el tamaño del efecto es lo más difícil de especificar. Suele

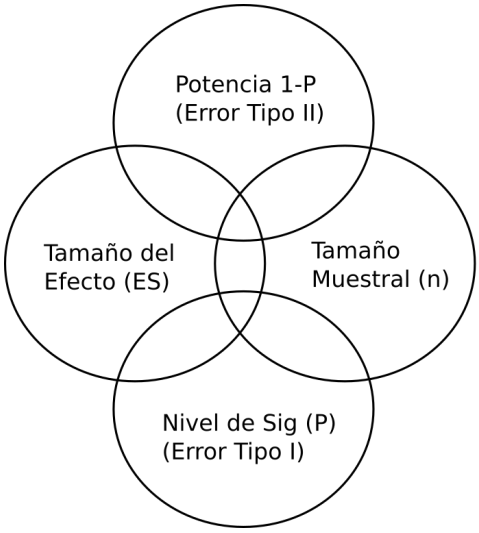


Figura 8.1: *Compromiso entre Potencia, Tamaño del Efecto, Tamaño muestra y nivel de significación.*

requerir conocimiento del tipo de datos, de la "experiencia" en investigaciones anteriores, etc... Trataremos este asunto al final del capítulo para centrarnos ahora en las funciones de la librería `pwr` (ver tabla 8.2).

Comenzamos instalando y cargando el paquete `pwr`.

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

8.1.2. t-test.

Emplearemos la función `pwr.t.test()`:

```
pwr.t.test(n= ,d= ,sig.level= ,power= ,type= ,alternative= )
* type = c("two.sample", "one.sample", "paired")
* alternative = c("two.sided","less","greater")
```

Para `sig.level`, '0.05' es el valor por defecto; `d` es el tamaño del efecto y lo definimos como "la diferencia de medias estandarizada",

Función	Descripción
pwr.2p.test()	2 proporciones (n iguales)
pwr.2p2n.test()	2 proporciones (n desiguales)
pwr.anova.test()	ANOVA de una vía balanceada
pwr.chisq.test()	Chi-cuadrado
pwr.f2.test()	Modelo Lineal General (MLG)
pwr.p.test()	Proprociones (una muestra)
pwr.r.test()	Correlación
pwr.t.test()	t.test (una y dos muestras, modelo dependiente)
pwr.t2n.test()	t.test (dos muestras, modelo independiente)

Tabla 8.2: *Funciones de la librería pwr.*

$$d = \frac{\mu_1 - \mu_2}{\sigma}$$

donde σ^2=varianza común

Ejemplo. Retomamos el ejemplo de los adolescentes que oían ACDC a toda caña. Tenemos un experimento 'two-tailed' y un test t de datos independientes.

- Tamaño del efecto: imaginemos que sabemos que el tiempo de reacción de los jóvenes tiene una desviación estándar de 1,25 seg. Y también vamos a suponer que una diferencia de 1 seg es ya diferencia suficiente, es importante. Con esto d=1/1,25=0,8 o mayor.
- Potencia: queremos estar al 90% seguros de que encontramos el efecto si este existe.
- Significancia: no queremos cometer más de un 5% de error de tipo I, es decir, queremos estar al 95% seguros de que los efectos que encontremos son reales. *¿Cuántos sujetos necesitamos para nuestro estudio?*

```
pwr.t.test(d=.8, sig.level=.05, power=.9, type="two.sample",
alternative="two.sided")
```

```
     Two-sample t test power calculation
```

```
              n = 33.8
              d = 0.8
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: n is number in *each* group
```

Ejemplo. Cambiamos la pregunta. Supongamos que ahora queremos detectar un 0.5 de diferencias en desviaciones estándares entre las dos poblaciones (tamaño del efecto). Queremos ampliar la seguridad de no cometer error de tipo I a 1/100 = 0,01. Además supongamos que sólo podemos plantearnos incluir 40 individuos en total (20 en cada grupo). ¿Cuál es la probabilidad de dar con un efecto significativo que sea real?

$d = 0.05 = 0.625/1.25 \qquad (\mu_1 - \mu_2 = 0.625)$

```
pwr.t.test(n=20, d=.5, sig.level=.01, type="two.sample",
                                  alternative="two.sided")

     Two-sample t test power calculation

              n = 20
              d = 0.5
      sig.level = 0.01
          power = 0.144
    alternative = two.sided

NOTE: n is number in *each* group
```

Así que tendremos solo un 14% de probabilidad de encontrar diferencias de medias de 0.625 o menores y un 86% de probabilidad de no encontrar un efecto real (obtenemos una potencia paupérrima).

Hemos estado suponiendo que los tamaños muéstrales son iguales. Si los tamaños los conocemos y son diferentes, entonces:

```
pwr.t2n.test(n1=, n2= , d= , sig.level = 0.05, power= ,
```

```
alternative =)
* alternative = c("two.sided", "less","greater"))
```

```
pwr.t2n.test(n1=20,n2=30, d=.5, sig.level=.01,  alternative="two.sided")
```

```
     t test power calculation

             n1 = 20
             n2 = 30
              d = 0.5
      sig.level = 0.01
          power = 0.183
    alternative = two.sided
```

Aumenta la potencia a 18%.

8.1.3. ANOVA

Para ANOVA emplearemos la función `pwr.anova.test()`.

```
pwr.anova.test(k = NULL, n = NULL, f = NULL, sig.level =
0.05, power = NULL)
```

Donde
k = número de grupos
n = es el tamaño de muestra común en cada grupo (la menor de las n's.)
f = tamaño del efecto para una ANOVA

$$f = \sqrt{\frac{\sum_{i=1}^{k} p_i \times (\mu_i - \mu)}{\sigma^2}}$$

donde
$p_i = \frac{n_i}{N_i}$
n_i = *nmero de observaciones en el grupo i*
N = *nmero total de observaciones*
μ_i = *media del grupo i*

$\mu = media\ total\ (grand\ mean)$
$\sigma^2 = varianza\ entre\ grupos$

Ejemplo. Queremos comparar 5 grupos y necesitamos saber el tamaño muestral necesario de cada grupo para tener un 80% de potencia estadística, cuando el tamaño del efecto buscado es de 0.25 y el nivel de significación sea de 0.05.

```
pwr.anova.test(k=5, f=.25, sig.level=.05, power=.8)
```

```
     Balanced one-way analysis of variance power calculation

              k = 5
              n = 39.2
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

Necesitamos pues 39 individuos en cada grupo, es decir $39 * 5 = 195$ individuos en total.

Ejercicio. Calcula el tamaño muestral si "buscáramos" efectos más grandes f = 0.5.

8.1.4. Correlación

Para calcular la potencia en correlaciones, empleamos la función `pwr.r.test()`.

```
pwr.r.test(n = NULL,r = NULL,sig.level = 0.05,power = NULL, alternative=
c("two.sided", "less","greater"))
```

`r` es el tamaño del efecto (coeficiente de correlación)

Ejemplo. Estamos estudiando la relación entre dos variables "dinero que uno gana" y "número de amigos/contactos en Google". Nuestra hipótesis es que

la correlación entre ambas variables es de 0.25 o mayor.

$$H_0 : \rho <= 0.25$$

$$H_a : \rho > 0.25$$

Entendemos que ρ es el coeficiente de correlación poblacional. Queremos una potencia del 90 % y una significación del 0.05. ¿Cuántas observaciones necesitamos para corroborar la hipótesis?

```
pwr.r.test(r=.25, sig.level=.05, power=.90, alternative="greater")
```

```
approximate correlation power calculation (arctangh transformation)

        n = 133
        r = 0.25
sig.level = 0.05
    power = 0.9
alternative = greater
```

Necesitamos 134 sujetos.

8.1.5. Modelos lineales

En modelos lineales, como la regresión lineal múltiple, emplearemos la función `pwr.f2.test()`.

```
pwr.f2.test(u= NULL, v= NULL, f2= NULL, sig.level=0.05, power=
NULL)
```

- $f^2 = \dfrac{R^2}{1-R^2}$
  donde R = *coeficiente de determinacin poblacional.*
- $f^2 = \dfrac{R^2_{AB} - R^2_A}{1-R^2_{AB}}$
  donde :
  R^2_A = *varianza explicitada en la poblacin por el conjunto de variables A.*
  R^2_{AB} = *varianza explicitada en la poblacin por el conjunto de variables A y B a la vez.*
- u, v : *Grados de libertad del numerador y denominador.*

Emplearemos la primera fórmula cuando evaluamos el impacto de un conjunto de predictores en una variable dependiente. La segunda fórmula es apropiada cuando evaluamos el impacto de un conjunto de predictores sobre otro conjunto de predictores.

Ejemplo. *Imaginemos que estamos interesados en saber cuándo el estilo de mando ('estilo') de un jefe afecta a la satisfacción laboral ('satisfacción'), más allá del 'sueldo' y el 'carácter' del empleado (supongamos que todo está medido en una encala cuantitativa).*

Sabemos que el "estilo"" del jefe está relacionado con 4 variables y "suelo" y "carácter" con tres.

Supongamos que, gracias a estudios previos, sabemos que "salario" y "carácter" explican el 30 % de la varianza en "satisfacción" laboral.

Pensamos que sería suficiente que "estilo" explicará al menos un 5 % de "satisfacción". Suponiendo una potencia del 90 % ¿cuántos individuos se necesitarían para asegurar esa contribución con un 90 % de confianza?
$u = 3$, número de predictores, menos el número de predicadores en el conjunto B
$f = \frac{0.35 - 0.30}{1 - 0.35} = 0.00769$

```
pwr.f2.test(u=3, f2=0.0769, sig.level=0.05, power=0.90)
```

```
     Multiple regression power calculation

              u = 3
              v = 184
             f2 = 0.0769
      sig.level = 0.05
          power = 0.9
```

Los grados de libertad en MLG se calculan así N - k - 1 (N =numero total de observaciones, k=número de variables indep), N - 7 - 1 = 185, así que la muestra requerida es N = 185 + 7 + 1 = 193.

8.1.6. Test de proporciones

Emplearemos la función `pwr.2p.test()`.

```
pwr.2p.test(h = NULL ,n = NULL ,sig.level = 0.05 ,power = NULL,
alternative= )
* alternative = c("two.sided","less","greater")
* h es el tamaño del efecto y n el tamaño muestral común.
```

$$h = 2arcsin\left(\sqrt{p_1}\right) - 2arcsin\left(\sqrt{p_2}\right)$$

Podemos emplear la función `ES.h(p1,p2)` para calcularlo.

Si los tamaños son diferentes emplearemos la función `pwr.2p2n.test()`.

```
pwr.2p2n.test(h= ,n1= ,n2= ,sig.level=0.05 ,power= , alternative=
)
```

Ejemplo. *Sabemos que un tratamiento es efectivo en un 60 % de la población. Un nuevo tratamiento, optimizado y mucho más caro será autorizado si es efectivo en un 65 % de la población. ¿Cuantos sujetos necesitamos si queremos hacer un estudio comparando ambos medicamentos?*

```
pwr.2p.test(h=ES.h(.65, .6), sig.level=.05, power=.9,alternative="greater")
```

```
     Difference of proportion power calculation for binomial distribution
     (arcsine transformation)

              h = 0.103
              n = 1604
      sig.level = 0.05
          power = 0.9
    alternative = greater

NOTE: same sample sizes
```

Necesitamos 1604 individuos en cada condición experimental, es decir, ¡¡3210 sujetos!!.

G. Étnico	Promociona	No Promociona
Caucásico	0.42	0.28
Negro	0.03	0.07
Hispano	0.1	0.1

Tabla 8.3: *Ejemplo* `pwr.chisq.test()`*, "grupo étnico" y promoción.*

Ejemplo. *Si la mejora del tratamiento 'asegura la farmacéutica que es de un 20 %' ¿cuántos individuos necesitamos para contrastar esta hipótesis?*

```
pwr.2p.test(h=ES.h(.80, .6), sig.level=.05, power=.9,alternative="greater")
```

Solución: tan sólo 88 individuos por grupo (N=176).

8.1.7. Test χ^2

Emplearemos la función `pwr.chisq.test()`.

```
pwr.chisq.test(w = NULL ,N = NULL ,df = NULL ,sig.level = 0.05
,power = NULL)
```

Donde w es el tamaño del efecto.

$$w = \sqrt{\sum_{i=1}^{m} \frac{(p0_i - p1_i)^2}{p0_i}}$$

donde
$p0_i$ = *probabilidad de la celda i – esima bajo la* H_0
$p1_i$ = *probabilidad de la celda i – esima bajo la* H_1
m = *nmero de celdas de la tabla de contingencia.*

Ejemplo. Queremos ver la relación entre "grupo étnico" y promoción.

Según la tabla 8.3, esperamos que el 42 % de la población sea caucásicos que promocionan ($0.42 = 0.7 \times 0.6$), el 7 % serán negros que no promocionan... Fijamos $\alpha = 0.05$ y la potencia deseada en el 90 % (0.9).

Los grados de libertad en una tabla de doble entrada son $(r-1)\times(c-1)$, donde r representa el número de niveles de la primera variable y c el de la segunda, $gl = (r-1)\times(c-1) = (3-1)\times(2-1) = 2\times 1 = 2$
Calculamos el tamaño del efecto:

```
P <- matrix(c(.42, .28, .03, .07, .10, .10), byrow=TRUE, nrow=3)
ES.w2(P)
```

```
[1] 0.185
```

Calculamos ahora el tamaño muestral que necesitamos:

```
pwr.chisq.test(w=.1853, df=2, sig.level=.05, power=.9)
```

```
     Chi squared power calculation

              w = 0.185
              N = 369
             df = 2
      sig.level = 0.05
          power = 0.9

NOTE: N is the number of observations
```

Así que necesitamos 368 individuos para detectar una relación entre estas dos variables.

8.1.8. Tamaños de los efectos

Como ya hemos dicho, lo más difícil suele ser determinar el tamaño del efecto. Suele requerir experiencia previa, haber realizado o estar en posesión de estudios previos en los que se tenga cierta seguridad de qué representan los parámetros que necesitamos. Pero, ¿qué hacemos cuando nos enfrentamos a una situación nueva?

Para estudios en Ciencias Sociales, Cohen[60] propuso unos límites para considerar efectos de diferentes tamaños: pequeños, medios y grandes.

Esta guía puede que no se adapte a un campo de estudio concreto. Lo que

Test	Medida ES	Pequeño	Mediano	Grande
T-tets	d	0.20	0.50	0.80
ANOVA	f	0.10	0.25	0.40
MLG	f2	0.02	0.15	0.35
Proporciones	h	0.20	0.50	0.80
Chi-2	w	0.10	0.30	0.50

Tabla 8.4: *Valores sugeridos por Cohen (1988) para diferentes medidas del tamaño del efecto y diferentes test.*

podemos hacer cuando no tenemos "ni idea", es variar los parámetros y observar que impacto tienen sobre los valores de potencia, tamaño de muestras, etc...

Podemos crear gráficos de potencia para analizar diferentes tamaños muéstrales y relacionarlos con el tamaño del efecto. Por ejemplo, para una ANOVA, el siguiente código genera la figura 8.2.

```
es <- seq(.1, .5, .01)
nes <- length(es)
samsize <- NULL
for (i in 1:nes){
  result <- pwr.anova.test(k=5, f=es[i], sig.level=.05, power=.9)
  samsize[i] <- ceiling(result$n)
}
plot(samsize,es, type="l", lwd=2, col="red",
ylab="Effect Size",
xlab="Tamaño muestral x grupo)",main="ANOVA de 1 via, Potencia=0.90y Alpha=.05")
```

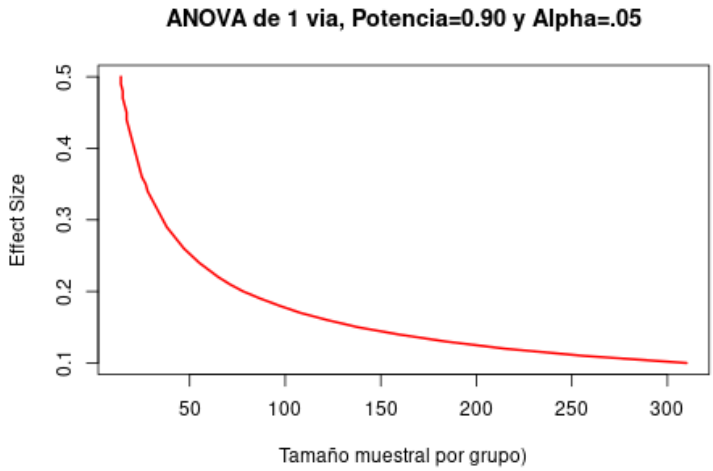


Figura 8.2: *Relación del tamaño muestral y el tamaño del efecto para una test ANOVA.*

9 Miscelánea visual

Aurora Gónzalez Vidal

9.1. Introducción a los sistemas gráficos en R

Para el desarrollo de este capítulo hemos utilizado como referencia bibliográfica básica el libro [61] y la web [62].

Los gráficos son una forma muy útil de estudiar los datos antes de analizarlos. Si no inspeccionamos los datos mediante un gráfico, realmente no sabremos cómo interpretar los resultados.

Si utilizamos el comando `demo(graphics)` podemos ver muchas de las posibilidades que nos ofrece R en lo que se refiere a la generación de gráficos.

Al realizar un gráfico en R, este es enviado a un dispositivo gráfico, que no es más que una ventana gráfica o un archivo.

Cómo citar este capítulo: Gónzalez Vidal A (2022). Miscelánea visual. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (365-460). Editum. Ediciones de la Universidad de Murcia.

Por otro lado existen diferentes sistemas gráficos que podemos utilizar: `base`, `lattice` o `ggplot2` y generalmente no podremos mezclar.

Además podemos diferenciar entre dos tipos de funciones gráficas

- Funciones de alto nivel que lanzan un nuevo `device gráfico`
- Funciones de bajo nivel que añaden elementos a una gráfica ya existente.

9.1.1. Base

Los gráficos se construyen poco a poco, tratando cada aspecto del gráfico separadamente a través de una serie de funciones; es más simple conceptualmente hablando y permite construir reflejando el proceso de pensamiento.

Los inconvenientes de este sistema son

- No puedes volver atrás una vez que has empezado (e.d. para ajustar márgenes); necesitamos pensarlo de antemano.
- Es difícil "traducirle" el gráfico a otros una vez hemos creado el gráfico
- El gráfico es sólo una serie de expresiones de R

```
library(datasets)
data(cars)
with(cars, plot(speed, dist))
```

9.1.2. Lattice

Los gráficos se crean normalmente mediante una única función, por lo que todos los parámetros del gráficos se tienen que especificar a la vez, lo que permite a R calcular automáticamente los espaciados y los tamaños de letra.

Muy útiles para gráficos condicionados: analizando cómo cambia `y` a través de los distintos niveles de `z`. Además es un buen sistema para poner varios gráficos en una mismo cuadro.

Sus inconvenientes son

- A veces es complicado especificar un gráfico completo en una sola llamada a una función

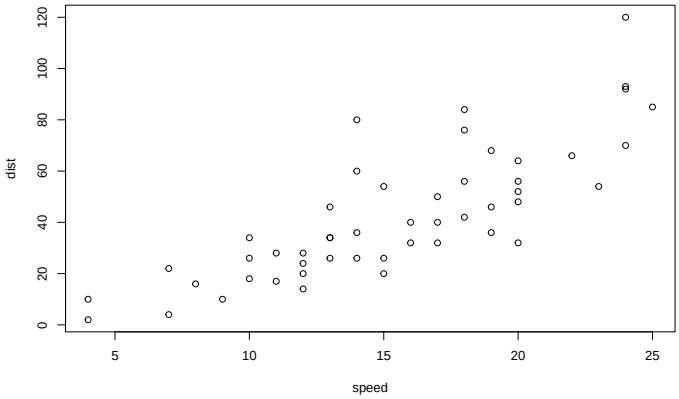


Figura 9.1: *Gráfico que relaciona velocidad y distancia del conjunto de datos cars*

- Los títulos, etiquetas, etc no son intuitivos
- Dificultad de uso y se necesita preparación intensa
- No se puede añadir elementos al gráfico una vez está creado

```
library(lattice)
df <- data.frame(state.x77, region = state.region)
xyplot(Life.Exp ~ Income | region, data = df, layout = c(4, 1))
```

9.1.3. ggplot2

Combina conceptos de los dos sistemas base y del `lattice`, pero usa una implementación diferente.

Se ocupa automáticamente con espaciados, texto, títulos pero además nos permite añadir elementos al gráfico.

Superficialmente es similar al sistema `lattice` pero generalmente es más fácil/intuitivo de usar

El modo por defecto toma muchas decisiones por el usuario, aunque es altamente personalizable.

```
library(ggplot2)
```

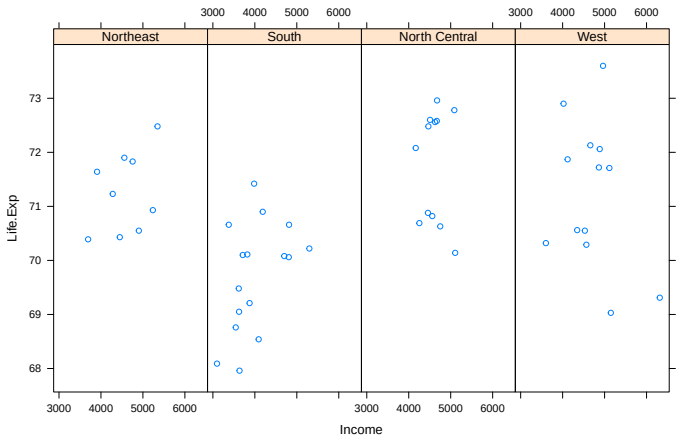


Figura 9.2: *Gráfico con lattice*

```
data(mpg)
# qplot( displ, hwy, data = mpg)
qplot(mpg$displ, mpg$hwy)
```

Como vemos los sistemas gráficos son muy diferentes entre sí y normalmente no podemos mezclarlos.

Podemos encontrar más información sobre los sistemas gráficos en [63].

9.2. Sistema base

El sistema gráfico básico es el más utilizado usualmente y es un sistema muy poderoso para la creación de gráficos de dos dimensiones. Está implementado en los siguientes paquetes [62]

- `graphics`: contiene las funciones del sistema de gráfico básico, incluyendo `plot`, `hist`, `boxplot` y otros muchos.
- `grDevices`: contiene los dispositivos gráficos como X11, PDF, PostScript, PNG, etc.

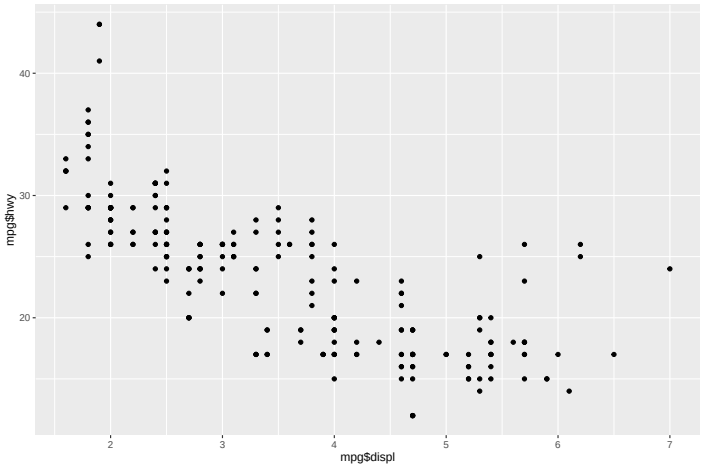


Figura 9.3: *Gráfico con ggplot*

9.2.1. Dispositivos gráficos (device)

Un dispositivo gráfico es el lugar donde hacemos que aparezca el gráfico

- Una ventana del ordenador
  - En Unix/Linux se lanza con `x11()`
  - En Windows se lanza con `windows()`
  - En Mac se lanza con `quartz()`
- Un archivo PDF
- Un archivo PNG o JPEG
- etc

Cuando hacemos un gráfico, debemos considerar para qué usaremos el gráfico para determinar a qué dispositivo gráfico debemos enviarlo. El lugar más común para enviar un gráfico es a una ventana y este es al que por defecto los manda R si no indicamos lo contrario.

La lista de dispositivos gráficos disponible en R se obtiene ejecutando el comando `?device`. Si abrimos más de un dispositivo gráfico, el último dispositivo que hayamos ejecutado se convierte en el dispositivo activo, sobre el cual se van a dibujar las gráficas que generemos. Con la función `dev.list()` se muestra una lista de todos los dispositivos abiertos.

9.2.1. Manejo de devices

Para saber cual es el dispositivo activo, cambiarlo o bien cerrarlo, utilizamos los siguientes comandos

```
dev.cur()  # nombre y número del device activo
dev.set(n)  # cambiarimos el device activo al n
dev.off()  # cerrar el device activo
dev.off(n)  # cerrar el device n
```

9.2.1. Exportar un gráfico

```
library(datasets)
pdf(file = "migrafico.pdf")  # abre un device PDF, crea
# 'migrafico.pdf' en el wd y lo guarda en un archivo
plot(faithful$eruptions, faithful$waiting)
dev.off()  # cierra el device PDF
# Cualquier gráfico que dibujemos ahora aparecerá en la pantalla
```

Podemos guardarlos gráficos como imágenes (.png y .jpeg) con las funciones `png( "migrafico.png" )` y `jpeg( "migrafico.jpg" )`.

9.2.2. Creación de gráficos en R

Hay dos fases en la creación de un gráfico en este sistema (base):

- Inicializar un gráfico (funciones de alto nivel)
- Añadir elementos a un gráfico existente (funciones de bajo nivel)

Funciones como `plot()`, `hist()` lanzan un `device` gráfico (si no está ya creado) y representan un nuevo gráfico en el dispositivo gráfico.

Estas funciones tienen muchos `argumentos` como poner un título, etiquetas a los ejes, etc. Para ver los argumentos utilizamos la ayuda de la función, `?plot`.

Además el sistema gráfico básico tiene muchos `parámetros` que podemos establecer y ajustar. Todos estos parámetros están documentados en ?par, los veremos más adelante.

9.2.3. Funciones gráficas de alto nivel

Veamos a continuación algunas de las funciones gráficas más utilizadas. Todas ellas admiten los siguientes argumentos generales:

- `add = FALSE` (por defecto): si es TRUE superpone al gráfico existente
- `axes = TRUE` (por defecto): si es FALSE no dibuja los ejes ni la caja del gráfico
- `type =`: especifica el tipo de gráfico, p = puntos (por defecto), l = líneas, b = puntos conectados por líneas, h = líneas verticales
- `xlim =`, `ylim =`: especifica los límites inferiores y superiores de los ejes
- `xlab =`, `ylab =`: añade etiquetas a los ejes
- `main =`: añade el título principal
- `sub =`: añade un subtitulo (letra más pequeña).

9.2.3. Gráficos de dispersión

- `plot(x)`: dibuja los valores de x (en el eje y) ordenados en el eje x.

```
set.seed(13614)
x <- rnorm(30)
plot(x)
```

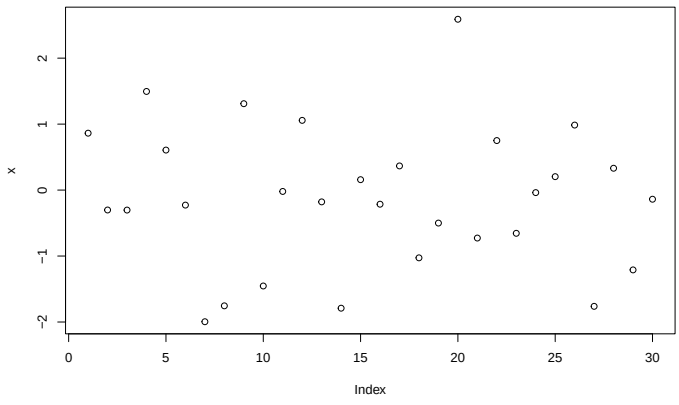


Figura 9.4: *Gráfico función plot() con un solo argumento*

- `plot( x, y )`: gráfico bivariado para los valores de x e y.

```
x <- 1:9
y <- c(3, 7, 2, 9, 13, 5, 12, 6, 1)  # create some data
plot(x, y)
```

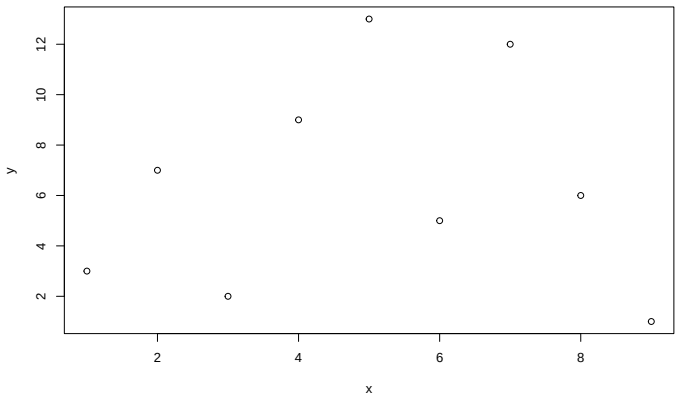


Figura 9.5: *Gráfico función plot() con un dos argumentos (x e y)*

El argumento `type` determina el tipo de puntos: líneas, puntos, ambos, etc.

```
plot(x, y, type = "b", main = "punto-línea", xlab = "eje abcisas",
ylab ="eje ordenadas")
```

Cuando cargamos la librería datasets, el conjunto de datos 'airquality' se puede utilizar (entre otros):

```
library(datasets)
colores <- factor(airquality$Month)
with(airquality, plot(Wind, Ozone, col = colores))
```

- `pairs(x)`: si x es una matriz o un `data.frame`, esta función dibuja todas las posibles gráficas bivariadas entre las columnas de x.

```
df <- mtcars[, c(1, 3, 5:6)]
pairs(df, panel = panel.smooth, main = "Matriz de dispersión")
```

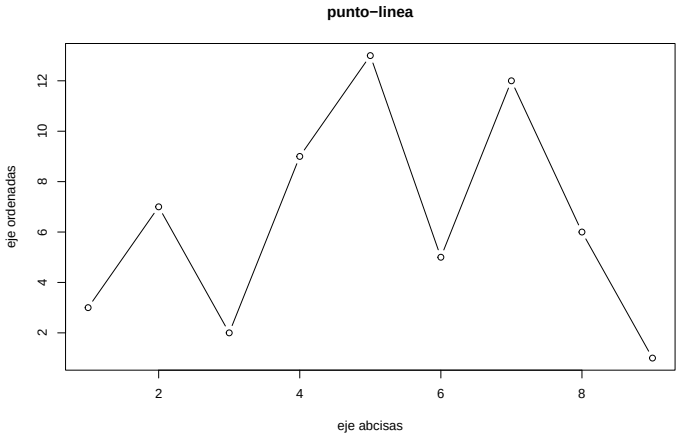


Figura 9.6: *Gráfico de dispersión con más argumentos*

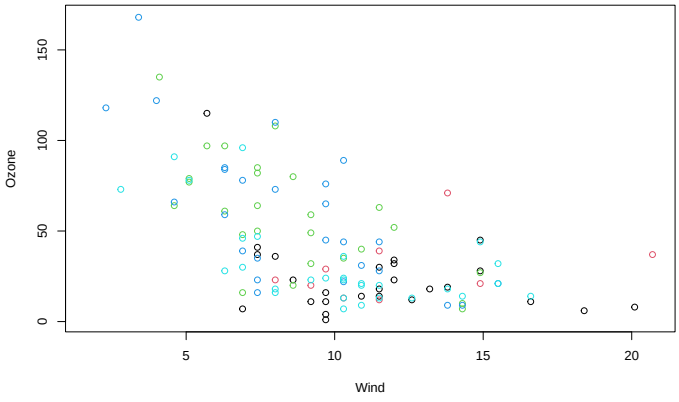


Figura 9.7: *Datos de airquality con colores*

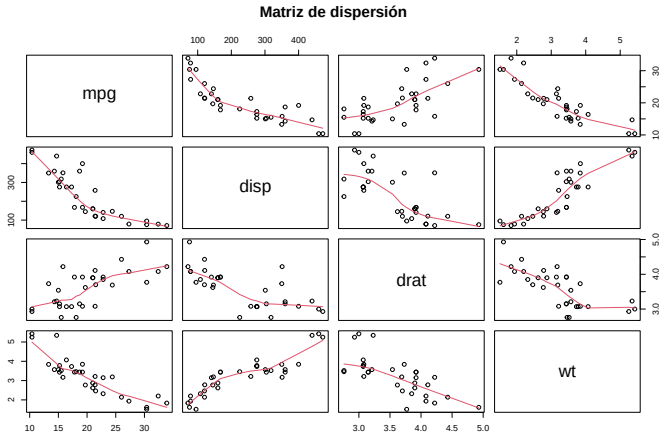


Figura 9.8: *Matriz de dispersión (pairs)*

9.2.3. Gráficos de barras

Para crear gráficos de de barras se utiliza la función `barplot(x)`, donde `x` es un vector o una matriz. Si `x` es un vector, los valores determinan las alturas de las barras y si es una matriz y `beside = FALSE` (por defecto) entonces cada barra corresponde a una columna de altura, con los valores de la columna dando las alturas de "sub-barras" apiladas. Si `x` es una matriz y `beside = TRUE`, entonces los valores de cada columna se yuxtaponen en lugar de apilarse.

Veamos un ejemplo de cada uno de estos casos [61].

- **Gráfico de barras simple**

```
x <- table(mtcars$gear)
barplot(x, main = "Gráfico de barras",
        names.arg = c("3 Gears", "4 Gears", "5 Gears"))
```

- **Gráfico de barras apilado**

```
x <- table(mtcars$vs, mtcars$gear)
barplot(x, main = "Gráfico de barras",
        horiz = TRUE, col = c("blue", "red"), legend = rownames(x))
```

- **Gráfico de barras agrupado**

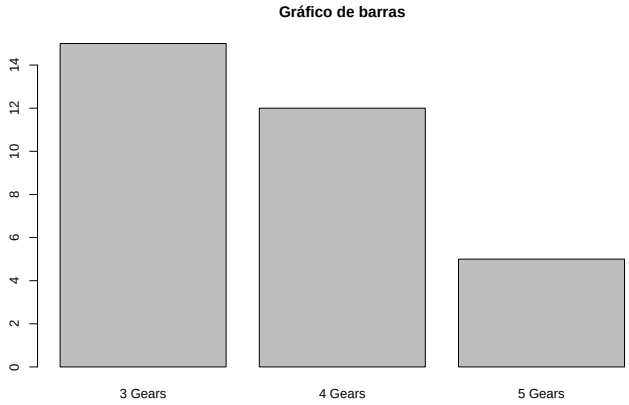


Figura 9.9: *Gráfico de barras simple*

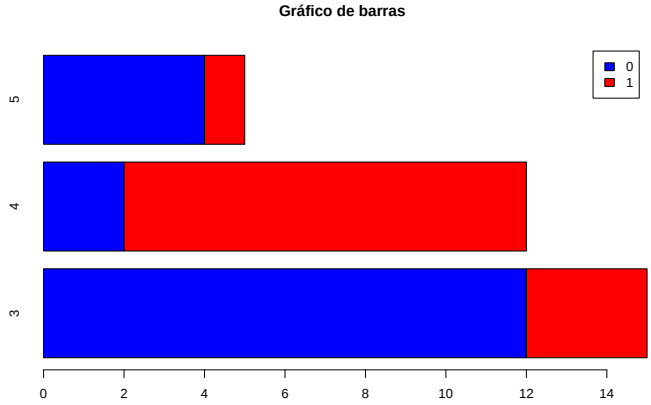


Figura 9.10: *Gráfico de barras apilado*

```
barplot(x, main = "Gráfico de barras", beside = TRUE,
col = c("blue", "red"), legend = rownames(x))
```

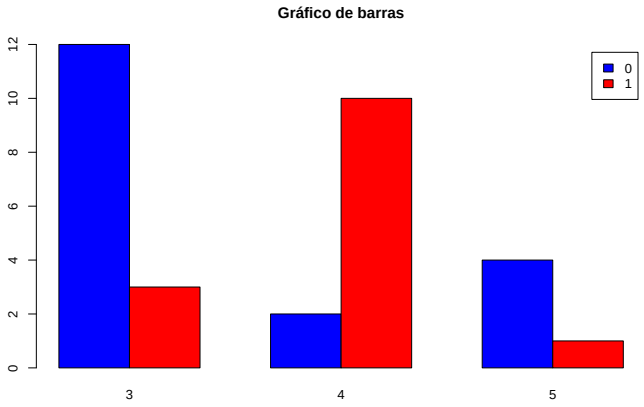


Figura 9.11: *Gráfico de barras agrupado*

9.2.3. Boxplot

Los diagramas de caja-bigote se pueden crear para variables individuales o para variables por grupo. El formato de la función es `boxplot(x,data =)`, donde `x` es una fórmula y `data` es el `data.frame` que contiene los datos.

Un ejemplo de una fórmula es `y ~ grupo` en el que se genera un diagrama de cajas separado para la variable numérica por cada valor de grupo.

La opción `varwidth = TRUE` hace la anchura del diagrama proporcional a los tamaños muéstrales y `horizontal = TRUE` invierte la orientación del eje.

- **Boxplot simple**

```
with(cars, boxplot(speed))
```

- **Boxplot múltiple**

```
airquality$Month <- factor(airquality$Month)
boxplot(Ozone ~ Month, data = airquality, xlab = "Mes", ylab = "Ozono")
```

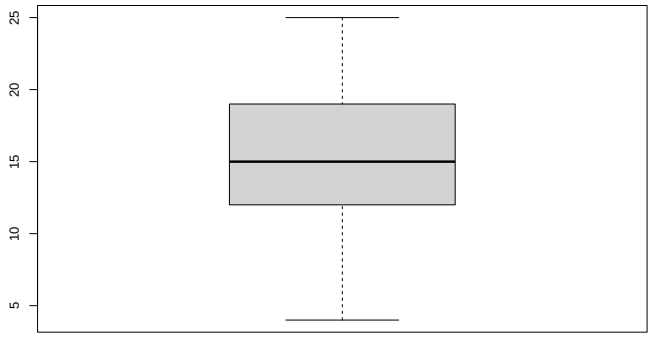


Figura 9.12: *Boxplot simple*

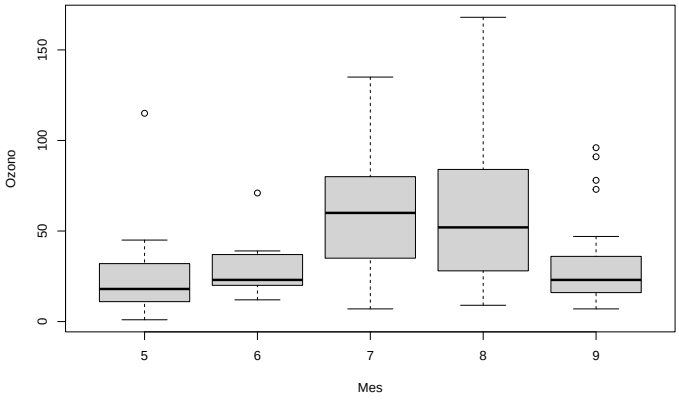


Figura 9.13: *Boxplot múltiple*

9.2.3. Histogramas

Podemos crear histogramas con la función `hist(x)`, donde `x` es un vector numérico. Con la opción `frec = FALSE` se representan las densidades en lugar de las frecuencias. La opción `breaks = n` controla el número de barras.

```
hist(airquality$Ozone, breaks = 13, col = "blue")
rug(airquality$Ozone)
```

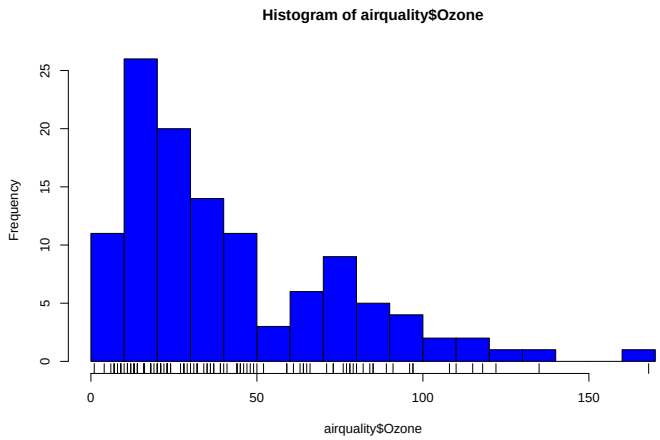


Figura 9.14: *Histograma simple*

9.2.3. Densidades

Tenemos que calcular la función de densidad mediante `density(x)` y después representarla con `plot()`.

```
dens <- density(cars$speed)
plot(dens, main = "Densidad de velocidad ")
```

Podemos rellenar el área que hay bajo la curva de densidad

```
plot(dens, main = "Función de densidad")
polygon(dens, col = "purple", border = "blue")
```

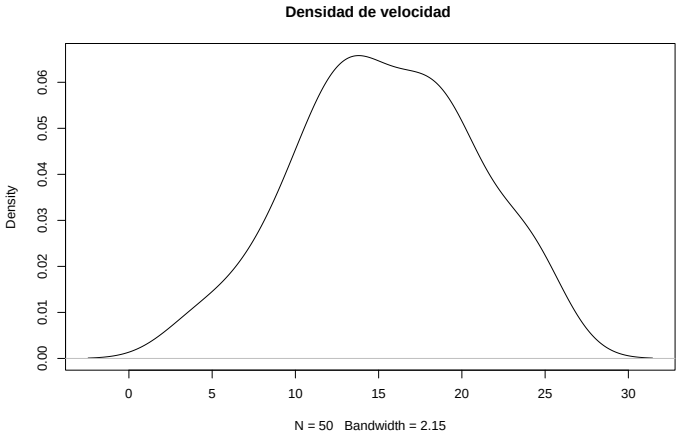


Figura 9.15: *Función de densidad*

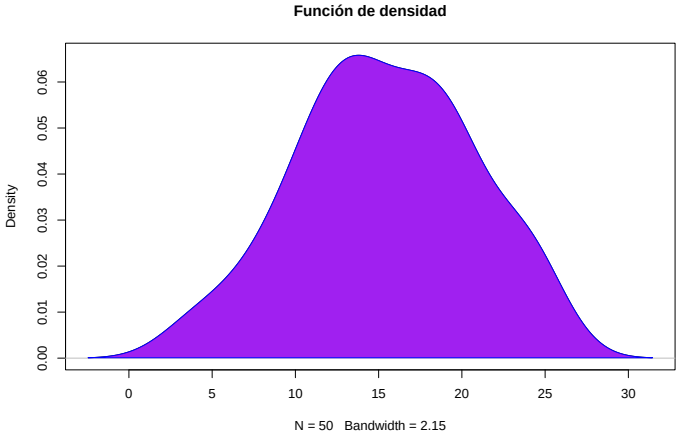


Figura 9.16: *Función de densidad coloreada*

9.2.3. Gráficos Q-Q plot (normalidad)

- `qqnorm(x)`: cuantiles de x con respecto a lo esperado bajo una distribución normal.

```
x <- rnorm(20)
qqnorm(x)
qqline(x)
```

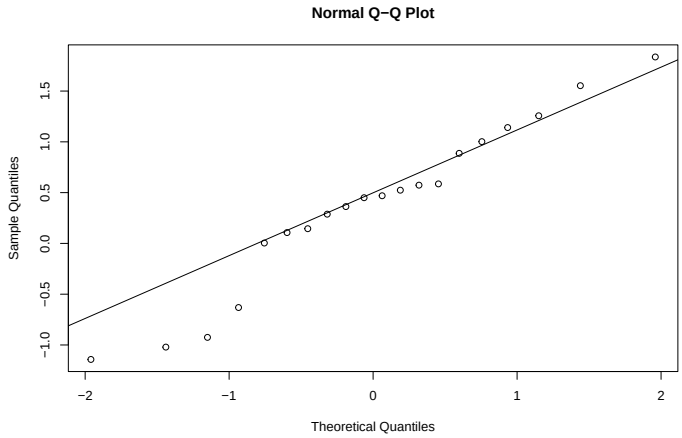


Figura 9.17: *qqnorm y qqline*

- `qqplot(x, y)`: cuantiles de y con respecto a los de x

```
x <- rnorm(30)
y <- rnorm(30)
qqplot(x, y, col = "blue", main = "Cuantiles")
```

9.2.3. Otras funciones

- `pie()`: gráfico de sectores.

```
precios <- c(0.23, 0.35, 0.14, 0.2, 0.23)
names(precios) <- c("manzana", "uva", "pera", "naranja", "cereza")
pie(precios, col = rainbow(length(names(precios))), main = "Precios frutas")
```

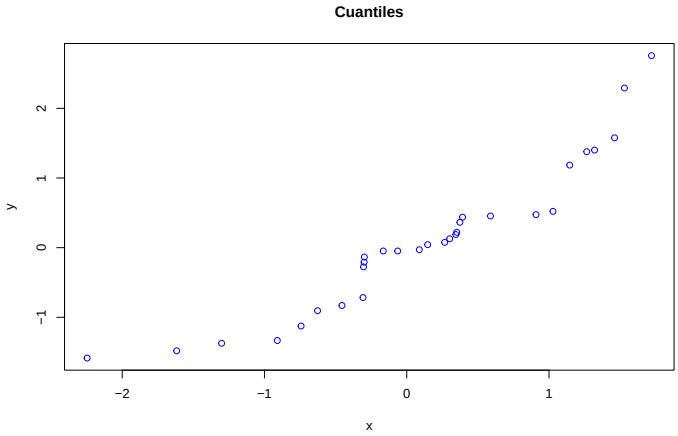


Figura 9.18: *qqplot*

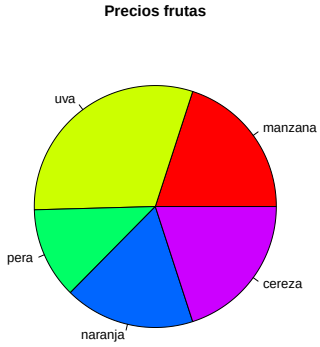


Figura 9.19: *Gráfico de tarta*

- `dotchart(x)`: si x es un `data.frame`, realiza gráficos apilados fila por fila y columna por columna.

```
data(cars)
with(cars, dotchart(speed))
```

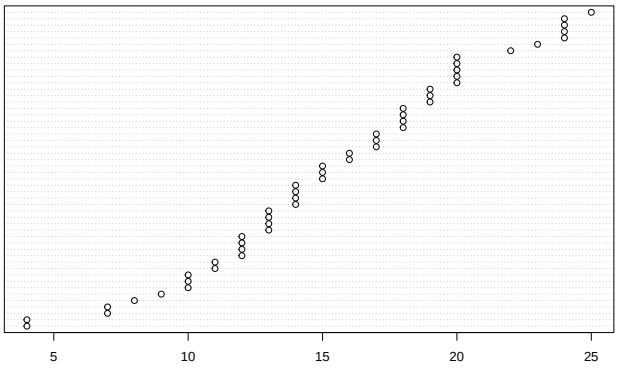


Figura 9.20: *Dotchart*

- `contour(x, y, z)`: gráfico de contornos (los datos son interpolados para dibujar las curvas), x e y deben ser vectores, z debe ser una matriz tal que `dim(z)=c(length(x), length(y))`.

```
x <- c(1, 3, 5, 7, 9)
y <- c(2, 4, 6, 8)
z <- matrix(runif(20, 1, 10), 5, 4)
contour(x, y, z)
```

- `filled.contour( x, y, z )`: igual que el anterior, pero las áreas entre contornos están coloreadas, y se dibuja una leyenda de colores.

```
filled.contour(x, y, z)
```

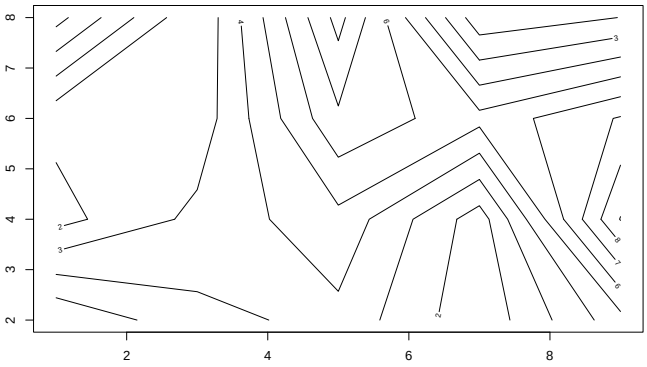


Figura 9.21: *Contour*

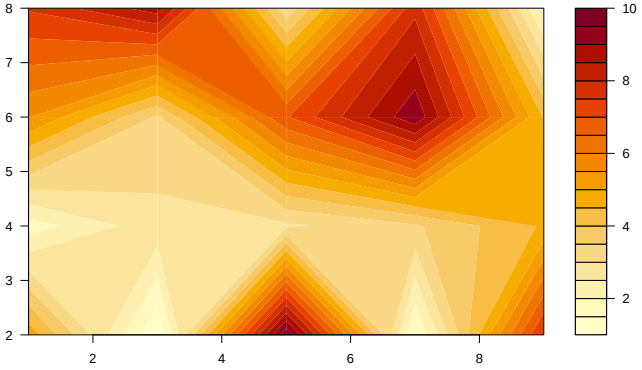


Figura 9.22: *Filled.contour*

9.2.4. Funciones de bajo nivel

Cuando tenemos ya lanzado un `device`, podemos añadirle una serie de "objetos" utilizando un conjunto de expresiones. Veamos los más importantes:

- `points( x, y )`: agrega puntos con coordenadas `(x,y)`

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Puntos")
points(0, 1.7, col = "red")
points(2, 2, col = "blue")
```

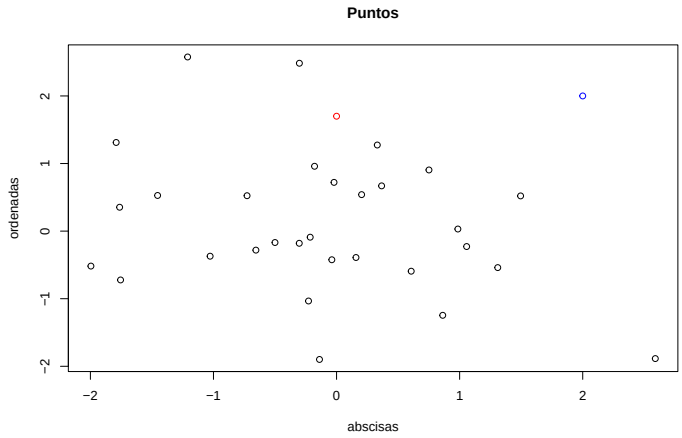


Figura 9.23: *Añadir puntos al plot*

- `lines( x, y )`: añade líneas en el vector de coordenadas `( x, y )`

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Líneas")
lines(c(1, 2), col = "blue")
```

- `text( x, y, labels, ...)`: agrega texto dado en labels (etiquetas) en las coordenadas `( x, y )`

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Texto")
text(-0.9, 1.5, "EFJ", col = "purple")
```

- `mtext( text, side, line, col, adj, ... )`
  - `side`: lado de la gráfica (1 = abajo, 2 = izquierda, 3 = arriba, 4 = derecha)

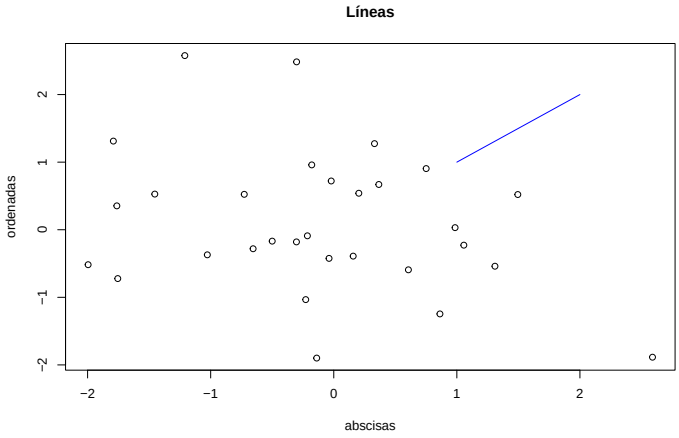


Figura 9.24: *Añadir líneas al plot*

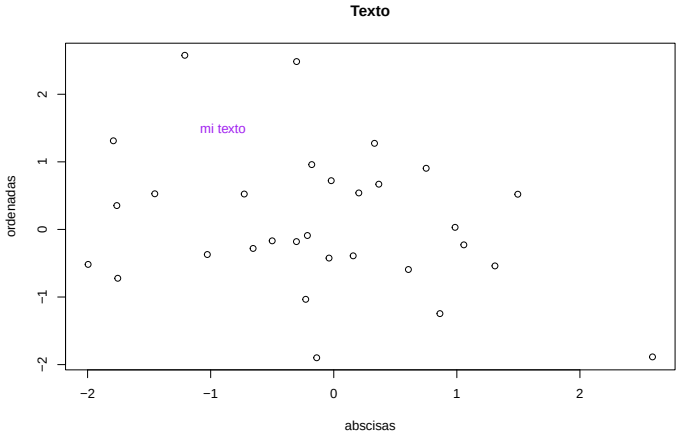


Figura 9.25: *Añadir texto al plot con text*

- `line`: línea de margen, empezando a contar de 0 hacia fuera
- `adj` : alínear el texto (0 = izquierda/abajo , 1 = arriba/derecha)

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Texto")
mtext("FEIR3", side = 3, col = "red", adj = 0)
```

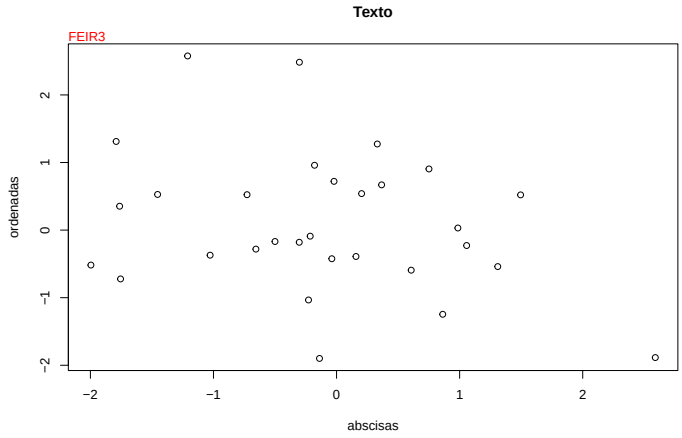


Figura 9.26: *Añadir texto al plot con mtext*

- `segments( x0, y0, x1, y1 )`: dibuja una línea desde el punto (`x0, y0`) a (`x1, y1`)

```
plot(x, y,xlab= "abscisas", ylab = "ordenadas", main= "Segmento",
             col= "purple",pch = 18)
segments(1, 2.5, 2.5, 0.5, col = "green2")
```

- `arrows( x0, y0, x1, y1 )`: igual que el anterior pero con flechas.

```
plot(x,y,xlab= "abscisas",ylab= "ordenadas", main = "Flecha", col= "blue")
arrows(1, 1, 2, 2, col = "red")
```

- `abline( a, b )`: dibuja una línea con pendiente b e intercepto a
- `abline( h = y )`: dibuja una línea horizontal en la ordenada y
- `abline( v = x )`: dibuja una línea vertical en la abscisa x

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Abline")
abline(h = 2, col = "red")
```

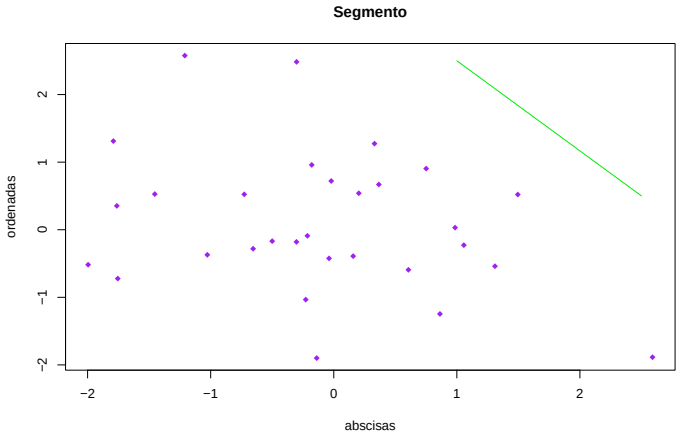


Figura 9.27: *Añadir segmentos al plot*

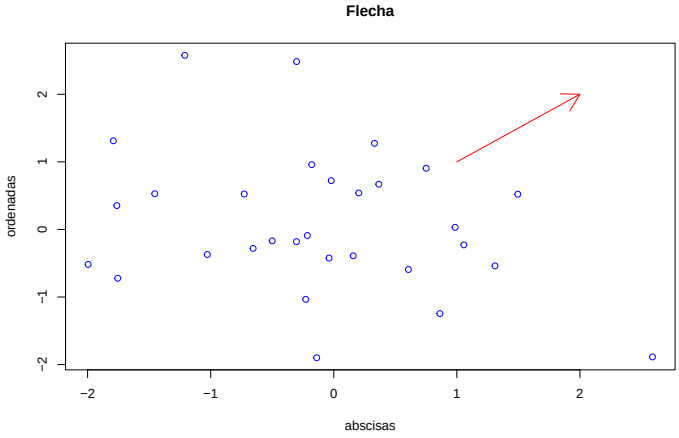


Figura 9.28: *Añadir flechas al plot*

```
abline(v = -1, col = "blue")
```

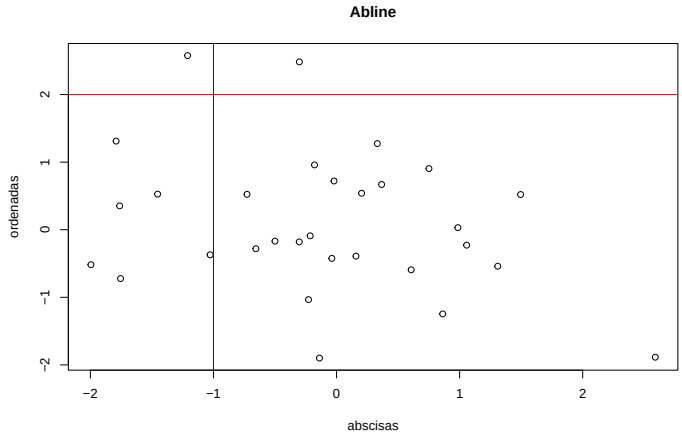


Figura 9.29: *Añadir línea horizontal y vertical al plot*

- `abline( modelo )`: dibuja la línea de regresión dada por `modelo`

```
modelo <- lm(x ~ y)
modelo
```

```
##
##  Call:
##  lm(formula = x ~ y)
##
##  Coefficients:
##  (Intercept)            y
##     -0.09172     -0.29505
```

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Recta regresión")
abline(modelo, col = "red")
```

- `rect( x1, y1, x2, y2 )`: dibuja un rectángulo donde las esquinas son x1 = izquierda, y1 = inferior, x2 = derecha, y2 = superior.

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Rectángulo")
rect(1.6, 2, 2.4, 1.3, border = "blue")
rect(-0.9, 1.9, -0.5, 1.2, col = "magenta")
```

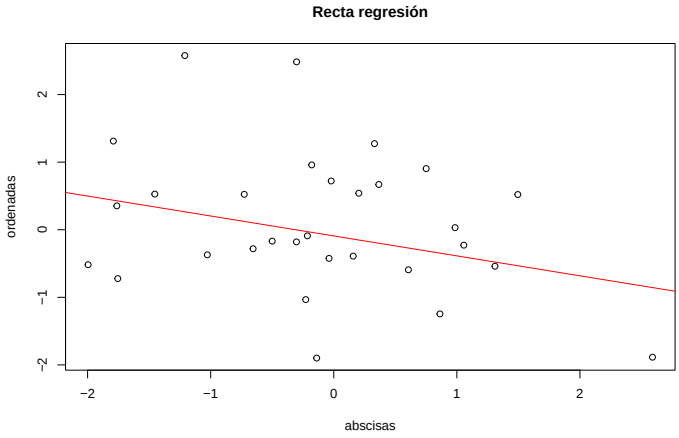


Figura 9.30: *Añadir la recta de regresión al plot*

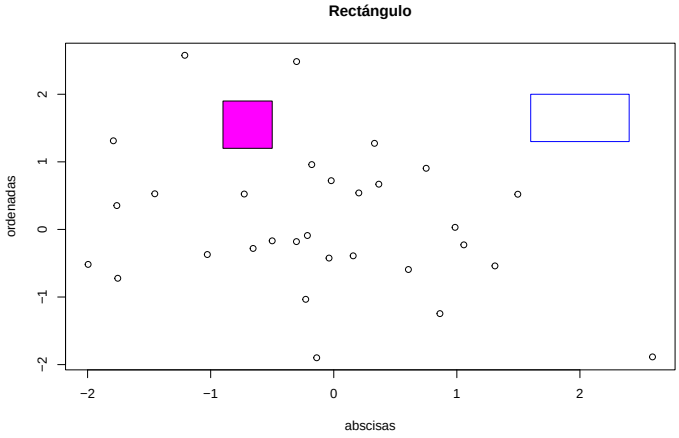


Figura 9.31: *Añadir rectángulos al plot*

- `title( )`: agrega todas las etiquetas al gráfico
  - main = "título principal"
  - sub = "sub-título"
  - xlab = "etiqueta eje x"
  - ylab = "etiqueta eje y"

- `axis( side, at=, labels=, pos=, lty=, col=, las=, tck=, ...)`
  - `side`: lugar del gráfico donde dibujar el eje (1=abajo, 2=izquierda, 3=arriba, 4=derecha)
  - `at`: vector numérico indicando dónde se deben dibujar las marcas
  - `labels` vector de caracteres con las etiquetas
  - pos las coordenadas donde se dibujará la línea del eje

```
plot(x, y, xlab = "", ylab = "", main = "")
axis(side = 3)
axis(side = 4)
```

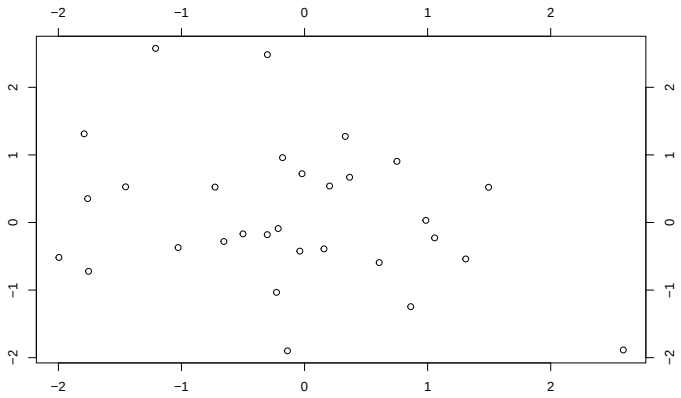


Figura 9.32: *Gráfico con todos los ejes etiquetados*

También podemos añadir expresiones matemáticas a una gráfica con el comando `text( x, y, expression( ) )`, donde la función expresión transforma su argumento en una ecuación matemática.

```
plot(x, y, xlab = "abscisas", ylab = "ordenadas", main = "Expresión")
eq <- expression(beta == over(1, sqrt(1 + x^2)))
text(1.3, 1.9, labels = eq, col = "blue")
```

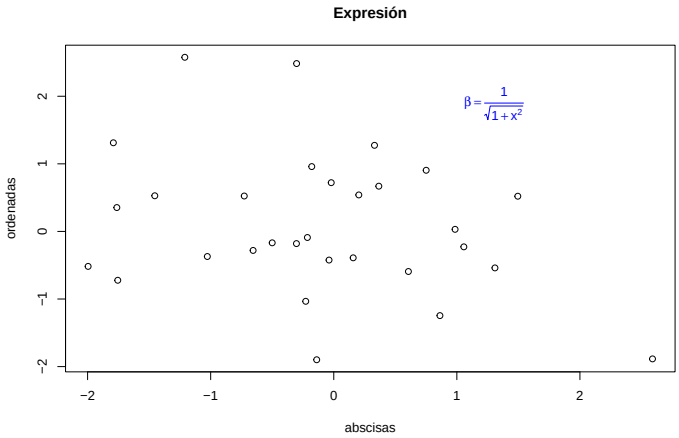


Figura 9.33: *Gráfico con expresión matemática*

- `legend( posición, titulo, leyenda, ... )` Otras opciones incluyen la leyenda son `bty` para el tipo caja, `bg` para color de fondo, `cex` para el tamaño, y `col` para el color del texto. Con la opción `horiz = TRUE` la leyenda horizontalmente en lugar de verticalmente.

```
data(mtcars)
df <- mtcars
boxplot(df$mpg ~ df$cyl, main = "Boxplot", yaxt = "n",
        xlab = "Millas", horizontal = TRUE,
    col = terrain.colors(3))
legend("topright", inset = 0.05, title = "Cilindros",
        c("4", "6", "8"), fill = terrain.colors(3),
    horiz = TRUE)
```

9.2.5. Parámetros gráficos

Podemos personalizar muchas características de nuestros gráficos (colores, ejes, títulos, fuentes de letra,...) a través de parámetros gráficos adicionales.

Una forma de especificar estas opciones es a través de la función `par()`. Si establecemos valores para los parámetros aquí, los cambios estarán en vigor durante el resto de la sesión o hasta que vuelvan a cambiar.

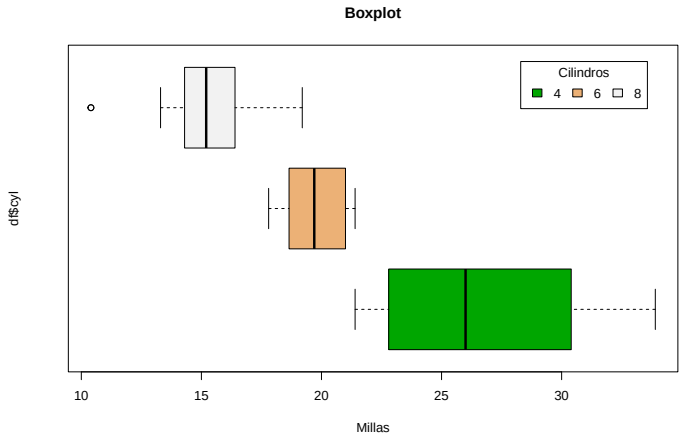


Figura 9.34: *Leyenda para boxplots*

La función toma la forma `par(nombreOpcion = valor, nombreOpcion = valor, ...)`.

```
par()  # vemos la configuración actual

opar <- par()  # hacemos una copia de la configuración actual
par(col.lab = "blue")  # etiquetas de x e y azules
hist(mtcars$mpg, main = "Histograma")  # histograma con las nuevas opciones
rug(mtcars$mpg)

par(opar)  # restaura las opciones originales
```

Una segunda opción para especificar parámetros gráficos es añadiendo los parámetros `OptionName = valor` directamente a los argumentos de una función de alto nivel. En este caso, las opciones sólo afectan a ese gráfico específico.

```
# Establecer un parámetro dentro de la función gráfica
hist(mtcars$mpg, col.lab = "red", col = "blue", main = "Histograma")
rug(mtcars$mpg)
```

Vemos a continuación las principales posibilidades que nos ofrece esta función.

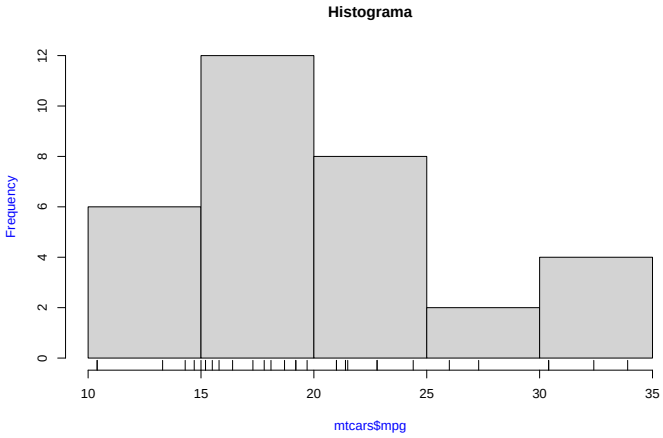


Figura 9.35: *Etiquetas de colores definidas con par()*

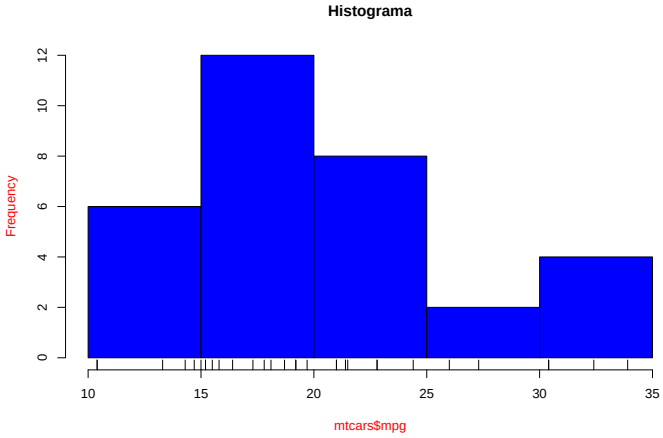


Figura 9.36: *Colores del gráfico definidos en la propia función*

9.2.5. Tamaño del texto y los símbolos

Podemos utilizar las siguientes opciones para controlar el tamaño del texto y de los símbolos en los gráficos.

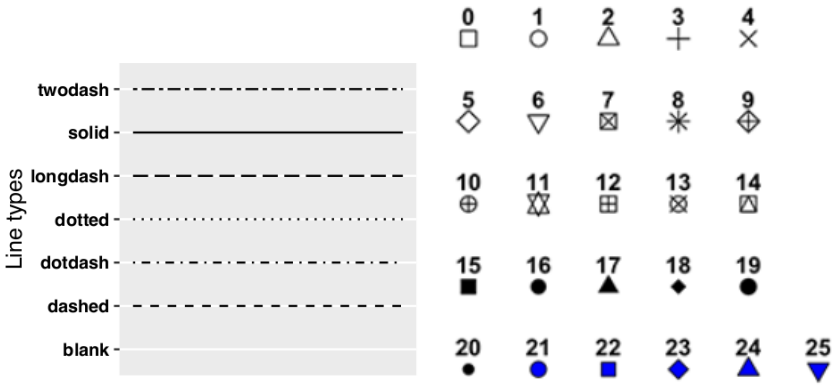
- `cex`: valor que escala el tamaño del texto y de los símbolos con respecto al valor por defecto. El valor por defecto es 1, un 1.5 significa un 50% más grande y 0.5 es un 50% más pequeño.
- `cex.axis`: aumento de los ejes con respecto al `cex`
- `cex.lab`: incremento de las etiquetas de x e y en relación a `cex`.
- `cex.main`: magnificación de los títulos relativos al `cex`.
- `cex.sub`: ampliación del subtitulo con respecto al `cex`.

9.2.5. Símbolos

Podemos personalizar las opciones de los puntos.

- `pch`: controlar el tipo de símbolo para trazar los puntos con un entero entre 1 y 25.
- `col`: determinar el color del borde
- `bg`: modificar el color de relleno

Los valores que los argumentos `lty` y `pch` pueden tomar son estos:



```
# Establecer un parámetro dentro de la función gráfica
plot(x, y, pch = 15, bg = 7, col = "blue", main = "Símbolos")
```

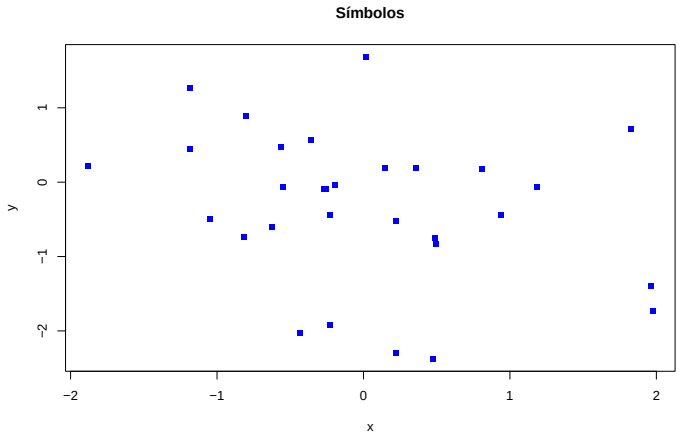


Figura 9.37: *Gráfico con puntos del símbolo 15*

9.2.5. Líneas

Podemos cambiar el trazado de las líneas mediante las siguientes opciones.

- `lty`: controla el tipo de las líneas
  - 1: solida
  - 2: quebrada
  - 3: punteada
  - 4: punto-línea
  - 5: línea larga-corta
  - 6: dos líneas cortas
- `lwd`: determinar el ancho de línea en relación con el valor por defecto (defecto=1)

```
plot(x, y, main = "líneas", type = "n")
abline(v = -1, lty = 1, col = "magenta")
abline(h = -1, lty = 2, col = "red")
abline(-1, 2, lty = 3, col = "blue", lwd = 2)
```

```r
abline(v = 1, lty = 5, col = "green3")
abline(1, -1, lty = 6, col = "purple", lwd = 2)
```

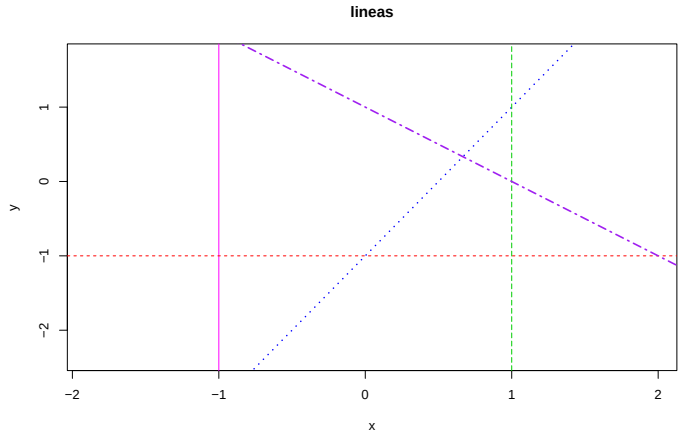


Figura 9.38: *Distintos trazados de línea*

9.2.5. Colores

- `col`: color por defecto del trazado
- `col.axis`: color para los ejes
- `col.lab`: color de las etiqueta de los ejes
- `col.main`: color del título principal
- `col.sub`: color para los subtítulos
- `fg`: color para el primer plano del gráfico (ejes, cajas) *bg: color de fondo

En R podemos especificar los colores mediante el índice, el nombre, código hexadecimal o RGB. Por ejemplo `col = 1`, `col = "white"`, y `col = "# FFFFFF"` son equivalentes.

Podemos ver toda la carta de colores con la función `colors()`, que devuelve el nombre de todos los colores disponibles.

```r
# Establecer un parámetro dentro de la función gráfica
hist(mtcars$mpg, col.lab = "red", col = "blue", col.axis = "brown",
        main = "Histograma")
rug(mtcars$mpg)
```

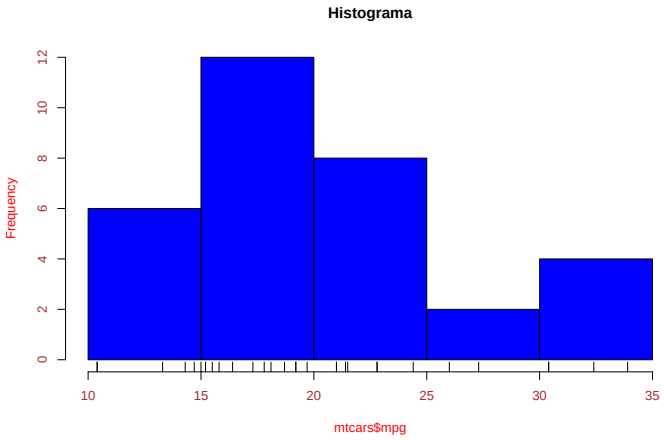


Figura 9.39: *Colores en los distintos elementos del gráfico*

También podemos crear un vector de n colores contiguos utilizando las funciones `rainbow( n )`, `heat.colors( n )`, `terrain.colors( n )`, `topo.colors( n )`, and `cm.colors( n )`.

9.2.5. Fuentes

Podemos configurar fácilmente el tamaño y el estilo de la fuente de la letra

- `font`: especifica la fuente para el texto
  - 1: texto plano
  - 2: negrita
  - 3: cursiva
  - 4 cursiva-negrita
- `font.axis`: fuente para los ejes
- `font.lab`: fuente para las etiquetas
- `font.main`: fuente para los títulos
- `font.sub`: fuente para los subtítulos

```
x <- c(5, 13, 9)
barplot(x, font = 4, font.main = 1, main = "Gráfico de barras", ,
        names.arg = c("mat ",
    "len", "ing"))
```

Figura 9.40: *Fuentes de texto en el gráfico*

9.2.5. Otras opciones

- `adj`: posición del texto
  - `adj=0` ajustado a la izquierda
  - `adj=0.5` texto centrado
  - `adj=1` ajustado a la derecha.
- `bty`: tipo de caja que rodea los gráficos. Las opciones son "`o`", "`l`", "`7`", "`c`", "`u`", "`]`" y "`n`" (suprime la caja).
- `las`: especifica la posición de las etiquetas de los ejes
  - 0: paralelo al eje (por defecto)
  - 1: horizontal
  - 2: perpendicular al eje
  - 3: vertical.
- `mar`: vector numérico que controla el espacio entre los ejes y el borde de la gráfica. Tiene la forma `c( inferior, izquierda, superior, derecha )` y los valores por defecto son `c( 5, 4, 4, 2 ) + 0.1`.
- `mfcol`: vector del tipo `c( nf, nc )` que divide la ventana gráfica en una matriz cn nf filas y nc columnas. Las gráficas se van dibujando sucesivamente

por columnas.

- `mfrow`: igual que el anterior pero las gráficas se dibujan por filas.
- `xaxt`: si `xaxt = "n"` el eje x se coloca pero no se muestra.
- `yaxt`: si `yaxt= "n"` el eje y se coloca pero no se muestra.

Veamos un ejemplo con la función `par( )`

```
x <- rnorm(20)
y <- rnorm(20)
par(col.axis = "blue", mar = c(4, 4, 2.5, 2.5), font = 2)
plot(x, y, xlab = "abscisas", ylab = "ordenadas", xlim = c(-2, 2),
ylim = c(-2, 2),
    pch = 20, col = "red", bty = "l", cex = 1.5)
title("Gráfico en R", font.main = 1, adj = 0)
```

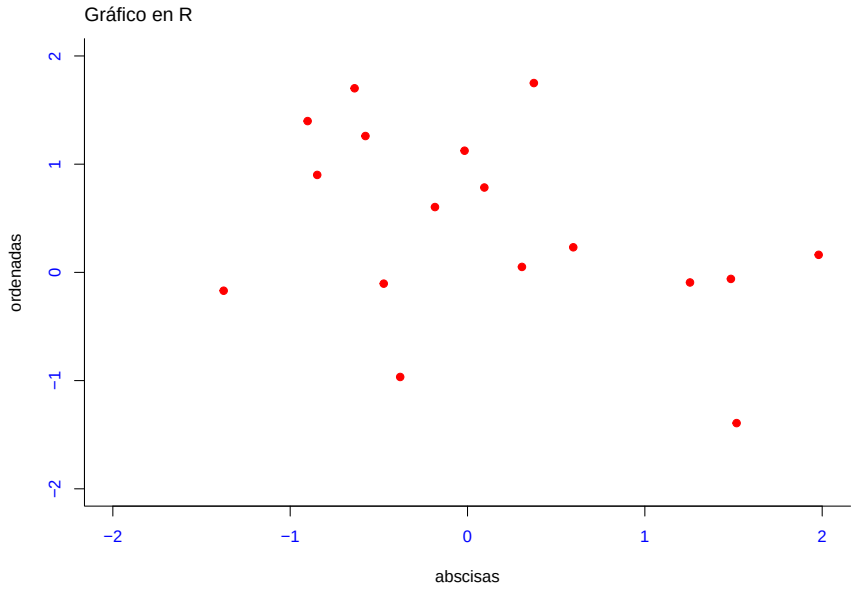


Figura 9.41: *Ejemplo de gráfico con la función par()*

9.2.6. Combinación de gráficos

Podemos combinar varias parcelas en una gráfica global, utilizando las funciones `par( )` o `layout( )`.

Con la función `par( )` podemos incluir la opción `mfrow = c( nfilas, ncolumnas )` para crear una matriz de nfilas x ncolumas gráficos que se introducen por filas. `mfcol = c( nfilas, ncolumas )` rellena la matriz por columnas.

Vemos primero un ejemplo con `mfrow = c( nfilas, ncolumnas )`

```
# 4 imagens ordenadas en 2 filas y 2 columnas
df <- mtcars
par(mfrow = c(2, 2))
plot(df$wt, df$mpg, main = "wt vs. mpg")
plot(df$wt, df$disp, main = "wt vs disp")
hist(df$wt, main = "Histograma de wt")
boxplot(df$wt, main = "Boxplot de wt")
```

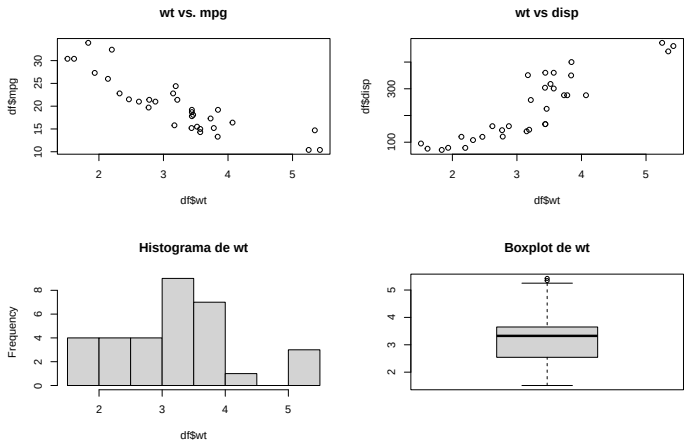


Figura 9.42: *Combinación de gráficos en grid 2×2*

y ahora otro con `mfcol = c(nfilas, ncolumas)`

```
# 3 figures arranged in 3 rows and 1 column
par(mfrow = c(3, 1))
hist(df$wt)
```

```
hist(df$mpg)
hist(df$disp)
```

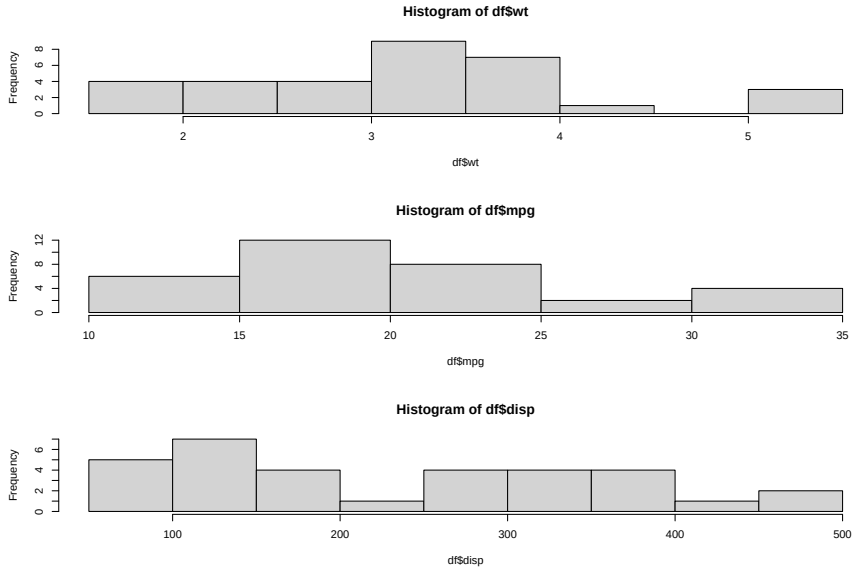


Figura 9.43: *Combinación de histogramas en grid 3×1*

La función `layout()` divide el dispositivo activo en varias partes donde se colocarán las gráficas de manera sucesiva. Esta función tiene como argumento principal una matriz de enteros indicando el número de figuras del gráfico.

```
# Una figura en la fila 1 y dos figuras en la fila 2
layout(matrix(c(1, 1, 2, 3), 2, 2, byrow = TRUE))
hist(df$wt)
hist(df$mpg)
hist(df$disp)
```

Por defecto `layout()` divide el dispositivo en dimensiones regulares, pero podemos incluir las opciones anchura y altura en la función de `layout()` para controlar el tamaño de cada figura con mayor precisión. Estas opciones tienen la forma

- `widths` = un vector de valores de las anchuras de las columnas
- `heights` = un vector de valores de las alturas de las filas.

```
dev.off()
```

```
##  null device
##            1
```

```
# Una figura en la fila 1 y dos figuras en la fila 2 La fila 1 tiene 1/3
# de la altura de 2 La columna 2 tiene 1/4 de la anchura de la 1
layout(matrix(c(1, 1, 2, 3), 2, 2, byrow = TRUE),

                widths = c(3, 1), heights = c(1,
    2))
hist(df$wt)
hist(df$mpg)
hist(df$disp)
```

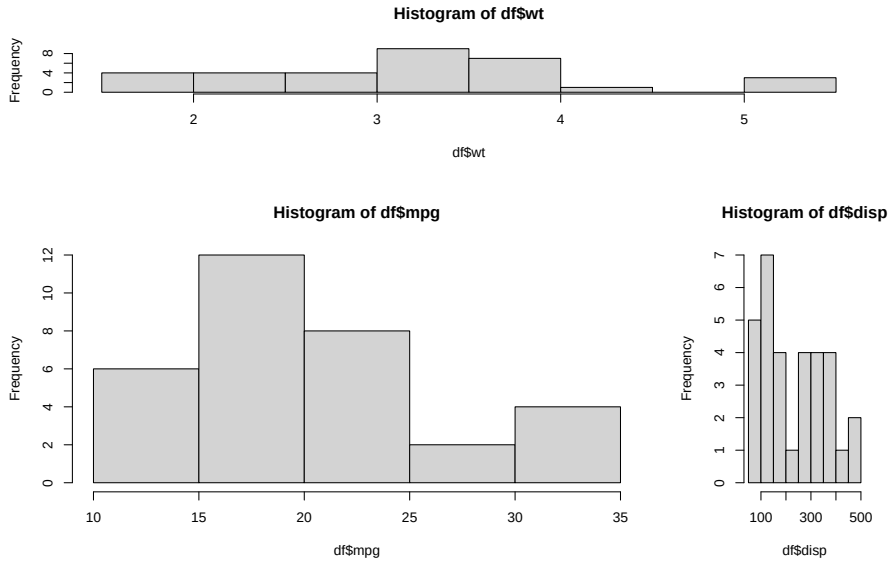


Figura 9.44: *Combinación de histogramas con layout()*

9.2.7. Gráficos más complejos

En este apartado hemos hecho uso tanto de [62] como de [64] para el desarrollo del mismo.

9.2.7. Gráfico de dispersión 3D

```
library(datasets)
data(airquality)
df <- airquality[airquality$Month == c(5, 6, 7), ]

library(scatterplot3d)
colores <- factor(df$Month)
scatterplot3d(df$Wind, df$Ozone, df$Solar.R, color = colores, pch = 18,
              main ="Gráfica de dispersión 3D")
```

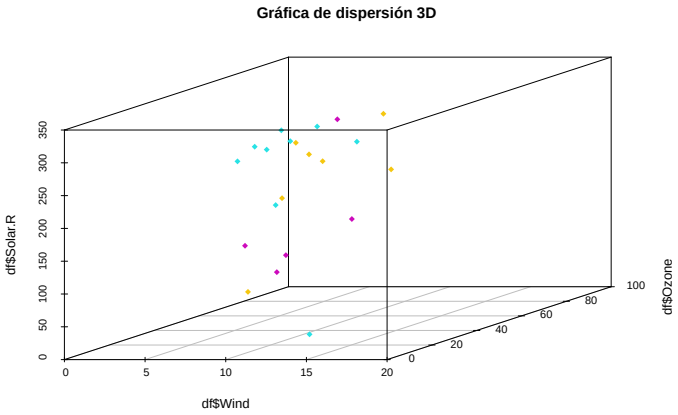


Figura 9.45: *Gráfica de dispersión scatterplot3d()*

9.2.7. Gráfico de barras horizontales

Mostramos directamente el código

```
groups <- c("cows", "sheep", "horses", "elephants", "giraffes")
males <- sample(1:10, 5)
females <- sample(1:10, 5)  # muestras aleatorias

par(mar = c(0.5, 5, 0.5, 1))  # margenes

plot.new()
```

```
plot.window(xlim = c(-10, 10), ylim = c(-1.5, 5.5))

ticks <- seq(-10, 10, 5)
y <- 1:5
h <- 0.2

lines(rep(0, 2), c(-1.5, 5.5), col = "grey")
segments(-10, y, 10, y, lty = "dotted")
rect(-males, y - h, 0, y + h, col = "dark grey")
rect(0, y - h, females, y + h, col = "light grey")
mtext(groups, at = y, adj = 1, side = 2, las = 2)
par(cex.axis = 0.5, mex = 0.5)
axis(1, at = ticks, labels = abs(ticks), pos = 0)

tw <- 1.5 * strwidth("females")
rect(-tw, -1 - h, 0, -1 + h, col = "dark grey")
rect(0, -1 - h, tw, -1 + h, col = "light grey")
text(0, -1, "males", pos = 2)
text(0, -1, "females", pos = 4)

box("inner", col = "grey")
```

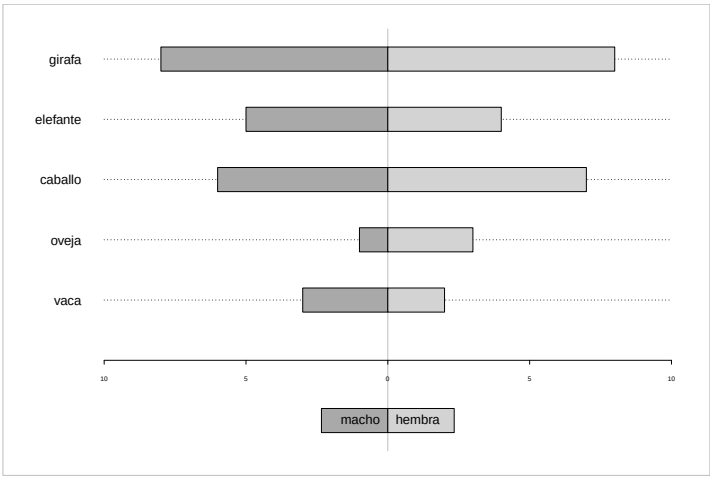


Figura 9.46: *Gráfico de barras horizontales*

9.2.7. Etiquetar casos

```
df <- LifeCycleSavings[1:9, ]
plot(df$sr ~ df$dpi, xlim = range(df$dpi), col = "red", pch = 18,
        xlab = "Renta",
    ylab = "Ahorros", main = "Ciclo de ahorros", data = df)

text(df$sr ~ df$dpi, labels = row.names(df), pos = 4, col = "blue")
```

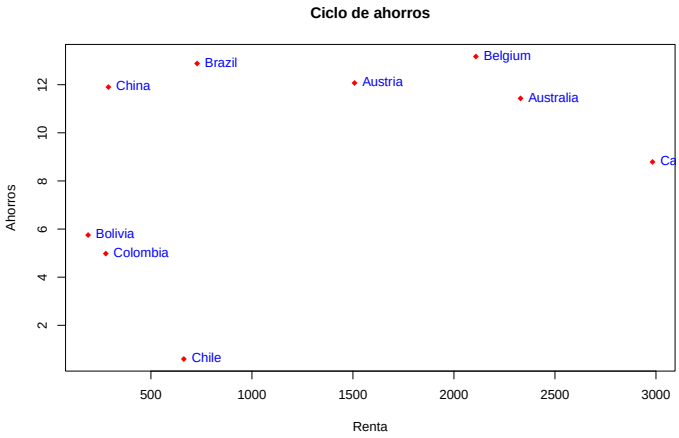


Figura 9.47: *Casos etiquetados*

9.2.7. Histograma con normal

```
# Add a Normal Curve (Thanks to Peter Dalgaard)
x <- mtcars$mpg
h <- hist(x, breaks = 10, col = "blue", xlab = "Velocidad",
        main = "Histograma curva Normal")
xfit <- seq(min(x), max(x), length = 40)
yfit <- dnorm(xfit, mean = mean(x), sd = sd(x))
yfit <- yfit * diff(h$mids[1:2]) * length(x)
lines(xfit, yfit, col = "red", lwd = 2)
```

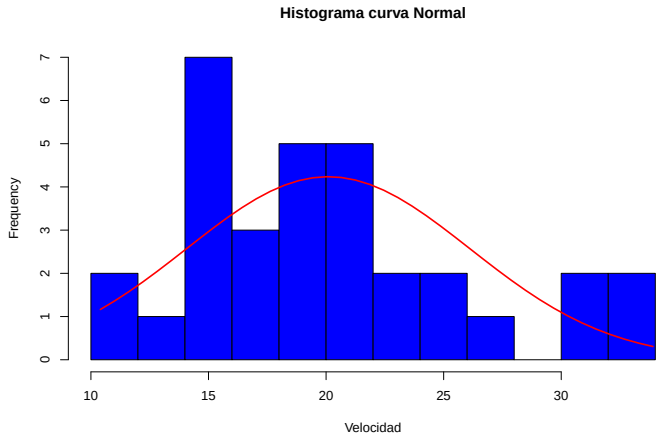


Figura 9.48: *Histograma con normal*

9.2.7. Histograma con boxplot

```
set.seed(4566)
df <- rnorm(100)
layout(mat = matrix(c(1, 2), 2, 1, byrow = TRUE), height = c(3, 1))
par(mar = c(3.1, 3.1, 1.1, 2.1))
hist(df, xlim = c(-4, 4), col = "pink")
boxplot(df, horizontal = TRUE, outline = TRUE, ylim = c(-4, 4),
        frame = FALSE, col = "green1", width = 10)
```

Ponemos el `boxplot` dentro del gráfico del histograma

```
par(mar = c(3.1, 3.1, 1.1, 2.1))
hist(df, xlim = c(-4, 4), col = "pink")
boxplot(df, horizontal = TRUE, outline = TRUE, ylim = c(-4, 4),
        frame = FALSE, col = "green1",
    add = TRUE)
```

Otra

```
# Add boxplots to a scatterplot
par( fig = c( 0, 0.8, 0, 0.8 ), new = TRUE )
plot( mtcars$wt, mtcars$mpg, xlab="Peso", ylab = "Velocidad" )
```

```
par( fig = c( 0, 0.8, 0.55, 1 ), new = TRUE )
boxplot( mtcars$wt, horizontal = TRUE, axes = FALSE )
par( fig = c( 0.65, 1, 0, 0.8), new = TRUE )
boxplot( mtcars$mpg, axes = FALSE )

mtext("Dispersión con boxplot", side = 3, outer = TRUE, line = -3 )
```

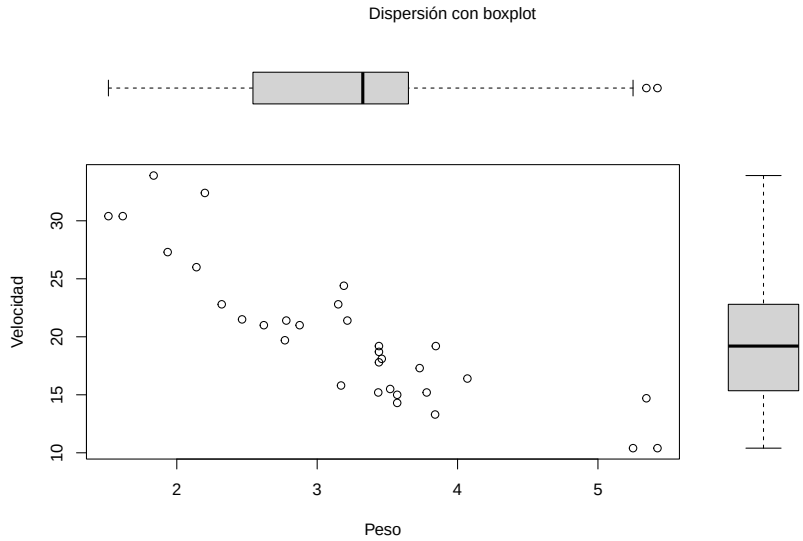


Figura 9.49: *Histograma con Boxplot*

9.3. Ggplot2

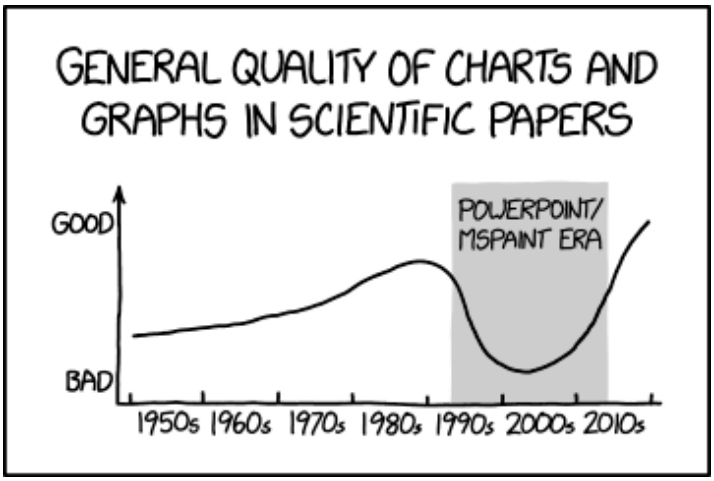
A diferencia de los gráficos con el paquete base donde creamos un gráfico a base de pasos sucesivos, añadiendo elementos a un `graphical device` ggplot2 se basada en una gramática de gráficos compuesta por partes independientes que pueden ser compuestas de muchas maneras diferentes.

ggplot2 fracciona un gráfico en tres partes fundamentales que son: `data`, `aesthetics` y `geometry`, las dos últimas se denominan `layers` (en general son elementos geométricos y transformaciones estadísticas). Esto nos permite crear gráficos de una forma interactiva, comenzando con los datos que queremos re-

presentar y añadiéndole sucesivamente `layers` que completan y dan forma al gráfico.

Como dice Wickhman: `reducir la distancia entre el gráfico pensado y el que estás creando en tu script`, nos permite crear gráficos utilizando la misma estructura de pensamiento que empleamos para diseñar un análisis. También nos obliga a pensar un gráfico antes de crearlo, pero esto mismo ocurre cuando queremos llevar a cabo un análisis.

Nota: La gg del nombre de ggplot2 viene de `Grammar of Graphics`.



9.3.1. La función `ggplot()`

```
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

- `data`: un `data frame`
- `aes`, aesthetics: se emplea para indicar los ejes x e y. También para controlar colores, tamaños, formas, alturas, etc....
- `geometry`: aquí indicaremos el tipo de gráfico (histograma, box plot, líneas,... densidades, puntos,...)

Para crear un gráfico con ggplot lo primero es definir el conjunto de datos, el sistema de coordenadas y entonces aplicar una geometría determinada. Este sería el esquema más básico: datos, geometría y

sistema de coordenadas (por defecto el sistema de coordenadas es el cartesiano).

Vamos a verlo paso a paso. Trabajaremos con el dataframe `mtcars`, y consideraremos la variable `cyl` y `gear` como factores:

```
library(ggplot2)
data(mtcars)
df <- mtcars[, c("mpg", "cyl", "wt", "gear")]
df$cyl <- factor(df$cyl)
df$gear <- factor(df$gear)

head(df)
```

```
##                    mpg cyl    wt gear
##  Mazda RX4         21.0   6 2.620    4
##  Mazda RX4 Wag     21.0   6 2.875    4
##  Datsun 710        22.8   4 2.320    4
##  Hornet 4 Drive    21.4   6 3.215    3
##  Hornet Sportabout 18.7   8 3.440    3
##  Valiant           18.1   6 3.460    3
```

9.3.1. i. Datos

Queremos representar el peso (`wp`) frente/relacionada con las millas/galon (`mpg`).
En una primera aproximación definimos la parte del **origen de datos** y los ejes en la función `ggplot()` con el argumento `aes()`:

```
ggplot(data = mtcars, aes(x = wt, y = mpg))
```

Observamos como en `aes` establecemos los ejes, pero en ningún momento hemos especificado un `layer` de geometría, así que no producirá más que un marco general (ejes cartesianos), sin una `geometría` especifica.

Este comportamiento nos da una idea de la forma interactiva de trabajar con `ggplot2`.

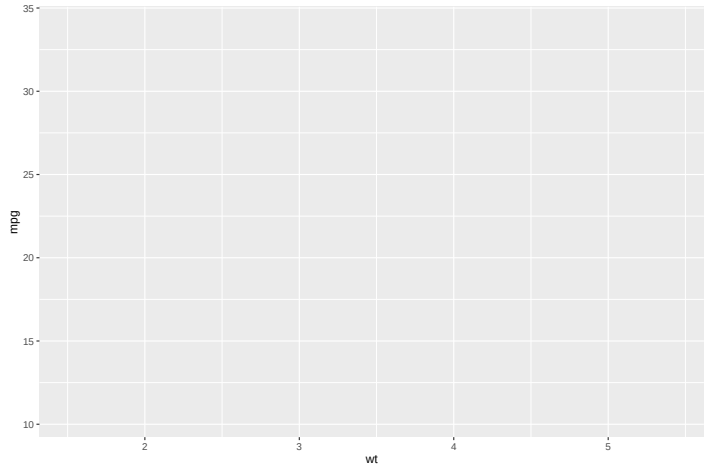


Figura 9.50: *Ejes cartesianos*

9.3.1. ii. Geometrías

Para añadir `layers` empleamos el signo + en la siguiente línea del layer:

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_point()
```

La función `geom_point()` admite muchos argumentos, como el tamaño de los puntos (`size`), la forma (`shape`), el color (`color`), etc..

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
        geom_point(size = 1.5, shape = 18, color = "red")
```

Nota: Hay `layers` más avanzados que hacen transformaciones de los datos: `stat_density()`, `stat_count()`,..

Prueba a emplear otros layers de geometrías como `geom_line()`:

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_line()
```

9.3.1. iii. Varias geometrías

Inlcuso podemos añadir más de un `layer` de geometría

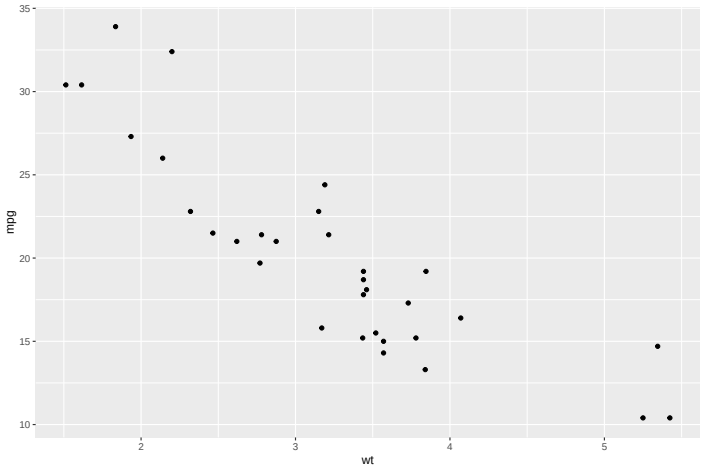


Figura 9.51: *Geom_point*

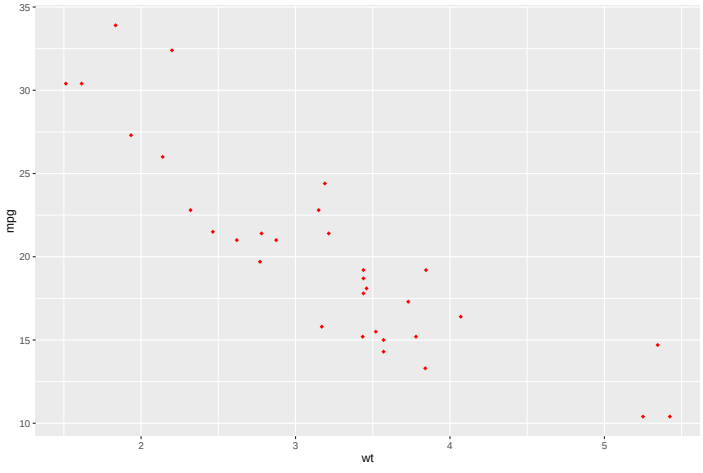


Figura 9.52: *Geom_point con colores*

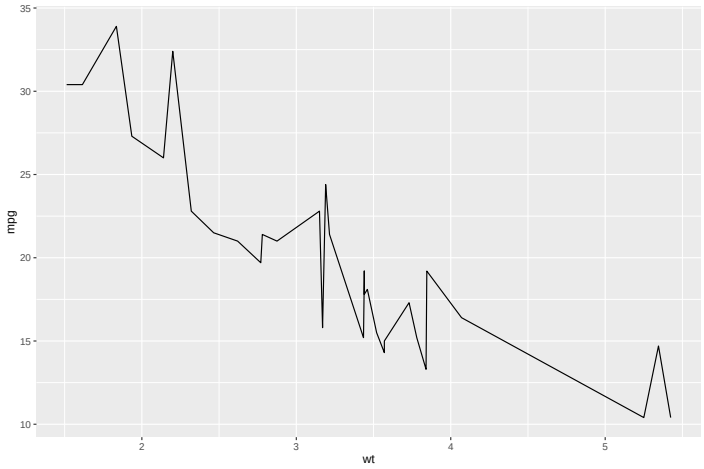


Figura 9.53: *Geom_line*

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_line(
        linetype = 2) + geom_point(col = "blue")
```

9.3.1. iv. Diferentes orígenes de datos

También es posible emplear diferentes orígenes de datos para cada layer. Aunque complica el gráfico y nos aleja del concepto principal, aquí presentamos un ejemplo:

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_point(
        col = "blue") + geom_line(data = head(mtcars),
    color = "red")
```

Observamos como la línea, `geom_line()` solo actúa sobre el subconjunto `head(mtcars)`.

También podemos hacer cálculos sencillos sobre las variables

```
ggplot(data = mtcars, aes(x = log(wt), y = log(mpg))) +
        geom_point(col = "blue")
```

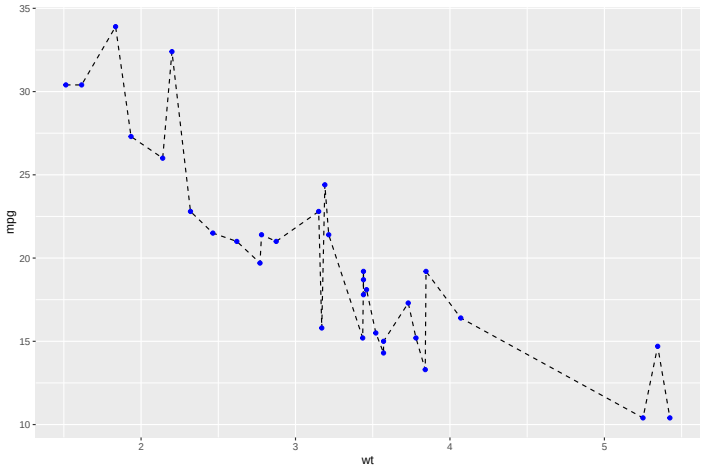


Figura 9.54: *Varios layers de geometrías*

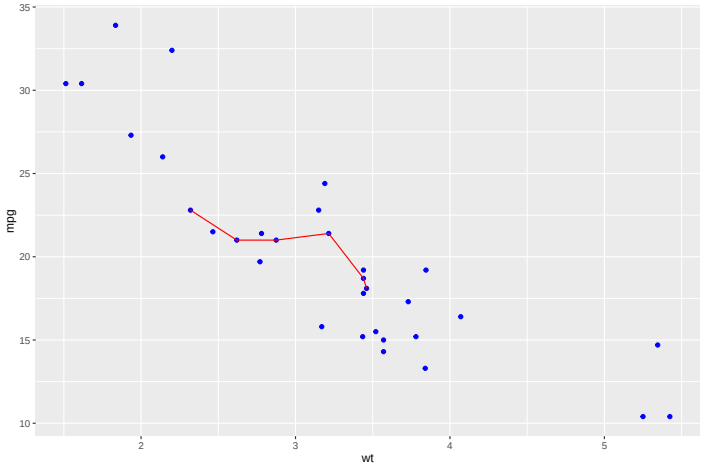


Figura 9.55: *Datos con fuentes diferentes*

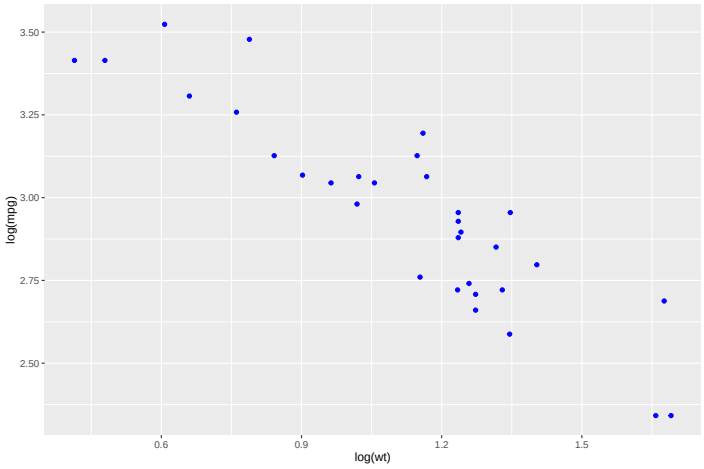


Figura 9.56: *Cálculos sobre las variables*

9.3.1. v. Sistemas de coordenadas

En ocasiones conviene parametrizar el sistema de coordenadas, por ejemplo para hacer zoom en una parte del gráfico (ver figura 9.57).

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
        coord_cartesian(xlim = c(2, 5)) + geom_line()
```

Vamos a limitar también el eje de OY entre 15 y 20:

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + coord_cartesian(xlim = c(2, 5),
          ylim = c(15,20)) + geom_line()
```

9.3.2. Refinando un gráfico: colour, shape y size

Podemos añadir más variables al gráfico introduciendo nuevos `atesthetics` en la llamada a `aes()`

Trabajamos con el conjunto de datos `mpg`;

```
data(mpg)
head(mpg)
```

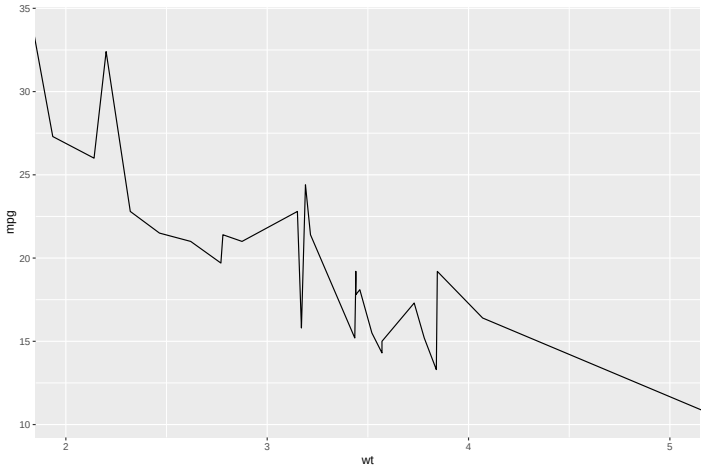


Figura 9.57: *Zoom en el eje OX*

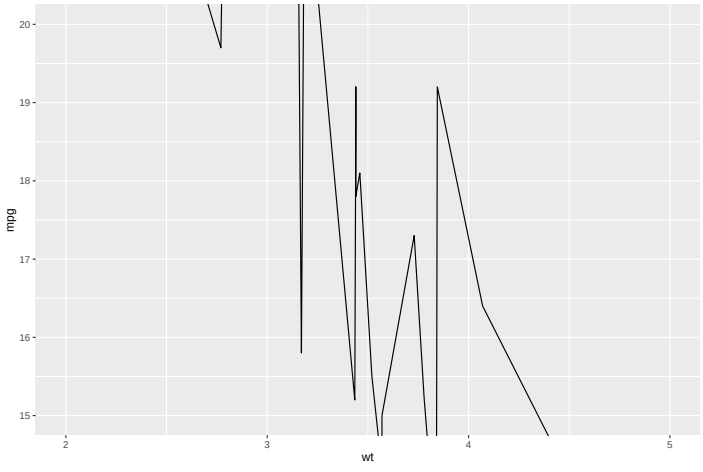


Figura 9.58: *Zoom en ambos ejes*

```
##  # A tibble: 6 x 11
##    manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##    <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
##  1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
##  2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
##  3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
##  4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
##  5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
##  6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

Por ejemplo, cada punto de un color según la clase (`class`) a la que pertenezca con `colour=class`:

```
ggplot(data = mpg, aes(cty, hwy, colour = class)) + geom_point()
```

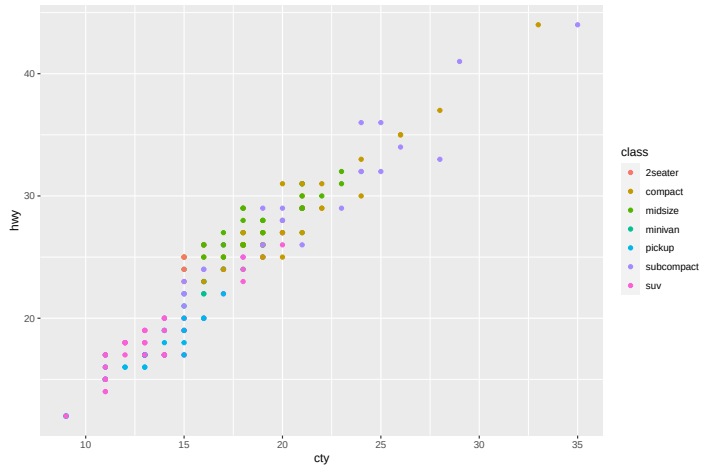


Figura 9.59: *Cada punto de un color*

Inlcuimos una forma según `drv` fijando `shape= drv`:

```
ggplot(data = mpg, aes(displ, hwy, colour = class, shape = drv))
        + geom_point()
```

más aún:

```
ggplot(data = mpg, aes(displ, hwy, colour = class, shape = drv, size = cyl))
        + geom_point()
```

Generalmente diferentes tipos de atributos estéticos trabajan mejor con ciertos tipos de variables, por ejemplo `color` y `shape` son apropiados para variables

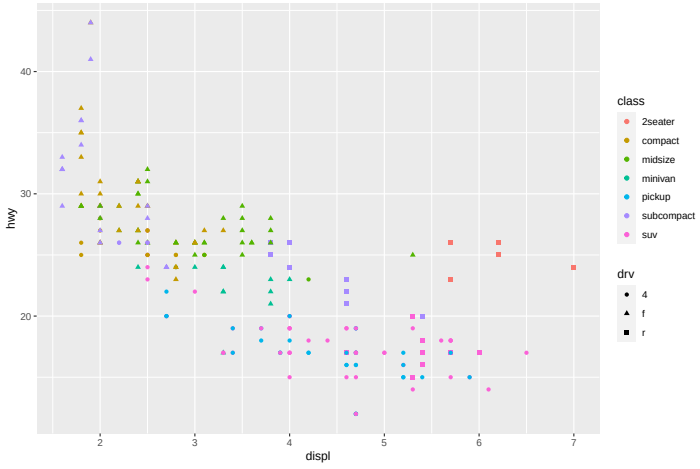


Figura 9.60: *Distintas formas de los puntos de acuerdo a una columna*

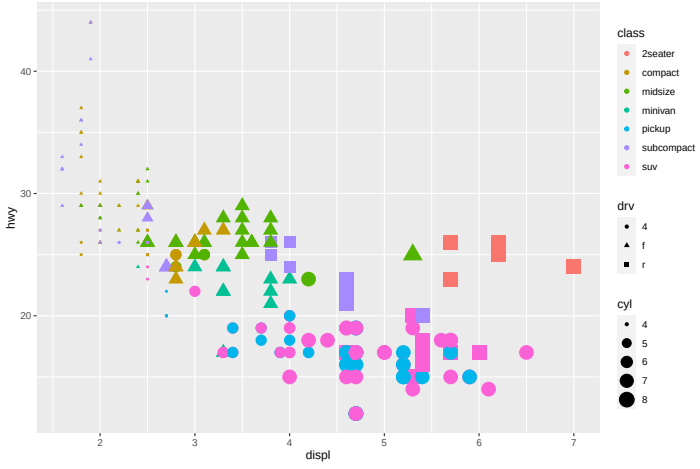


Figura 9.61: *Distintos colores, formas y tamaños*

categóricas, size con continuas.

Como dice Hadley Wickham cuando se utiliza elementos estéticos (aesthetics) en un gráfico, menos es normalmente más. Es difícil ver las relaciones simultáneas entre el color, la forma y el tamaño, por lo que hay que ser prudente a la hora de utilizar demasiados aesthetics simultaneamente. En lugar de tratar de hacer una gráfico muy complejo que muestre todo a la vez, hay que intentar quizás crear una serie de gráficos simples que cuenten una historia, llevando al lector de la ignorancia al conocimiento.

> when using aesthetics in a plot, less is usually more
> (Hadley Wickham)

Agrupamientos: Como estamos viendo, para añadir más información a un gráfico, en general lo que se hace es añadir más variables en los aesthetics como son: colour, shape, fill, linetype, fill, linetypes y más abruptamente con facets.

En ocasiones también podemos emplear el argumento group que suele requerir variables discretas para que ggplot particione con respecto a esa variable. Así la variable de group debe tener un valor diferente para cada grupo. Como decíamos antes diferentes argumentos trabajan mejor con diferentes geometrías y las geometrías se comportan mejor o peor dependiendo del tipo de datos: continuos, discretos, mixtos,...

Otro ejemplo:

```
ggplot(data = df, aes(mpg, wt, group = cyl, color = cyl)) + geom_line()
```

Observamos como un gráfico de líneas hace una línea diferente para cada "grupo". Las hemos coloreado para mejor entendimiento del ejemplo. Pero un efecto parecido podemos obtener jugando con el tipo de línea.

```
ggplot(data = df, aes(mpg, wt, linetype = cyl)) + geom_line()
```

9.3.3. Facetting

El enfoque facetting nos permite crear matrices de gráficos según una variable categórica. facetting creará una tabla de gráficos, un gráfico para cada

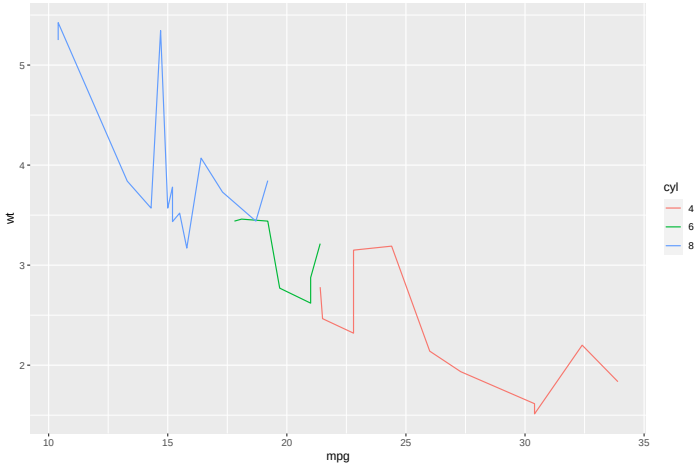


Figura 9.62: *Agrupamientos - colores*

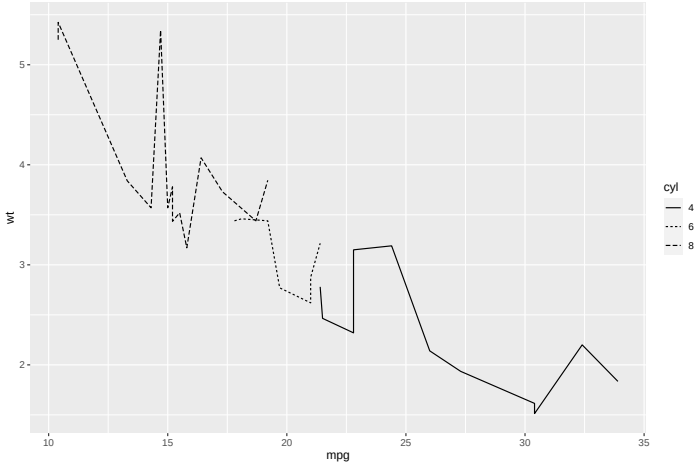


Figura 9.63: *Agrupamientos - tipo de línea*

subconjunto. El gráfico se divide en tantas columnas como niveles hay en la variable class con facet_grid(.~class)

```
ggplot(data = mpg, aes(displ, hwy)) + geom_point() + facet_grid(. ~ class)
```

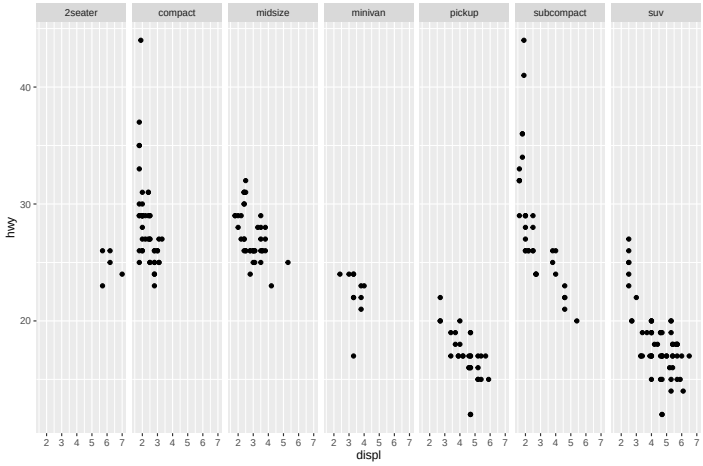


Figura 9.64: *facet_grid por columnas*

En tantas filas como `class` con `facet_grid(class~.)`

```
ggplot(data = mpg, aes(displ, hwy)) + geom_point() + facet_grid(class ~ .)
```

En tantas filas como `cyl` y columnas como `class` con `facet_grid(cyl~class)`

```
ggplot(data = mpg, aes(displ, hwy)) + geom_point() + facet_grid(cyl ~ class)
```

9.3.4. Customización plus

9.3.4. i. Líneas auxiliares

Podemos añadir líneas verticales u horizontales a un gráfico empleando el layer `geom_abline()` y sus derivados: `geom_hline()` y `geom_vline()`.

Añadimos una línea que pasa por el punto (0,10) y tiene pendiente m=3, otra vertical que pasa por el punto (3,0) y una horizontal que pasa por el punto (25,0)

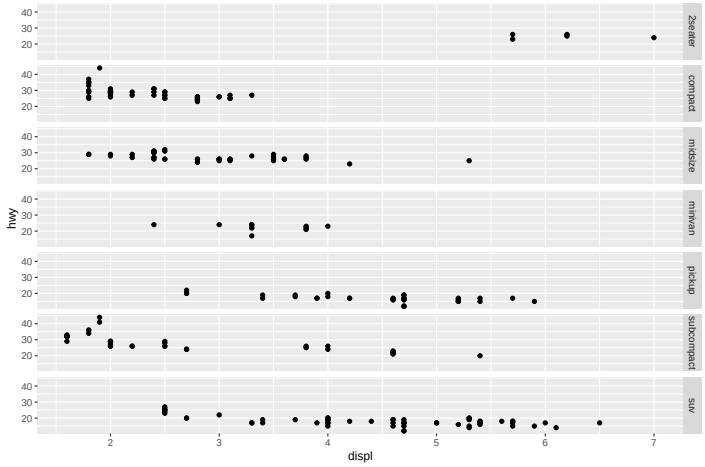


Figura 9.65: *facet_grid por filas*

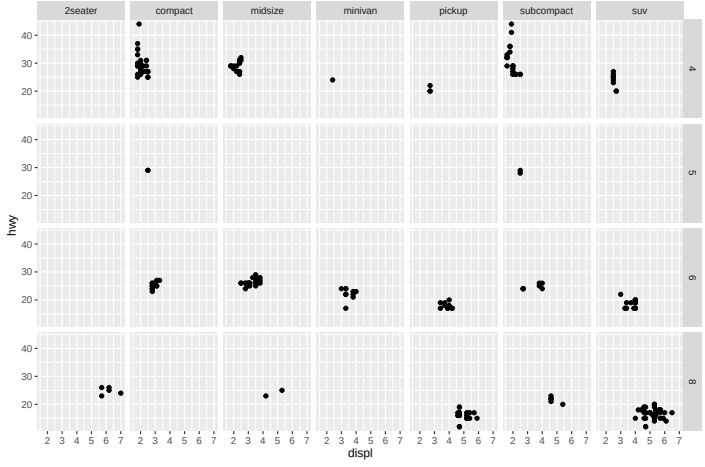


Figura 9.66: *facet_grid por filas y columnas*

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_line() +
geom_abline(aes(intercept = 10, slope = 3), col = "red", linetype = 2) +
geom_vline(aes(xintercept = 3), col = "blue", linetype = 3) +
geom_hline(aes(yintercept = 24), col = "purple", linetype = 4)
```

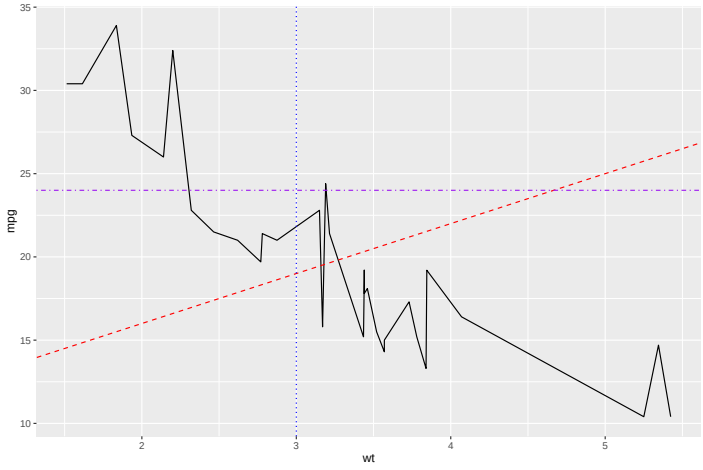


Figura 9.67: *Varias líneas*

9.3.4. Segmentos

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_line() +
geom_segment(aes(x = 2, y = 15, xend = 3, yend = 15))
```

Podemos rápidamente convertir un segmento en una flecha:

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) + geom_line() +
geom_segment(aes(x = 2, y = 15, xend = 3, yend = 15),
        arrow = arrow(length = unit(0.5, "cm")))
```

9.3.4. iii. Etiquetado de ejes

En ocasiones debemos de etiquetar el gráfico y los ejes. Para eso disponemos de funciones específicas que sobreescriben lo parametrizado por defecto.

- `p + ggtitle("Titulo principal")`: añade un título principal

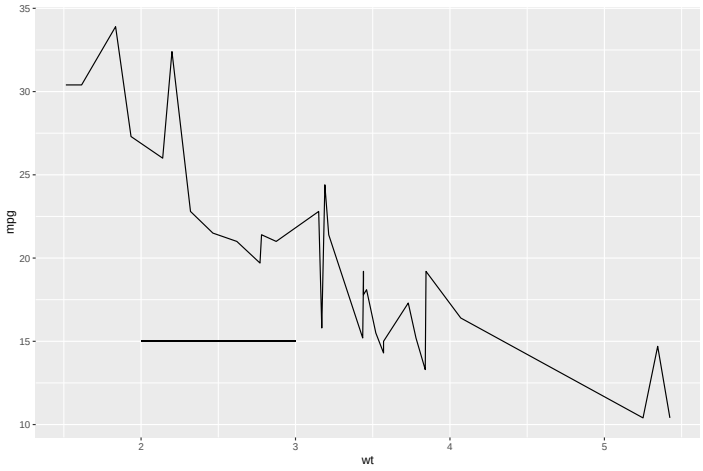


Figura 9.68: *Segmento*

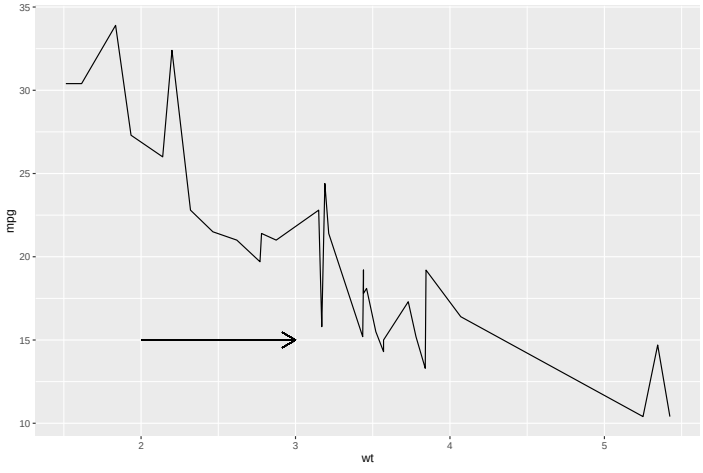


Figura 9.69: *Flecha*

- `p + xlab("Eje OX")`: Cambia solo la etiqueta del eje de las Xs
- `p + ylab("Eje OY")`: Cambia solo la etiqueta del eje de las Xs
- `p + labs(title = "Titulo", x = "Eje OX", y = "Eje OY")`: Todo a la misma vez (esta es la forma más usada)

Se puede obligar un salto de línea empleando "\n"

```
pp <- ggplot(data = df, aes(x = cyl, y = mpg, group = gear, fill = gear)) +
geom_boxplot()
pp
```

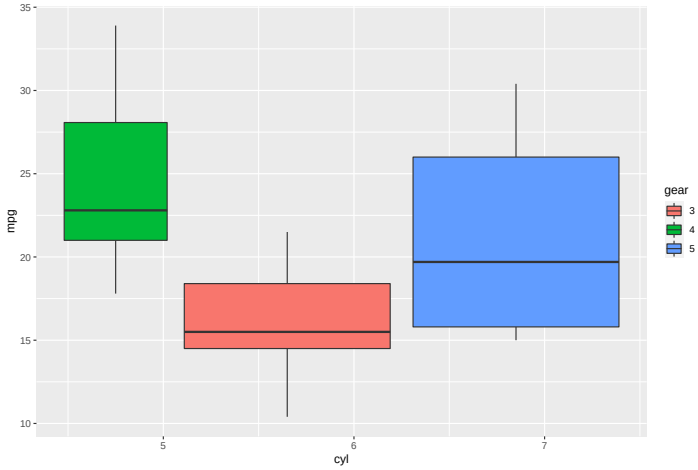


Figura 9.70: *Boxplot sencillo*

```
pp <- pp + labs(title = "Un gráfico cualquiera", x = "Eje OX: cyl"
, y = "Eje OY: mpg", subtitle = "Un subtitulo..")
pp
```

Podemos también cambiar el `tema` de los ejes, incluido el tipo de letra con `face`, que puede tomar los valores: "`plain`", "`italic`", "`bold`" y "`bold.italic`".

```
pp <- pp + theme(plot.title = element_text(color = "red", size = 12, face =
"bold.italic"), axis.title.x = element_text(color = "blue"
, size = 12, face = "bold"), axis.title.y = element_text(color = "#993333",
        size = 12, face = "italic"))
pp
```

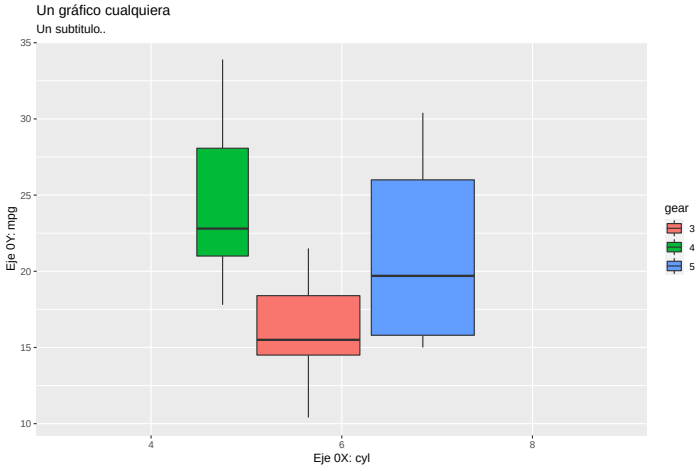


Figura 9.71: *Boxplot con subtítulo y ejes personalizados*

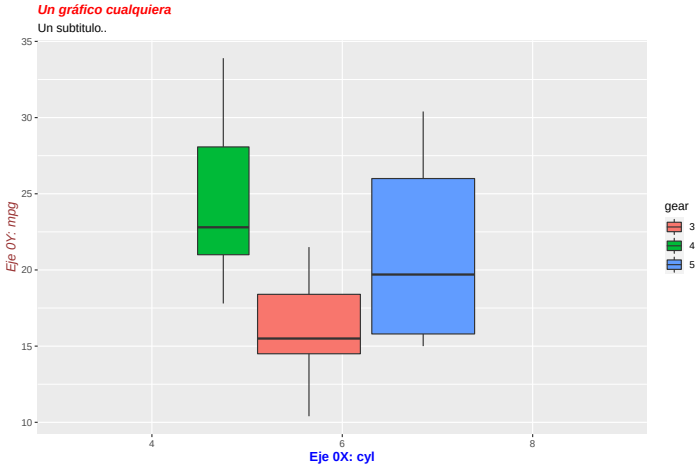


Figura 9.72: *Boxplot con subtítulo y ejes personalizados de colores*

Nota: `element_blank()` oculta la etiqueta cuando empleamos `theme()`

9.3.4. iv. Leyendas

Podemos modificar elementos de la leyenda que ggplot ofrece por defecto. Por ejemplo el título, puesto que crea la leyenda según la variable que pongamos en `fill`.

```
pp + labs(fill = "Nº de \nvelocidades")
```

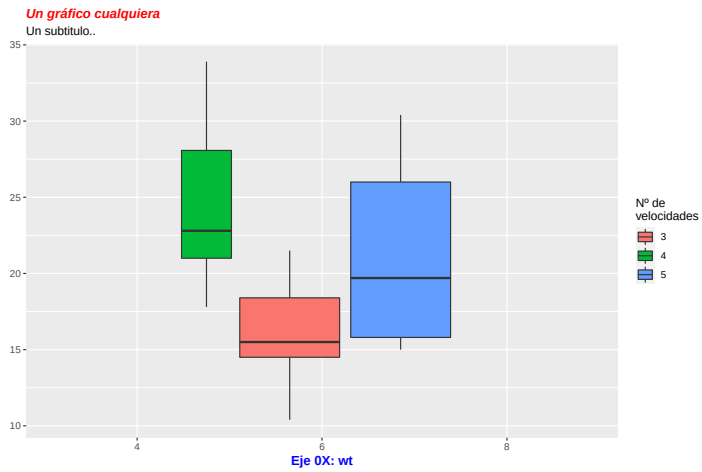


Figura 9.73: *Leyenda con salto*

También se puede cambiar la posición: "left","top", "right", "bottom", "none". O podemos ser más precisos empleando un vector con componentes que toman valores entre 0 y 1: c(0,0) corresponde a la "parte inferior izquierda" y c(1,1) corresponde a la "parte superior derecha".

```
pp + theme(legend.position = "top")
```

```
pp + theme(legend.position = c(0.13, 0.2))
```

Para no incluir leyenda se puede utilizar la sentencia `theme(legend.position = "none")`

```
pp + theme(legend.position = "none")
```

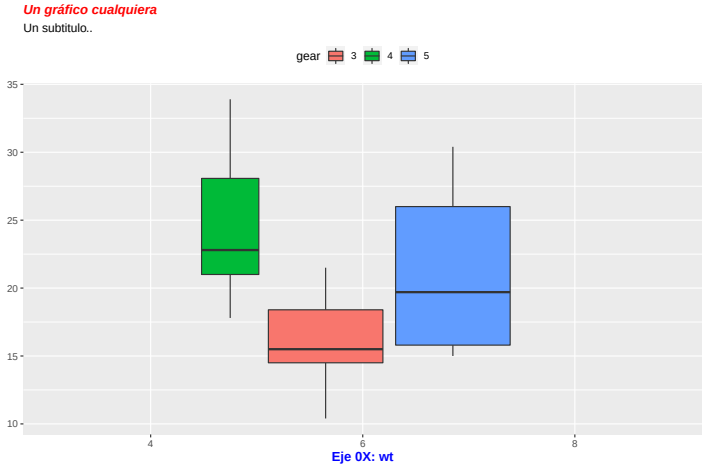


Figura 9.74: *Leyenda arriba*

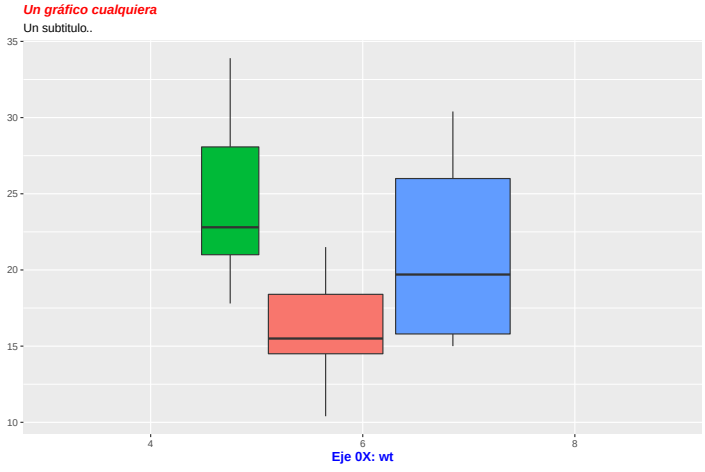


Figura 9.75: *Leyenda posición concreta*

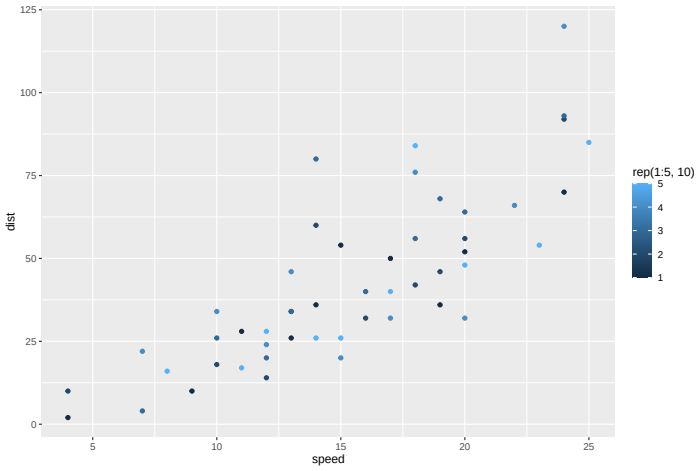


Figura 9.76: *Sin leyenda*

9.3.5. Límites de los ejes

Hay varias formas de hacerlo, algunas implican el recortado de datos y otras no. Lo más recomendable en general es no recortar los datos.

1. `p + coord_cartesian(xlim = c(5, 20), ylim = (0, 50))`: Así hacemos un zoom en el gráfico, sin recortar los datos.
2. Con recorte de los datos (elimina los puntos de datos no vistos)

- `p + xlim(5, 20) + ylim(0, 50)`
- `p + scale_x_continuous(limits = c(5,20)) + scale_y_continuous(limits = c(0, 50))`

3. Expandir los límites del gráfico con datos: `expand_limits()` crea una delgada envoltura alrededor de geom_blank() que facilita añadir datos al gráfico.

- `p + expand_limits(x = 0, y = 0)`: ajustar el corte de los ejes x e y a (0,0)
- `p + expand_limits(x = c(5, 50), y = c(0, 150))`

Probamos estas funciones con el siguiente gráfico de puntos:

```
data(cars)
q <- ggplot(cars, aes(x = speed, y = dist, col = rep(1:5, 10))) +
geom_point()
```

q

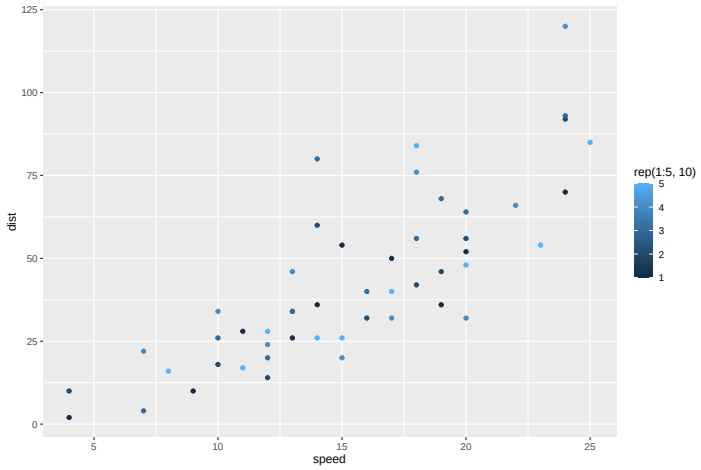


Figura 9.77: *Gráfico sin límites en los ejes*

```
q <- ggplot(cars, aes(x = speed, y = dist, col = rep(1:5, 10))) +
geom_point()
q + coord_cartesian(xlim = c(5, 20), ylim = c(0, 50)) +
xlab("coord_cartesian")
```

9.3.6. Miscelánea

9.3.6. Polígonos

```
library("maps")
spain = map_data("world", region = "Spain")
ggplot(spain, aes(x = long, y = lat, group = group)) + geom_polygon(
fill = "white",
    colour = "black")

puntos <- data.frame(x = c(2, 5, 7), y = c(3, 5, 3))
ggplot(puntos, aes(x = x, y = y)) + geom_polygon(colour = "red", alpha = 0.5)
```

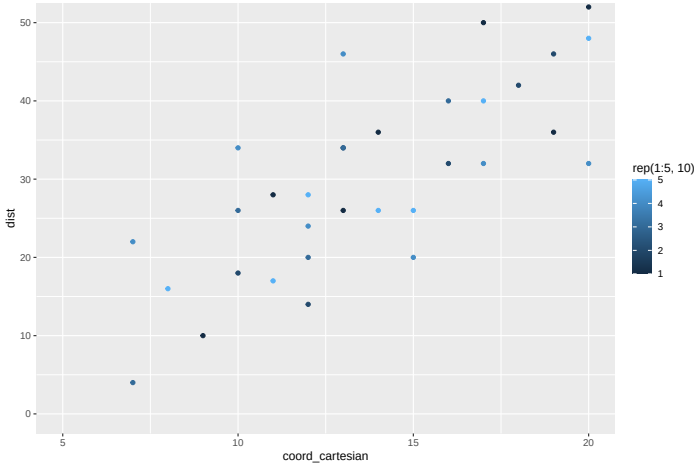


Figura 9.78: *Gráfico con límites en los ejes*

Figura 9.79: *Mapa de España*

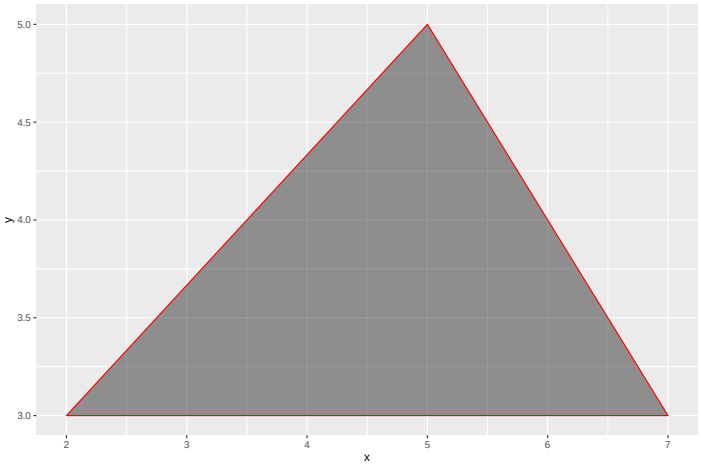


Figura 9.80: *Polígono*

9.3.7. Colores

Por la importancia que tiene el color en un gráfico vamos a ahondar un poco más en este `aesthetics` ya que con muy poco más los resultados pueden ser muchísimo mejores.

```
ToothGrowth$dose <- factor(ToothGrowth$dose)
bp <- ggplot(ToothGrowth, aes(x = dose, y = len)) + geom_boxplot()
bp
```

Podemos usar los argumentos estándar de la geometría concreta que estemos empelando y asignarle colores:

```
bp + geom_boxplot(fill = "steelblue", color = "red")
```

O colorear según alguna variable del conjunto de datos para añadir más información

```
bp + geom_boxplot(aes(fill = dose))
```

La luminosidad y la intensidad de color de los colores por defecto puede ser modificado usando las funciones `scale_hue()`:

```
bp + geom_boxplot(aes(fill = dose)) + scale_fill_hue(l = 40, c = 35)
```

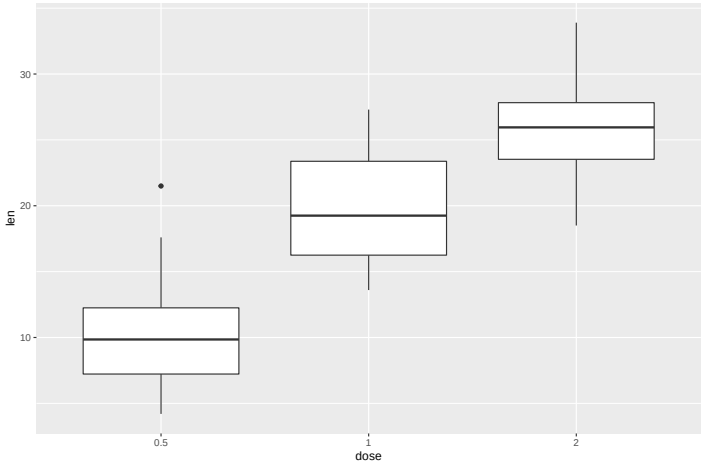


Figura 9.81: *Boxplot sin color*

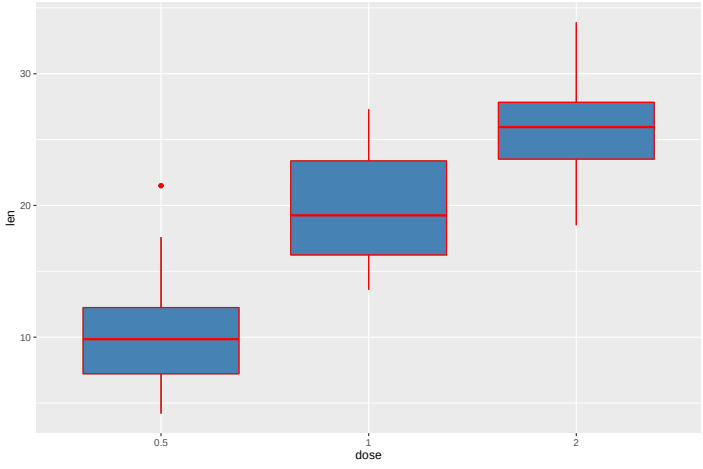


Figura 9.82: *Boxplot con colores de relleno y línea*

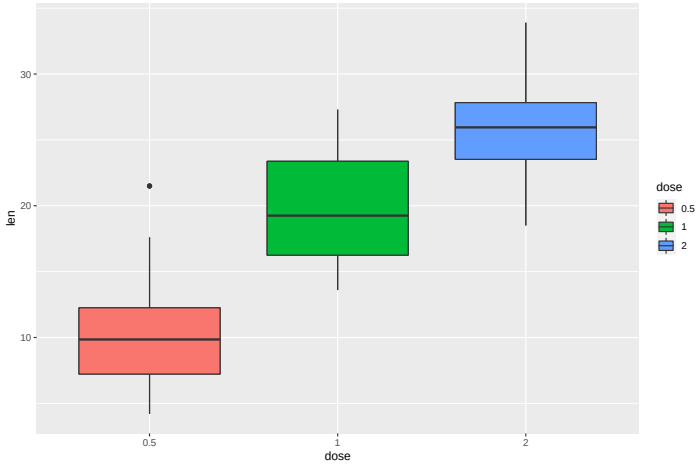


Figura 9.83: *Colores basados en variable*

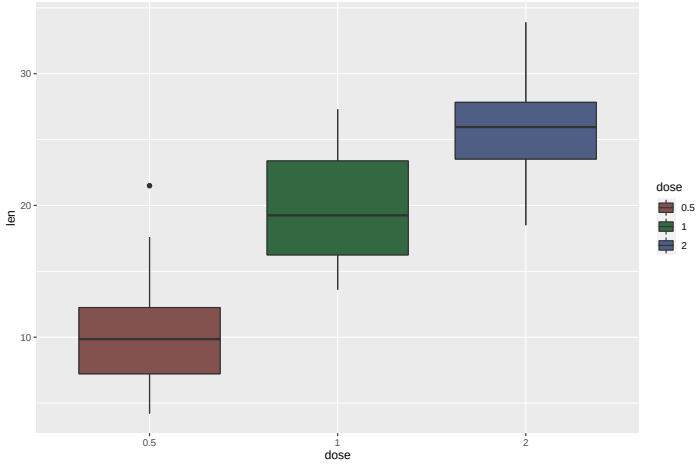


Figura 9.84: *Luminosidad de los colores ajustada*

Aunque también podemos asignar colores manualmente:

```
bp + geom_boxplot(aes(fill = dose)) + scale_fill_manual
(values = c("#999999", "#E69F00",
    "#56B4E9"))
```

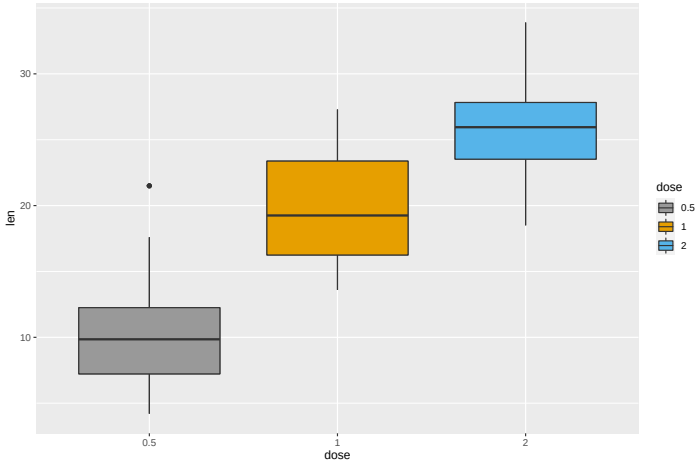


Figura 9.85: *Colores RGB manualmente*

9.3.7. Paletas de color

Una opción muy interesante es aplicar paletas que combinan colores y que están estudiadas para que queden bien:

```
library(RColorBrewer)
```

```
bp + geom_boxplot(aes(fill = dose)) + scale_fill_brewer(palette = "Dark2")
```

En el caso de gráficos de líneas o puntos en lugar de `scale_fill_brewer()` emplearmos `scale_color_brewer()`

Nos pude ser de utilidad el portal COLORBREWER 2.0 para seleccionar paletas según los criterios que deseemos, por ejemplo “6 colores divergentes” o “6 colores en secuenciales”,...

Estas son unas paletas aptas para daltónicos:

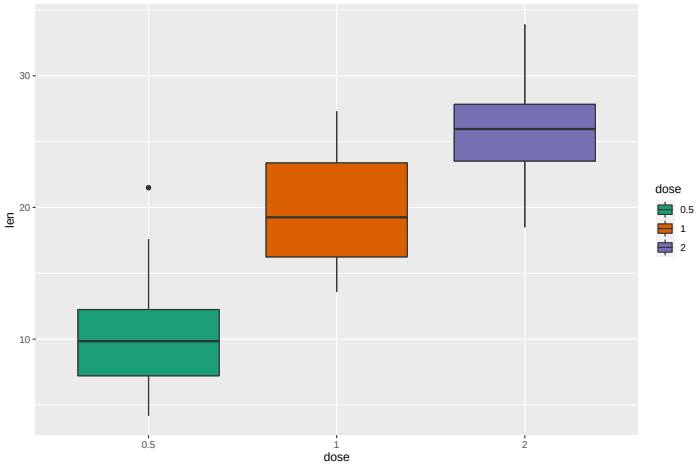


Figura 9.86: *Colores con paleta Dark2*

```
# contiene el verde
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442",
"#0072B2", "#D55E00", "#CC79A7")
# con coor negro
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442",
"#0072B2", "#D55E00", "#CC79A7")
```

```
scale_fill_manual(values=cbPalette)
```

o así

```
scale_colour_manual(values=cbPalette)
```

Una opción muy socorrida es emplear escalas de grises:

```
bp + geom_boxplot(aes(fill = dose)) + scale_fill_grey(start = 0.8, end = 0.2)
```

Igualmente para gráficos de líneas o puntos emplear en lugar de `scale_fill_grey()` `scale_color_grey()`

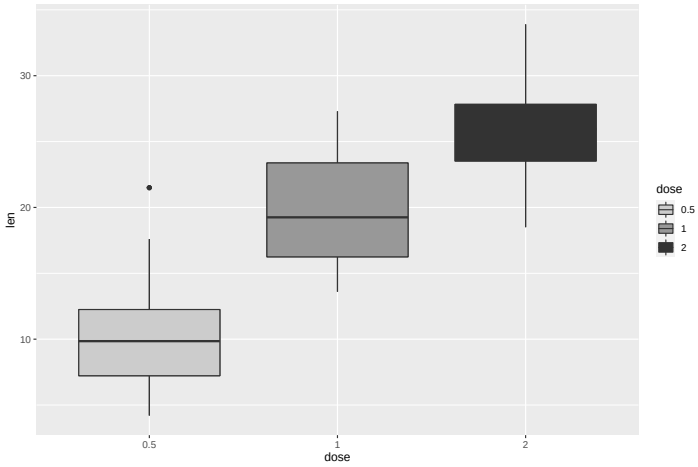


Figura 9.87: *Escala de grises*

También podemeos especificar un gradiente entre dos colores:

- `scale_color_gradient()`, `scale_fill_gradient()` para gradientes secuenciales entre dos colores
- `scale_color_gradient2()`, `scale_fill_gradient2()` para gradientes divergente
- `scale_color_gradientn()`, `scale_fill_gradientn()` para gradientes secuenciales entre n colores

```
class(mtcars$qsec)
```

```
##  [1] "numeric"
```

```
sp2 <- ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point(aes(color = qsec))
sp2
```

```
sp2 + scale_color_gradient(low = "blue", high = "red")
```

```
# divergente con un punto medio, la media por ejemplo
mid <- mean(mtcars$qsec)
sp2 + scale_color_gradient2(midpoint = mid, low = "blue",
mid = "white" , high = "red", space = "Lab")
```

Gradientes entre varios colores:

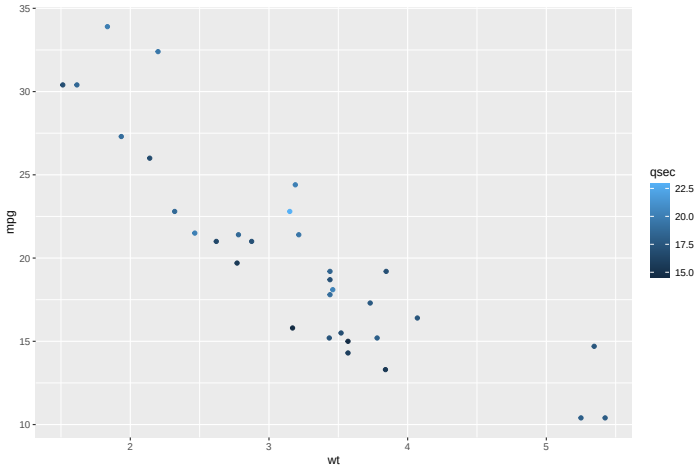


Figura 9.88: *Gradiente de colores*

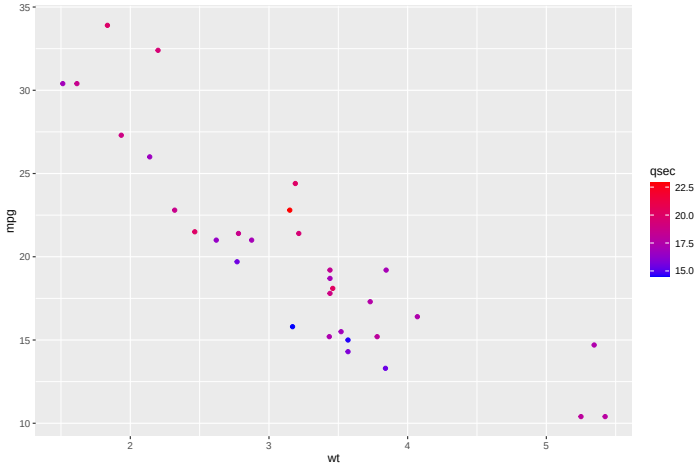


Figura 9.89: *Gradiente de colores con low y high*

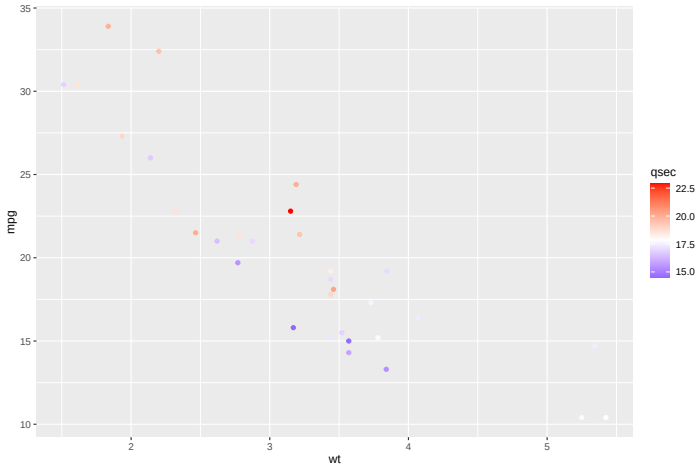


Figura 9.90: *Gradiente de colores divergente*

```
sp2 <- ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point(aes(color = qsec))
sp2 + scale_color_gradientn(colours = rainbow(5))
```

9.3.8. Dónde seguir

Hay muchos sitios donde encontrar buen material sobre ggplot, nosotros recomendamos el libro de Hadley Wickham por que es fácil de leer, muy didáctico, asequible y se aprende mucho. Un libro intermedio es el de Alboukadel Kassambara que es un manual de acceso rápido a muchos tipos de gráficos. Para ir tirando en momentos de apuros es muy útil la cheat sheet de Rstudio y el portal de STHDA.

- Data Visualization with ggplot2: CHEAT SHEET
- El libro de Hadley Wickham es magnífico y muy asequible [65]: ggplot2, Elegant Gaphics for Data Analysis
- El libro de Alboukadel Kassambara es un manual de acceso rápido a muchos tipos de gráficos [66]: Guide to Create Beautiful Graphics in R:, http://www.sthda.com/english/download/3-ebooks/5-guide-to-create-beautiful-graphics-in-r-book/
- STHDA Statistical tools for high-throughput data analysis. Web interesante con lo esencial de ggplot2: http://www.sthda.com/english/wiki/ggplot2-

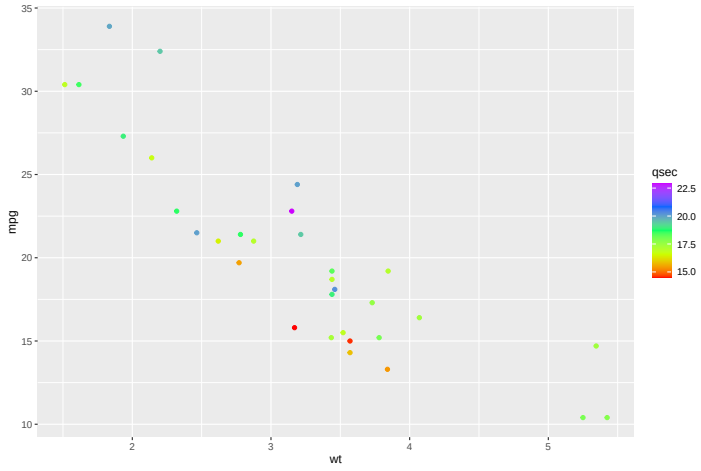


Figura 9.91: *Gradiente de varios colores*

essentials

- El libro de Winston Chang Cookbook for R e s un manual magnifico de fácil y rápido acceso [67]: http://www.cookbook-r.com/Graphs/

9.4. Shiny: aplicaciones web con R

Shiny es un paquete de R que permite crear aplicaciones web interactivas que cuentan con todos los elementos de R. Para aquellos que tienen conocimiento de R, es muy sencillo crear una aplicación en cuestión de horas [1]. Para crear una aplicación mínima, no se necesita conocimientos de HTML (`HyperText Markup` Language), CSS (`Cascading Style Sheets`) o `JavaScript` y sus dependencias. Además, no es necesario pensar en los elementos técnicos para hacerla accesible en la web como, por ejemplo, el puerto, ya que shiny se encarga de esos detalles si no se cambian las opciones por defecto. Éstas son algunas de las razones principales por las cuales shiny se ha vuelto tan popular a lo largo de los años, ya que con poco esfuerzo se pueden crear pruebas de concepto de un producto, mostrar algoritmos o presentar resultados de investigación de forma elegante a través de interfaces de usuario accesibles, reproducibles y amigables [68].

9.4.1. Componentes de shiny

El primer paso para disfrutar de shiny consiste en instalar el paquete que está disponible en CRAN:

```
install.packages("shiny")
```

Las aplicaciones Shiny tienen dos partes mínimas:

1. Una interfaz de usuario `ui`
2. Un script de servidor o secuencia de comandos de servidor `server`

Estas partes pueden encontrarse en el mismo script o estar separadas en dos scripts con nombres fijos: `ui.R` y `server.R`. En este caso se ha elegeido la segunda opción para ilustrar los ejemplos con mayor claridad. Una aplicación Shiny es un directorio que contiene estos scripts y otros ficheros adicionales (conjuntos de datos, fichero donde se definen funciones no dinámicas, etc)

El código mínimo para crear una aplicación con un título, panel lateral y panel principal es el que sigue:

- `ui.R`

```
shinyUI(pageWithSidebar(headerPanel("Titulo descriptivo"),
sidebarPanel(), mainPanel()))
```

- `server.R`

```
shinyServer(function(input, output) {
})
```

9.4.1. Elementos para entrada de datos

Control deslizante: Un control deslizante permite que el usuario seleccione entre un intervalo de valores moviendo un control de posición por una pista. En shiny se crean con la función `sliderInput` que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
sliderInput(inputId, label, min, max, value, step = NULL, animate = FALSE)
```

Sus características incluyen:

- La posibilidad de introducir un único valor y rangos
- Formatos customizaods (por ejemplo para entradas relativas al dinero)
- Pueden ser animados y recorrer los valores de forma automática (argumento `animate`)

Algunos ejemplos son:

```
library(shiny)

shinyUI(pageWithSidebar(

  headerPanel("Barras"),

  sidebarPanel(sliderInput("enteros", "Enteros:",
                           min=0, max=1000, value=500),

               sliderInput("decimales", "Decimales:",
                           min = 0, max = 1, value = 0.5, step= 0.1),

               sliderInput("rango", "Rango:",
                           min = 1, max = 1000, value = c(200,500)),

               sliderInput("animacion", "Animacion:", 10, 200, 10, step = 10,
                           animate=animationOptions(loop=T)),
    ),

  mainPanel()
))
```

Figura 9.92: *Barras deslizantes*

Botón circular: Un botón circular es un tipo de selector que da una lista de opciones entre las cuales solo se puede seleccionar una. En shiny se crean con la función `radioButtons` que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
radioButtons(inputId, label, choices, selected = NULL)
```

Un ejemplo es el siguiente, donde la distribución "Exponencial" está elegida por defecto.

```
radioButtons("dist", "Tipo de distribución:",
c(Normal = "norm", Uniforme = "unif",
    `Log-normal` = "lnorm", Exponencial = "exp"), "Exponencial")
```

Selección múltiple: Un cuadro de selección múltiple es un tipo de selector que da una lista de opciones entre las cuales se pueden seleccionar varias. En shiny se crean con la función `selectInput` que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
selectInput(inputId, label, choices, multiple = FALSE)
```

```
selectInput("variable", "Variable:", c(Seleccion1 = "s1",
Seleccion2 = "s2", Seleccion3 = "s3"),multiple = TRUE)
```

Entrada numérica

```
numericInput("obs", "Numero de observaciones:", 10)
```

Entrada de texto

```
helpText("aclaraciones")
```

9.4.1. Elementos para visualización (salida)

El flow de Shiny es como sigue: el `ui.R` recoge los parámetros necesarios mediante la expresión principal `input`. En el script `server.R` se llevan a cabo los cómputos necesarios utilizando dichas entradas para proveer de un elemento como un gráfico, una tabla, o cualquier salida que están referenciadas a partir del objeto `output` y las funciones `renderYYY` donde YYY es el tipo de objeto a devovler. Por último el `ui.R` utiliza funciones específicas del tipo `YYYOutput` para mostrar dichos resultados por pantalla. En concreto, algunas posibilidades son:

Server	Ui	Crea
renderImage	imageOutput	Imagen
renderPlot	plotOutput	Grafico
renderTable	tableOutput	Tabla
renderText	textOutput	Texto

Server	Ui	Crea
	htmlOutput	HTML
	verbatimTextOutput	Texto

Gráficos

```
# server.R
shinyServer(function(input, output) {

    output$gra1 <- renderPlot({
        dist <- rnorm(input$enteros)
        hist(dist)

    })

})
```

Se ha llamado gra1 a la variable que es el gráfico. En en interior se crea una distribución de el número de muestras que se introducen mediante la barra enteros y se crea un histograma que es lo que se mostrará. Este histograma se debe llamar en la ui:

```
# ui.R
shinyUI(pageWithSidebar(headerPanel("Gráfico"),
sidebarPanel(sliderInput("enteros",
    "Enteros:", min = 0, max = 1000, value = 500), ),
    mainPanel(plotOutput("gra1"))))
```

9.4.1. Lectura de ficheros de datos

También es posible leer ficheros de datos para utilizarlos en la aplicación shiny. Podemos tener predefinidos los argumentos de la función `read.csv()` si estamos seguros de que los datos objeto de la aplicación a desarrollar tienen siempre el mismo formato. Si no, es posible seleccionar mediante los distintos elementos para entrada de datos las características del dataset. En el siguiente

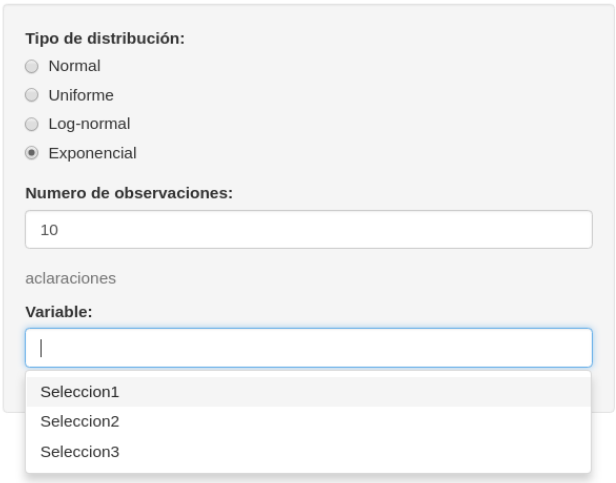


Figura 9.93: *Varios elementos para la entrada de datos*

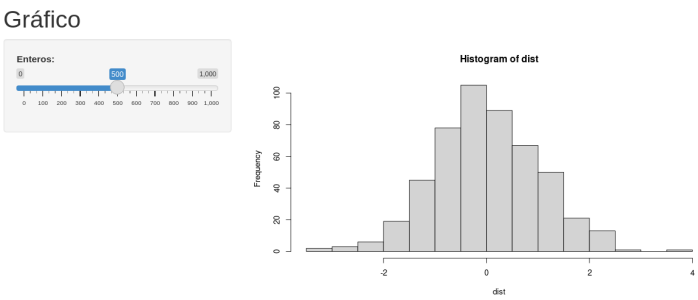


Figura 9.94: *Shiny con gráfico*

ejemplo mostramos cómo leer un fichero de datos y cómo mostrar su contenido con `renderTable`.

```
shinyUI(pageWithSidebar(

  headerPanel("Lectura de datos"),

  sidebarPanel(
    h4("Cargar fichero CSV"),
    fileInput('file1', '',
              accept=c('text/csv', 'text/comma-separated-values,text/plain',
              '.csv')),

    checkboxInput('header', 'Header (el csv tiene nombres de variables)',
    TRUE),

    radioButtons('sep', 'Separador:',
                 c(Tabulador=';',
                   'Punto y coma'='\t'
                 )),

    radioButtons('dec', 'Separador de decimales:',
                 c('Punto'=',',
                   'Coma'='.'))

  ),

  mainPanel(
    tabsetPanel(

      tabPanel("CSV",
               h4("Vista del fichero CSV"),
               tableOutput('contents')
      )
    )
    )
))
```

```
shinyServer(function(input, output) {
    output$contents <- renderTable({
        inFile <- input$file1
        if (is.null(inFile))
            return(NULL)
        read.csv(inFile$datapath, header = input$header, dec = input
        $dec, sep = input$sep)
    })

})
```

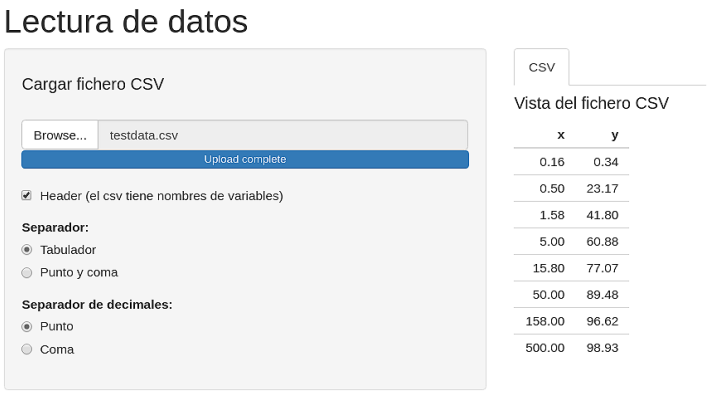


Figura 9.95: *Lectura de csv*

9.4.1. Reactividad

El modelo de reactividad que utiliza Shiny es el siguiente: hay una fuente reactiva, un conductor reactivo y un punto final de la reactividad. La fuente reactiva suele ser lo que el usuario introduce y el punto de parada lo que se muestra por pantalla. A lo que el usuario introduce se accede con el objeto input y a lo que se muestra por pantalla con el objeto output. Veamos un ejemplo que ya hemos usado:

```
output$gra1 <- renderPlot({
    dist <- rnorm(input$enteros)
```

```
    hist(dist)
})
```

El objeto `output$distPlot` es un punto final de la reactividad, y usa la fuente reactiva `input$obs`. Cuando `input$obs` cambia, a `output$distPlot` se le notifica que necesita ejecutarse de nuevo.

Conductores reactivos

También es posible crear componentes reactivos que conecten los inputs y los outputs. En el siguiente ejemplo hemos creado un objeto reactivo `datos` que genera datos que siguen una distribución que el usuario selecciona a través del radioButton `dist` y cuya muestra tiene tantos elementos como el usuario haya especificado en el numericInput `obs`.

```
# server.R
shinyServer(function(input, output) {

    datos <- reactive({
        dist <- switch(input$dist, norm = rnorm, unif = runif,
lnorm = rlnorm, exp = rexp,             rnorm)

        dist(input$obs)
    })

    output$plot <- renderPlot({
        hist(datos(), main = paste("r", input$dist, "(", input$obs, ")",
        sep = ""))
    })
})
```

```
# ui.R
shinyUI(pageWithSidebar(
  headerPanel("Reactividad y RadioButtons"),
  sidebarPanel(
    radioButtons("dist", "Tipo de distribucion:",
                 c("Normal" = "norm",
                   "Uniforme" = "unif",
                   "Log-normal" = "lnorm",
```

```
                    "Exponencial" = "exp"), selected= "Exponencial"),

    numericInput("obs", "Numero de observaciones:", 10),

  ),

  mainPanel(
      tabPanel('Histograma distribucion RadioButton',
               'Plot', plotOutput("plot")
    )
  )
  ))
```

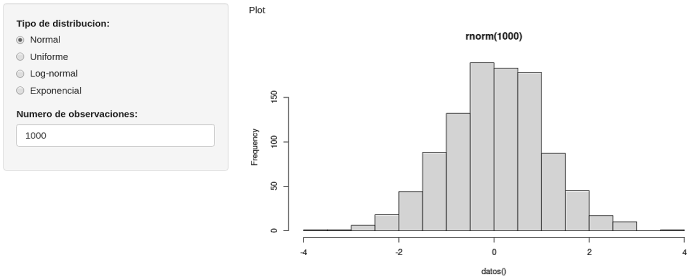


Figura 9.96: *Distribución cambiante*

Controlar la reactividad

Shiny permite controlar la reactividad a través de los actionButtons. Se pueden modificar las entradas sin obtener una respuesta hasta que se apriete dicho botón.

Hemos creado un panel nuevo dentro del mainPanel y éste contendrá, además del `plot` previo, un actionButton con la etiqueta `Presiona`.

```
shinyUI(pageWithSidebar(
  headerPanel("Controlar reactividad"),
  sidebarPanel(
    radioButtons("dist", "Tipo de distribucion:",
                 c("Normal" = "norm",
                   "Uniforme" = "unif",
```

```
                    "Log-normal" = "lnorm",
                    "Exponencial" = "exp"), selected= "Exponencial"),
                numericInput("obs", "Numero de observaciones:", 10),
  ),
  mainPanel(
      tabPanel('Histograma distribucion RadioButton',
               'Plot', plotOutput("plot"), actionButton("botonReac",
               "Presiona")    )
  )

  ))
```

A continuación, hacemos referencia a ese botón para cada una de las expresiones reactivas anteriores, que se aislarán con la función `isolate`:

```
shinyServer(function(input, output) {
    datos <- reactive({
        if (input$botonReac == 0)
            return(dist(rexp(input$obs)))
        isolate({
            dist <- switch(input$dist, norm = rnorm, unif = runif,
lnorm = rlnorm,
                exp = rexp, rnorm)

            dist(input$obs)
        })
    })

    output$plot <- renderPlot({
        if (input$botonReac == 0)
            return(NULL)
        isolate({
            hist(datos(), main = paste("r", input$dist, "(", input$obs, ")",
            sep = ""))
        })
    })
})
```

9.4.2. Publicación de la aplicación en la web

Después del desarrollo de una aplicación Shiny, suele ser interesante publicarla para su explotación científica o empresarial. Existen soluciones que ofrece Rstudio, con diferente complejidad y niveles de libertad que interesa conocer para poder exponer la aplicación en forma de página web: - shinyapps.io - Shiny Server - RStudio Connect

Daremos una introducción muy breve a cada uno de ellos y enlaces para que el lectro pueda indagar en profundidad

9.4.2. Shinyapps.io

La forma más sencilla para convertir una aplicación Shiny en una página web es utilizar shinyapps.io, que es el servicio de `hosting` que ofrece Rstudio.

Shinyapps.io te permite subir la aplicación directamente desde la sesión de R a un servidor que es mantenido por Rstudio. Hay un control casi completo sobre la aplicación, incluyendo la administración del servidor.

Shinyapps.io es una `platform as a service` (PaaS) para hospedar aplicaciones web Shiny en la nube. Lo único que se necesita es: - Un entorno de desarrollo de R, como RStudio IDE - La última versión del paquete rsconnect

En la web shinyapps.io en el apartado "Dashboard" te puedes inscribir. Shinyapps.io genera de forma automática un token que el paquete rsconnect utiliza para acceder a la cuenta.

```
rsconnect::setAccountInfo(name = "<ACCOUNT>", token = "<TOKEN>",
secret = "<SECRET>")
```

Para desplegar la aplicación utiliza `deployApp()` como sigue

```
library(rsconnect)
deployApp()
```

Para más información sobre este método, consulta la página https://shiny.rstudio.com/articles/shinyapps.html

9.4.2. Shiny Server

Shiny server construye un servidor web diseñado para hospedar aplicaciones Shiny. Es gratuito, de código abierto y está disponible en GitHub.

Para usar el Shiny Server, es necesario tener un servidor Linux que tenga soporte explícito para U Ubuntu 12.04 or superior (64 bit) y CentOS/RHEL 5 (64 bit). Si no se está utilizando una distribución con soporte explícito, se puede aún así utilizar construyéndolo desde el paqeute fuente.

En el mismo Shiny Server se pueden hospedar múltiples aplicaciones Shiny. Para ver instrucciones detalladas para su instalación y configuración, recomendamos la guía Shiny Server https://docs.rstudio.com/shiny-server.‘

9.4.2. RStudio Connect

Si utilizas Shiny en un entorno con fines lucrativos, es posible que desee dotarse de las herramientas de servidor que vienen con la mayoría de los programas de servidor de pago, como

- Soporte SSL
- Herramientas de administrador
- Soporte prioritario

Para ello, la plataforma de publicación RStudio Connect puede ser una solución. Esta herramienta permite compartir aplicaciones Shiny, informes R Markdown, cuadros de mando, gráficos, Jupyter Notebooks y más. Con RStudio Connect se puede programar la ejecución de informes y políticas de seguridad flexibles.

Tabla 9.2: *Comparativa entre herramientas para publicación en la nube de la aplicación Shiny [1]*

Categoria	Descripcion	ShinyServer	RstudioConnect	Shinyapps.io
Overview	Licencia comercial (no AGPL)		X	X
	Soporte de RStudio		X	X
	Despliegue de aplicaciones Shiny en la web	X	X	X
	Publicación mediante un botón desde el IDE de RStudio		X	X
	Despliegue y acceso a aplicaciones Shiny, cuadros de mando, informes R Markdown, gráficos estáticos y APIs en un solo lugar		X	

Categoria	Descripcion	ShinyServer	RstudioConnect	Shinyapps.io
	Despliegue y acceso a contenido basado en Python, incluyendo Flask, Dash, Streamlit y Bokeh		X	
	Actualizaciones programadas y distribución de informes		X	
	Contenido autogestionado: vea y gestione lo que ha publicado o a lo que pueden acceder otros		X	solo publishers
	Controladores profesionales: conéctese a algunas de las bases de datos más populares		X	
Seguridad y autenticación	Proteger con contraseña las aplicaciones		X	X*
	Despliegue de aplicaciones Shiny detrás de cortafuegos	X	X	
	Acceso controlado mediante SSL y LDAP, Active Directory, Google OAuth, PAM, autenticación proxy o contraseñas		X	
Tuneo y escalado	Escala de aplicaciones a través de múltiples procesos de R		X	X
	Procesos R persistentes para tiempos de carga más rápidos		X	
Métricas y gestión	Métricas de rendimiento y recursos		X	X
	Punto final de comprobación de la salud		X	

9.5. Tablas

En este apartado vamos a utilizar el paquete `kableExtra` para diseñar vistosas tablas con los datos que pueden ser incorporadas a nuestros informes, tesis, aplicaciones web y demás trabajos de investigación y productos.

El objetivo de `kableExtra` es construir tablas complejas y manipular estilos de tablas. Este paquete utiliza la sintaxis pipes`%>%` y verbaliza las funciones de forma que se pueden añadir capas, de una forma similar a `ggplot`.

Cuando no se especifique el formato, `kableExtra` genera una tabla markdown y deja que pandoc automáticamente realice la conversión de markdown a HTML/PDF.

Nota: Aunque no es esltrictamente necesario, si deseas conocer qué es pandoc sigue el siguiente enlace https://ull-esit-dsi-1617.github.io/estudiar-las-rutas-en-expressjs-alberto-diego/Diego/Pandoc/queespandoc.html

Para empezar, cargamos la librería `kableExtra`.

```
library(kableExtra)
```

Vamos a crear el dataset cuya tabla será manipulada a lo largo de este apartado. Para ello, guardamos en la variable `subC02` las 5 primeras filas y las 4 primeras columnas del dataset `C02`.

```
subC02 <- C02[1:5, 1:4]
```

9.5.1. Estilos

Utilizamos la función kable solo con la tabla `subC02`, sin ningún argumento extra para observar el estilo básico html.

```
kable(subC02)
```

Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	95
Qn1	Quebec	nonchilled	175
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

Como puedes observar, es un estilo demasiado crudo y que no provoca interés al lector.

Para controlar el estilo se utiliza `kable_styling()`

```
kable_styling(kable(subC02))
```

Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	95
Qn1	Quebec	nonchilled	175
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

O lo que es equivalente, usando pipes

```
subCO2 %>%
    kable() %>%
    kable_styling()
```

Algunos de los argumentos de estilo son:

- `full_width`: anchura completa (de tipo booleano: TRUE o FALSE)
- `position`: lugar que ocupa en el documento (opciones: left, center, right, float_left and float_right)
- `font_size`: tamaño de la fuente (número entero)

En el siguiente ejemplo podemos ver la misma tabla pero sin la anchura completa, ocupando la posición izquierda y cuya fuente es 7.5.

```
subCO2 %>%
    kable() %>%
    kable_styling(full_width = F, position = "left", font_size = 7.5)
```

Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	95
Qn1	Quebec	nonchilled	175
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

Nota: Hay más argumentos de estilo, se recomienda continuar con el argumento `bootstrap_options` e indagar en la ayuda de la función `?kable_stylig` para más información.

9.5.2. Especificaciones en filas y columnas

Es posible también ajustar el ancho de las columnas y filas, darles color y customizarlas con las funciones `column_spec` y `row_spec`.

Por ejemplo, coloreamos de naranja la segunda columna de nuestra tabla y, además, le asignamos 3 cm de anchura con los siguientes argumentos `column`, `width` y `background`.

```
kable(subC02) %>%
    kable_styling(full_width = F) %>%
    column_spec(column = 2, width = "3cm", background = "orange")
```

Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	95
Qn1	Quebec	nonchilled	175
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

Una tabla que utiliza el `column_spec` es la siguiente:

```
kable(subC02) %>%
    kable_styling(full_width = F) %>%
    column_spec(column = 2, width = "3cm", background = "red"
    , color = "pink", italic = F) %>%
    column_spec(column = 4, background = "yellow", color = "black"
    , strikeout = T)
```

Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	~~95~~
Qn1	Quebec	nonchilled	~~175~~
Qn1	Quebec	nonchilled	~~250~~
Qn1	Quebec	nonchilled	~~350~~
Qn1	Quebec	nonchilled	~~500~~

Se puede realizar lo mismo para las filas con la función `row_spec`. Además, hay una funcionalidad extra. Las cabeceras o nombres de las variables se consideran la fila número 0 y también se pueden customizar.

```
kable(subCO2) %>%
    kable_styling(full_width = F) %>%
    row_spec(row = 0, color = "blue", angle = -15) %>%
    row_spec(row = 2, background = "green", bold = T, color = "white")
```

Plant	*Type*	*Treatment*	*conc*
Qn1	Quebec	nonchilled	95
Qn1	**Quebec**	**nonchilled**	**175**
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

9.5.3. Agrupar filas y columnas

9.5.3. Agrupar columnas

Algunas tablas requieren múltiples cabeceras para mostrar subgrupos de los datos. Para ello, se puede utilizar `add_header_above()`.

Tan solo hay que indicar los nuevos nombres de las columnas y cuántas columnas pertenecen a cada uno de ellos, por ejemplo se pueden crear 2 grupos de nombres A y B con 2 y 2 columnas respectivamente de la siguiente forma:

```
kable(subCO2) %>%
    kable_styling() %>%
    add_header_above(c(A = 2, B = 2))
```

A		B	
Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	95
Qn1	Quebec	nonchilled	175
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

Continuamos con este ejemplo añadiendo otra cabecera que agrupa las 3 primeras columnas por un lado y la última por otro:

```
kable(subCO2) %>%
    kable_styling() %>%
    add_header_above(c(A = 2, B = 2)) %>%
    add_header_above(c(GR1 = 3, GR2 = 1))
```

GR1			GR2
A		B	
Plant	Type	Treatment	conc
Qn1	Quebec	nonchilled	95
Qn1	Quebec	nonchilled	175
Qn1	Quebec	nonchilled	250
Qn1	Quebec	nonchilled	350
Qn1	Quebec	nonchilled	500

9.5.3. Agrupar filas mediante etiquetas

Si lo que se desea agrupar son las filas, se utiliza la función `group_rows` cuyos argumentos básicos por defecto son: el nombre que se quiere dar, la fila donde empieza el grupo y la fila donde termina, por ejemplo:

```
kable(CO2[1:10, 1:5]) %>%
    kable_styling(full_width = F) %>%
    group_rows(index = c(` ` = 3, `Grupo 1` = 4, `Grupo 2` = 3))
```

Plant	Type	Treatment	conc	uptake
Qn1	Quebec	nonchilled	95	16.0
Qn1	Quebec	nonchilled	175	30.4
Qn1	Quebec	nonchilled	250	34.8
Grupo 1				
Qn1	Quebec	nonchilled	350	37.2
Qn1	Quebec	nonchilled	500	35.3
Qn1	Quebec	nonchilled	675	39.2
Qn1	Quebec	nonchilled	1000	39.7
Grupo 2				
Qn2	Quebec	nonchilled	95	13.6
Qn2	Quebec	nonchilled	175	27.3
Qn2	Quebec	nonchilled	250	37.1

9.5.3. Agrupar filas mediante celdas múltiples

La función `collapse_rows` permite mostrar información de forma estructurada en múltiples filas. Veámoslo con un ejemplo. El siguiente conjunto de datos está en formato long, esto es, los datos de una fila pertenecen a un grupo que está especificando en alguna columna y ese grupo se repite varias veces. Además, una segunda columna indica el subgrupo:

```
kable(collapse_rows_dt) %>%
    kable_styling(full_width = F)
```

C1	C2	C3	C4
a	c	1	0
a	d	2	0
a	d	3	0
a	d	4	1
b	c	5	1
b	c	6	0
b	d	7	1

Una forma en la que se puede querer mostrar la información es sin dichas repeticiones, juntando las columnas de acuerdo a los grupos o valores repetidos. Esto se puede llevar a cabo señalando como argumento cuáles de las columnas indican los grupos en la función `collapse_rows`:

```
kable(collapse_rows_dt) %>%
    kable_styling(full_width = F) %>%
    collapse_rows(columns = 1:2)
```

C1	C2	C3	C4
a	c	1	0
	d	2	0
		3	0
		4	1
b	c	5	1
		6	0
	d	7	1

El argumento que modifica la posición de las etiquetas es `valign`. Para hacer que se encuentren en la parte de arriba, y no en el centro como se muestran por defecto, se utiliza el siguiente código:

```
kable(collapse_rows_dt) %>%
    kable_styling(full_width = F) %>%
    collapse_rows(columns = 1:2, valign = "top")
```

C1	C2	C3	C4
a	c	1	0
	d	2	0
		3	0
		4	1
b	c	5	1
		6	0
	d	7	1

10 Manipulación de datos

Aurora Gónzalez Vidal

10.1. Manipulación básica de dataframes

La primera sección de este capítulo recopila conceptos básicos que ya se han visto a lo largo del libro. Sin embargo, los hemos incluido por completitud del tema.

10.1.1. El dataframe

R es un lenguaje de programación estadístico y para hacer estadística necesitamos trabajar con conjuntos de datos (*datasets*). Estos conjuntos de datos se componen normalmente de:

- Observaciones (o instancias).
- Variables (asociadas a dicha observaciones).

El `dataframe` es la estructura fundamental para manipular conjuntos de datos en R.

Cómo citar este capítulo: Gónzalez Vidal A (2022). Manipulación de datos. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (461-502). Editum. Ediciones de la Universidad de Murcia.

El dataframe se utiliza para guardar tablas de datos. Se puede considerar una lista de vectores de igual longitud que no tienen por qué ser del mismo tipo.

La Figura 10.1 es un ejemplo de cómo se ve un dataframe: un conjunto de datos ordenado por filas y columnas donde cada fila es un sujeto, las columnas son las variables y a lo largo de las filas se pueden observar los valores que toma cada sujeto para cada una de las variables.

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2
7	Qn1	Quebec	nonchilled	1000	39.7
8	Qn2	Quebec	nonchilled	95	13.6
9	Qn2	Quebec	nonchilled	175	27.3
10	Qn2	Quebec	nonchilled	250	37.1

Figura 10.1: *Ejemplo de dataframe y subconjunto del mismo (rectángulo)*

Como se puede observar, algunas variables son cualitativas (como `Treatement`, que es una cadena de caracteres) y otras son cuantitativas (como conc, que es un número entero), por lo que en un dataframe las columnas no tienen por qué ser del mismo tipo entre ellas.

Asimismo, en rojo se selecciona un subconjunto que comprende las filas de la 3 a la 7 (`3:7` en R) y las columnas de la 2 a la 4 (`2:4` en R).

10.1.2. Crear, utilizar y leer dataframes

10.1.2. data.frame() para crear conjuntos de datos

Como vemos en el siguiente ejemplo, la función para crear un dataframe es `data.frame()`:

```
num <- c(1,2,3) # u otros números cualesquiera
letras <- c("aa", "bb", "cc")
logicos <- c(TRUE, FALSE, TRUE)
```

```
df <- data.frame(letras, logicos, num)
df
```

```
  letras logicos num
1     aa    TRUE   1
2     bb   FALSE   2
3     cc    TRUE   3
```

La longitud de los vectores que componen el dataframe debe ser la misma. Para comprobar la longitud de un vector se utiliza la función `length()`:

```
length(letras)
```

```
[1] 3
```

10.1.2. Importar conjuntos de datos incluidos en R

Hay muchos conjuntos de datos que están incluidos en los diferentes paquetes de R. En el paquete básico `base` que se instala automáticamente en R hay algunos. Estos se pueden ver escribiendo `data()` como sigue:

```
data()
```

En este capítulo vamos a utilizar reiteradamente dos conjuntos de datos que están incluidos en R: **CO2** y **mtcars**.

`CO2` contiene información sobre la resistencia al frío de una especie de césped:

- Plant: Identificador de cada planta
- Type: lugar de origen de la planta
- Treatment: Helada o no helada
- conc: Concentración de CO2 en el ambiente
- uptake: Absorción del CO2

Para información más detallada sobre `CO2` visita: https://stat.ethz.ch/R-manual/R-patched/library/datasets/html/zCO2.html.

`mtcars` contiene información sobre coches relativa a:

- mpg: Consumo (miles per gallon)

- cyl: Número de cilindros
- disp: Desplazamiento
- hp: Caballo de vapor
- drat: Eje trasero
- wt: Peso (1000 lib)
- qsec: Tiempo que tarda en recorrer 1/4 de milla desde reposo (seg)
- vs: Forma del motor
- am: Tipo de transmisión
- gear: Engranajes
- carb: Número de carburadores

Para información más detallada sobre `mtcars` visita: https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html.

10.1.2. read.table() para leer conjuntos de datos

Por último, también se puede leer un conjunto de datos que se encuentre en la nube o en una carpeta de tu ordenador usando la función `read.table()`. Esta función tiene varios argumentos, a continuación se ven los esenciales y para un recordatorio más extenso leer `?read.table`.

Si el conjunto de datos `misdatos.csv` tiene este formato al abrirlo con el bloc de notas u otro editor de texto:

```
id;edad;gasto
1;27;5.5
2;33;8.2
3;18;1
4;65;5.1
```

Es porque tiene cabecera, es decir, la primera línea contiene los nombres de las variables (`header=T`), el separador es el punto y coma (sep=";") y los decimales se señalan con un punto (`dec="."`):

```
df <- read.table(file="misdatos.csv", header=T, sep=";", dec=".")
```

10.1.3. Funciones básicas de descripción

Vamos a utilizar las funciones básicas de descripción de conjuntos de datos para descubrir información sobre el conjunto de datos **CO2** del paquete `datasets`.

- `head()`
- `tail()`
- `summary()`
- `str()`
- `nrow()`, `ncol()`, `dim()`

Si no recordamos para qué sirven las funciones anteriores, podemos utilizar el comando `help()` o `?` para desplegar la ayuda:

```
?head
```

Usando una sola función, mostramos cuántas filas y columnas tiene el dataset CO2:

```
dim(CO2)
```

```
[1] 84  5
```

Usando una sola función, mostramos estadísticas básicas del dataset CO2 (frecuencias para variables cualitativas y medias y otros parámetros para variables cuantitativas):

```
summary(CO2)
```

```
    Plant             Type         Treatment       conc
 Qn1    : 7   Quebec     :42   nonchilled:42   Min.   :  95
 Qn2    : 7   Mississippi:42   chilled   :42   1st Qu.: 175
 Qn3    : 7                                   Median : 350
 Qc1    : 7                                   Mean   : 435
 Qc3    : 7                                   3rd Qu.: 675
 Qc2    : 7                                   Max.   :1000
 (Other):42
    uptake
 Min.   : 7.7
 1st Qu.:17.9
```

```
 Median :28.3
 Mean   :27.2
 3rd Qu.:37.1
 Max.   :45.5
```

Usando una sola función, describimos el tipo de las variables del dataset CO2:

```
str(CO2)
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':  84 obs. of  5 variables:
 $ Plant    : Ord.factor w/ 12 levels "Qn1"<"Qn2"<"Qn3"<..: 1 1 1 1 1 1 1 2 2 2 ...
 $ Type     : Factor w/ 2 levels "Quebec","Mississippi": 1 1 1 1 1 1 1 1 1 1 ...
 $ Treatment: Factor w/ 2 levels "nonchilled","chilled": 1 1 1 1 1 1 1 1 1 1 ...
 $ conc     : num  95 175 250 350 500 675 1000 95 175 250 ...
 $ uptake   : num  16 30.4 34.8 37.2 35.3 39.2 39.7 13.6 27.3 37.1 ...
 - attr(*, "formula")=Class 'formula'  language uptake ~ conc | Plant
  .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "outer")=Class 'formula'  language ~Treatment * Type
  .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
  ..$ x: chr "Ambient carbon dioxide concentration"
  ..$ y: chr "CO2 uptake rate"
 - attr(*, "units")=List of 2
  ..$ x: chr "(uL/L)"
  ..$ y: chr "(umol/m^2 s)"
```

10.1.4. Filtrado de datos

Para seleccionar columnas de un dataframe tenemos las siguientes opciones:

- $: seguido del nombre de una variable selecciona solo una columna (el tabulador será de gran ayuda para autocompletar el nombre de la variable)

```
mtcars$mpg
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3
[14] 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3
[27] 26.0 30.4 15.8 19.7 15.0 21.4
```

Utilizamos el $ para seleccionar la columna `disp` del dataframe `mtcars` y calcula su media:

```
mean(mtcars$disp)
```

```
[1] 231
```

A partir de ahora, cuando seleccionemos y mostremos las columnas utilizaremos head por razones de espacio.

- []:
  - se puede utilizar con las posiciones de las columnas a seleccionar
    ```
    head(mtcars[c(1,3)])
    ```
    ```
                      mpg disp
    Mazda RX4         21.0  160
    Mazda RX4 Wag     21.0  160
    Datsun 710        22.8  108
    Hornet 4 Drive    21.4  258
    Hornet Sportabout 18.7  360
    Valiant           18.1  225
    ```
  - o con los nombres de las columnas a seleccionar

```
head(mtcars[c("mpg","disp")])
```

```
                  mpg disp
Mazda RX4         21.0  160
Mazda RX4 Wag     21.0  160
Datsun 710        22.8  108
Hornet 4 Drive    21.4  258
Hornet Sportabout 18.7  360
Valiant           18.1  225
```

Utilizamos el [] para seleccionar las columnas 1 y 5 del dataset mtcars y calculamos estadísticos de dichas columnas seleccionadas.

```
summary(mtcars[c(1,5)])
```

```
      mpg             drat      
 Min.   :10.4   Min.   :2.76  
 1st Qu.:15.4   1st Qu.:3.08  
 Median :19.2   Median :3.69  
 Mean   :20.1   Mean   :3.60  
```

```
 3rd Qu.:22.8   3rd Qu.:3.92
 Max.   :33.9   Max.   :4.93
```

Para seleccionar filas tan solo tendremos que determinar el rango de filas que se desea mostrar:

```
mtcars[1:4,]
```

```
               mpg cyl disp  hp drat   wt qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.62 16.5  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.88 17.0  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.32 18.6  1  1    4    1
Hornet 4 Drive 21.4  6  258 110 3.08 3.21 19.4  1  0    3    1
```

Nota: Recuerda que con la notación `[]` se considera que se seleccionan `[rangofilas, rangocolumnas]`. Vacío significa "todas", o sea:

- `[rangofilas,]`: las filas de ese rango y todas las columnas
- `[,rangocolumnas]`: todas las filas y las columnas de ese rango.
- Si tan solo escribimos un rango sin coma, se seleccionarán las columnas: `[columnas]`. Veámoslo con un ejemplo:

  Las 3 primeras filas:

```
mtcars[1:3,]
```

```
               mpg cyl disp  hp drat   wt qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.62 16.5  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.88 17.0  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.32 18.6  1  1    4    1
```

Las tres primeras columnas:

```
head(mtcars[,1:3])
```

```
                  mpg cyl disp
Mazda RX4         21.0   6  160
Mazda RX4 Wag     21.0   6  160
Datsun 710        22.8   4  108
Hornet 4 Drive    21.4   6  258
```

```
Hornet Sportabout 18.7   8  360
Valiant           18.1   6  225
```

De nuevo las 3 primeras columnas:

```
head(mtcars[1:3])
```

```
                  mpg cyl disp
Mazda RX4         21.0   6  160
Mazda RX4 Wag     21.0   6  160
Datsun 710        22.8   4  108
Hornet 4 Drive    21.4   6  258
Hornet Sportabout 18.7   8  360
Valiant           18.1   6  225
```

Además de rangos numéricos, se pueden utilizar expresiones para hacer el filtrado. Por ejemplo, para seleccionar las filas para las cuales la variable `hp` es mayor que 200 hay que utilizar la expresión `mtcars$hp >200` dentro de los corchetes [ ], como sigue:

```
mtcars[mtcars$hp >200 , ]
```

```
                   mpg cyl disp  hp drat   wt qsec vs am gear carb
Duster 360        14.3   8  360 245 3.21 3.57 15.8  0  0    3    4
Cadillac Fleetwood 10.4   8  472 205 2.93 5.25 18.0  0  0    3    4
Lincoln Continental 10.4   8  460 215 3.00 5.42 17.8  0  0    3    4
Chrysler Imperial  14.7   8  440 230 3.23 5.34 17.4  0  0    3    4
Camaro Z28         13.3   8  350 245 3.73 3.84 15.4  0  0    3    4
Ford Pantera L     15.8   8  351 264 4.22 3.17 14.5  0  1    5    4
Maserati Bora      15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
```

Seleccionamos los valores de `cyl` (columna) para los cuales la variable `disp` es inferior a 100.

```
mtcars[ mtcars$disp <100 , ]
```

```
               mpg cyl disp  hp drat   wt qsec vs am gear carb
Fiat 128      32.4   4 78.7  66 4.08 2.20 19.5  1  1    4    1
Honda Civic   30.4   4 75.7  52 4.93 1.61 18.5  1  1    4    2
Toyota Corolla 33.9   4 71.1  65 4.22 1.83 19.9  1  1    4    1
```

```
Fiat X1-9       27.3   4 79.0  66 4.08 1.94 18.9  1  1    4    1
Lotus Europa    30.4   4 95.1 113 3.77 1.51 16.9  1  1    5    2
```

Por último, creamos un nuevo dataframe a partir del filtrado de otro es sencillo, simplemente asignándolo a un objeto. Por ejemplo, `mtcarsF` puede ser `mtcars` filtrado:

```
mtcarsF <- mtcars[mtcars$hp >200 , ]
head(mtcarsF)
```

```
                   mpg cyl disp  hp drat   wt qsec vs am gear carb
Duster 360        14.3   8  360 245 3.21 3.57 15.8  0  0    3    4
Cadillac Fleetwood 10.4   8  472 205 2.93 5.25 18.0  0  0    3    4
Lincoln Continental 10.4   8  460 215 3.00 5.42 17.8  0  0    3    4
Chrysler Imperial  14.7   8  440 230 3.23 5.34 17.4  0  0    3    4
Camaro Z28         13.3   8  350 245 3.73 3.84 15.4  0  0    3    4
Ford Pantera L     15.8   8  351 264 4.22 3.17 14.5  0  1    5    4
```

10.2. Manipulación avanzada de dataframes

En la sección anterior de este capítulo hemos empleando la librería base y los operadores más clásicos para manipular conjuntos de datos basados en sus índices. Vamos ahora a ahondar en la manipulación de dataframes con funciones de más alto nivel.

Trabajaremos con las funciones que nos permiten tratar valores faltantes, ampliar un dataframe, por filas o columnas: rbind(), cbind(), merge() y nos centraremos en funciones más allá de los operadores basados en índices del data.frame como son las funciones: aggregate() y la familia de funciones apply.

10.2.1. Limpieza del dataframe con funciones base

10.2.1. Cambiar los nombres de filas y columnas

Los dataframes también pueden tener un atributo names que indica los nombres de sus columnas (si los tiene). En general, entendemos las columnas como variables:

```
names(mtcars)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
[10] "gear" "carb"
```

names(mtcars) es un vector de caracteres que incluye los nombres del conjunto de datos mtcars. Este vector es modificable. Cambiamos el nombre de la segunda columna por su equivalente en español (cilindrada).

```
names(mtcars)[2] = "cilindrada"
```

De la misma naturaleza son los nombres de las columnas:

```
rownames(mtcars)
```

```
 [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
 [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
 [7] "Duster 360"          "Merc 240D"           "Merc 230"
[10] "Merc 280"            "Merc 280C"           "Merc 450SE"
[13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetwood"
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
[31] "Maserati Bora"       "Volvo 142E"
```

10.2.1. Valores faltantes

Los valores faltantes son uno de los quebraderos de cabeza de los investigadores ya que pueden alterar el resultado de los experimentos. R nos apoya en la

tarea de detectarlos, imputarlos o borrarlos con las siguientes funciones.

Para comprobar si hay valores faltantes se puede utilizar la (a estas alturas ya famosa) función `summary()`.

```
coches <- read.table("coches.csv",
                     sep=";", head=T)
summary(coches)
```

```
      mpg             cyl             disp             hp
 Min.   :10.4   Min.   :4.00   Min.   : 71   Min.   : 52
 1st Qu.:15.4   1st Qu.:4.00   1st Qu.:121   1st Qu.: 97
 Median :19.2   Median :6.00   Median :225   Median :123
 Mean   :20.1   Mean   :6.14   Mean   :233   Mean   :150
 3rd Qu.:22.8   3rd Qu.:8.00   3rd Qu.:334   3rd Qu.:180
 Max.   :33.9   Max.   :8.00   Max.   :472   Max.   :335
                NA's   :3      NA's   :1     NA's   :3
      drat             wt             qsec             vs
 Min.   :2.76   Min.   :1.51   Min.   :14.5   Min.   :0.000
 1st Qu.:3.08   1st Qu.:2.58   1st Qu.:16.9   1st Qu.:0.000
 Median :3.69   Median :3.33   Median :17.7   Median :0.000
 Mean   :3.60   Mean   :3.22   Mean   :17.9   Mean   :0.438
 3rd Qu.:3.92   3rd Qu.:3.61   3rd Qu.:18.9   3rd Qu.:1.000
 Max.   :4.93   Max.   :5.42   Max.   :22.9   Max.   :1.000

       am              gear            carb
 Min.   :0.000   Min.   :3.00   Min.   :1.00
 1st Qu.:0.000   1st Qu.:3.00   1st Qu.:2.00
 Median :0.000   Median :4.00   Median :2.00
 Mean   :0.414   Mean   :3.69   Mean   :2.81
 3rd Qu.:1.000   3rd Qu.:4.00   3rd Qu.:4.00
 Max.   :1.000   Max.   :5.00   Max.   :8.00
 NA's   :3       NA's   :3
```

Para obtener una tabla que indique el número de valores faltantes por columna, podemos combinar las funciones `colSums()` y `is.na()`:

```
colSums(is.na(coches))
```

```
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
   0    3    1    3    0    0    0    0    3    3    0
```

Utilizamos las funciones `which()` y `is.na()` para encontrar las filas en las que la varible `hp` tiene los valores faltantes.

```
which(is.na(coches$hp))
```

```
[1] 1 5 8
```

Si queremos imputar los valores faltantes con un valor fijo solo tenemos que seleccionarlos y asignar dicho valor a la selección como sigue:

```
cochesI <- coches
cochesI[is.na(cochesI)] <- 10
```

Calculamos la mediana de la columna 2 (`cyl` o `cilindrada`) e imputamos los valores faltantes de esa columna con ella.

```
mediana <- median(coches$cyl)
coches$cyl[is.na(coches$cyl)] <- mediana
```

La función `complete.cases()` devuelve un vector lógico indicando qué filas del conjunto de datos están completas, es decir, no tienen valores faltantes.

```
complete.cases(coches)
```

```
 [1] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[12]  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
[23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

Utilizamos la función `complete.cases()` para crear un nuevo dataframe, basado en `coches` pero que se haya deshecho de las filas donde hubiera algún valor faltante. Comprobamos que ya no hay ningún valor faltante en el nuevo dataframe como hemos aprendido anteriormente.

```
cochesLimpio <- coches[complete.cases(coches),]
colSums(is.na(cochesLimpio))
```

```
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
   0    0    0    0    0    0    0    0    0    0    0
```

10.2.1. Combinar dataframes

Ahora vamos a aprender a unir dataframes. Trabajaremos con estos conjuntos de datos:

```
paises1 <- read.table("countries_sub1.csv",
                                  sep=";", head=T)
paises2 <- read.table("countries_sub2.csv",
                                  sep=";", head=T)
paises3 <- read.table("countries_sub3.csv",
                                  sep=";", head=T)
paises4 <- read.table("countries_sub4.csv",
                                  sep=";", head=T)
```

Cuyo contenido se puede consultar en la Figura 10.2.

paises1

Country	MigracionNeta
Argentina	0.61
Alemania	2.18

paises2

Country	MigracionNeta
Rusia	1.00
España	0.99
Grecia	2.35

paises3

Pais	MigracionNeta
Grecia	2.35
Honduras	-1.99

paises4

Country	RatioNacimientos
España	10.06
Argentina	16.73
Rusia	9.95

Figura 10.2: *Dataframes de paises1-4*

Cuando trabajamos con datos que proceden de diversas fuentes, nos puede interesar unirlos. Los datos a unir deben tener una estructura común: mismas variables (incluyendo nombre y tipo) o misma longitud.

- Añadir más filas: `rbind()` (*row bind*) sirve para anexar las filas de 2 dataframes.

```
paises <- rbind(paises1, paises2)
```

Country	MigracionNeta
Argentina	0.61
Alemania	2.18
Rusia	1.00
España	0.99
Grecia	2.35

Tabla 10.1: *Tabla rbind paises1 y paises2*

Si deseamos unir el conjunto anterior, `paises` con `paises3`, ocurre el siguiente error:

```
rbind(paises, paises3)
```

"Error in match.names(clabs, names(xi)) : names do not match previous names"

Es muy importante leer los errores, en este caso nos indica que no se puede realizar la unión de las filas de ambos conjuntos de datos porque los nombres de las columnas de ambos no coinciden, como comprobamos a continuación:

```
colnames(paises)
```

```
[1] "Country"       "MigracionNeta"
```

```
colnames(paises3)
```

```
[1] "Pais"          "MigracionNeta"
```

La primera columna tiene el nombre en español y en inglés respectivamente, podemos solucionarlo de la siguiente manera:

```
colnames(paises) <- colnames(paises3)
elbueno <- rbind(paises, paises3)
```

- Añadir más columnas `cbind()` (*column bind*) y `merge()` sirven para anexar las columnas de 2 dataframes que tienen las mismas variables.

```
paises5 <- cbind(paises2, paises4)
```

Country	MigracionNeta	Country	RatioNacimientos
Rusia	1.00	España	10.06
España	0.99	Argentina	16.73
Grecia	2.35	Rusia	9.95

Tabla 10.2: *Paises5*

¡Cuidado!. En este caso, `cbind` devuelve una tabla que no sigue los principios `tidy` ya que la variable `Country` aparece repetida.

Para evitar este problema al combinar 2 dataframes se utiliza la función

`merge()`. Esta función cuenta con un argumento `by` donde se indica la variable (o clave) por la que unir los dataframes.

En este caso, la clave por la que se desea unir los dataframes es la variable `Country` y los dataframes solo tienen dos sujetos en común: España y Rusia.

```
merge(paises2,paises4,by="Country")
```

Country	MigracionNeta	RatioNacimientos
España	0.99	10.06
Rusia	1.00	9.95

Tabla 10.3: *Merge de paises2 y paises4*

Dependiendo de los resultados que se quieran obtener hay muchas formas de unir las columnas de un dataframe.

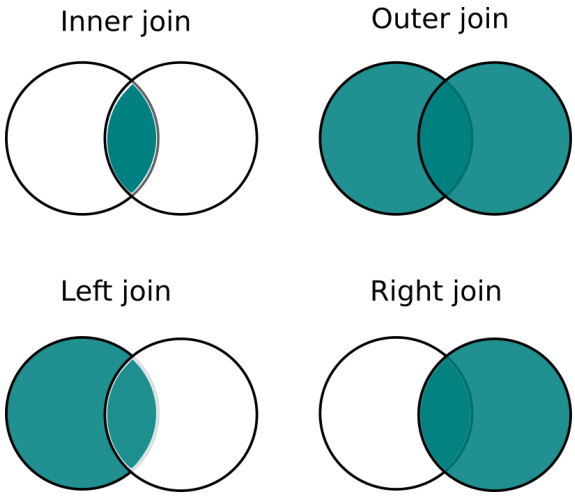


Figura 10.3: *Tipos de join*

Inner Join	merge(df1, df2, by='clave')	solo sujetos comunes
Outer Join	merge(df1, df2, by='clave', all=TRUE)	todos los sujetos
Left outer	merge(df1, df2, by='clave', all.x=TRUE)	todos del izdo. y comunes con el dcho.
Rigth outer	merge(df1, df2, by='clave', all.y=TRUE)	todos del dcho. y comunes con el izdo.

Veamos todas las posibles uniones en las Tablas 10.4, 10.5 y 10.6:

```
merge(paises2, paises4, by='Country', all=T)
```

Country	MigracionNeta	RatioNacimientos
Argentina	NA	16.73
España	0.99	10.06
Grecia	2.35	NA
Rusia	1.00	9.95

Tabla 10.4: *all=T*

```
merge(paises2, paises4, by='Country', all.y=T)
```

Country	MigracionNeta	RatioNacimientos
Argentina	NA	16.73
España	0.99	10.06
Rusia	1.00	9.95

Tabla 10.5: *all.y=T*

```
merge(paises2, paises4, by='Country', all.x=T)
```

Country	MigracionNeta	RatioNacimientos
España	0.99	10.06
Grecia	2.35	NA
Rusia	1.00	9.95

Tabla 10.6: *all.x=T*

10.2.2. Aggregate

La palabra agregar tiene varios significados, entre los que se incluye "añadir unas cosas o personas a otras del mismo tipo o unir varias cosas similares".

`Aggregate()` es una función base de R que sirve para agregar un dataframe utilizando una función especificada en el argumento `FUN` en las columnas del dataframe definidas por el argumento by. El argumento by tiene que ser una lista.

Por ejemplo, el conjunto de datos `mtcars` tiene una columna dedicada a la cilindrada (`cyl`) que toma los valores del conjunto {4,6,8}, como se ve en la Figura

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	**6**	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	**6**	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	**4**	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	**6**	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	**8**	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	**6**	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	**8**	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	**4**	146.7	62	3.69	3.190	20.00	1	0	4	2

Figura 10.4: *Subconjunto de mtcars*

Vamos a utilizar la función `aggregate()` para calcular la media de todas las variables del dataset `mtcars` para cada uno de los valores de la variable `cyl`. Para ellos los argumentos son:

- `mtcars`: el conjunto de datos
- `by = list(cyl)`: una lista con el nombre de la variable por la que queremos aplicar la función
- `FUN = mean`: la función media

```
aggMtcars <-aggregate(mtcars, by = list(mtcars$cyl), FUN = mean)
aggMtcars
```

```
  Group.1  mpg cyl disp    hp drat   wt qsec    vs    am gear carb
1       4 26.7   4  105  82.6 4.07 2.29 19.1 0.909 0.727 4.09 1.55
2       6 19.7   6  183 122.3 3.59 3.12 18.0 0.571 0.429 3.86 3.43
3       8 15.1   8  353 209.2 3.23 4.00 16.8 0.000 0.143 3.29 3.50
```

En la tabla anterior se muestran las medias para **todas las columnas** de `mtcars` agrupándolas por `cyl`.

A continuaciÍon, realizaremos ejercicios en los que se agregarán distintas columnas del conjunto de datos CO2. Una vista reducida del conjunto se ve en la Tabla 10.7.

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
22	Qc1	Quebec	chilled	95	14.2
23	Qc1	Quebec	chilled	175	24.1
43	Mn1	Mississippi	nonchilled	95	10.6
44	Mn1	Mississippi	nonchilled	175	19.2
64	Mc1	Mississippi	chilled	95	10.5
65	Mc1	Mississippi	chilled	175	14.9

Tabla 10.7: *Vista reducida del conjunto de datos CO2*

Calculamos la media de absorción (`uptake`) por origen de planta (`Type`) usando `aggregate()`.

```
aggregate(CO2$uptake, by = list(CO2$Type), FUN=mean)
```

Ahora calculamos el máximo de absorción (`uptake`) por origen de planta y por planta (`Type`, `Plant`) usando `aggregate()`.

```
aggregate(CO2$uptake, by = list(CO2$Type, CO2$Plant), FUN=mean)
```

Construimos una tabla de estadísticos descriptivos personalizada. Se desea obtener la suma, media y desviación estándar de absorción (`uptake`) por tipo de planta (`Type`).

```
f.custom <- function(x){
  c(Suma = sum(x),
    Media = mean(x),
    SD = sd(x))
}
aggregate(CO2$uptake, by = list(CO2$Type), FUN = f.custom)
```

Los valores de entrada en aggregate se pueden especificar también como una fórmula.

Las fórmulas en R relacionan conceptos usando la virgulilla ~. A la izquierda se encuentra la variable independiente y a la derecha la o las dependientes que se relacionan usando el signo más.

Veamos un ejemplo de cómo usar la fórmula en `aggregate`.

```
aggregate(CO2$uptake, by = list(CO2$Type, CO2$Plant), FUN=max)
```

```
       Group.1 Group.2    x
1       Quebec     Qn1 39.7
2       Quebec     Qn2 44.3
3       Quebec     Qn3 45.5
4       Quebec     Qc1 38.7
5       Quebec     Qc3 41.4
6       Quebec     Qc2 42.4
7  Mississippi     Mn3 28.5
8  Mississippi     Mn2 32.4
9  Mississippi     Mn1 35.5
10 Mississippi     Mc2 14.4
11 Mississippi     Mc3 19.9
12 Mississippi     Mc1 22.2
```

```
aggregate(uptake ~ Type+Plant, data=CO2, FUN=max)
```

```
          Type Plant uptake
1       Quebec   Qn1   39.7
2       Quebec   Qn2   44.3
3       Quebec   Qn3   45.5
4       Quebec   Qc1   38.7
5       Quebec   Qc3   41.4
6       Quebec   Qc2   42.4
7  Mississippi   Mn3   28.5
8  Mississippi   Mn2   32.4
9  Mississippi   Mn1   35.5
10 Mississippi   Mc2   14.4
11 Mississippi   Mc3   19.9
12 Mississippi   Mc1   22.2
```

Con esta estructura, se evita llamar al dataframe cada vez que invocas una variable.

Las fórmulas no se pueden usar como entrada de todas las funciones de R, pero son parte fundamental de este lenguaje.

10.2.3. Algunos componentes de la familia Apply: apply(), lapply() y tapply()

Este apartado versa sobre el uso de la función `apply` en R y algunas variantes en los conjuntos de datos.

10.2.3. La familia apply()

La familia `apply()` pertenece al paquete `base` de R y se utiliza para manipular porciones de los datos de una forma repetitiva. Estas funciones permiten cruzar los datos de diversas formas evitando utilizar bucles, que son menos eficientes y el código es más engorroso.

Si no sabes lo que son los bucles en programación, no son necesarios para seguir el material, pero puedes completar la información con https://www.datacamp.com/community/tutorials/tutorial-on-loops-in-r.

La familia incluye las fuciones `apply()`, `lapply()` , `sapply()`, `vapply()`, `mapply()`, `rapply()`, y `tapply()`. Se utilizarán dependiendo de la estructura de los datos que se quieran manipular y del formato de salida que se necesite.

10.2.3. apply

`apply()` opera en vectores y matrices. Sus argumentos básicos son `apply(X, MARGIN, FUN)`, donde:

- X es un vector o una matriz
- MARGIN es una variable que define cómo se aplica la función:
  - `MARGIN = 1 --> por filas`
  - `MARGIN = 2 --> por columnas`
  - `MARGIN = c(1,2) por filas y columnas`
- FUN es la función que se desea aplicar a los datos

Como todas las filas o todas las columnas se van a ver afectadas por la misma función, éstas tienen que ser del mismo tipo. Para ello crearemos un

dataframe muy simple:

```
df<- matrix(c(1:10, 1:10, 1:10), nrow = 10, ncol = 3)
df
```

```
      [,1] [,2] [,3]
 [1,]    1    1    1
 [2,]    2    2    2
 [3,]    3    3    3
 [4,]    4    4    4
 [5,]    5    5    5
 [6,]    6    6    6
 [7,]    7    7    7
 [8,]    8    8    8
 [9,]    9    9    9
[10,]   10   10   10
```

Y, a continuación mostramos la suma por filas y por columnas respectivamente utilizando la función `apply`:

```
apply(df, 1, sum)
```

```
 [1]  3  6  9 12 15 18 21 24 27 30
```

```
apply(df, 2, sum)
```

```
[1] 55 55 55
```

La función `sum` ya existe en R, pero en algunas ocasiones será útil definir nuestras propias funciones. Utilizamos la función `st.err` definida a continuación para calcular el error estándar por columnas de `df`:

```
st.err <- function(x){
  sd(x)/sqrt(length(x))
}

apply(df, 2 , st.err)
```

```
[1] 0.957 0.957 0.957
```

Utilizamos la función apply para calcular el rango de las variables numéri-

cas del dataset CO2:

```
apply (CO2[c(4,5)], 2, range)
```

```
     conc uptake
[1,]   95    7.7
[2,] 1000   45.5
```

10.2.3. lapply()

Veamos el funcionamiento de la función `lapply()` con este simple ejemplo:

```
lapply(1:3, function(x) x^2)
```

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

[[2]]
[1] 4

[[3]]
[1] 9
```

Lapply devuelve una lista de la misma longitud de su argumento de entrada.

Los dataframes son listas de vectores del mismo tamaño, por lo tanto, `lapply()` resulta muy útil para aplicar una función a las columnas de un dataframe.

El conjunto de datos starwarsjuguetes presenta la siguiente codificación:

genero	1:femenino, 2:masculino
jedi	1:jedi, 2:no jedi
especie	1:humano, 2:droide, 3:yoda, 4:wookiee
arma	1:dinamita, 2:rifleBayesta, 3:sable de luz, 4:sinArmas

```
starwars <- read.table("starwarsjuguetes.csv", sep=";", head=T)
summary(starwars)
```

```
   nombre              genero         altura          peso
 Length:8           Min.   :1.00   Min.   :0.66   Min.   : 17.0
 Class :character   1st Qu.:2.00   1st Qu.:1.36   1st Qu.: 44.8
 Mode  :character   Median :2.00   Median :1.70   Median : 76.0
                    Mean   :1.88   Mean   :1.55   Mean   : 64.9
                    3rd Qu.:2.00   3rd Qu.:1.80   3rd Qu.: 77.8
                    Max.   :2.00   Max.   :2.28   Max.   :112.0
     jedi         especie          arma
 Min.   :1.00   Min.   :1.00   Min.   :1.00
 1st Qu.:1.00   1st Qu.:1.00   1st Qu.:1.75
 Median :2.00   Median :1.50   Median :3.00
 Mean   :1.62   Mean   :1.88   Mean   :2.62
 3rd Qu.:2.00   3rd Qu.:2.25   3rd Qu.:3.25
 Max.   :2.00   Max.   :4.00   Max.   :4.00
```

Las variables género, `jedi`, `especie` y `arma` son categóricas, sin embargo, se tratan como numéricas ya que no lo hemos especificado de otra forma al leer el conjunto de datos. Además de convertirlas de forma individual en factores, podemos hacerlo de forma más inteligente, esto es, utilizando la función `lapply()` en las columnas seleccionadas:

```
starwars[c(2,5:7)] <- lapply(starwars[c(2,5:7)], as.factor)
summary(starwars)
```

```
   nombre          genero     altura          peso        jedi
 Length:8          1:1    Min.   :0.66   Min.   : 17.0   1:3
 Class :character  2:7    1st Qu.:1.36   1st Qu.: 44.8   2:5
 Mode  :character         Median :1.70   Median : 76.0
                          Mean   :1.55   Mean   : 64.9
                          3rd Qu.:1.80   3rd Qu.: 77.8
                          Max.   :2.28   Max.   :112.0
 especie arma
 1:4     1:2
 2:2     2:1
 3:1     3:3
 4:1     4:2
```

Utilizamos la función `levels()` para renombrar los niveles de una de las variables categóricas de acuerdo a la tabla de codificación anterior y mostramos un gráfico de barras de ella:

```
levels(starwars$genero) <- c("fem", "masc")
%barplot(table(starwars$genero))
```

10.2.3. tapply()

La función `tapply` es útil para partir un vector por grupos, realizar cálculos en cada uno de esos grupos y devolver los resultados en forma de tabla. Se pueden utilizar más de un factor para definir los grupos.

```
tapply(CO2$uptake, CO2$Type, mean)
```

```
     Quebec Mississippi
       33.5        20.9
```

10.2.4. Diferencias entre aggregate y tapply()

`Aggregate` está diseñado para aplicar una función a múltiples columnas y devolver un dataframe con una fila por categoría.

`Tapply` está diseñado para utilizar como entrada un solo vector y los resultados se devuelven en forma de tabla.

```
aggregate(CO2$uptake, by = list(CO2$Type, CO2$Plant), FUN=max)
```

```
       Group.1 Group.2    x
1       Quebec     Qn1 39.7
2       Quebec     Qn2 44.3
3       Quebec     Qn3 45.5
4       Quebec     Qc1 38.7
5       Quebec     Qc3 41.4
6       Quebec     Qc2 42.4
7  Mississippi     Mn3 28.5
8  Mississippi     Mn2 32.4
9  Mississippi     Mn1 35.5
```

```
10 Mississippi     Mc2 14.4
11 Mississippi     Mc3 19.9
12 Mississippi     Mc1 22.2
```

```
tapply(CO2$uptake, INDEX=list(CO2$Type, CO2$Plant), max)
```

```
             Qn1  Qn2  Qn3  Qc1  Qc3  Qc2  Mn3  Mn2  Mn1  Mc2  Mc3
Quebec      39.7 44.3 45.5 38.7 41.4 42.4   NA   NA   NA   NA   NA
Mississippi   NA   NA   NA   NA   NA   NA 28.5 32.4 35.5 14.4 19.9
             Mc1
Quebec        NA
Mississippi 22.2
```

10.3. Tidyverse: dplyr y tidyr

10.3.1. DplyR

El popular programador de R **Hadley Wickham** ha creado numerosos paquetes entre los que se encuentran los dos que vamos a trabajar en este apartado: `dplyr`, `tidyr`. Aquí podéis encontrar el manifiesto del universo tidy o `tidyverse` donde los paquetes se construyen para que tengan sintonía los unos con los otros: https://cran.rstudio.com/web/packages/tidyverse/vignettes/manifesto.html

El paquete `dplyr` contiene una colección de funciones para realizar operaciones de manipulación de datos comunes como: filtrar por fila, seleccionar columnas específicas, reordenar filas, añadir nuevas filas y agregar datos.

Además, también contiene funciones que sirven para realizar una tarea que se denomina: *split-apply-combine*. Esto es, básicamente, calcular estadísticos (y otros) por grupos.

- Split: cortar por grupo
- Apply: aplicar la función
- Combine: combinar los grupos para dotarles de una estructura.

Esta tarea es muy común, ya la hemos visto y se puede realizar también con funciones que ya conocemos: aggregate y las de la familia apply.

La mayor ventaja de dplyr es que está orientado a dataframes. Además tiene otras ventajas que iremos descubriendo.

Cargamos el paquete `dplyr` en el espacio de trabajo:

```
library("dplyr")
```

Muchas de estas operaciones se pueden realizar de forma alternativa con funciones básicas de R, sin embargo, la sintaxis de dplyr:

- Es más sencilla
- Es más consistente
- Está enfocada a datasets en lugar de a vectores

Además, combinándolo con la sinxtaxis `pipes` se extrae todo su pontecial [69].

10.3.1. Funciones importantes a recordar

Los siguientes son los verbos imprescindibles del paquete:

- `select()`: seleccionar columnas
- `filter()`: filtrar filas
- `arrange()`: reorderar filas
- `mutate()`: crear nuevas columnas

- `summarise()`: resumir los valores
- `group_by()`: permite operaciones por grupos del tipo *split-apply-combine*

Seleccionamos las 3 primeras columnas del dataset `mtcars` y mostramos la cabecera:

```
head( select( mtcars, mpg, cyl, disp ) )
```

```
                  mpg cyl disp
Mazda RX4         21.0   6  160
Mazda RX4 Wag     21.0   6  160
Datsun 710        22.8   4  108
Hornet 4 Drive    21.4   6  258
Hornet Sportabout 18.7   8  360
Valiant           18.1   6  225
```

Utilizando la ayuda `?select_helpers`, se pueden ver todas las opciones que hay para realizar selecciones.

```
?select_helpers
```

A continuación, algunos ejemplos:

- Seleccionamos las columnas que empiezan por d

```
select( mtcars, starts_with( "d" ) )
```

- Seleccionamos las columnas que terminan por p

```
select( mtcars, ends_with( "p" ) )
```

Otras funcionalidades con `select` incluyen la eliminación de columnas utilizando el signo menos y la selección de columnas en cuyo nombre exista cierta cadena de caracteres:

```
head(select( mtcars, -drat, -am ))
```

```
                  mpg cyl disp  hp   wt qsec vs gear carb
Mazda RX4         21.0   6  160 110 2.62 16.5  0    4    4
Mazda RX4 Wag     21.0   6  160 110 2.88 17.0  0    4    4
Datsun 710        22.8   4  108  93 2.32 18.6  1    4    1
```

```
Hornet 4 Drive    21.4   6  258 110 3.21 19.4  1    3    1
Hornet Sportabout 18.7   8  360 175 3.44 17.0  0    3    2
Valiant           18.1   6  225 105 3.46 20.2  1    3    1
```

```
head(select( mtcars, contains( "a" ) ))
```

```
                  drat am gear carb
Mazda RX4         3.90  1    4    4
Mazda RX4 Wag     3.90  1    4    4
Datsun 710        3.85  1    4    1
Hornet 4 Drive    3.08  0    3    1
Hornet Sportabout 3.15  0    3    2
Valiant           2.76  0    3    1
```

Veamos ahora la función filter:

```
head(filter( mtcars, mpg > 20, gear == 4))
```

```
                mpg cyl  disp  hp drat   wt qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.62 16.5  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.88 17.0  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.32 18.6  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.19 20.0  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.15 22.9  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.20 19.5  1  1    4    1
```

Seleccionamos los sujetos con tipo de transmisión (`am`) 1 que, además, tienen 6 cilindros o menos.

```
filter( mtcars, am == 1, cyl<=6)
```

Seleccionamos los sujetos que bien consumen menos de 21 mpg o bien tienen menos de 3 carburantes (`carb`) y menos de 4 engranajes (`gear`)

```
filter( mtcars, mpg<21 | carb <=2, gear <4)
```

Veamos otras funcionalidadesd del paquete `dplyr`:

Ordenamos por cilindrada (`cyl`) y por desplazamiento (`disp`):

```
arrange(mtcars, cyl, disp)
```

Creamos una nueva columna que indique los kilogramos que pesa el coche, sabiendo que 1 libra = 0.45 kg. La variable `wt` indica el peso en libras:

```
mutate(mtcars, kg = 1000*wt*0.45)
```

La función `summarise()` agrupa los valores en una tabla de acuerdo a la función que indiquemos. Calculamos la media de `disp` usando la función summarise:

```
summarise(mtcars, mean = mean(disp))
```

Habitualmente se utiliza junto a `group_by()`:

```
summarise(group_by(mtcars, cyl), max = max(disp))
```

```
# A tibble: 3 x 2
    cyl   max
  <dbl> <dbl>
1     4  147.
2     6  258
3     8  472
```

10.3.2. *Piping* datos

Pipes utiliza el resultado de una función como entrada de la siguiente. La función *pipe*, que pertenece al paquete `magrittr` se escribe así:

%>%.

Puede parecer rara al principio, pero necesitábamos una función definida

por >, que implica que es una cadena (una sucesión de órdenes) y en R, para que un símbolo defina una función debe estar escrito entre porcentajes.

```
mtcars %>%
  select( mpg:disp )%>%
  head
```

```
                   mpg cyl disp
Mazda RX4         21.0   6  160
Mazda RX4 Wag     21.0   6  160
Datsun 710        22.8   4  108
Hornet 4 Drive    21.4   6  258
Hornet Sportabout 18.7   8  360
Valiant           18.1   6  225
```

Para entenderlo bien, vamos a leer en voz alta la secuencia anterior:

Cogemos el dataset `mtcars`, seleccionamos la columnas de mpg a disp e imprimimos las primeras filas

Como vemos, sucesivamente se encadenan funciones `dplyr`, lo cual nos permite manipular los datos paso a paso, al igual que pensamos, evitando la creación de variables temporales por en medio para capturar la salida. Esto funciona porque **la salida de las funciones `dplyr` es un dataframe** y **el primer argumento de las funciones `dplyr` es un dataframe**.

Veamos esta secuencia:

```
head(select(select(mtcars, contains("a")), -drat, -am))
```

```
                  gear carb
Mazda RX4            4    4
Mazda RX4 Wag        4    4
Datsun 710           4    1
Hornet 4 Drive       3    1
Hornet Sportabout    3    2
Valiant              3    1
```

La reescribimos utilizando pipes:

```
mtcars %>%
```

```
  select(-drat,-am )%>%
  select(contains("a"))%>%
  head
```

Hacerlo sin pipes no es un error. El problema es que hay que leerlo de dentro a fuera y es sencillo equivocarse (paréntesis que se pierden, argumentos que se separan de su función...). Además, esto es un ejemplo bastante sencillo pero para operaciones complejas, escribir código tal y como lo pensamos, esto es, de forma secuencial puede ser más sencillo.

Otra secuencia de funciones es la siguiente

```
mtcars_filtered = filter(mtcars, wt > 1.5)
mtcars_grouped = group_by(mtcars_filtered, cyl)
summarise(mtcars_grouped, mn = mean(mpg), sd = sd(mpg))
```

que se puede reescribir con pipes como sigue:

```
mtcars %>%
  filter(wt > 1.5)%>%
  group_by(cyl) %>%
  summarise(mean = mean(disp), n = n())
```

Otro ejemplo de cómo reescribir una secuencia en pipes es el que sigue:

```
my_cars <- mtcars[, c(1:5, 7)]
my_cars <- my_cars[my_cars$disp > mean(my_cars$disp), ]
my_cars <- colMeans(my_cars)
my_cars
```

```
my_cars <- mtcars %>%
  select(c(1:5, 7)) %>%
  filter(disp > mean(disp)) %>%
  colMeans()
my_cars
```

Utilizamos la función `cor()` para calcular la correlación entre las variables de mtcars.

```
mtcars %>% cor()
```

```
        mpg    cyl   disp     hp    drat     wt    qsec     vs
mpg   1.000 -0.852 -0.848 -0.776  0.6812 -0.868  0.4187  0.664
cyl  -0.852  1.000  0.902  0.832 -0.6999  0.782 -0.5912 -0.811
disp -0.848  0.902  1.000  0.791 -0.7102  0.888 -0.4337 -0.710
hp   -0.776  0.832  0.791  1.000 -0.4488  0.659 -0.7082 -0.723
drat  0.681 -0.700 -0.710 -0.449  1.0000 -0.712  0.0912  0.440
wt   -0.868  0.782  0.888  0.659 -0.7124  1.000 -0.1747 -0.555
qsec  0.419 -0.591 -0.434 -0.708  0.0912 -0.175  1.0000  0.745
vs    0.664 -0.811 -0.710 -0.723  0.4403 -0.555  0.7445  1.000
am    0.600 -0.523 -0.591 -0.243  0.7127 -0.692 -0.2299  0.168
gear  0.480 -0.493 -0.556 -0.126  0.6996 -0.583 -0.2127  0.206
carb -0.551  0.527  0.395  0.750 -0.0908  0.428 -0.6562 -0.570
          am   gear    carb
mpg   0.5998  0.480 -0.5509
cyl  -0.5226 -0.493  0.5270
disp -0.5912 -0.556  0.3950
hp   -0.2432 -0.126  0.7498
drat  0.7127  0.700 -0.0908
wt   -0.6925 -0.583  0.4276
qsec -0.2299 -0.213 -0.6562
vs    0.1683  0.206 -0.5696
am    1.0000  0.794  0.0575
gear  0.7941  1.000  0.2741
carb  0.0575  0.274  1.0000
```

Por último, introducimos la función `n()` que es específica de `summarise()`, `mutate()` y `filter()` y sirve para contar la frecuencia de apariciones.

```
mtcars %>%
  filter(wt > 1.5)%>%
  group_by(cyl) %>%
  summarise(mean = mean(disp), n = n())
```

```
# A tibble: 3 x 3
    cyl  mean     n
  <dbl> <dbl> <int>
1     4  105.    11
2     6  183.     7
```

```
3     8  353.     14
```

10.3.3. TidyR

Cargamos el paquete `tidyr` en el espacio de trabajo:

```
library("tidyr")
```

El paquete `tidyr` se centra en dos funciones:

- `pivot_longer()`
- `pivot_wider()`

Y posee otras dos también muy importantes:

- `separate()`
- `unite()`

10.3.3. Pares key-value

Para comenzar, vamos a exponer con un ejemplo la idea de pares *key-value* ya que es una terminología utilizada recurrentemente en las ayudas relacionadas con este paquete.

El siguiente conjunto de datos muestra el consumo de electricidad diario en 3 estancias de una casa:

```
set.seed(1)
consumo <- data.frame(
```

```
   fecha = as.Date('2009-01-01') + 0:9,
   habitacion = rnorm(10, 20, 1),
   despacho= rnorm(10, 20, 2),
   cocina = rnorm(10, 20, 4)
 )
consumo
```

```
        fecha habitacion despacho cocina
1  2009-01-01       19.4     23.0   23.7
2  2009-01-02       20.2     20.8   23.1
3  2009-01-03       19.2     18.8   20.3
4  2009-01-04       21.6     15.6   12.0
5  2009-01-05       20.3     22.2   22.5
6  2009-01-06       19.2     19.9   19.8
7  2009-01-07       20.5     20.0   19.4
8  2009-01-08       20.7     21.9   14.1
9  2009-01-09       20.6     21.6   18.1
10 2009-01-10       19.7     21.2   21.7
```

La primera fila muestra 3 datos de consumo, una por estancia. Los números son los valores. ¿Qué valor van con qué estancia? Para saberlo miramos a los nombres de las columnas. Estos nombres son la *clave* o *key*. El valor o *value* 19.37355 está relacionado con la clave **habitación**, por lo tanto es un par **key-value**. Para esa primera fila, hay tres pares *key-value* que están relacionados de forma única con la fecha 2009-01-01.

10.3.3. Formatos de tabla ancha (*wide*) y larga (*long*)

Wide y *Long* son términos que se utilizan para describir dos formas de presentar los datos en una tabla o en un dataframe.

- *Wide*: Cada variable de los datos se presenta en una sola columna

persona	edad	peso
Alba	25	65
Perico	26	70
Maripili	27	83

Tabla 10.8: *Ejemplo de tabla wide*

- *Long*: Una columna contiene todos los valores y otra columna contiene los nombres de todas las variables.

persona	variable	valor
Alba	edad	25
Alba	peso	65
Perico	edad	26
Perico	peso	70
Maripili	edad	27
Maripili	peso	83

Tabla 10.9: *Ejemplo de tabla long*

10.3.3. Pivot_longer

La función `pivot_longer` sustitye a `gather`. Ésta coge muchas columnas y las hace colapsar en los pares key-value, duplicando las columnas como se necesite.

Ahora habrá una variable que estará formada por los nombres de las columnas y todos los valores se encontrarán en una única fila.

Existe la opción de colapsar todas las columnas con el el parámetro `cols=everything()`

```
mtcarsG <- mtcars  %>% pivot_longer(cols = everything())
```

O especificar aquellas columnas que se deseen colapsar, para pasar de formato long a wide:

```
pivot_longer(mtcars, cols = c("mpg", "cyl", "carb"))
```

```
## # A tibble: 96 x 10
##     disp    hp  drat    wt  qsec    vs    am  gear name  value
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <dbl>
##  1   160   110  3.9   2.62  16.5     0     1     4 mpg    21
##  2   160   110  3.9   2.62  16.5     0     1     4 cyl     6
##  3   160   110  3.9   2.62  16.5     0     1     4 carb    4
##  4   160   110  3.9   2.88  17.0     0     1     4 mpg    21
##  5   160   110  3.9   2.88  17.0     0     1     4 cyl     6
##  6   160   110  3.9   2.88  17.0     0     1     4 carb    4
##  7   108    93  3.85  2.32  18.6     1     1     4 mpg    22.8
##  8   108    93  3.85  2.32  18.6     1     1     4 cyl     4
##  9   108    93  3.85  2.32  18.6     1     1     4 carb    1
## 10   258   110  3.08  3.22  19.4     1     0     3 mpg    21.4
## # ... with 86 more rows
```

Sin embargo, en otras ocasiones no será tan directo. Si se hace para el conjunto consumo se poroducirá un error.

```
consumo  %>% pivot_longer( cols = everything())
```

La columna fecha no queremos que se una a los atributos, sino que se mantenga como columna individual, por lo que la podemos quitar de los elementos a tener en cuenta.

```
consumosPorEstancia <- consumo %>% pivot_longer(cols = -c(fecha))
```

10.3.3. Pivot_wider

Con la función pivot_wider, que sustituye a spread podemos volver a estado anterior:

```
head(consumosPorEstancia)
```

```
## # A tibble: 6 x 3
##   fecha      name       value
##   <date>     <chr>      <dbl>
## 1 2009-01-01 habitacion  19.4
## 2 2009-01-01 despacho    23.0
## 3 2009-01-01 cocina      23.7
```

```
## 4 2009-01-02 habitacion  20.2
## 5 2009-01-02 despacho    20.8
## 6 2009-01-02 cocina      23.1
```

```
pivot_wider(consumosPorEstancia, names_from = name, values_from = value)
```

```
## # A tibble: 10 x 4
##    fecha      habitacion despacho cocina
##    <date>          <dbl>    <dbl>  <dbl>
##  1 2009-01-01       19.4     23.0   23.7
##  2 2009-01-02       20.2     20.8   23.1
##  3 2009-01-03       19.2     18.8   20.3
##  4 2009-01-04       21.6     15.6   12.0
##  5 2009-01-05       20.3     22.2   22.5
##  6 2009-01-06       19.2     19.9   19.8
##  7 2009-01-07       20.5     20.0   19.4
##  8 2009-01-08       20.7     21.9   14.1
##  9 2009-01-09       20.6     21.6   18.1
## 10 2009-01-10       19.7     21.2   21.7
```

Para pasar mtcarsG a formato long de nuevo

Vemos que hay un problema, y es que necesita un conjunto de identificadores.

Para que funcione, reharemos el conjunto `mtcarsG` a partir de mtcars añadiendo lo que necesites ahora para poder aplicar `pivot_wider`

```
# vamos a crear una columna identificadora que se denomina `car`
#y que tiene los nombres de las filas
mtcars2 <- mtcars
mtcars2$car <- rownames(mtcars2)
# a este conjunto le aplicamos el pivot_longer pero quitando
# el identificador
mtcarsG <- mtcars2  %>% pivot_longer(cols = -c(car))

# y ahora sí podemos revertirlo (gracias a que hay una columna de
# identificadores denomiada car)
mtcarsSpread <- mtcarsG %>% pivot_wider(names_from = name, values_from=value)
```

```
head(mtcarsSpread)
```

```
## # A tibble: 6 x 12
##   car            mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mazda RX4     21       6   160   110  3.9   2.62  16.5     0     1     4     4
## 2 Mazda RX4 W~  21       6   160   110  3.9   2.88  17.0     0     1     4     4
## 3 Datsun 710    22.8     4   108    93  3.85  2.32  18.6     1     1     4     1
## 4 Hornet 4 Dr~  21.4     6   258   110  3.08  3.22  19.4     1     0     3     1
## 5 Hornet Spor~  18.7     8   360   175  3.15  3.44  17.0     0     0     3     2
## 6 Valiant       18.1     6   225   105  2.76  3.46  20.2     1     0     3     1
```

Con la función spread podemos volver a estado anterior:

```
head(consumosPorEstancia)
```

```
       fecha   estancia cons
1 2009-01-01 habitacion 19.4
2 2009-01-02 habitacion 20.2
3 2009-01-03 habitacion 19.2
4 2009-01-04 habitacion 21.6
5 2009-01-05 habitacion 20.3
6 2009-01-06 habitacion 19.2
```

```
spread(consumosPorEstancia, estancia, cons)
```

```
        fecha cocina despacho habitacion
1  2009-01-01   23.7     23.0       19.4
2  2009-01-02   23.1     20.8       20.2
3  2009-01-03   20.3     18.8       19.2
4  2009-01-04   12.0     15.6       21.6
5  2009-01-05   22.5     22.2       20.3
6  2009-01-06   19.8     19.9       19.2
7  2009-01-07   19.4     20.0       20.5
8  2009-01-08   14.1     21.9       20.7
9  2009-01-09   18.1     21.6       20.6
10 2009-01-10   21.7     21.2       19.7
```

Si pasamos mtcarsG a formato long de nuevo como sigue:

```
head(mtcarsG)
mtcarsSpread <- mtcarsG %>% spread(key = atributo, value=valor)
```

Encontramos el siguiente error "Each row of output must be indetified by a unique combination of keys".

Necesita una columna con identificadores para que funcione correctamente. Creamos una columna identificadora que se denomina car y que tiene los nombres de las filas

```
mtcars2 <- mtcars
mtcars2$car <- rownames(mtcars2)
```

A a este conjunto le aplicamos el gather pero quitando el identificador, y ahora sí podemos revertirlo (gracias a que hay una columna de identificadores denomiada car).

```
mtcarsG <- mtcars2 %>% gather(atributo, valor,-car)
mtcarsSpread <- mtcarsG %>% spread(key = atributo, value=valor)
head(mtcarsSpread)
```

```
                 car am carb cyl disp drat gear  hp  mpg qsec vs   wt
1        AMC Javelin  0    2   8  304 3.15    3 150 15.2 17.3  0 3.44
2 Cadillac Fleetwood  0    4   8  472 2.93    3 205 10.4 18.0  0 5.25
3         Camaro Z28  0    4   8  350 3.73    3 245 13.3 15.4  0 3.84
4  Chrysler Imperial  0    4   8  440 3.23    3 230 14.7 17.4  0 5.34
5          Datsun 710  1    1   4  108 3.85    4  93 22.8 18.6  1 2.32
6   Dodge Challenger  0    2   8  318 2.76    3 150 15.5 16.9  0 3.52
```

10.3.3. Unite

Sirve para **pegar** múltiples columnas en una sola:

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat   wt qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.62 16.5  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.88 17.0  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.32 18.6  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.21 19.4  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.44 17.0  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.46 20.2  1  0    3    1
```

```
mtcarsU <- unite(mtcars, "vs_am", c("vs","am"))
head(mtcarsU)
```

```
                   mpg cyl disp  hp drat   wt qsec vs_am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.62 16.5   0_1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.88 17.0   0_1    4    4
Datsun 710        22.8   4  108  93 3.85 2.32 18.6   1_1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.21 19.4   1_0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.44 17.0   0_0    3    2
Valiant           18.1   6  225 105 2.76 3.46 20.2   1_0    3    1
```

Juntamos las dos columnas `vs` y `am` utilizando los dos puntos como separador y, además, evitando que se borren las columnas originales

```
mtcarsU2 <- unite(mtcars, "vs:am", c("vs","am"), sep=":", remove=F)
head(mtcarsU2)
```

10.3.3. Separate

Convierte una única columa en múltiples separando los caracteres que componen a la inicial.

Para ilustrar la función `separate()` vamos a utilizar un dataset sencillo:

```
df <- read.table("longlat.txt",
                                  sep=";", head=T)
df
```

```
        Date Time Accident.Type                Location
1 07/01/2012 1630            PD  (39.26699, -76.560642)
2 07/02/2012 1229            PD (39.000549, -76.399312)
3 07/02/2012 1229            PD  (39.00058, -76.399267)
4 07/02/2012  445            PI   (39.26367, -76.56648)
5 07/02/2012  802            PD (39.240862, -76.599017)
6 07/02/2012  832            PD   (39.27022, -76.63926)
```

La intención es separar la columna `Location` en dos: `lat` y `long`. Para ello, primero eliminamos los paréntesis como sigue:

```
df2 <-df
```

```
df2$Location <- gsub("[()]", "", df2$Location)
df2
```

```
        Date Time Accident.Type              Location
1 07/01/2012 1630            PD  39.26699, -76.560642
2 07/02/2012 1229            PD 39.000549, -76.399312
3 07/02/2012 1229            PD  39.00058, -76.399267
4 07/02/2012  445            PI   39.26367, -76.56648
5 07/02/2012  802            PD 39.240862, -76.599017
6 07/02/2012  832            PD   39.27022, -76.63926
```

Fíjese el lector que hemos creado df2, un dataset intermedio (no es necesario, solo por claridad).

Y después utilizamos la función `separate()`, que tiene los siguientes argumentos

- `col`: la columna a partir
- `into`: los nombres que van a tener las nuevas columnas
- `sep`: el carácter o expresión que separa

Lo intentamos con la longitud y latitud:

```
separate(df2, col = Location, into = c("lat","long"), sep = ",")
```

```
        Date Time Accident.Type       lat        long
1 07/01/2012 1630            PD  39.26699  -76.560642
2 07/02/2012 1229            PD 39.000549  -76.399312
3 07/02/2012 1229            PD  39.00058  -76.399267
4 07/02/2012  445            PI  39.26367   -76.56648
5 07/02/2012  802            PD 39.240862  -76.599017
6 07/02/2012  832            PD  39.27022   -76.63926
```

11 Trabajar con Likert

Francisco Javier Ibáñez López

11.1. Introducción

La finalidad de este capítulo es aconsejar y orientar a los investigadores en el diseño, validación, interpretación y análisis de cuestionarios elaborados con el fin de evaluar la opinión y el conocimiento sobre distintas temáticas de un determinado grupo de personas.

Los cuestionarios son instrumentos de investigación muy empleados hoy en día y que tienen por objetivo identificar el grado de acuerdo o desacuerdo de los participantes con cada una de las preguntas que se plantean. El tipo de cuestionario más empleado en Ciencias Sociales suele ser la escala tipo Likert [70], ya que a través de cuestiones con una escala ordenada y unidimensional como respuesta, permite medir actitudes y conocer el grado de aceptación sobre una determinada afirmación.

Este tipo de escala surgió en 1932, cuando Rensis Likert (1903-1981) publicó un informe en el que explicaba cómo usar un tipo de instrumento para

Cómo citar este capítulo: Ibáñez López F. J. (2022). Trabajar con Likert. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (503-532). Editum. Ediciones de la Universidad de Murcia.

la medición de actitudes [71]. En este método se supone que todos los ítems miden con la misma intensidad la actitud, la opinión, el comportamiento o las características que se desean medir y es el sujeto el que asigna una puntuación, normalmente de 1 a 5, en función de su punto de vista sobre la afirmación que sugiere el ítem [72]. La actitud final que se asigna al sujeto será la media de la puntuación que este otorgue a cada uno de los ítems del cuestionario.

Con respecto a cómo validar un cuestionario, se expondrán diferentes tipos de validación, como la validación de contenido por expertos, o la evaluación de la fiabilidad y la consistencia interna. En este sentido, siempre se recomienda hacer un estudio piloto, para poder comprobar la adecuación de los ítems empleados a la correcta resolución de los problemas de investigación planteados y poder corregir posibles defectos de forma y contenido antes de lanzarse a realizar el trabajo de campo y tener todos los datos recogidos.

Además, en función del marco y contexto en el que se plantea la investigación, se debe calcular el tamaño muestral conveniente para que los resultados puedan ser considerados como significativos, siempre y cuando se cumplan los principios de representatividad y aleatoriedad en la población analizada dentro del estudio.

Por otro lado, para la validación de constructo, además del Análisis Factorial (AF) cuyas bases y completa ejecución se desarrollan en este capítulo, se llevan a cabo y se presentan modelajes mediante técnicas más avanzadas, como las Ecuaciones Estructurales (SEM), técnica ampliamente utilizada en los últimos años y demandada cada vez más por los investigadores. Existen librerías relacionadas con los SEM, `semPlot` o `semTools` entre otras, que nos permiten generar figuras en las que se representan las covarianzas entre las variables latentes y la influencia que ejerce cada variable latente sobre sus respectivas variables observadas. En ellas, también se representa el error asociado a cada variable observada. Durante el desarrollo de este capítulo se avanzará en el conocimiento y la formación en estas nuevas opciones de representación y análisis con el uso del software *libre R*.

Por otro lado, dentro del análisis de datos e interpretación de los resultados estadísticos, se emplearán librerías como `likert`, `psych`, `lavaan` o `psy` como herramientas que pueden ayudar a conseguir gráficos descriptivos muy potentes, no solo de todos los ítems por separado, sino incluso clasificados según variables sociodemográficas.

Por último, se detalla cómo ejecutar la inferencia estadística sobre los ítems buscando diferencias significativas en función de las variables sociodemográficas.

Es importante, por tanto, realizar un proceso exhaustivo en la elaboración, recogida y procesamiento de datos cuyo instrumento de campo sea un cuestionario. Así, es necesario realizar una correcta validación del instrumento utilizado, que permitirá obtener resultados más concluyentes y fiables. Además, contando con el potente software estadístico R, se pueden implementar presentaciones, análisis descriptivos, análisis estadísticos y análisis de fiabilidad de los datos más potentes y elaborar figuras de mayor calidad gráfica en comparación con otros paquetes estadísticos y que pueden resultan de gran utilidad en la elaboración de informes.

11.2. Consejos para la elaboración de una escala Likert

[73] ofrece una breve lista de las características descriptivas básicas que deben presentare estas escalas:

1. La escala contiene varios ítems.
2. Los niveles de respuesta están dispuestos horizontalmente.
3. Los niveles de respuesta están fijados con enteros consecutivos.
4. Los niveles de respuesta también son fijados con etiquetas verbales que connotan graduaciones más o menos espaciadas uniformemente.
5. Las etiquetas verbales son bivalentes y simétricas sobre un medio neutro.
6. En el uso de una escala Likert, la escala siempre mide la actitud en términos de nivel de acuerdo/desacuerdo con una declaración de objetivos.

Se debe tener en cuenta que las escalas que utilizan alternativas de respuesta que no están vinculadas con el acuerdo o desacuerdo con los ítems, no son escalas Likert en sentido original. No obstante, es frecuente que se les denomine escalas tipo Likert por generalización [74]. Según [75], en la construcción de una escala tipo Likert, se deberían contemplar las siguientes fases:

1. Preparación de los ítems iniciales.
2. Administración de ítems a una muestra de sujetos (pilotaje).
3. Asignación de las puntuaciones a los ítems.

4. Asignación de las puntuaciones a los sujetos.
5. Análisis y selección de los ítems.
6. Análisis de la fiabilidad de la escala.
7. Preparación de la aplicación de la versión final de la escala.

Todo ello debe realizarse teniendo en cuenta los siguientes criterios:

1. Evitar los ítems que se refieren al pasado en lugar de al presente.
2. No usar ítems que den demasiada información sobre hechos o aquellos que puedan ser interpretados como tales.
3. Eludir los ítems ambiguos o que hacen referencia a dos términos diferentes.
4. Evitar los ítems irrelevantes con respecto a la actitud que quiere medir.
5. Los ítems en la escala deben formularse según expresen actitudes o juicios favorables o desfavorables con respecto a la actitud.
6. Eliminar los ítems con los cuales todos o prácticamente nadie concuerda.
7. Realizar una formulación con un lenguaje claro, simple y directo.
8. Solamente en casos excepcionales deberían exceder las 20 palabras.
9. Deben contener sólo una frase lógica.
10. Omitir los ítems que incluyan palabras como "todos", "siempre" o "nadie".
11. De ser posible deben formularse con frases simples y no compuestas.
12. Usar palabras que el participante pueda comprender.
13. Evitar las negaciones, más aún las dobles negaciones.
14. Combinar ítems formulados positivamente y negativamente en una proporción aproximada de 50 % positivos, 50 % negativos.

Tradicionalmente, las encuestas siempre se realizaban en papel y de forma presencial, con el consiguiente despliegue de medios materiales y humanos para su implementación. Hoy en día, con las nuevas tecnologías, los cuestionarios suelen ser administrados por medios telemáticos, lo que permite ahorrar tiempo en recopilar la información y hacerlo de una forma más rápida y eficiente. Además, se consigue que las encuestas lleguen a un mayor número de sujetos por medio del correo electrónico, las páginas web o las redes sociales. Así, existen diferentes plataformas para diseñar y administrar un cuestionario tipo escala Likert:

- Survey Monkey
- Survio
- Type Form
- Google Forms

- QuestionPro
- Microsoft Forms
- LimeSurvey
- Encuestas de la Universidad de Murcia

Cada una de ellas tiene sus ventajas y desventajas, por lo que antes de realizar la recogida de datos, se debería planificar y seleccionar cuidadosamente la plataforma con la que se va a administrar la encuesta.

11.3. Fiabilidad y consistencia interna

La fiabilidad y consistencia interna de un instrumento puede definirse como el grado de precisión que ofrece a la hora de realizar la medición para la que está confeccionado; para ser fiable, una escala debe tener la capacidad de exhibir resultados consistentes en mediciones sucesivas del mismo fenómeno.

Puede obtenerse mediante un coeficiente de confiabilidad, el cual corresponde a un índice que, bajo la forma de proporción, da cuenta de la razón entre la varianza de la puntuación verdadera de la escala y la varianza total.

Por lo tanto, tiene como objetivo determinar, probabilísticamente, el grado de variación atribuible a errores aleatorios o causales no vinculados a la construcción del instrumento. Garantiza la consistencia expresada en la determinación del grado de error contenido en la aplicación de una escala, y por tanto, en la medición del fenómeno.

El error puede ser entendido como el componente de la puntuación observada en la medición que no se relaciona con la capacidad que posee quien la responde. De esta manera, siendo X una puntuación observada, T una puntuación verdadera y E el error, el hecho de que una puntuación observada sea igual a la puntuación verdadera más el error puede expresarse como: $X = T + E$.

Para el estudio de la fiabilidad de una escala se presentan a continuación cuatro índices diferentes:

- Alfa de Cronbach (α).
- Fiabilidad Compuesta (CR).
- Varianza Media Extractada (AVE).

- Omega de McDonald (ω).

Es muy importante señalar que la fiabilidad de la escala debe obtenerse siempre con los datos de cada muestra para garantizar la medida fiable del constructo en la muestra concreta de investigación.

11.3.1. Alfa de Cronbach

El método de consistencia interna basado en el Alfa de Cronbach permite estimar la fiabilidad de un instrumento de medida compuesto por un conjunto de ítems de tipo escala Likert, que esperamos midan la misma dimensión teórica (el mismo constructo). De esta manera son sumables en una puntuación única que mide un rasgo que es importante en la construcción teórica del instrumento.

Como criterio general, como se puede comprobar en [76], se tienen las siguientes recomendaciones para interpretar el resultado el coeficiente Alfa de Cronbach:

- Coeficiente $\alpha < 0.5$ es inaceptable
- Coeficiente $0.5 \leq \alpha < 0.6$ es pobre
- Coeficiente $0.6 \leq \alpha < 0.7$ cuestionable
- Coeficiente $0.7 \leq \alpha < 0.8$ es aceptable
- Coeficiente $0.8 \leq \alpha < 0.9$ es bueno
- Coeficiente $\alpha > 0.9$ es excelente

Se debe calcular el coeficiente de Alfa de Cronbach para cada uno de los bloques (constructos) y, posteriormente, se puede calcular también el Alfa de Cronbach general para todo el cuestionario.

Este cálculo se realiza con la función `alpha()` del paquete `psych` del programa R:

alpha(bloque, check.keys = TRUE)$total$raw_alpha

- `check.keys`: detecta automáticamente y cálcula el inverso de los ítems codificados negativamente (cuestiones formuladas en negativo).
- La función genera una lista con diferentes datos. Es por eso que añadiendo **$total$raw_alpha** accedemos al valor del alfa de Cronbach.

11.3.2. Fiabilidad Compuesta y Varianza Media Extractada

Asimismo, en las escalas con naturaleza ordinal de la matriz de correlaciones de datos, se considera importante ofrecer los datos de fiabilidad compuesta para cada una de las dimensiones críticas, pues de este modo se analizan las relaciones entre las respuestas a los ítems y la variable latente medida, así como la varianza extraída para estudiar la validez de la escala.

El coeficiente de Fiabilidad Compuesta (`CR - Composite Reliability`) se considera más adecuado que el Alfa de Cronbach porque no depende del número de atributos asociados a cada concepto. Se considera que debe tomar un valor mínimo de 0.70 [77].

La varianza extraída o varianza media extractada (`AVE - Average Variance Extracted`), por su parte, refleja la cantidad total de la varianza de los indicadores recogida por el constructo latente. Cuanto mayor sea su valor, más representativos son los indicadores de la dimensión crítica en la que cargan. En general, se sugiere que su valor debe exceder del 0.50 [77].

De igual forma que con el Alfa de Cronbach, primero se calculan estos coeficientes para cada uno de los constructos y, después, de forma general para todo el cuestionario.

Para el cálculo de estos índices se usan las funciones `cfa()` y `standardizedSolution()` del paquete `lavaan` para obtener las cargas factoriales estandarizadas y se emplean las siguientes fórmulas:

```
comp <- NULL
ave <- NULL

items <- paste( names(bloq), collapse = "+")
model <- paste( "Val", items, sep = "=~")
# model
fit <- lavaan::cfa(model, data = bloq1, check.gradient = FALSE)
sl <- lavaan::standardizedSolution(fit)
sl <- sl$est.std[sl$op == "=~"]
# sl
```

```
re <- 1 - sl^2
comp[ 1 ] <- sum(sl)^2/ (sum(sl)^2 + sum(re))
ave[ 1 ] <- sum(sl^2) / (sum(sl^2) + sum(re))
```

11.3.3. Omega de McDonald

Otros autores proponen el coeficiente Omega, también conocido como Rho de Jöreskog, pues este no se ve afectado ni por el número de items ni por el número de alternativas de respuesta ni por la proporción de la varianza del test [78]. El coeficiente Omega está basado en las cargas factoriales, que son la suma ponderada de las variables estandarizadas.

Como criterio general, se tienen las siguientes recomendaciones para interpretar el coeficiente Omega:

- Coeficiente $\omega < 0.65$ es inaceptable
- Coeficiente $0.65 \leq \omega < 0.7$ es cuestionable
- Coeficiente $0.7 \leq \omega < 0.9$ es aceptable
- Coeficiente $\omega > 0.9$ es excelente

Como en los índices anteriores, primero se calcula para cada uno de los bloques y después de forma general para todo el cuestionario.

Se calcula con la función `omega()` del paquete `psych`:

psych::omega(bloq, nfactors = 1, plot = FALSE)$omega.tot

- `nfactors`: número de factores por bloque (suele ser 1).
- Añadiendo **$omega.tot** a la función obtenemos el valor de la Omega.

11.4. Validación de un cuestionario

Los Estándares para *Pruebas Educativas y Psicológicas* propuestos por la *American Educational Research Association* (AERA), la *American Psychological Association* (APA) y el *National Council on Measurement in Education* (NCME) establecen criterios para el desarrollo y la evaluación de cuestionarios, así como una serie de pautas para establecer la validez de la interpretación en su realización y en

sus resultados. Se debe, por tanto, comprobar la validez de contenido, la validez de constructo y la validez de criterio, desglosadas en cinco fuentes de evidencia: el contenido de la prueba, los procesos de respuesta, la estructura interna, las relaciones con otras variables y las consecuencias de las pruebas [79].

Es fundamental, al menos, comprobar la validez de contenido y la validez de constructo.

11.4.1. Validez de contenido

Trata de medir el grado en el que el contenido de la prueba refleja el propósito de la misma y representa los conocimientos, las competencias y las habilidades que se desean cuantificar [80]. Esta evidencia se recaba generalmente a través de expertos en la temática que indicarán la relevancia, coherencia, claridad y suficiencia de las cuestiones planteadas en el cuestionario para conseguir el objetivo de investigación (*juicio de expertos*). No está definido el número de expertos necesario y suficiente, pues depende de las características de la prueba y el análisis estadístico correspondiente, pero suele oscilar entre 3 y 10 expertos.

11.4.2. Validez de constructo

La validez de constructo está basada en la estructura interna. Se busca analizar un constructo para visualizar las distintas dimensiones que componen un concepto, mediante la identificación de propiedades o variables latentes (factores). Cada factor es representado por los indicadores que alcanzan mayores correlaciones. Se debe comprobar la homogeneidad (relación recíproca entre los indicadores de un constructo), la unidimensionalidad (el conjunto de ítems o indicadores que mide una única dimensión), la convergencia (diferentes medidas de un concepto o ítem que proporcionan resultados semejantes) y la divergencia (muestra bajos niveles de correlación con conceptos diferentes o ítems).

Se recomienda que se usen modelos de Análisis Factorial (tanto exploratorio, `AFE`, como confirmatorio, `AFC`) y que se consideren también otros índices de ajuste tales como el de ajuste comparativo (`CFI`), el de ajuste no normado (`TLI`) y el error cuadrático medio de aproximación (`RMSEA`) para su interpretación [80].

11.4.2. Validez de Constructo con Análisis Factorial (AF)

Con el fin de ver el papel (cargas factoriales, comunalidades...) que cada ítem juega en cada factor teórico, o *bloque*, y saber qué ítems son los que más varianza explican, se ejecuta un AFE (*Análisis Factorial Exploratorio*, en inglés EFA) para cada conjunto de variables (ítems). Esto permite ver si algún ítem no está aportando nada al modelo y se debería eliminar.

Es aconsejable realizar las siguientes comprobaciones antes de proceder con el Análisis Factorial:

- Observar la matriz de correlaciones en busca de variables poco o demasiado correlacionadas.
- Comprobar el resultado del *Test de Bartlett*.
- Interpretar el *Coeficiente Kaiser-Meyer-Olkin (KMO)* en cada bloque.
- Calcular el determinante de la matriz de correlaciones en cada bloque.

Matriz de correlaciones. Se puede trabajar indistintamente con el conjunto de datos o con la matriz de correlaciones, pero en cualquier caso, siempre con variables numéricas, no con factores. Se aconseja trabajar con la matriz de correlaciones, ya que así se evita recalcular dicha matriz cada vez que se emplee alguna función durante el análisis. Para obtener la matriz de correlaciones, se empelará la función `cor()`.

Solo con esta matriz ya se puede extraer información importante de los datos que servirá para realizar el análisis correctamente. El propósito principal del Análisis Factorial es encontrar constructos formados por variables que correlacionan bien entre ellas, pero no perfectamente. Por lo tanto, antes de llevar a cabo el análisis, se recomienda examinar la matriz de correlaciones en busca de variables que no correlacionen bien con alguna otra, es decir, con coeficientes de correlación menores que 0.3 y variables que correlacionen demasiado bien con otras, es decir, variables que tengan algún coeficiente de correlación mayor que 0.9.

```
df <- na.omit(df)
corrMatrix <- cor(df)
options("max.print" = 4000)
round(corrMatrix, 3)
cat("Nº de variables que no correlacionan bien: ",
```

```
sum( sapply(as.data.frame(corrMatrix), function(x) all(x < 0.3)) ) )

cat("Nº de variables que podrían causar multicolinealidad: ",
sum( sapply(as.data.frame(corrMatrix), function(x) any(x >= 0.9 & x != 1)) ) )
```

Las primeras se deberían eliminar del análisis y las segundas se pueden mantener, pero teniendo en cuenta que quizá causen problemas de multicolinealidad.

Test de Bartlett. Se requiere que al ejecutar el test de Bartlett, el resultado salga significativo ($p - valor < 0.05$), es decir, que la matriz no sea similar a la matriz identidad. Este test se ejecuta mediante la función `cortest.bartlett()` del paquete `psych` sobre la matriz de correlación:

cortest.bartlett(corrMatrix, nrow(df))

Análisis factorial para cada uno de los bloques. Para cada uno de los bloques del cuestionario se calculará la medida de la adecuación muestral de *Kaiser-Meyer-Olkin (KMO)*. Esta medida contrasta si las correlaciones parciales entre las variables son pequeñas. Se obtiene con la función `KMO()` del paquete `psych` para la matriz de correlación de cada bloque. Cuanto más cercano a 1 sea el valor del *KMO*, más alta es la relación entres las variables. En el siguiente ejemplo, aplicado sobre un bloque de 10 cuestiones, se obtiene un KMO de 0.82:

```
Warning in system("timedatectl", intern = TRUE): comando ejecutado
'timedatectl' tiene estatus 1
```

```
library(psych)
corMat <- cor( bloq1 )
KMO( corMat )
```

```
Kaiser-Meyer-Olkin factor adequacy
Call: KMO(r = corMat)
Overall MSA =  0.82
MSA for each item =
Ítem1  Ítem2  Ítem3  Ítem4  Ítem5  Ítem6  Ítem7  Ítem8  Ítem9 Ítem10
0.81   0.84   0.82   0.85   0.68   0.70   0.90   0.80   0.87   0.83
```

También se realiza un Análisis Factorial para cada bloque, mediante un *Análisis de Componentes Principales* y rotación de *varimax*. Se ejecutará mediante

la función `fa()` del paquete `psych` sobre la matriz de correlación del bloque, añadiendo como argumento `nfactors = 1` pues se quiere considerar un único factor en cada bloque.

```
sol <- fa(r = corMat, nfactors = 1, rotate = "varimax", fm = "pa");
sol
```

```
Factor Analysis using method =  pa
Call: fa(r = corMat, nfactors = 1, rotate = "varimax", fm = "pa")
Standardized loadings (pattern matrix) based upon correlation matrix
PA1    h2   u2 com
Ítem1  0.61 0.374 0.63   1
Ítem2  0.56 0.313 0.69   1
Ítem3  0.60 0.363 0.64   1
Ítem4  0.29 0.082 0.92   1
Ítem5  0.29 0.086 0.91   1
Ítem6  0.32 0.103 0.90   1
Ítem7  0.53 0.285 0.72   1
Ítem8  0.53 0.281 0.72   1
Ítem9  0.67 0.454 0.55   1
Ítem10 0.60 0.357 0.64   1

PA1
SS loadings    2.70
Proportion Var 0.27

Mean item complexity =  1
Test of the hypothesis that 1 factor is sufficient.

The degrees of freedom for the null model are  45  and the
                                        objective function was  2.07
The degrees of freedom for the model are 35  and the
                                        objective function was  0.41

The root mean square of the residuals (RMSR) is  0.07
The df corrected root mean square of the residuals is  0.08
```

```
Fit based upon off diagonal values = 0.93
Measures of factor score adequacy
PA1
Correlation of (regression) scores with factors   0.90
Multiple R square of scores with factors          0.80
Minimum correlation of possible factor scores     0.61
```

La salida muestra las cargas factoriales (PA1), las comunalidades (h2) y la proporción de varianza explicada (0.27), entre otros datos.

Análisis factorial general. Cuando previamente no están establecidos los bloques, se hace un estudio exploratorio inicial. Se comprueba primero el número teórico de factores de la base de datos con la función `parallel()` del paquete `nFactors`.

```
library(nFactors)
ev <- eigen(cor(df)) # Obtención de los autovalores
ap <- nFactors::parallel(subject=nrow(df),var=ncol(df),rep=100,cent=.05)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
nfac <- nS$Components$noc
plotnScree(nS,xlab = "Número de Componentes", ylab = "Autovalores",
main = "Solución por autovalores para determinar
            el número de factores o componentes")
```

En este ejemplo, la función devuelve un número de 5 factores (dimensiones).

Se calculará igualmente el KMO de la base completa y haremos un análisis factorial de forma general, indicando en este caso como argumento `nfactors` el número de factores que hayan salido anteriormente. De esta manera se puede observar cómo se repartirían los ítems en las diferentes dimensiones. Estas agrupaciones deberán ser interpretadas y recolocadas por parte de los investigadores, pero siempre teniendo en cuenta el tanto por ciento de la varianza del cuestionario que explican.

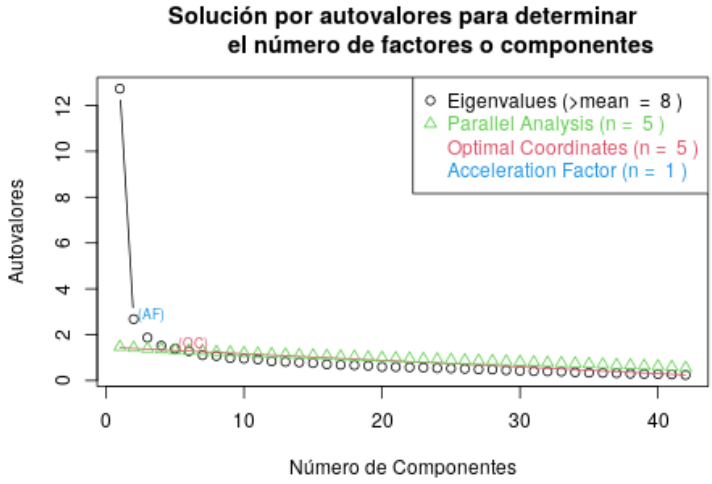


Figura 11.1: *Gráfico de perfil (Scree plot).*

11.4.2. Validez de Constructo con Modelos de Ecuaciones Estructurales

Establecidas las dimensiones de un cuestionario, estas se pueden confirmar mediante un Análisis Factorial Confirmatorio (AFC) implementado mediante un Modelo de Ecuaciones Estructurales (*Structural Equation Modeling*, *SEM*).

Especificación del modelo para cada bloque. Mediante un Modelo de Ecuaciones Estructurales se está ejecutando una regresión logística, donde la variable dependiente sería el constructo (cada bloque) y las cuestiones serían las variables independientes. Al calcular el modelo se estiman los valores para esos coeficientes en la regresión. Se utiliza la función `cfa()` de la librería `lavaan` [81].

Una vez establecida la estructura teórica hay que validarla. Para ello se comparará la *matriz de covarianza derivada* de las variables observadas y la *matriz de covarianzas reproducida* por el modelo. Así, se contrastará la hipótesis de que la diferencia entre la matriz procedente de los datos recogidos con el cuestionario y la matriz teórica definida en el modelo conceptual no es estadísticamente significativa.

Al tratarse de una escala Likert, no se cumple el supuesto de normalidad,

por lo que se opta por hacer una estimación robusta del estadístico χ^2 mediante el estimador DWLS (*Diagonal Weighted Least Squares*), es decir, un estimador ponderado de mínimos cuadrados [82].

```
items <- paste(names(bloq1), collapse = " + ")
fitBloq1 <- paste("Cont1", items, sep = " =~ ")
fitBloq1 <- cfa( fitBloq1, data = bloq1, ordered = names( bloq1 ),
missing = "pairwise", std.lv = TRUE )
show( fitBloq1 )
```

```
lavaan 0.6-9 ended normally after 13 iterations

Estimator                                         DWLS
Optimization method                           NLMINB
Number of model parameters                        45

Number of observations                           643
Number of missing patterns                         1

Model Test User Model:
Standard      Robust
Test Statistic                           287.317     402.529
Degrees of freedom                            35          35
P-value (Chi-square)                       0.000       0.000
Scaling correction factor                              0.722
Shift parameter                                        4.309
simple second-order correction
```

Si se obtiene un p-valor significativo para este estimador, se debe interpretar que el modelo no se ajusta bien a los datos. Aunque, cabe destacar que este resultado es preliminar, pues este estadístico es muy sensible a mínimas diferencias y la decisión final estará basada también en el cálculo de otros índices de ajuste.

Se pueden obtener también los **coeficientes estimados** para las variables indicadoras del modelo con la función `parameterEstimates()` (también del paquete `lavaan`) sobre los datos del `AFC`. De esta manera se puede observar la estimación de los coeficientes, el error estándar y el p-valor para la hipótesis nula

H_0: "el coeficiente es igual a cero en la población". Que una variable tenga un p-valor asociado significativo significaría que, en mayor o menor medida, aporta algo al modelo.

```
param <- parameterEstimates( fitBloq1 )
param [1:10,]
```

```
lhs op    rhs   est    se     z pvalue ci.lower ci.upper
1  Cont1 =~  Ítem1 0.715 0.025 28.97      0    0.667    0.764
2  Cont1 =~  Ítem2 0.679 0.032 21.47      0    0.617    0.741
3  Cont1 =~  Ítem3 0.669 0.027 24.96      0    0.617    0.722
4  Cont1 =~  Ítem4 0.428 0.056  7.71      0    0.319    0.537
5  Cont1 =~  Ítem5 0.393 0.037 10.58      0    0.320    0.466
6  Cont1 =~  Ítem6 0.411 0.038 10.84      0    0.337    0.486
7  Cont1 =~  Ítem7 0.602 0.031 19.65      0    0.542    0.662
8  Cont1 =~  Ítem8 0.614 0.030 20.80      0    0.556    0.672
9  Cont1 =~  Ítem9 0.746 0.024 30.72      0    0.698    0.793
10 Cont1 =~ Ítem10 0.684 0.026 26.36      0    0.633    0.734
```

A continuación, se muestra una figura con la definición del Modelo de Ecuaciones Estructurales con la función `semPaths()` del paquete `semPlot`.

```
semPaths( fitBloq1,  what = "est", intercepts = FALSE,
layout = "circle",
thresholds = FALSE,
edge.label.cex = 0.8,
edge.color = "black",
edge.width = 0.5,
label.cex = 0.6,
label.scale = F,
sizeMan = 5,
sizeLat = 5,
color = list(
lat = rgb(245, 253, 118, maxColorValue = 255),
man = rgb(155, 253, 175, maxColorValue = 255)
                )
# ,mar = c(15, 1, 5, 1) # c(bottom, left, top, right)
           )
```

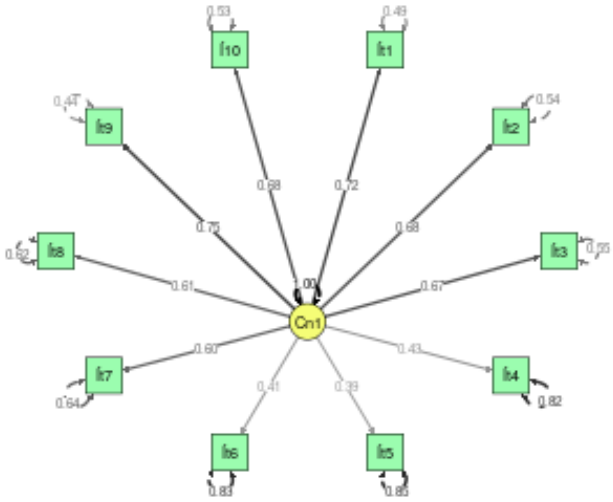


Figura 11.2: *Modelo de Ecuaciones Estructurales para un bloque.*

En esta figura, las flechas bidireccionales representan las covarianzas entre las variables latentes (elipses) y las flechas unidireccionales simbolizan la influencia que ejerce cada variable latente (constructo) sobre sus respectivas variables observadas (ítems). Por último, las flechas bidireccionales que aparecen encima de los cuadrados (ítems) muestran el error asociado a cada variable observada.

Medidas de ajuste. Antes de mostrar los distintos índices de ajuste, cabe destacar que existe cierta controversia respecto a ellos, ya que no hay un acuerdo establecido en la comunidad científica respecto a su uso.

Algunos autores no creen que estos índices añadan nada al análisis [83] y que solo debería ser interpretada la Chi-cuadrado (de esta manera se evita que los investigadores puedan exponer que no tienen un mal modelo debido a que estos ajustes están mal especificados).

Otros autores [84] abogan por el uso con precaución de las medidas de ajuste, pues los límites de estas pueden ser engañosos si se da un mal uso de las mismas.

Índices de ajuste incremental. Se deben comprobar los valores del índice `TLI` (*Tucker-Lewis Index*), también conocido como `NNFI` (*Non-Normed Fit Index*) y del índice `CFI` (*Comparative Fit Index*).

Se calcula con la función `fitMeasures()` del paquete `lavaan`, añadiendo `fit.measures = c('tli', 'cfi')` como argumento en la función.

Estos índices toman valores entre 0 y 1, donde valores cercanos a 1 indican un buen ajuste [85]:

- `TLI/CFI > 0.95`: el ajuste es bueno.
- `TLI/CFI > 0.90`: el ajuste es moderado.
- `TLI/CFI < 0.90`: hay un ajuste pobre.

```
fitMeasures( fitBloq1, fit.measures = c("tli", "cfi") )
```

```
tli   cfi
0.925 0.941
```

Índice de ajuste absoluto. También se debe comprobar el valor de `RMSEA` (*Error Medio Cuadrático de Aproximación*) que mide la diferencia absoluta de la estructura de relaciones entre el modelo teórico propuesto y los datos observados, teniendo en cuenta el número de estimadores y el tamaño muestral [86].

Se obtiene con la función `fitMeasures()` del paquete `lavaan`, añadiendo

`fit.measures = c('rmsea','rmsea.ci.lower','rmsea.ci.upper')`

como argumento en la función para así ver también los valores mínimos y máximos del mismo.

Este índice también toma valores entre 0 y 1, donde valores cercanos a 0 indican un buen ajuste.

- `RMSEA < 0.01`: el ajuste entre el modelo y los datos es excelente.
- `RMSEA < 0.05`: el ajuste entre el modelo y los datos es muy bueno.
- `RMSEA < 0.1`: hay un buen ajuste entre el modelo de medición y la estructura de datos.
- `RMSEA > 0.1`: hay un ajuste pobre. No se puede aceptar.

```
fitMeasures( fitBloq1, fit.measures = c( "rmsea", "rmsea.ci.lower",
```

```
"rmsea.ci.upper") )
```

```
rmsea rmsea.ci.lower rmsea.ci.upper
0.106          0.095          0.117
```

Re-especificación del modelo. En el caso de que el modelo no se ajuste bien a los datos del bloque, se debería proceder a eliminar las variables que menos aporten al modelo, observando el error interno de cada variable, para intentar mejorarlo. Se deben eliminar inicialmente las variables con error interno mayor que 0.80, para después volver a llevar a cabo el análisis del bloque y comprobar si los índices de ajuste han mejorado.

11.5. Análisis de un cuestionario

Una vez elaborado y validado el cuestionario tipo escala Likert, se debe proceder a su aplicación sobre la muestra determinada en la investigación para la obtención de los datos. Es recomendable incluir en el propio instrumento, o hacerlo mediante un documento anexo, el consentimiento informado para las personas que van a participar aportando sus datos. En él, informaremos al participante del objetivo u objetivos de investigación, los investigadores o el grupo u organismo que lleva a cabo el estudio y se reflejarán los derechos como participante a la protección y confidencialidad de todos sus datos, a finalizar el estudio y/o a abandonarlo.

Obtenidos los datos, llega el momento de su análisis descriptivo e inferencial, en el que se seguirá la siguiente secuencia:

- Tratamiento de valores perdidos o Missings.
- Análisis descriptivo de las variables del estudio.
- Análisis descriptivo de los cruces según variables sociodemográficas.
- Análisis inferencial. Posthoc.

11.5.1. Tratamiento de valores perdidos

La base de datos completa obtenida mediante la administración del cuestionario tendrá en las columnas las variables de estudio (sociodemográficas e ítems)

y en las filas las encuestas completadas por los participantes, es decir, los casos. El primer paso será proceder a tratar los posibles valores perdidos o *missings*.

Un valor perdido es un valor que el participante, al rellenar un cuestionario, no completa y deja vacío. Si el cuestionario se administra de forma electrónica, se puede "forzar" a los participantes a contestar todas las cuestiones evitando, de esta forma, la obtención de valores perdidos.

Como norma general, se deben eliminar todos aquellos cuestionarios que tengan uno o varios valores perdidos. Pero esto a veces no es posible, porque implica perder un gran número de cuestionarios para hacer el estudio. Por ello, se pueden realizar varias acciones dependiendo del valor perdido que sea detectado.

Normalmente, las variables independientes de un cuestionario, esto es, las variables sociodemográficas (sexo, edad, lugar, profesión, categoría profesional,...), suelen ser variables categóricas obligatorias y deben ser completadas para poder seguir completando el cuestionario. Cuando el cuestionario se aplica de manera presencial mediante lápiz y papel, puede ser que alguna de estas variables quede sin completar. En estos casos en los que no se cumplimentan estas variables, se debe dar por perdido el cuestionario completo, si bien es cierto, que algunas variables podrían ser completadas de manera aleatoria (por ejemplo, asignar de forma aleatoria el sexo a un participante). El software estadístico R permite realizar estas acciones.

En cuanto a las variables dependientes sin completar, es decir, las cuestiones tipo escala Likert, los procedimientos que se pueden realizar en el caso de obtener valores perdidos son varios:

- Se debe establecer un número máximo de valores perdidos en un mismo cuestionario y proceder a eliminar el cuestionario completo cuando se rebasa ese límite. Este número, dependerá del número total de cuestionarios, para no eliminar una cantidad excesiva de cuestionarios incompletos. Por ejemplo, se puede establecer eliminar aquellos cuestionarios que tengan cinco o más cuestiones sin completar.
- Se puede reemplazar los valores perdidos por valores aleatorios generados por R o por el valor de la mediana del resto de valores obtenidos en esa misma cuestión, solución esta última que es más rigurosa y óptima.
- En ocasiones, si el tamaño muestral es suficientemente grande, se puede trabajar haciendo el descriptivo con el total de respuestas obtenidas de cada

ítem.

11.5.2. Cálculo del tamaño muestral

Cuando se planifica un experimento, hay que prestar atención a tres aspectos principalmente:

- El nivel de confianza (α): probabilidad de cometer error de tipo I (se comete error de tipo I o α, cuando 'se afirma' que sí hay diferencias y en realidad no las hay).
- La varianza estimada de la población (diversidad de opiniones). Cuanto mayor sea la diversidad esperada mayor será el número de sujetos necesarios en la muestra.
- El margen de error que estamos dispuestos a aceptar.

Teniendo en cuenta estos aspectos, se puede calcular el tamaño muestral distinguiendo entre dos tipos de poblaciones: infinitas (tamaño desconocido, muy grande) y finitas (tamaño reducido y conocido).

11.5.2. Población infinita

Para este tipo de poblaciones utilizaremos la siguiente fórmula [87]:

$$N = \frac{z^2 pq}{e^2}$$

donde:

- z corresponde al nivel de confianza ($\alpha = 0.05$, $z = 1.96$).
- pq es la varianza de la población, siendo p la proporción de respuestas en una categoría y q la proporción de respuestas en la otra.
- e es el error muestral (posible) que cometeríamos al extrapolar los resultados.

De aquí en adelante, se considera que $p = q$, es decir, que la mitad de los encuestados contesta una categoría y el resto a la otra categoría, siendo este el caso de mayor diversidad posible. Así, para realizar el cálculo del tamaño muestral, se supondrá un $\alpha = 0.05$ ($z = 1.96$), $p = q = 0.5$ y un error determinado (si se opta por un error del 3 %, se utilizará 0.03).

En R, se puede ejecutar sencillamente esta función:

```
taMuestInf <- function( z, p, q, e){
    N <- ( z*z*p*q)/(e*e)
return( N )
}
```

11.5.2. Población finita

Para el cálculo del tamaño muestral en una población finita conocida, se empleará la siguiente fórmula [87]:

$$n = \frac{N}{1 + \frac{e^2(N-1)}{Z^2 pq}}$$

donde:

- n es el tamaño de la muestra que se desea conocer.
- N es el tamaño conocido de la población.
- z corresponde al nivel de confianza ($\alpha = 0.05$, $z = 1.96$).
- pq es la varianza de la población.
- e es el error muestral (posible) que cometeríamos al extrapolar los resultados.

Se debe tener en cuenta que para poblaciones mayores de 30.000 sujetos, esta fórmula no aporta mucho y debemos utilizar la fórmula para poblaciones infinitas.

En R, se puede ejecutar sencillamente esta función:

```
taMuestFin <- function( N, z, p, q, e){
    fr<- ( e*e*(N-1) )/(z*z*p*q)
    n <- N / (1 + fr )
return( n )
}
```

11.5.3. Análisis descriptivo de las variables de estudio

Enmarcados en la parte de análisis descriptivo de una escala Likert, primeramente se deben ofrecer resultados mediante tablas de frecuencias con respecto a las variables sociodemográficas del cuestionario y sus respectivas gráficas.

En las variables de tipo cuantitativo, como por ejemplo la edad, es recomendable hacer una categorización de la variable, estableciendo rangos de clasificación de las respuestas obtenidas, como muestra el ejemplo (cuadro 1 y figura 2).

Tabla 11.1: *Individuos por edad.*

Edad	n	pct
Menos de 25	1	0.1277
Entre 26 y 35	75	9.5785
Entre 36 y 50	342	43.6782
Entre 51 y 70	365	46.6156
Total	783	100.0000

Con respecto a los ítems del cuestionario (variables dependientes), se realiza un análisis descriptivo mediante la siguiente fórmula generada *ad hoc* y que devuelve la tabla 11.2, en la que se proporcionan estadísticos descriptivos como el mínimo, el máximo, media, mediana, desviación típica y tantos por ciento de respuesta en cada uno de los posibles niveles de contestación de la pregunta.

```
descriptivosItems <- function(datos,v, nombres= names(datos)[v], nBloque= NULL,
etiqueta = paste0("Des cuestiones bloque ", nBloque,"."), tabla = FALSE){
  frame <- data.frame(NULL)
  t <- 1
for (i in v) {
    vAux <- na.omit(datos[,i])
    suma <- as.vector(summary(vAux))
    sumatantos <- signif(suma/length(vAux),3)*100
    vAux <- as.numeric(vAux)
    qq <- c(length(vAux),min(vAux),max(vAux),mean(vAux),median(vAux),
```

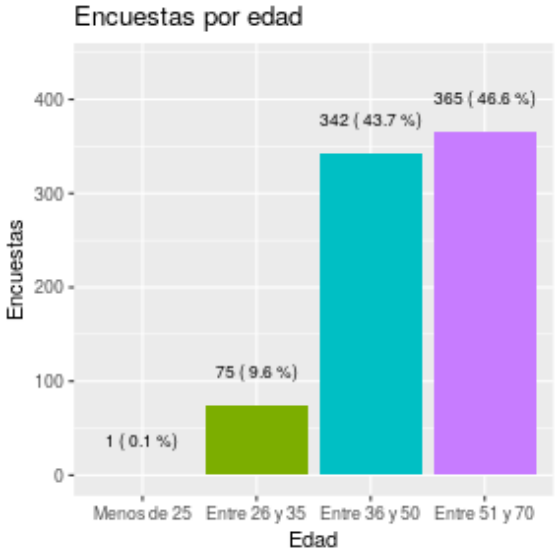


Figura 11.3: *Gráfico de barras de individuos por edad.*

```
                                ,sumatantos,sd(vAux))
for(k in 1:length(qq)){
      frame[t,k] <- qq[k]
    }
    t <- t + 1
  }
  jj <- as.data.frame(frame)
  nItems <- ncol(jj) - 6 #Cuantos niveles tienen los items
colnames(jj) <- c("N", "Mín", "Máx", "Media", "Mediana",
                                      , paste0("%",1:nItems), "Sd")
  colTotal <- c(mean(jj$N), min(jj$Mín), max(jj$Máx),
                               sapply(jj[,4:ncol(jj)], mean, na.rm = TRUE))
  jj[nrow(jj)+1, ] <- colTotal
rownames(jj) <- c(nombres, "Total cuestiones")
print(kbl(jj,  booktabs = T, digits = 2,align = "ccccccccc",
                                 caption = etiqueta ))if(tabla) return(jj)
}

indBloque <- 6:15
```

Tabla 11.2: *Descriptivo de las cuestiones del bloque.*

	N	Mín	Máx	Media	Med	%1	%2	%3	%4	%5	Sd
I1	782	1	5	4.31	4.0	0.13	1.41	8.57	47.2	42.7	0.70
I2	777	1	5	4.57	5.0	0.13	0.13	3.09	35.6	61.0	0.57
I3	781	1	5	4.00	4.0	1.41	4.48	19.50	41.9	32.8	0.91
I4	777	3	5	4.83	5.0	0.00	0.00	0.64	15.7	83.7	0.39
I5	773	1	5	4.23	4.0	0.13	2.07	10.30	49.3	38.2	0.73
I6	780	1	5	4.06	4.0	0.90	3.97	17.60	43.1	34.5	0.87
I7	775	2	5	4.37	4.0	0.00	0.90	6.84	46.5	45.8	0.65
I8	779	1	5	4.17	4.0	0.51	1.93	14.80	45.6	37.2	0.78
I9	777	1	5	4.40	4.0	0.39	0.77	7.08	42.1	49.7	0.69
I10	777	1	5	4.24	4.0	0.77	2.06	12.20	42.6	42.3	0.80
Total	778	1	5	4.32	4.2	0.44	1.77	10.06	41.0	46.8	0.71

```
names(df)[indBloque] <- c("I1", "I2", "I3", "I4", "I5", "I6", "I7",
                          "I8", "I9", "I10" )

descriptivosItems(df, 6:15)
```

También se pueden hacer gráficos con los ítems de un bloque, mediante la función `plot()` y el paquete `likert` tal y como muestra la Figura 3:

```
library(likert)
itemsAux <- likert(df[indBloque])

plot( itemsAux, centered = TRUE, group.order = colnames ( itemsAux$items ),
legend.position = "right", wrap = 40, digits = 3 )  +
theme ( axis.text.x = element_text ( size =11 ),
axis.text.y = element_text ( size = 11, hjust = 0 ),
legend.text = element_text ( size = 11 ),
legend.title = element_text ( size = 11 ) ) +
ggtitle("Cuestiones valoración colaboración familia centro")+
ylab("Porcentaje")+
        guides(fill=guide_legend(title="Respuesta"))
```

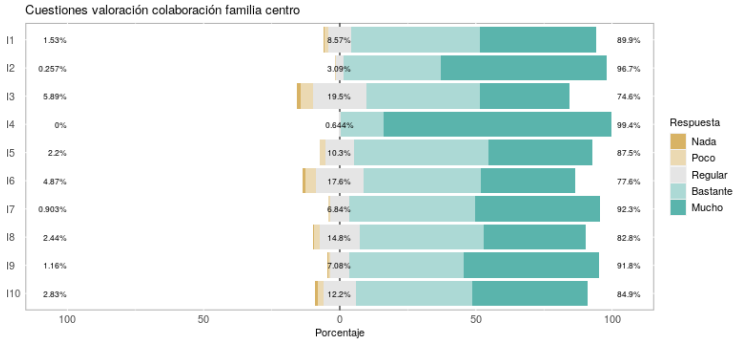


Figura 11.4: *Descriptivo cuestiones bloque 1.*

11.5.4. Análisis descriptivo de los cruces según variables sociodemográficas

Además, normalmente, se suelen realizar cruces de las variables dependientes con respecto a las variables sociodemográficas. Se calculan tanto tablas como gráficos nuevamente. El cuadro 3 muestra los cruces de los ítems con la variable sociodemográfica categorizada de la edad.

```
descripCrucesItems <- function(vble, factor, base = df){
  base %>%
group_by(.data[[factor]], .drop = FALSE)  %>%
select(.data[[vble]]) %>%
na.omit() %>%
summarise( N = length(as.numeric(.data[[vble]])),
            Mín = min(as.numeric(.data[[vble]])),
            Máx = max(as.numeric(.data[[vble]])),
Media = mean(as.numeric(.data[[vble]])),
Mediana = median(as.numeric(.data[[vble]])),
"%1" = (summary(.data[[vble]])/length(.data[[vble]])*100)[1],
"%2" = (summary(.data[[vble]])/length(.data[[vble]])*100)[2],
"%3" = (summary(.data[[vble]])/length(.data[[vble]])*100)[3],
"%4" = (summary(.data[[vble]])/length(.data[[vble]])*100)[4],
"%5" = (summary(.data[[vble]])/length(.data[[vble]])*100)[5],
# "%6" = (summary(.data[[vble]])/length(.data[[vble]])*100)[6],
# "%7" = (summary(.data[[vble]])/length(.data[[vble]])*100)[7],
```

```
Sd = sd(as.numeric(.data[[vble]])),
            Cuestión= vble ) %>%
select(c(ncol(.), 1:(ncol(.)-1) ) )
}

dfAux <- names(df[,indBloque]) %>%
lapply(descripCrucesItems, factor) %>%
bind_rows()
names(dfAux)[2] <- titFactorCorto

kbl(dfAux, booktabs = T, digits = 2, longtable = TRUE,
caption = "Ítems según edad" ) %>%
# column_spec(2, width = "13em")%>%
kable_styling(position = "center", latex_options = c("repeat_header")) %>%
collapse_rows(1:2,latex_hline = "major", valign = "top")
```

Cuestión	Edad	N	Mín	Máx	Media	Mediana	%1	%2	%3	%4	%5	Sd
I1	Menos de 25	1	5	5	5.00	5	0.00	0.00	0.00	0.0	100.0	NA
	Entre 26 y 35	75	2	5	4.39	4	0.00	1.33	5.33	46.7	46.7	0.66
	Entre 36 y 50	341	1	5	4.30	4	0.29	1.17	8.80	47.8	41.9	0.70
	Entre 51 y 70	365	2	5	4.30	4	0.00	1.64	9.04	46.9	42.5	0.70
I2	Menos de 25	0	Inf	-Inf	NaN	NA	NaN	NaN	NaN	NaN	NaN	NA
	Entre 26 y 35	75	3	5	4.64	5	0.00	0.00	2.67	30.7	66.7	0.54
	Entre 36 y 50	340	2	5	4.56	5	0.00	0.29	2.94	36.8	60.0	0.57
	Entre 51 y 70	362	1	5	4.57	5	0.28	0.00	3.31	35.6	60.8	0.59
I3	Menos de 25	1	5	5	5.00	5	0.00	0.00	0.00	0.0	100.0	NA
	Entre 26 y 35	74	1	5	3.99	4	1.35	2.70	21.62	44.6	29.7	0.87
	Entre 36 y 50	342	1	5	4.00	4	1.46	3.51	20.76	41.8	32.5	0.90
	Entre 51 y 70	364	1	5	4.00	4	1.37	5.77	17.86	41.5	33.5	0.93
I4	Menos de 25	1	4	4	4.00	4	0.00	0.00	0.00	100.0	0.0	NA
	Entre 26 y 35	75	3	5	4.79	5	0.00	0.00	1.33	18.7	80.0	0.44
	Entre 36 y 50	341	3	5	4.81	5	0.00	0.00	0.88	17.3	81.8	0.42
	Entre 51 y 70	360	3	5	4.86	5	0.00	0.00	0.28	13.3	86.4	0.35

Figura 11.5: *Aspecto de la tabla Ítems según edad.*

Finalmente, mediante el siguiente código se obtiene el gráfico de la figura 11.6, que muestra los cruces nuevamente:

```
items <- cbind(df[,indBloque], socdem=df[,factor])
items <- items[!is.na(items[ncol(items)]),]
ind <- 1

items1.1 <- likert(items[,ind:length(indBloque)],grouping = items[,ncol(items)]
```

```
gg<- plot( items1.1, grouping = df$Genero,  centered = TRUE,digits= 3)+
           theme ( axis.text.x = element_text ( size =11 ),
axis.text.y = element_text ( size = 11, hjust = 0 ),
legend.text = element_text ( size = 9 ),
legend.title = element_text ( size = 9 ) ) +
ggtitle("Ítems según edad") +
ylab("Porcentaje")+
guides(fill=guide_legend(title="Respuesta"))

print(gg)
```

11.5.5. Análisis inferencial y post-hoc

Con respecto al análisis inferencial, se emplean técnicas no paramétricas, por ser las más potentes y robustos para datos ordinales [88]. Así, para variables con dos categorías se usa la *U de Mann-Whitney* y para comparación con más de dos niveles, se usa la *K de Kruskal-Wallis* (ANOVA no paramétrica), tomando como nivel de significación 0.05. Para el post-hoc, se usa el *Pairwise Wilcoxon Rank Sum Test con corrección de Bonferroni*.

En R, la función `wilcox.test()` ejecuta la *U de Mann-Whitney*, la función `kruskal.test()` ejecuta la *K de Kruskal-Wallis* y, por último, el post-hoc se realiza mediante la función `pairwise.wilcox.test()`:

```
wilcox.test(as.numeric(df$I1)~df$b..Género)
```

```
Wilcoxon rank sum test with continuity correction

data:  as.numeric(df$I1) by df$b..Género
W = 67414, p-value = 0.02
alternative hypothesis: true location shift is not equal to 0
```

En este caso, se obtiene $p - valor < 0.05$, por lo que se puede afirmar que existen diferencias significativas según género en el ítem 1.

```
kruskal.test(df$I1~df$a..Edad)
```

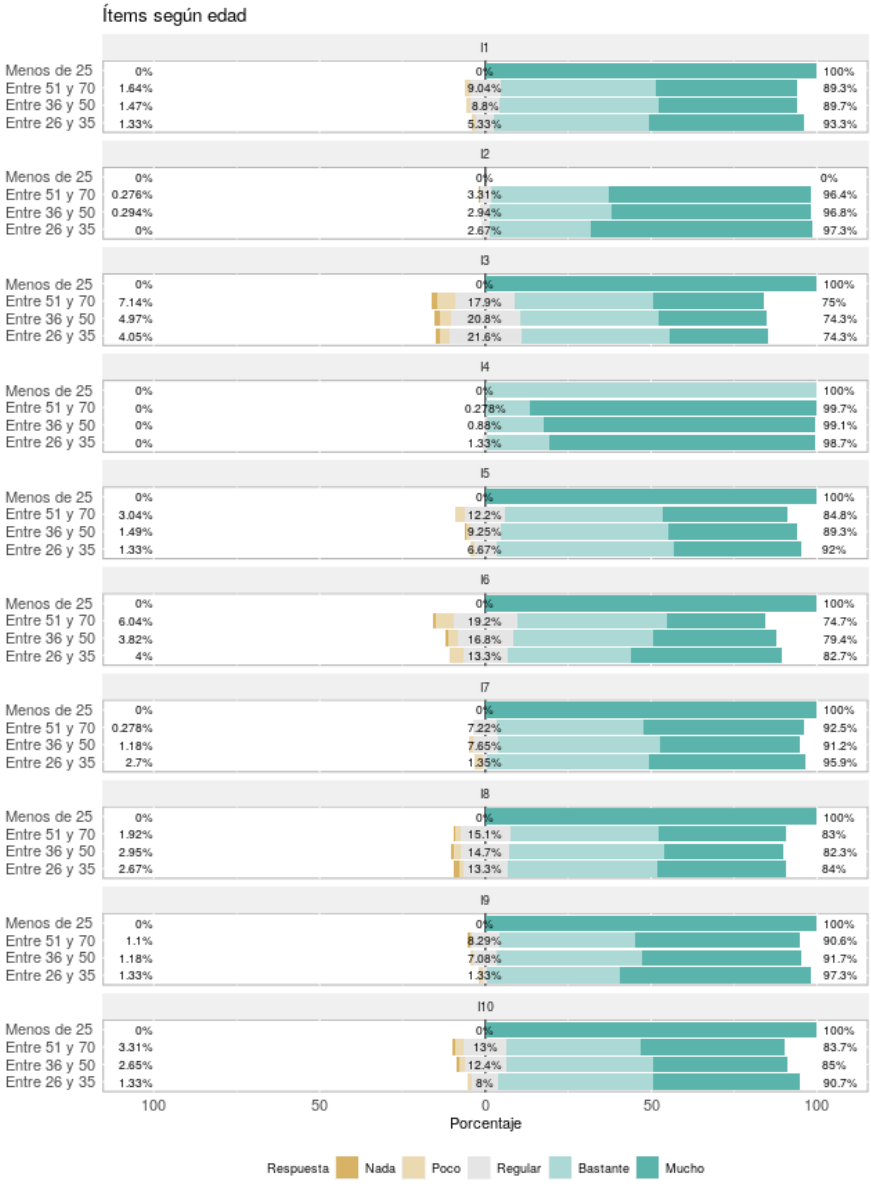


Figura 11.6: *Ítems según edad.*

```
Kruskal-Wallis rank sum test

data:  df$I1 by df$a..Edad
Kruskal-Wallis chi-squared = 2, df = 3, p-value = 0.5
```

Sin embargo, en el mismo ítem, no existen diferencias según edad.

12 Análisis multivariante: principales técnicas

Mª. Francisca Carreño Fructuoso
Fernándo Pérez Sanz
José Antonio Palazón Ferrando

Hablamos de observaciones multivariantes cuando los objetos de estudio son descritos por un gran número de variables. Las técnicas multivariantes son una constante en los trabajos descriptivos y se pueden considerar imprescindibles en diversos campos de investigación. Aportan, por un lado, la capacidad de la ordenación para reducir la dimensión de la matriz de datos y simplificar su lectura eliminando la redundancia, y por otro la posibilidad de organizar grupos de elementos, variables o unidades de muestreo, con características homogéneas mediante la clasificación.

El material que sigue, incluido el próximo tema, están relacionados con la utilización de técnicas multivariantes de ordenación y clasificación.

1. Estudiar la organización más adecuada de la información en los distintos tipos de matrices y tablas de datos.

2. Interpretación de los resultados proporcionados por las distintas técnicas

Cómo citar este capítulo: Carreño Fructuoso M. F., Pérez Sanz, F. y Palazón Ferrando J. A.(2022). Análisis multivariante: principales técnicas. En Maurandi-López, A. y González-Vidal, A. (Ed). *Análisis de datos y métodos estadísticos con R* (533-586). Editum. Ediciones de la Universidad de Murcia.

multivariantes estudiadas y evaluación de la aplicabilidad de las mismas.

3. Valorar el uso combinado de las técnicas para optimizar los resultados.

12.1. Medidas de asociación y relación entre muestras y variables

Para analizar las relaciones, esto es la semejanza o coincidencia, entre las objetos (unidades de muestreo) y entre las variables se recurre a funciones —índices, similaridades o distancias— que nos permiten medir la semejanza o desemejanza hay entre ellas. Este tipo de funciones depende del tipo de variables que constituyen la matriz de datos, así, encontramos índices para el caso de variables cualitativas o cuantitativas.

Para dos objetos son idénticos el valores de los índice de similaridad es 1 y son podemos afirmar que son totalmente distintos cuando este índice vale 0.

En el caso de distancias el valor de máxima coincidencia entre dos objetos se corresponde con el valor 0 (dos objetos idénticos). Los valores de distancia aumentan conforme aumentan las discordancias entre los valores observados en las variables que describen a los dos objetos.

En muchos se calcula la medida de asociación para todos los pares de objetos o variables y se obtiene así una matriz de asociación, matriz de distancias, matriz de correlaciones, ... (según el caso); en ellas se describe la relaciones entre todos los objetos. Se trata de una matriz cuadrada y simétrica donde la diagonal estará ocupada por el valor que indique la máxima semejanza.

12.1.1. Datos cualitativos y datos binarios

En el caso de datos cualitativos se utilizan índices de coincidencias que relacionan el número de valores coincidentes en las dos objetos con el de las coincidencias posibles.

Para datos binarios se pueden considerar tanto las afinidades entre las filas como las afinidades entre las columnas de la matriz de datos. Estas afinidades

pueden medirse con los mismos índices. Entre ellos cabe destacar: el índice de Jaccard y el índice de coincidencia simple. Ambos son índices de similaridad y toman un valor cero cuando los elementos comparados no coinciden en absoluto y el valor aumenta con el número de coincidencias. Se calculan tras construir una tabla comparando los dos vectores de presencias–ausencias que proporciona:

		1	0	
Objeto i	1	a	b	$a+b$
	0	c	d	$c+d$
		$a+c$	$b+d$	$a+b+c+d$

siendo: a: las dobles presencias, b: las presencia en el objeto i y ausencia en el j, c: las presencia en el objeto j y ausencia en el i, y d: dobles ausencias.

A partir de esta información se pueden calcular los distintos índices, entre ellos el índice de Jaccard (I_J):

$$I_J = \frac{a}{a+b+c}$$

que corresponde al cociente entre los caracteres comunes a los dos objetos y los caracteres presentes en ambos objetos. El índice de coincidencia simple (I_{CS}):

$$I_{CS} = \frac{a+d}{a+b+c+d}$$

que a diferencia del índice de Jaccard incluye en el denominador el número de dobles ausencias.

Otros índices clásicos en la literatura ecológica son:

- Czekanowski: $\frac{2a}{2a+b+c+d}$
- Ochiai: $\frac{2a}{\sqrt{(a+b)(a+c)}}$
- Mozley: $\frac{a(a+b+c+d)}{(a+b)(a+c)}$

Muchos índices de similaridad adquieren propiedades de distancia con una simple transformación, por ejemplo:

$$d_J = \sqrt{1 - I_J} \qquad d_{CS} = \sqrt{1 - I_{CS}}$$

Esta misma transformación es válida para los restantes índices mencionados salvo para el de Mozley.

Tal como se recoge en la *Task view* de *Environmetrics* [89] las funciones que proporcionan una matriz cuadrada y simétrica con las comparaciones por pares de elementos son:

- `dist()` del paquete estándar `stats`
- `daisy()` del paquete `cluster`
- `vegdist()` del paquete `vegan`
- `dsvdis()` del paquete `labdsv`
- `Dist()` del paquete `amap`
- `distance()` del paquete `ecodist`
- Varias funciones del paquete `ade4`

12.1.2. Ejemplo

Considerando la matriz de datos que recoge el fichero `drapaces.dat`:

```
x <- read.table( "datos/drapaces.dat" )
```

```
options( width = 100)
head( x )
```

```
           rupícola forestal rural diurna nocturna mamíferos aves reptiles invertebrados
buhoReal          1        0     0      0        1         1    0        0             0
buhoChico         0        1     1      0        1         1    0        0             0
cárabo            1        1     0      0        1         1    0        0             0
lechuza           0        0     1      0        1         0    0        0             1
mochuelo          0        0     1      0        1         0    0        0             1
autillo           0        1     1      0        1         0    0        0             1
```

```
x[ 1:2, ]
```

```
           rupícola forestal rural diurna nocturna mamíferos aves reptiles invertebrados
buhoReal          1        0     0      0        1         1    0        0             0
buhoChico         0        1     1      0        1         1    0        0             0
```

```
xm <- as.matrix( x[ 1:8, ] )
```

```
Ij <- vegan::vegdist( xm, method = "jaccard", diag = TRUE, upper = TRUE )
round( Ij, 2 )
```

	buhoReal	buhoChico	cárabo	lechuza	mochuelo	autillo	aguilaReal	aguilaPerdicera
buhoReal	0.00	0.60	0.25	0.80	0.80	0.83	0.60	0.60
buhoChico	0.60	0.00	0.40	0.60	0.60	0.40	0.86	0.86
cárabo	0.25	0.40	0.00	0.83	0.83	0.67	0.67	0.67
lechuza	0.80	0.60	0.83	0.00	0.00	0.25	1.00	1.00
mochuelo	0.80	0.60	0.83	0.00	0.00	0.25	1.00	1.00
autillo	0.83	0.40	0.67	0.25	0.25	0.00	1.00	1.00
aguilaReal	0.60	0.86	0.67	1.00	1.00	1.00	0.00	0.00
aguilaPerdicera	0.60	0.86	0.67	1.00	1.00	1.00	0.00	0.00

```
Isor <- vegan::vegdist( xm, method = "jaccard"  )
```

```
( a <- x & x )
```

	rupícola	forestal	rural	diurna	nocturna	mamíferos	aves	reptiles	invertebrados
buhoReal	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
buhoChico	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
cárabo	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
lechuza	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE
mochuelo	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE
autillo	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE
aguilaReal	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE
aguilaPerdicera	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE
halconPeregrino	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
aguilaCalzada	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
aguilaCulebrera	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
ratonero	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
azor	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
gavilan	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
alcotan	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
cernícaloVulgar	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
cernícaloPrimilla	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
aguiluchoCenizo	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE

```
( b <- x & !x )
```

	rupícola	forestal	rural	diurna	nocturna	mamíferos	aves	reptiles	invertebrados
buhoReal	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
buhoChico	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
cárabo	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
lechuza	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
mochuelo	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
autillo	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
aguilaReal	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
aguilaPerdicera	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
halconPeregrino	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
aguilaCalzada	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
aguilaCulebrera	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
ratonero	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

```
azor                FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
gavilan             FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
alcotan             FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
cernícaloVulgar     FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
cernícaloPrimilla   FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
aguiluchoCenizo     FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
```

```
( c <- !x[ , ] & !x[ , ] )
```

```
                  rupícola forestal rural diurna nocturna mamíferos  aves reptiles invertebrados
buhoReal             FALSE     TRUE  TRUE   TRUE    FALSE     FALSE  TRUE    TRUE          TRUE
buhoChico             TRUE    FALSE FALSE   TRUE    FALSE     FALSE  TRUE    TRUE          TRUE
cárabo               FALSE    FALSE  TRUE   TRUE    FALSE     FALSE  TRUE    TRUE          TRUE
lechuza               TRUE     TRUE FALSE   TRUE    FALSE      TRUE  TRUE    TRUE         FALSE
mochuelo              TRUE     TRUE FALSE   TRUE    FALSE      TRUE  TRUE    TRUE         FALSE
autillo               TRUE    FALSE FALSE   TRUE    FALSE      TRUE  TRUE    TRUE         FALSE
aguilaReal           FALSE     TRUE  TRUE  FALSE     TRUE     FALSE FALSE    TRUE          TRUE
aguilaPerdicera      FALSE     TRUE  TRUE  FALSE     TRUE     FALSE FALSE    TRUE          TRUE
halconPeregrino      FALSE     TRUE  TRUE  FALSE     TRUE      TRUE FALSE    TRUE          TRUE
aguilaCalzada         TRUE    FALSE  TRUE  FALSE     TRUE      TRUE FALSE   FALSE          TRUE
aguilaCulebrera       TRUE    FALSE  TRUE  FALSE     TRUE      TRUE  TRUE   FALSE          TRUE
ratonero              TRUE    FALSE  TRUE  FALSE     TRUE     FALSE FALSE   FALSE          TRUE
azor                  TRUE    FALSE  TRUE  FALSE     TRUE      TRUE FALSE    TRUE          TRUE
gavilan               TRUE    FALSE  TRUE  FALSE     TRUE      TRUE FALSE    TRUE          TRUE
alcotan               TRUE    FALSE FALSE  FALSE     TRUE      TRUE  TRUE    TRUE         FALSE
cernícaloVulgar       TRUE     TRUE FALSE  FALSE     TRUE      TRUE  TRUE    TRUE         FALSE
cernícaloPrimilla     TRUE     TRUE FALSE  FALSE     TRUE      TRUE  TRUE    TRUE         FALSE
aguiluchoCenizo       TRUE     TRUE FALSE  FALSE     TRUE      TRUE FALSE    TRUE         FALSE
```

```
( d <- x[ , ] & !x[ , ] )
```

```
                  rupícola forestal rural diurna nocturna mamíferos  aves reptiles invertebrados
buhoReal             FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
buhoChico            FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
cárabo               FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
lechuza              FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
mochuelo             FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
autillo              FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
aguilaReal           FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
aguilaPerdicera      FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
halconPeregrino      FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
aguilaCalzada        FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
aguilaCulebrera      FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
ratonero             FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
azor                 FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
gavilan              FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
alcotan              FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
cernícaloVulgar      FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
cernícaloPrimilla    FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
aguiluchoCenizo      FALSE    FALSE FALSE  FALSE    FALSE     FALSE FALSE    FALSE         FALSE
```

vegandist dissimilarity index, partial match to “manhattan”, “euclidean”, “canberra”, “clark”, “bray”, “kulczynski”, “jaccard”, “gower”, “altGower”, “mo-

risita", "horn", "mountford", "raup", "binomial", "chao", "cao", "mahalanobis", "chisq" or "chord"

12.1.3. Datos cuantitativos

En el caso matrices donde las columnas se corresponden con variables cuantitativas, las columnas y las filas a unidades de muestreo ambas tienen una naturaleza muy distinta. En ellas es preciso utilizar distintas medidas para relacionar las variables (la matriz de varianzas-covarianzas o la de correlaciones) y para las unidades de muestreo (matriz de distancias euclídeas).

La distancia euclídea entre dos objetos se calcula recurriendo al teorema de Pitágoras. En la figura 12.1 se representan dos puntos (unidades de muestreo) para las que se han medidos las variables x y y, obteniendo dos puntos (x_i, y_i) (x_j, y_j) situados a una distancia de:

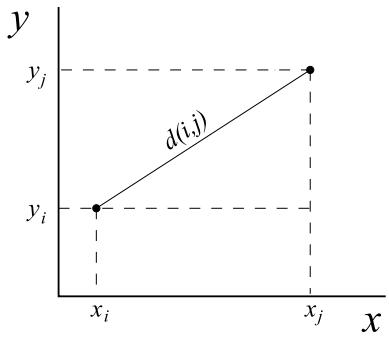


Figura 12.1: *Distancia Euclídea.*

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Es ella se refleja el caso particular para dos variables pero puede generalizarse a p variables de la siguiente forma:

$$d(i,j) = \sqrt{\sum_{k=1}^{p} (x_{i,k} - x_{j,k})^2}$$

Este es el caso de la distancia euclídea, pero se pueden utilizar otros tipos de distancias — por ejemplo, la distancia de Camberra, de Bray-Curtis, de Mahalanobis, etc.— que se basan en distintas propiedades del espacio de referencia no necesariamente euclídeo.

Además de la función elegida para medir las relaciones entre objetos y variables, las unidades en que se expresan las distintas variables pueden deformar estas relaciones. Estandarizar los datos de la matriz consiste en realizar la siguiente transformación para cada una de las columnas:

$$z_i = \frac{x_i - \bar{x}}{s_x}$$

siendo x_i los datos de una variable, $\bar{x}$ la media de la variable, s_x la desviación típica.

Columnas =/= filas

12.2. Clasificación

La clasificación es una de las actividades naturales de la mente humana [90]. Esencialmente consiste en asignar objetos o individuos a grupos de características homogéneas. Desde un punto de vista formal, podemos decir que, tras una clasificación obtenemos una nueva variable en la que se indica el grupo al que pertenece cada objeto clasificado. En la práctica cotidiana pueden darse dos situaciones:

1. Que los grupos en los que han de clasificarse los objetos de estudio ya existan: los grupos se han construido con independencia de los datos que van ha ser clasificados. La clasificación tiene, entonces, por objetivo decidir la pertenencia de cada individuo a uno u otro grupo —por ejemplo, si tenemos muestras procedentes de diversos lugares se asignaran por sus propiedades ambientales, composición florística o faunística a paisajes más o menos arquetípicios: matorrales, pastos, arenales, roquedos, pinares, etc.

2. Que los grupos no tengan una existencia *a priori*. Es entonces cuando la propia estructura de los datos (relaciones entre variables y muestras) la que determina quienes forman los grupos.

Se habla, respectivamente, de clasificación supervisada y de clasificación automática o clasificación no supervisada.

Cosideraremos aquí este segundo caso. Se trata de encontrar criterios objetivos para construir, automáticamente, grupos coherentes que nos permitan comprender la naturaleza de las posibles relaciones entre los objetos estudiados.

Entre las técnicas de clasificación automáticas cabe hacer dos grandes grupos: técnicas jerárquicas y las técnicas no jerárquicas. Las primeras producen una jerarquía de grupos que se organizan en subgrupos y estos, a su vez, en otros subgrupos. Se suelen representar por un árbol o dendrograma. En la no jerárquicas el resultado es una simple partición del conjunto de partida que se expresa mediante una variable cualitativa.

12.2.1. Clasificación no jerárquica

Existen diversos métodos y procedimientos para obtener una clasificación no jeraráquica. Para realizar un partición puede utilizarse una técnica clásica: K–means. Los resultados de la aplicación son los siguientes:

- Tamaño de los grupos (*clusters of sizes*)
- Centroides de los grupos (*cluster means*)
- Vector de clasificación (*clustering vector*)
- Medida de la varianza de cada grupo (*within cluster sum of squares*)

12.2.2. Por ejemplo: se clasifica un matriz de 100 objetos y 7 variables.

```
> kmeans(x,5)
K-means clustering with 5 clusters of sizes 21, 29, 23, 11, 16

Cluster means:
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
1 0.3293575 0.4807936 0.3401971 0.1624955 0.2906600 0.4856729 0.7309041
2 0.7595576 0.4587316 0.6889348 0.3283694 0.6626351 0.3961455 0.4795150
3 0.4563900 0.7868006 0.2638582 0.5762417 0.5649295 0.4677766 0.2356105
4 0.7133598 0.3469634 0.2364236 0.8299893 0.4498997 0.3125114 0.7340628
```

```
5 0.2435797 0.5615208 0.7463154 0.5768740 0.3592200 0.4157756 0.5182983

Clustering vector:
  [1] 1 3 1 3 1 5 1 1 2 2 1 1 1 3 4 3 3 1 2 1 3 2 5 5 5 1 3 1 5 3 5 3 1 2 2 5 3
 [38] 2 4 5 4 2 3 2 3 2 2 4 3 2 2 3 2 2 4 5 2 3 5 5 3 3 2 3 5 1 2 1 1 2 2 5 3 2
 [75] 2 4 4 1 4 2 1 4 2 2 3 3 2 2 2 1 1 3 3 4 5 5 1 5 4 2

Within cluster sum of squares by cluster:
[1]  7.930911 11.174670  8.634846  4.091959  6.496671

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

Los grupos son etiquetados con valores de 1 a 5 (número de grupos deseado). El grupo 1 tiene 21 elementos, el 2: 29, el 3: 23, ... (cluster y size)

En relación a los centroides, las 5 filas describen los centroides de cada uno de los grupos (centers).

El primer elemento de la matriz esta en el grupo 1, el segundo en el grupo 3, el tercero en el 1, ..., el último esta en el grupo 2

El grupo más compacto es el 4, con una menor suma de cuadrados.

Cuando los grupos no están claramente definidos y existen objetos entre ellos si repetimos el análisis los resultados pueden ser distintos. Estas diferencias se deben a que, a pesar de ser una técnica convergente, el inicio puede llevar a grupos claramente distintos o bien a grupos similares pero en los que los objetos situados en las fronteras puede aparecer con "compañeros".

Por ejemplo, en el caso anterior, al repetir la clasificación y cruzar las variables de clasificación se obtiene:

	1	2	3	4	5
1	0	11	5	4	1
2	11	3	3	12	0
3	4	0	1	2	16
4	1	1	6	2	1
5	1	6	3	3	3

Las filas describen los grupos del primer análisis y las columnas los del segundo. Así, se aprecia claramente que muchos de los elementos del grupo 1 del primer análisis han formado un grupo que se ha etiquetado como 2. Puede apreciarse también la "infidelidad" de muchos objetos que forman parte de otros

grupos en el segundo análisis.

En este caso interpretamos que no se trata de grupos claros. En otro ejemplo, como los datos de iris, los resultados:

	1	2	3	4	5
1	0	0	0	37	0
2	0	0	24	0	0
3	0	12	0	0	0
4	0	0	0	0	50
5	25	0	0	2	0

Muestran que en ambos casos los mismos elementos se agrupan juntos en los dos análisis, esto es, los grupos son muy claros.

12.2.3. Clasificación jerárquica aglomerativa

Entre las técnicas jerárquicas se puede hablar de dos grandes tipos: las aglomerativas —que inicialmente toman a cada objeto de estudio como un grupo para llegar a un grupo único— y las divisivas —que inicialmente consideran a todos los objetos como un sólo grupo llegando finalmente a tantos grupos como objetos se tenga—.

Para realizar una clasificación aglomerativa jerárquica es preciso calcular, en primer lugar, una matriz de distancias —también puede utilizarse índices de similaridad o disimilaridad— y agrupar los objetos que más se parecen (distancia menor), a continuación de forma sistemática se irán uniendo los objetos restantes a otros objetos o a grupos previamente construidos.

Precisamente, dentro de este tipo de clasificación se diferencian las distintas métodos por el criterio para medir la distancia entre objetos y grupos y entre grupos (criterio del vecino más próximo, del centroide, de Ward, . . .), en la figura 12.2 se muestran gráficamente el significado del algunos de ellos.

En las figuras 12.3 y 12.4 se refleja paso a paso el procedimiento de clasificación jerárquica por el método del vecino más próximo (*single linkage*). Se representa gráfica el proceso de construcción de un dendrograma considerando una distancia euclídea y el método de agregación del vecino más próximo y se

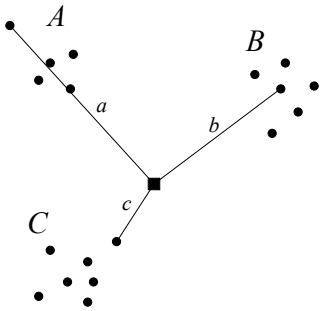


Figura 12.2: *Distancias.*

muestra como se construye el dendrograma que representa la jerarquía de las fusiones, esto es, el orden en que se producen.

Finalmente para obtener la clasificación es necesario realizar un corte del árbol, asignando a los elemento a cada uno de los grupos que se definen por el nodo inmediatamente por debajo del corte. Existe dos criterios básicos para realizar el corte: a) fijando la distancia de corte b) determinado el número de grupos que se desea obtener en la figura 12.5 se muestran gráficamente.

12.3. Ordenación

Cuando queremos representar la distribución de los valores de una única variable para una serie de individuos, un sólo eje es suficiente para ello. Si además representar una segunda variable, necesitamos dos ejes, lo cuál nos permite ver la relación entre estas variables. Pero si queremos representar 3, 4,. . . , 500 variables, se requerirían tatos ejes como variables, y esto es algo imposible de representar y de interpretar. Por tanto hay que recurrir a una serie de técnicas de reducción de dimensiones que nos permitan representar con el menor número de "variables" (dimensiones) posibles, la mayor parte de la información contenida en los datos.

Las técnicas de ordenación se aplican cuando se quiere explicar la mayor parte de la variabilidad existente en los datos, reduciendo lo máximo posible el número de variables a tener en cuenta. Se intenta reducir las dimensiones de los datos originales, creando nuevas variables que permitan caracterizar los datos

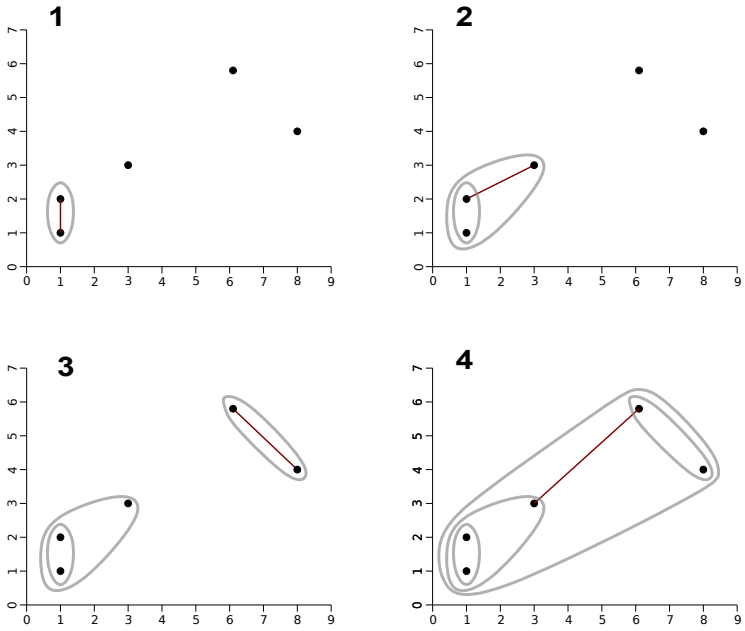


Figura 12.3: *clasificación jerárquica, grupos.*

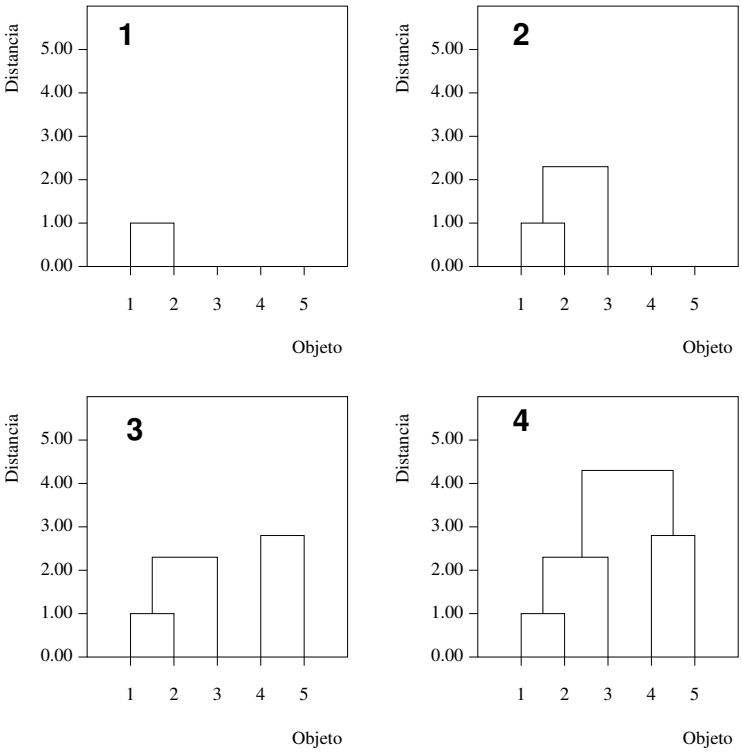


Figura 12.4: *clasificación jerárquica, dendrograma.*

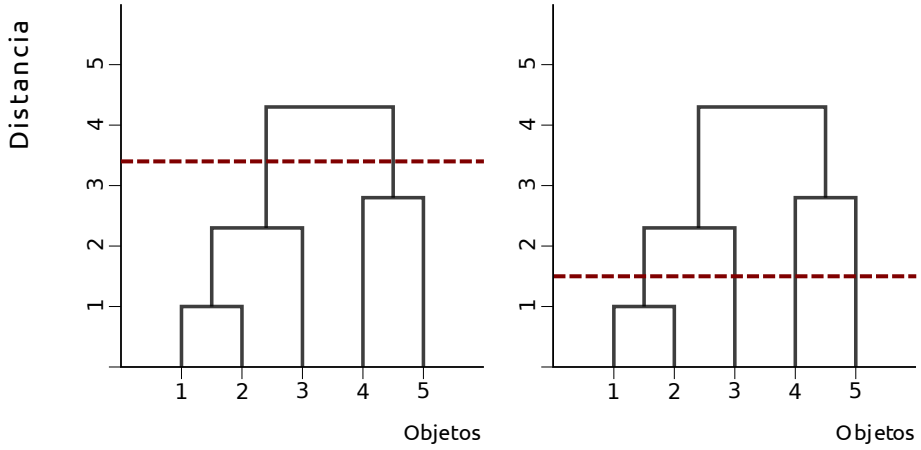


Figura 12.5: *Corte del árbol de clasificación.*

con el mínimo número de variables posible y de esta manera poder interpretar las relaciones existentes entre los individuos basadas en todas estas variables.

12.3.1. Análisis de componentes principales

Se trata de una técnica factorial y en adelante nos referiremos a ella por su acrónimo inglés: PCA (**Principal component analysis**).

Se trata de una técnica factorial que permite el la detección de gradientes con datos procedentes de un gradiente corto. Desde un punto de vista práctico conviene utilizar la matriz de datos estandarizada para obtener unos resultados más fácilmente interpretables. El procedimiento de básico del análisis se esquematiza en la figura 12.6 con un ejemplo sobre dos dimensiones que puede trasladarse a n dimensiones sin mucho esfuerzo.

Desde un punto de vista matemático, el cálculo de las nuevas coordenadas se basa en la diagonalización de la matriz de covarianzas, en el caso de datos estandarizados esta es la matriz de correlaciones.

Es, por tanto, una técnica de reducción de dimensiones consistente en extraer información importante de las variables originales, transformándolas en nuevas variables llamadas comúnmente dimensiones o factores, resultantes de la combinación lineal de las variables originales, de forma que cada nueva variable contiene una porción de la información que las variables originales contienen.

Estos factores (nuevas variables), a diferencia de las variables originales, tiene la característica de ser incorreladas entre ellas y por tanto, desde un punto de vista geométrico son ortogonales.

12.3.2. La matriz de datos

Disponemos de una matriz de objetos, unidades de muestreo (filas), por variables (columnas). Aunque cualquier matriz de estas características puede ser filtrada por PCA.

Por ejemplo la siguiente matriz de datos:

	A	B	C	D	E	F

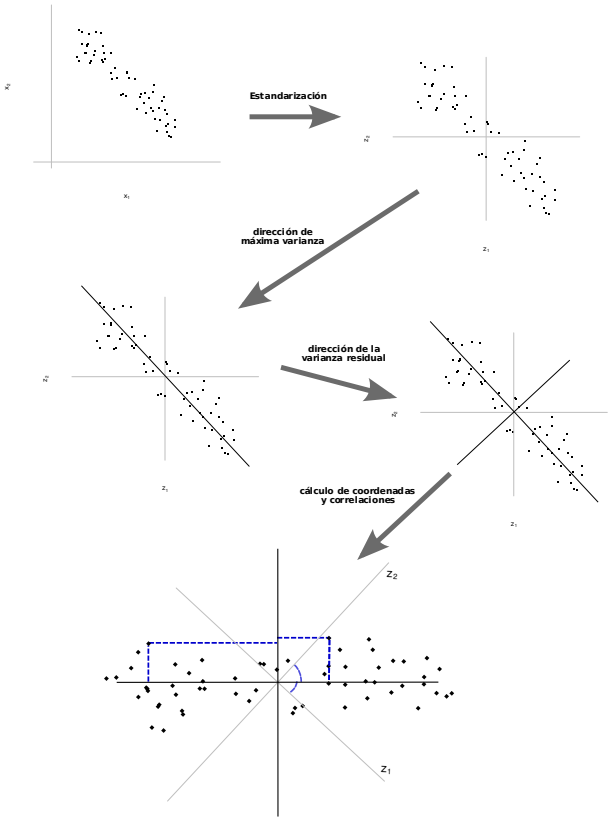


Figura 12.6: *ESquema de Análisis de componentes principales.*

```
1  6.6 6.5 2.8 5.0 2.2 2.8
2  7.0 5.1 3.0 4.9 2.1 1.1
3  6.7 5.5 3.1 5.1 2.1 1.7
4  6.0 9.5 2.1 4.9 2.9 6.5
5  6.5 7.4 2.4 4.9 2.4 4.0
6  6.6 6.1 2.7 5.1 2.1 2.2
7  6.3 8.3 2.3 4.9 2.6 5.0
8  6.5 7.0 2.6 5.0 2.5 3.6
9  7.0 4.5 2.9 5.0 1.9 0.5
10 6.2 7.9 2.6 5.1 2.6 4.6
11 6.3 9.0 2.4 4.9 2.5 5.8
```

Que representa 6 especies y 11 unidades de muestreo, procede del muestreo sobre un gradiente artificial, en ella es difícil apreciar la presencia del gradiente. En la siguiente tabla se recogen las correlaciones entre las variables. En la siguiente tabla, puede observarse como hay fuertes correlaciones entre, por ejemplo, la variable A y las variables B, E y F.

```
          A          B          C          D          E          F
A  1.0000000 -0.9545308  0.8508682  0.1598083 -0.9357039 -0.9564952
B -0.9545308  1.0000000 -0.9114388 -0.4135952  0.9398828  0.9990009
C  0.8508682 -0.9114388  1.0000000  0.5367073 -0.8728827 -0.9114707
D  0.1598083 -0.4135952  0.5367073  1.0000000 -0.3806817 -0.4117391
E -0.9357039  0.9398828 -0.8728827 -0.3806817  1.0000000  0.9485190
F -0.9564952  0.9990009 -0.9114707 -0.4117391  0.9485190  1.0000000
```

Más adelante veremos cómo el resultado de un PCA devuelve "nuevas variables" incorreladas.

Algunos conceptos relacionados con el PCA en particular y con algunas técnicas de ordenación factorial en general son:

Matriz de correlaciones como hemos visto anteriormente, la matriz de correlaciones entre las variables es un buen punto de partida para ver las relaciones existentes entre las variables.

Factores, dimensiones o componentes principales se denominan así a cada una de las nuevas variables, fruto de la combinación lineal de las variables originales.

Matriz factorial es una tabla que contiene los coeficientes factoriales. Correlacio-

nes entre factores y variables originales.

Autovalores, valores propios o eigenvalues son las varianzas de los factores

Autovectores, vectores propios o eigenvectors son las columnas de la matriz factorial. Cada columna es un vector.

Un punto importante a tener en cuenta, es la escala de las variables. En muchas ocasiones cada variable puede representar una característica particular y estar medida en unidades muy diferentes al resto. Por ejemplo, si tenemos en cuenta los valores biométricos de una serie de individuos, podríamos estar midiendo el peso en kilogramos, la altura en centímetros, la anchura de hombros también en centímetros, etc. Esto significa que mientras que el peso puede tener valores digamos entorno a los 70 kg, la altura los entorno a 165 cm, y la anchura de hombros entorno a los 60 cm. Cada uno con una varianza y una media diferente. Esto significa que algunas variables tendrán más influencia sobre los nuevos componentes que otras. Para evitar esto es conveniente normalizar o tipificar los datos. Una normalización muy común es el escalado y centrado, de manera que todas las variables quedan con media 0 y deviación típica 1.

12.3.3. Autovalores

Como se ha comentado anteriormente, los autovalores muestran las inercias, es decir la varianza acumulada por cada componente principal. Es una medida de la importancia relativa de cada componente. En la siguiente tabla se muestran los autovalores (varianzas), así como los porcentajes de varianza de cada componente, y la varianza acumulada. Puede observarse como el componente 1 (4.9) expresa aproximadamente 5 veces variabilidad que el segundo componente (0.91).

```
ypca <- FactoMineR::PCA(y, scale.unit = T, graph = F)
ypca$eig
```

```
        eigenvalue percentage of variance cumulative percentage of variance
comp 1 4.8999651689          81.666086148                           81.66609
comp 2 0.9116844492          15.194740821                           96.86083
comp 3 0.1115216628           1.858694379                           98.71952
comp 4 0.0704005939           1.173343232                           99.89286
comp 5 0.0058318688           0.097197813                           99.99006
comp 6 0.0005962564           0.009937607                          100.00000
```

Puede observarse cómo el componente 1 tiene una varianza de 4.9 que representa el 81.67 % de la varianza total. También puede verse que con 3 componentes

estaríamos representando el 98.72 de la varianza total, con lo que se podría asumir una mínima pérdida de información, a cambio de ganar en representatividad e interpretabilidad.

En la figura 12.7 se muestra un gráfico de sedimentación que permite visualizar los valores de la varianza de cada componente,

```
factoextra::fviz_screeplot( ypca, ncp = 6 )
```

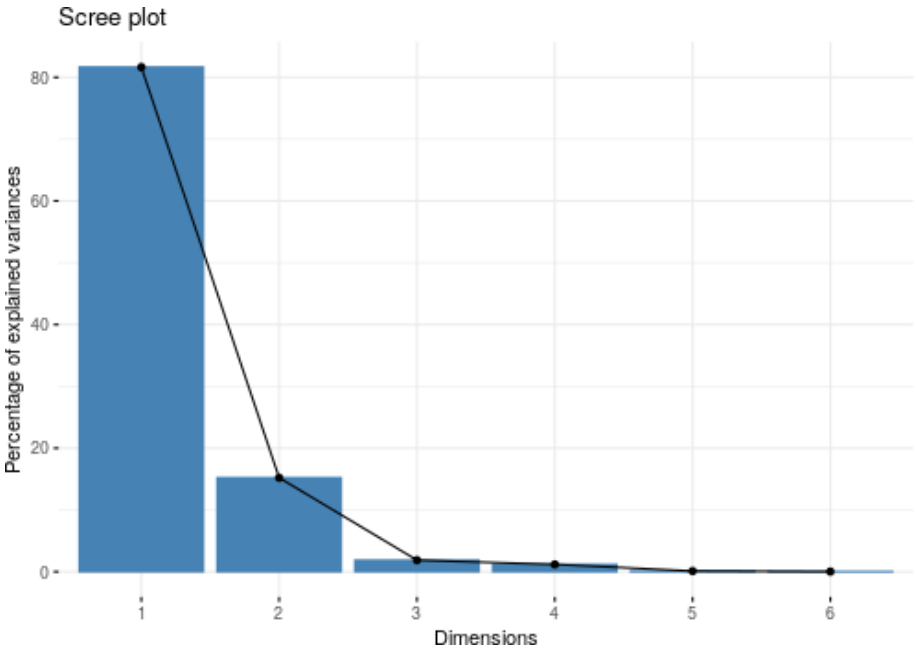


Figura 12.7: *Gráfico de sedimentación*

12.3.4. Coordenadas de individuos

La nueva tabla de datos resultante de aplicar PCA y por tanto de "girar" los ejes mediante combinación lineal de las variables originales, corresponden a las coordenadas de los individuos sobre el plano factorial (tabla 12.2). Son, por tanto, los valores que toman las nuevas variables sobre los individuos.

```
knitr::kable( ( round( ypca$ind$coord, 3 ) ),
              format = "simple",
```

```
    caption = "Coordenadas individuos")
```

Tabla 12.2: *Coordenadas individuos*

Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
-0.944	0.082	0.110	0.237	-0.098
-2.517	-1.558	0.502	-0.152	0.009
-2.485	1.080	0.414	0.135	-0.071
4.018	-0.117	0.017	-0.270	-0.104
0.892	-0.925	-0.313	-0.018	-0.003
-1.468	1.068	-0.643	0.087	0.007
2.130	-0.591	-0.120	-0.082	-0.043
0.287	0.267	0.136	-0.364	0.159
-3.253	-0.616	-0.430	-0.109	0.010
1.114	1.830	0.223	-0.120	0.036
2.227	-0.520	0.105	0.656	0.100

Tal como se menciono anteriormente (sección 12.3.2) las nuevas variables están incorreladas entre si, lo que significa que la tabla de correlaciones debería ser 0 salvo en la diagonal, que representa la correlación de una variable consigo misma, y por tanto su valor será 1, tal como se muestra en la tabla 12.3.

```
round( cor( ypca$ind$coord ), 3 )
```

Tabla 12.3: *Correlaciones de los factores*

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Dim.1	1	0	0	0	0
Dim.2	0	1	0	0	0
Dim.3	0	0	1	0	0
Dim.4	0	0	0	1	0
Dim.5	0	0	0	0	1

12.3.5. Otros parámetros de interés sobre los individuos

Hay una serie de parámetros que se obtienen al realizar un PCA, algunos de ellos útiles e interesantes para comprender los nuevos datos.

Los cos2 de los individuos, representan la distancia entre el punto (individuo) y su proyección sobre el eje y mide la contribución de cada factor a cada individuo. Las `dist`, son las distancias de cada individuo al origen de coordenadas teniendo en cuenta todas las dimensiones.

12.3.6. Variables de PCA

De forma equivalente a los individuos, se obtienen una serie de parámetros descriptivos de las nuevas variables. Destacamos entre ello las correlaciones y las coordenadas, que en el caso de datos tipificados son equivalentes (coordenadas = correlaciones). Éstas representan la correlación entre las variables originales y las nuevas variables.

```
round( ypca$var$cor, 3 )
```

```
   Dim.1  Dim.2  Dim.3  Dim.4  Dim.5
A -0.943 -0.327 -0.006 -0.017  0.063
B  0.989  0.065  0.034  0.129  0.017
C -0.949  0.122  0.282  0.072 -0.010
D -0.473  0.879 -0.064 -0.014  0.020
E  0.964  0.093  0.157 -0.193  0.016
F  0.991  0.069  0.046  0.105  0.028
```

Esta tabla es muy útil a la hora de interpretar los resultados, ya que nos está indicando la relación entre las variables originales, cuyo significado suele se claro y directo (altura, peso, temperatura, etc.), con los factores, cuyo significado es, de alguna manera abstracto, pues recoge información de todas las variables originales.

La tabla de cos2 nos indica la proporción de variable que es explicada por cada factor. La suma de cada fila es 1. Este es también un buen indicador del

significado que puede tener los nuevos ejes.

Si nos fijamos en la primera fila de (tab:varcos2), se puede observar que el 88.8 % de la variable A queda explicado por la primera componente, el 10.7 % por la segunda componente y así sucesivamente. La suma total será 1 (o 100 %)

```
round( ypca$var$cos2, 3 )
```

Tabla 12.4: *cos2 de las variables*

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Dim.1	1.000	-0.995	0.165	0.438	0.121
Dim.2	-0.995	1.000	-0.258	-0.466	-0.062
Dim.3	0.165	-0.258	1.000	0.029	-0.387
Dim.4	0.438	-0.466	0.029	1.000	-0.413
Dim.5	0.121	-0.062	-0.387	-0.413	1.000

La tabla de contrib nos indica cuánto contribuye cada variable a un factor concreto. Es una información complementaria a la aportada por los cos2que ayuda a entender las relaciones entre variables y factores.

En este caso nos vamos a centrar en la primera columna de la tabla 12.5. Vemos que el 18.13 % de la primera componente (Dim.1) queda explicada por la variable A, el 19.95 % por la variable B, etc. La suma total de cada columna es igual a 100. Dicho de otra manera, de la información que contiene la primera componente un 18.13 % se lo aporta A, un 19.95 % se lo aporta B y así sucesivamente.

```
round( ypca$var$contrib, 3 )
```

Tabla 12.5: *contribuciones de las variables*

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Dim.1	1.000	-0.995	0.165	0.438	0.121
Dim.2	-0.995	1.000	-0.258	-0.466	-0.062
Dim.3	0.165	-0.258	1.000	0.029	-0.387
Dim.4	0.438	-0.466	0.029	1.000	-0.413
Dim.5	0.121	-0.062	-0.387	-0.413	1.000

12.3.7. Representación gráfica de los individuos

La representación de los individuos en el plano factorial, muestra las relaciones de éstos con los diferentes factores o componentes. En el gráfico del 12.8 se representan los individuos en el plano factorial. Por ejemplo, puede observarse cómo el individuo 4 esta altamente relacionado de forma positiva con la primera dimensión, es decir el factor 1 explica muy bien el comportamiento de este individuo. De manera análoga, el individuo 2 está relacionado negativamente tanto con la primera, como con la segunda dimensión, de forma que ambas explican aproximadamente con la misma intensidad el comportamiento de este individuo. Cuando más cerca está un individuo del origen de coordenadas, menos relación existe entre éste y los factores, y por tanto menos explican éstos el comportamiento del individuo

```
factoextra::fviz_pca_ind( ypca )
```

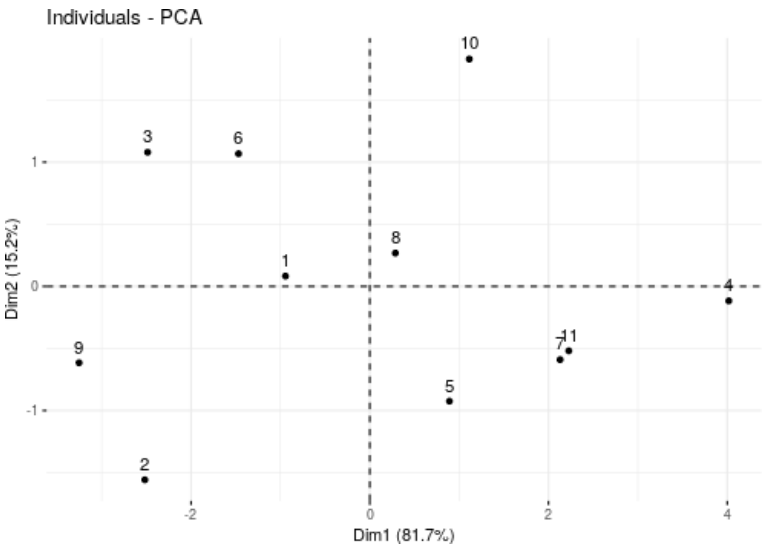


Figura 12.8: *PCA individuos*

12.3.8. Representación gráfica de las variables

Este tipo de gráficos ayuda a relacionar las variables originales con los factores o componentes. Las variables se representan mediante vectores con origen

en el origen de coordenadas y cuya dirección estará determinada por el nivel de correlación con el factor y su sentido si dicha correlación es positiva o negativa. En la figura 12.9 se observa que las variables `B`, `E` y `F` tienen una alta correlación positiva con la primera dimensión (eje horizontal). Por otro lado, la variable `D` tiene una correlación mas alta con la segunda componente y la variable `C` por su parte una alta correlación negativa con la primera componente. Véase tablas de coordenadas para entender el sentido (el signo indica el sentido) y tabla de cos2 para entender la magnitud y dirección.

```
factoextra::fviz_pca_var( ypca )
```

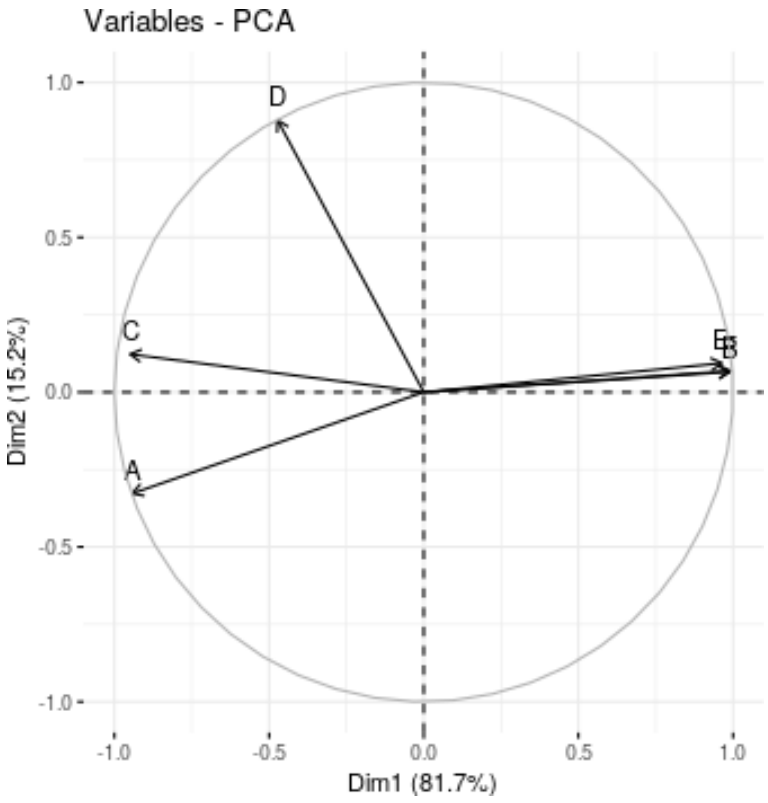


Figura 12.9: *PCA variables*

12.3.9. Gráfico biplot

Finalmente, el gráfico biplot (figura 12.10) agrupa las dos representaciones anteriores en una. Hay que tener en cuenta un detalle y es que para hacer el gráfico más legible se comente permite una pequeña licencia, ya que los vectores de las variables superan el valor de la unidad en el sistema de coordenadas en el que están representadas. Algo que es incorrecto, ya que el valor total máximo que puede alcanzar cada variable es 1.

```
factoextra::fviz_pca_biplot( ypca )
```

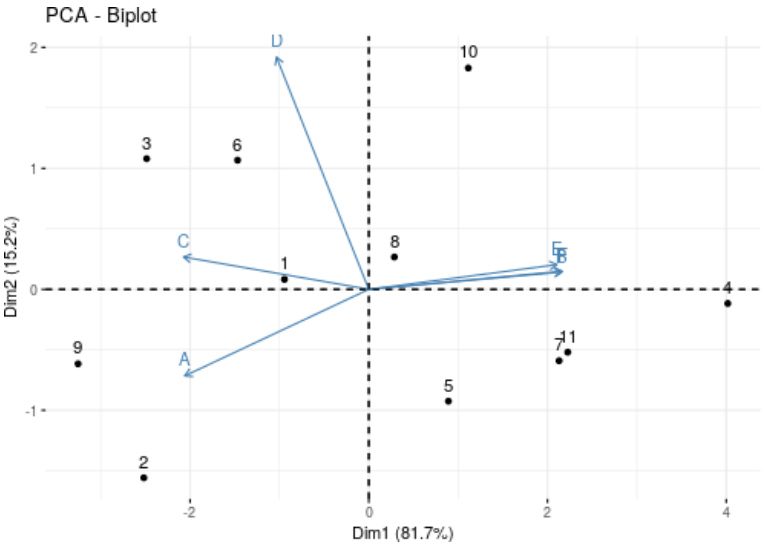


Figura 12.10: *PCA biplot*

12.3.10. Análisis de correspondencias

El análisis de correspondencias, en adelante nos referiremos por su acrónimo inglés CA (Correspondence Analysis), es una técnica, que al igual que el PCA, trata de resumir gran cantidad de información en un pequeño número de variables. En este caso, y a diferencia de PCA que trataba con variables continuas, aquí se va a tratar con variables categóricas.

Nos podemos encontrar con dos casos: el CA en el que se trabaja con una tabla de 2 vías, y el MCA (análisis de correspondencias múltiples) donde se trabaja con una tabla de múltiples vías. Esta técnica, en general, permite analizar patrones de asociación entre variables categóricas.

Nos centraremos en el caso de una tabla de dos vías por ser el más sencillo de entender. En este caso nos encontramos con dos variables categóricas, cada una de ellas con un número variable de categorías. Trabajaremos con un ejemplo para facilitar la interpretación de los resultados.

Los datos corresponden al dataset smoke, donde se representan en una matriz de contingencia de dos variables a 193 individuos pertenecientes a diferentes categorías profesionales (*Senior Manager* (SM), *Junior Employed* (JE), *Senior Employed* (SE) y *Secretaries* (SC)) ; y su grado de tabaquismo.(*none*, *light*, *medium*, *heavy*).

Se busca determinar si existe relación entre el puesto desempeñado y el grado de tabaquismo, y cómo es esa relación.

La tabla de contingencia, a diferencia de la tabla de datos, como la vista en el apartado de PCA, muestra los datos resumidos. Aquí tendríamos 2 variables, puesto de trabajo y tabaquismo, cada una con diferentes categorías, y un total de 193 observaciones.

Puesto que se quiere comprobar si existe relación entre las variables, al hacer un CA, se realiza un test de χ^2, cuya hipótesis nula H_0 es la independencia de las dos variables. Es decir, si las variables son independientes, no existirá relación entre ellas y por tanto la distribución de las observaciones en la tabla, debería desviarse poco o nada de distribución teórica. Si al realizar el test se obtiene un p-valor por encima del punto de corte establecido (digamos 0.05), no podremos rechazar la hipótesis nula y por tanto concluiremos que las variables son independientes.

Para realizar el CA y obtener posible asociación entre categorías, a cada una de éstas, se le asocia un punto en un espacio R^n, normalmente $n = 2$. De esta forma las relaciones distancia entre categorías reflejan la dependencia y semejanza existente entre ellas. Generalmente como medida de distancia se emplea la distancia de χ^2, y su proyección en el plano factorial se realiza mediante un escalado multidimensional (MDS, por sus siglas del inglés), que es una variante o

Tabla 12.6: *Tabla de frecuencias*

	none	light	medium	heavy	TotRows	tot
SM	4	2	3	2	**11**	$N1\cdot$
JM	4	3	7	4	**18**	$N2\cdot$
SE	25	10	12	4	**51**	$N3\cdot$
JE	18	24	33	13	**88**	$N4\cdot$
SC	10	6	7	2	**25**	$N5\cdot$
Totcols	**61**	**45**	**62**	**25**	**193**	
tot	$N_{\cdot 1}$	$N_{\cdot 2}$	$N_{\cdot 3}$	$N_{\cdot 4}$		$N_{\cdot\cdot}$

caso particular del PCA cuando la matriz de datos es una matriz de distancias.

En definitiva, y para simplificar los conceptos, al hacer un CA, estamos haciendo un test de χ^2 para comprobar la independencia de las variables a la vez que obtenemos una matriz de distancias de χ^2, la cuál nos va a servir para representar las categorías en un plano factorial mediante un MDS.

12.3.11. Tabla de frecuencias

La tabla 12.6 de frecuencias recoge el número de observaciones de cada cruce de categorías y además el número total de observaciones para cada categoría de cada variable, es decir totales por filas ($N_{i\cdot}$) y columnas ($N_{\cdot j}$). A partir de esta tabla se obtienen los perfiles filas y los perfiles columnas.

Tabla 12.7: *Perfil fila*

	none	light	medium	heavy	TotFreq
SM	0.36	0.18	0.27	0.18	**1**
JM	0.22	0.17	0.39	0.22	**1**
SE	0.49	0.20	0.24	0.08	**1**
JE	0.20	0.27	0.38	0.15	**1**
SC	0.40	0.24	0.28	0.08	**1**

12.3.12. Tabla de frecuencias condicionadas por filas (perfiles fila)

Esta tabla (Tabla 12.7) es el resultado de dividir el valor de cada celda n_{ij} por el total de su fila $N_{i\cdot}$. Para interpretar los valores de la tabla, hay que leer los datos por filas. Por ejemplo para la categoría `SM` de la variable de puestos de trabajo, diríamos que hay un 36 % de individuos que pertenecen a la categoría none, un 18 % pertenece a la categoría `light`, un 27 % a la `medium` y un 18 % a la categoría heavy. De esta manera tendríamos una visión de la distribución de cada categoría fila a lo largo de las categorías columna.

12.3.13. Tabla de distribuciones condicionadas por columnas (perfiles columna)

La tabla de frecuencias condicionadas por columnas (Tabla 12.8), análogamente a lo visto en el apartado anterior, resulta de dividir el valor de cada celda n_{ij} (número de observaciones para cruce de categorías) por el total de observaciones de su columna $N_{\cdot j}$. Cogiendo como ejemplo la categoría none de nivel de tabaquismo, diríamos que tenemos dentro de los empleados que no fuman, el 7 % pertenece a la categoría `SM`, otro 7 % a la categoría `JM`, un 41 % a `SE`, un 30 % a `JE` y finalmente un 16 % a la categoría `SC`.

Tabla 12.8: *Perfil columna*

	none	light	medium	heavy
SM	0.07	0.04	0.05	0.08
JM	0.07	0.07	0.11	0.16
SE	0.41	0.22	0.19	0.16
JE	0.30	0.53	0.53	0.52
SC	0.16	0.13	0.11	0.08
TotFreq	**1.00**	**1.00**	**1.00**	**1.00**

12.3.14. Parámetros de interés en CA

Al realizar un CA, nos encontramos con una serie de parámetros resultantes de dicho análisis.

```
yca <- FactoMineR::CA(smk, graph = FALSE)
FactoMineR::summary.CA(yca)
```

```
Call:
FactoMineR::CA(X = smk, graph = FALSE)

The chi square of independence between the two variables is equal to 16.44164 (p-value =  0.1718348 ).

Eigenvalues
                       Dim.1   Dim.2   Dim.3
Variance               0.075   0.010   0.000
% of var.             87.756  11.759   0.485
Cumulative % of var.  87.756  99.515 100.000

Rows
         Iner*1000    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
SM     |     2.673 | -0.066  0.330  0.092 |  0.194 21.356  0.800 |  0.071 69.433  0.107 |
JM     |    11.881 |  0.259  8.366  0.526 |  0.243 55.115  0.465 | -0.034 25.619  0.009 |
SE     |    38.314 | -0.381 51.201  0.999 |  0.011  0.300  0.001 | -0.005  1.698  0.000 |
JE     |    26.269 |  0.233 33.097  0.942 | -0.058 15.177  0.058 |  0.003  1.205  0.000 |
SC     |     6.053 | -0.201  7.006  0.865 | -0.079  8.052  0.133 | -0.008  2.045  0.001 |

Columns
         Iner*1000    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
none   |    49.186 | -0.393 65.400  0.994 |  0.030  2.934  0.006 | -0.001  0.061  0.000 |
light  |     7.059 |  0.099  3.085  0.327 | -0.141 46.317  0.657 |  0.022 27.282  0.016 |
medium |    12.610 |  0.196 16.562  0.982 | -0.007  0.174  0.001 | -0.026 51.140  0.017 |
heavy  |    16.335 |  0.294 14.954  0.684 |  0.198 50.575  0.310 |  0.026 21.517  0.005 |
```

En primer lugar podemos ver el valor del estadístico χ^2 (16.44) junto a su

p-valor (0.172). Al ser éste mayor de 0.05 no se puede rechazar la hipótesis nula, y por tanto, se asume la independencia de las 2 variables, por lo tanto no tendría sentido continuar con el análisis.

A pesar de la independencia de variables, continuaremos con el análisis, ya que estos datos sirven para ilustrar los resultados.

Otro de los parámetros que devuelve, son los autovalores o `eigenvalues`, que proporcionan la varianza acumulada por cada nuevo eje o factor. Se puede observar en los resultados del ejemplo, que sólo aparecen 3 dimensiones o ejes. Esto es correcto, ya que el número máximo de dimensiones en un CA es igual al número de categorías menos una ($n_{categorias} - 1$), de la variable que menos categorías tenga.

En el ejemplo, vemos que entre las 2 primeras dimensiones queda explicada el 99.5% de la varianza total.

Finalmente tenemos una serie de datos similares a los obtenidos en PCA (cos2, contr). Aquí hay un concepto que no aparecía en PCA, el concepto de inercias, que es similar al concepto de varianza. En este caso las inercias se interpretan como las varianzas de cada categoría explicadas por cada dimensión.

12.3.15. Representación gráfica de CA

La representación gráfica de un CA (figura 12.11), proyecta las categorías de las variables sobre el plano factorial. De esta manera se puede observar las relaciones (si existen) entre categorías de las diferentes variables. Esta relación es más estrecha, cuanto más cerca están dos categorías.

En nuestro ejemplo vemos que el hecho de no fumar (none) podría estar más relacionado con la categoría profesional SE, mientras que los individuos que más fuman se relacionan más con la categoría profesional JM.

Vimos anteriormente que estas variables no eran dependientes, según el test de la χ^2. Esto explica que no se vea una relación perfecta entre las categorías de ambas variables.

```
factoextra::fviz_ca(yca)
```

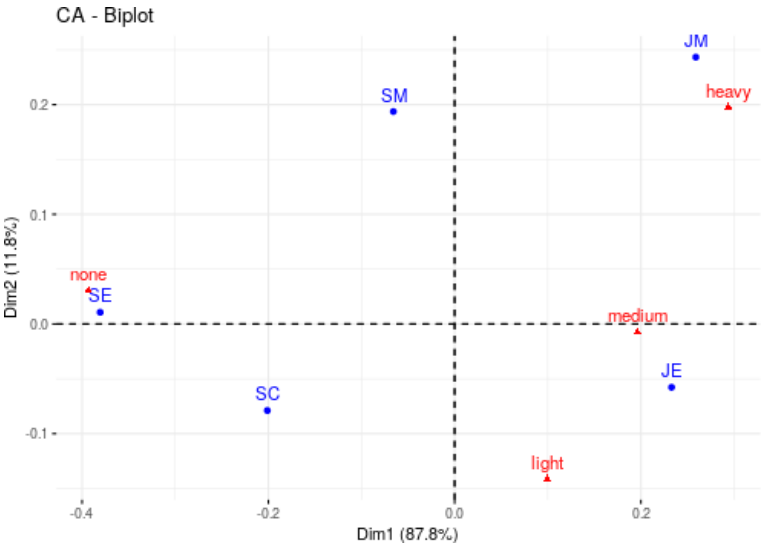


Figura 12.11: *CA individuos*

12.4. Combinación de técnicas. Ejemplo de aplicación.

Cuando nos enfrentamos al problema de identificar el papel de las distintas variables bioclimáticas en un área determinada nos surgen distintas preguntas que puede ser relevantes para la elaboración de modelos de distribución potencial.

¿Podemos definir distintas regiones bioclimáticas en la PI.? Si es así ¿cuántas regiones podríamos definir? ¿Cómo podríamos identificar el papel de las variables bioclimáticas en la Península Ibérica? ¿Es posible una buena representación local, por ejemplo en una provincia, cuando consideramos toda la PI. en la definición de la regiones? ¿Qué nos aportaría este conocimiento en el desarrollo de los modelos de distribución potencial?

```
nKlases <- 200
nGrupos <- 30
```

Para intentar contestar a estas preguntas y otras relacionadas con el análisis de variables ambientales o predictores vamos a realizar un procedimiento clásico:

- Ordenación: Ordenación, mediante PCA.
  Siendo los píxeles los individuos descritos por las variables bioclimáticas.
- Clasificación: Sobre las dimensiones relevantes obtenidas en la ordenación.

  1. No jerárquica para obtener los grandes grupos. Consideramos los los píxeles como los individuos y las coordenadas en los nuevos ejes como las variables. Recurrimos a *K–means*. asumiendo un gran numero de ellos: 200.
  2. Jerárquica para "reducir" el número de grupos reuniendo los más homogéneos. Los individuos a clasificar son los grupos (descritos por los centroides). Utilizamos el criterio de agregación de mínima varianza o criterio de Ward, haciendo un corte de árbol para generar 30.

12.4.1. Datos y funciones

Los datos proceden de la base de datos de *Bioclim*. Para el análisis utilizaremos la función `PCA` (librería `FactoMiner`), en la ordenación, y las funciones `kmeans` y `hclust` (librería `stats`) para la clasificación.

Además de la librería `FactoMineR` necesitamos, para simplificar y realzar algunos gráficos la librería `factoextra` y para manejar los datos espaciales la librería `terra`.

Los datos se leen como un objeto `SpatRaster` y lo etiquetamos con el nombre de `raster` para generalizar y hacer más legible el código. De cara a una mejor visualización de los datos bioclimáticos cambiamos los nombre originales de las capas, de la forma `wc2.1_30s_bio_1` a `wc2.1_30s_bio_19` por otra menos farragosa y homogénea `Bio01` a `Bio19`.

```
library( terra )
library( FactoMineR )
library( factoextra )
raster <- rast( "datos/bioclim.tif" )
names( raster ) <- paste0( "Bio", formatC( 1:19, width = 2, flag = "0"  ) )
```

12.4.2. Ordenación: PCA

```
ndimensiones <- 6
```

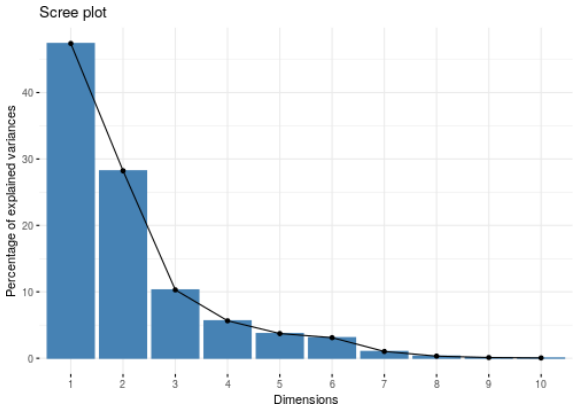
Para realizar el análisis simplemente fijamos el número de dimensiones a retener 6, que ya hemos determinado en ensayos previos. Por otro lado evitamos la representación del *biplot* dado el gran número de puntos a representar (1602720).

```
options( width = 120 )
pcaRaster <- PCA( raster, graph = FALSE, ncp = ndimensiones )
summary( pcaRaster )
```

```
Call:
PCA(X = raster, ncp = ndimensiones, graph = FALSE)


Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7   Dim.8   Dim.9  Dim.10  Dim.11  Dim.12
Variance               9.003   5.362   1.956   1.073   0.707   0.590   0.194   0.061   0.021   0.009   0.007   0.006
% of var.             47.386  28.223  10.294   5.645   3.718   3.104   1.023   0.321   0.112   0.049   0.039   0.032
Cumulative % of var.  47.386  75.609  85.903  91.548  95.267  98.371  99.393  99.714  99.826  99.875  99.913  99.946
                      Dim.13  Dim.14  Dim.15  Dim.16  Dim.17  Dim.18  Dim.19
Variance               0.005   0.002   0.001   0.001   0.001   0.000   0.000
% of var.              0.025   0.012   0.007   0.006   0.003   0.001   0.000
Cumulative % of var.  99.971  99.983  99.990  99.996  99.999 100.000 100.000


Individuals (the 10 first)
          Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
987    |  5.684 | -4.762  0.000  0.702 |  1.932  0.000  0.116 | -1.862  0.000  0.107 |
988    |  5.641 | -4.738  0.000  0.705 |  1.896  0.000  0.113 | -1.825  0.000  0.105 |
989    |  5.647 | -4.754  0.000  0.709 |  1.845  0.000  0.107 | -1.799  0.000  0.102 |
990    |  5.648 | -4.759  0.000  0.710 |  1.823  0.000  0.104 | -1.807  0.000  0.102 |
991    |  5.624 | -4.752  0.000  0.714 |  1.784  0.000  0.101 | -1.777  0.000  0.100 |
992    |  5.644 | -4.750  0.000  0.708 |  1.797  0.000  0.101 | -1.796  0.000  0.101 |
993    |  5.609 | -4.672  0.000  0.694 |  1.819  0.000  0.105 | -1.837  0.000  0.107 |
994    |  5.578 | -4.652  0.000  0.695 |  1.784  0.000  0.102 | -1.800  0.000  0.104 |
995    |  5.584 | -4.637  0.000  0.690 |  1.788  0.000  0.102 | -1.817  0.000  0.106 |
996    |  5.586 | -4.636  0.000  0.689 |  1.742  0.000  0.097 | -1.820  0.000  0.106 |

Variables (the 10 first)
         Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
Bio01 |  0.717  5.715  0.515 |  0.630  7.393  0.396 | -0.202  2.093  0.041 |
Bio02 |  0.716  5.694  0.513 | -0.414  3.195  0.171 |  0.182  1.686  0.033 |
Bio03 |  0.159  0.282  0.025 |  0.483  4.351  0.233 | -0.254  3.303  0.065 |
Bio04 |  0.547  3.324  0.299 | -0.721  9.702  0.520 |  0.305  4.763  0.093 |
Bio05 |  0.953 10.080  0.908 |  0.014  0.004  0.000 |  0.117  0.700  0.014 |
Bio06 |  0.286  0.908  0.082 |  0.902 15.173  0.814 | -0.268  3.683  0.072 |
Bio07 |  0.665  4.909  0.442 | -0.624  7.257  0.389 |  0.296  4.473  0.087 |
Bio08 |  0.482  2.575  0.232 |  0.154  0.444  0.024 | -0.735 27.643  0.541 |
Bio09 |  0.524  3.045  0.274 |  0.416  3.230  0.173 |  0.254  3.311  0.065 |
Bio10 |  0.903  9.047  0.815 |  0.295  1.618  0.087 | -0.052  0.140  0.003 |
```

Los resultados proporcionan información sobre los valores propios, *eigenvalues*, tanto en su valor como en la proporción del total de la varianza asociada individual o acumulada. Podemos representarla gráficamente.

```
fviz_eig( pcaRaster )
```

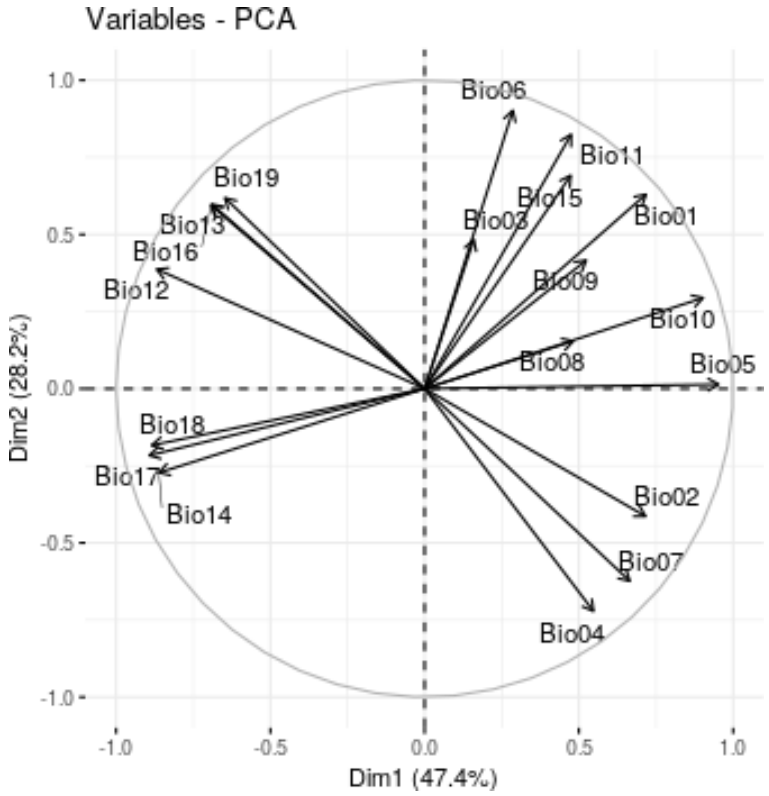


Sustituiremos las variables iniciales por tantas dimensiones como expliquen un valor acumulado por encima de un umbral dado, que puede variar en función de la naturaleza de los datos.

Las coordenadas de las variables en las nuevas variables (dimensiones, ejes, componentes, factores ...) y de los píxeles; en ambos casos sólo aparecen las primeros diez elementos de la lista.

De los resultados no interesa conocer cuales son las relaciones entre los ejes y las nuevas variables, cosa que de forma indirecta también nos informa de las relaciones entre las propias variables. Además nos interesa conocer la varianza recogida por las dimensiones o ejes resultantes; esta información aparece en la tabla siguiente, donde se recogen las coordenadas de las variables con los ejes, es decir, sus correlaciones. Para poder realizar la interpretación adecuadamente recurrimos a la representación gráfica o al análisis de los valores tabulados.

```
fviz_pca_var( pcaRaster, repel = TRUE  )
```



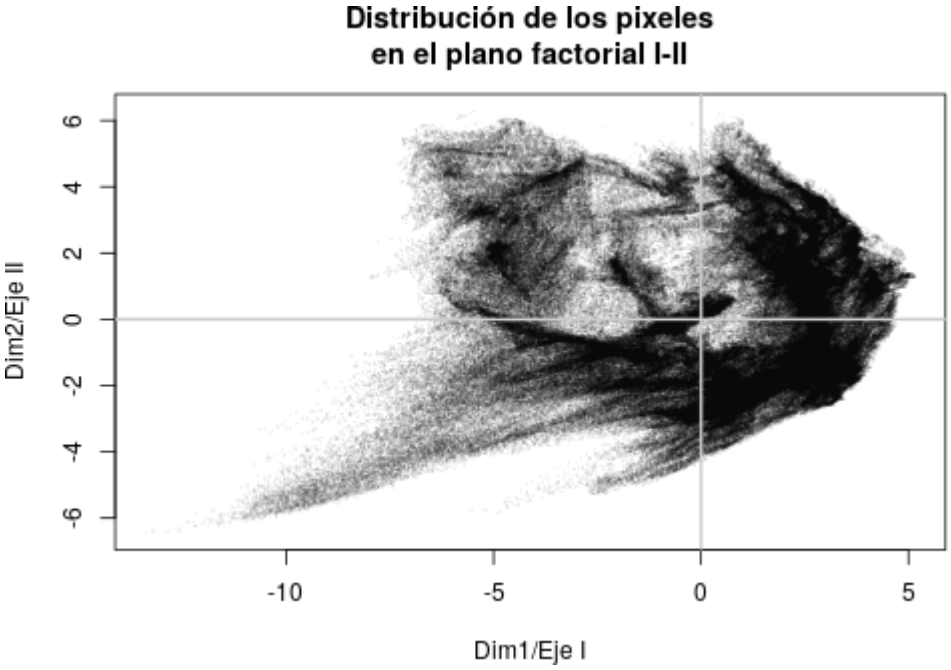

```
#| class.source: "small"
knitr::kable( round( pcaRaster$var$coord, 2 ),
              caption = "Correlaciones de los ejes con las                variables
```

Para hacernos una idea de la distribución de los píxeles en el plano factorial para las dos primeras dimensiones nos conformaremos con representar una fracción del total de los píxeles analizados (sólo un cincuenta ciento de ellos elegidos aleatoriamente).

```
plot( pcaRaster$ind$coord[ sample( nrow( pcaRaster$ind$coord ),
                                   nrow( pcaRaster$ind$coord ) / 2 ), ],
      cex = 2, pch = ".", col = grey( 0, 0.05 ),
      xlab = "Dim1/Eje I", ylab = "Dim2/Eje II",
      main = "Distribución de los pixeles\nen el plano factorial I-II" )
abline( h = 0, v = 0, col = "grey80", lwd = 2 )
```

Tabla 12.9: *Correlaciones de los ejes con las variables originales.*

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6
Bio01	0.72	0.63	-0.20	-0.05	0.18	-0.03
Bio02	0.72	-0.41	0.18	0.51	0.13	0.06
Bio03	0.16	0.48	-0.25	0.76	-0.16	0.26
Bio04	0.55	-0.72	0.31	-0.05	0.26	-0.11
Bio05	0.95	0.01	0.12	0.10	0.24	-0.04
Bio06	0.29	0.90	-0.27	-0.14	0.02	0.01
Bio07	0.66	-0.62	0.30	0.19	0.20	-0.05
Bio08	0.48	0.15	-0.74	0.06	0.14	-0.30
Bio09	0.52	0.42	0.25	-0.30	0.21	0.56
Bio10	0.90	0.29	-0.05	-0.06	0.28	-0.07
Bio11	0.47	0.82	-0.28	-0.03	0.08	0.02
Bio12	-0.87	0.39	0.21	0.09	0.20	-0.04
Bio13	-0.69	0.60	0.31	0.13	0.14	-0.15
Bio14	-0.86	-0.27	-0.29	0.06	0.23	0.14
Bio15	0.47	0.69	0.46	0.05	-0.14	-0.14
Bio16	-0.69	0.59	0.34	0.12	0.13	-0.13
Bio17	-0.89	-0.22	-0.25	0.02	0.27	0.10
Bio18	-0.88	-0.18	-0.30	0.08	0.27	0.01
Bio19	-0.65	0.62	0.40	0.09	0.11	-0.06



12.4.3. Visualizando los resultados del análisis

Dada la naturaleza espacial de la información la presentación de los resultados en forma de mapa puede ayudar a la interpretación de una manera más efectiva que el plano factorial.

Llevaremos los resultados obtenidos para las coordenadas de los píxeles a una nueva capa por cada eje. Debemos prestar especial atención a la "desaparición" en los resultados de los píxeles que presenten el valor `NA` en cualquiera de las capa iniciales. Esto nos obliga a "ubicar" cada pixel en su lugar en el ráster, cosa relativamente sencilla ya que disponemos del identificador de cada uno de ellos. Como se puede ver en los resultados del PCA los diez primeros píxeles de los que tenemos información se identifican con el número de posición el ráster.

Creamos dos capas, para los dos primeros ejes, con la geometría de los datos de entrada y los valores correlativos del 1 al número de pixeles de la capa.

```
rasterPCA <- raster[[ 1 ]]
```

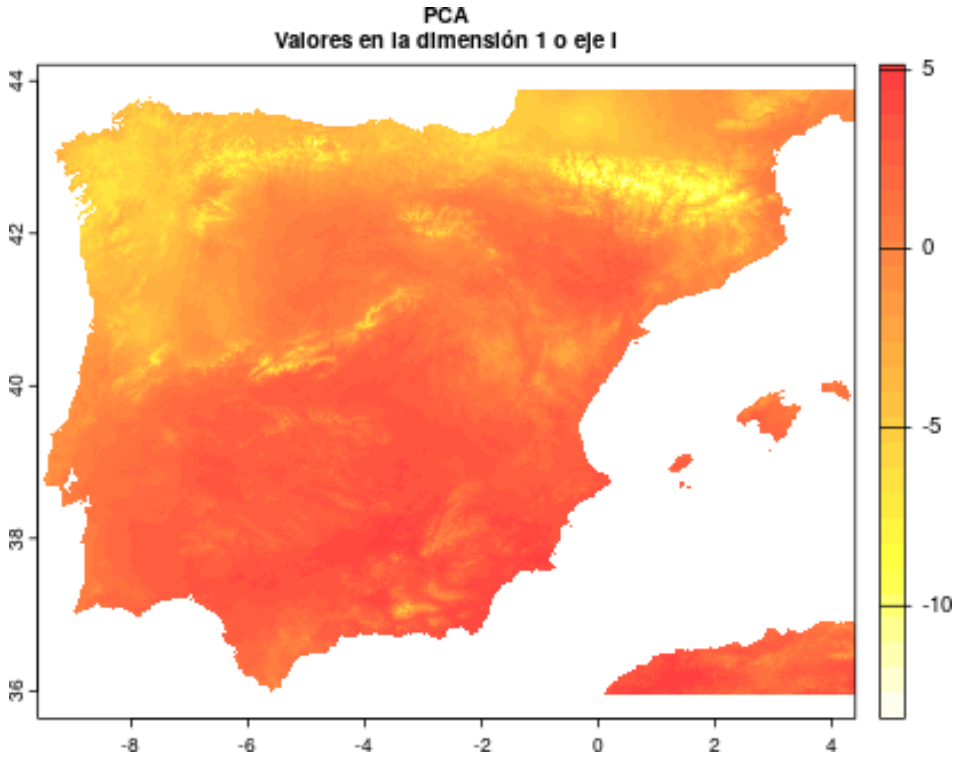
```
rasterPCA[] <- 1:( ncell( rasterPCA ) )
names( rasterPCA ) <- "rasterPCA01"

rasterPCA2 <- rasterPCA
names( rasterPCA2 ) <- "rasterPCA02"
```

Para realizar la transferencia de los resultados a las nuevas capas utilizamos un vector la misma logitud del número de píxeles de la capa, y con valor NA. Después ponemos los valores en función de nombre (etiqueta) del píxel en los resultados.

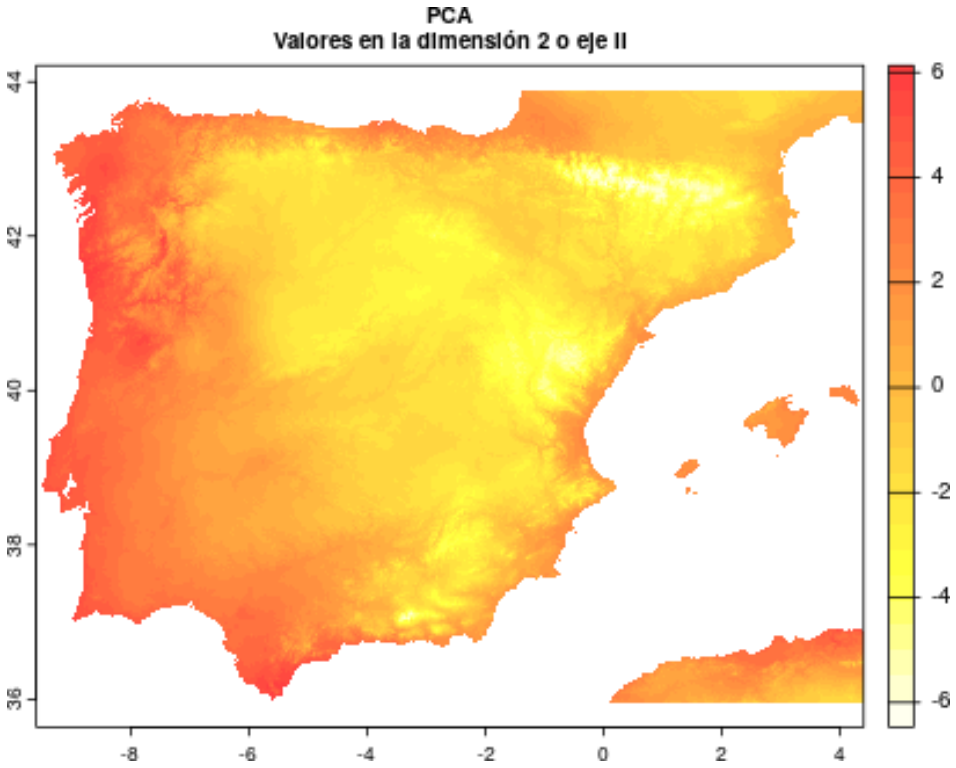
```
valorEje <- rep( NA, ncell( rasterPCA ) )
valorEje[ as.integer( rownames( pcaRaster$ind$coord ) ) ] <-
 pcaRaster$ind$coord[ , 1 ]
values( rasterPCA )  <- valorEje

valorEje2 <- rep( NA, ncell( rasterPCA ) )
valorEje2[ as.integer( rownames( pcaRaster$ind$coord ) ) ] <-
 pcaRaster$ind$coord[ , 2 ]
values( rasterPCA2 )  <- valorEje2
```

Podemos ver los resultados que muestras claramente el sentido de los dos primeros ejes.

```
plot( rasterPCA, col = heat.colors( 26, rev = TRUE, alpha = 0.75 ),
      main = "PCA\nValores en la dimensión 1 o eje I" )
```

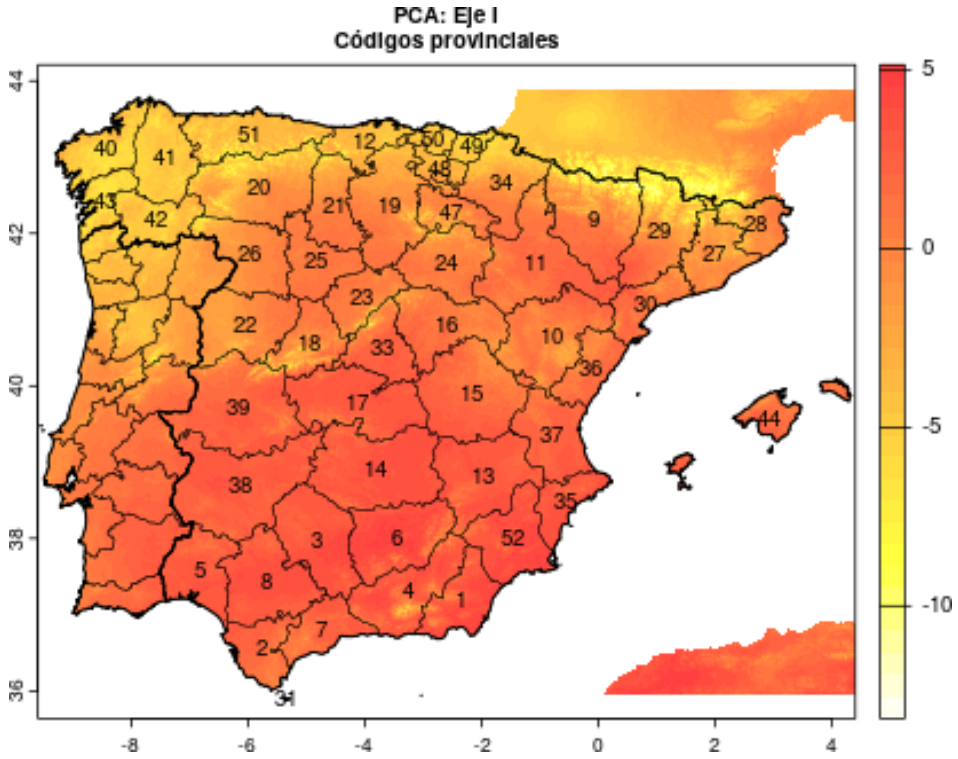



```
plot( rasterPCA2, col = heat.colors( 26, rev = TRUE, alpha = 0.75 ),
      main = "PCA\nValores en la dimensión 2 o eje II" )
```



Para la identificación del comportamiento espacial podemos ayudarnos con, por ejemplo, los límites provinciales.

```
plot( rasterPCA, main = "PCA: Eje I\nCódigos provinciales", col =
     heat.colors( 30, rev = TRUE, alpha = 0.75 ) )
lines( vect( "datos/shp/gadm40_ESP_2.shp" ), lwd = 0.5 )
lines( vect( "datos/shp/gadm40_PRT_1.shp" ), lwd = 0.5 )
lines( vect( "datos/shp/gadm40_ESP_0.shp" ), lwd = 1.4 )
lines( vect( "datos/shp/gadm40_PRT_0.shp" ), lwd = 1.4 )
text( vect( "datos/shp/gadm40_ESP_2.shp" ), cex = 0.8 )
```



12.4.4. Clasificación

Para generar una clasificación de todos los píxeles incluidos en el PCA (1026205)

Esencialmente, con la clasificación vamos a obtener como principal resultado una nueva variable que indique el grupo al que pertenece el píxel. Este resultado lo llevaremos a una capa para poder visualizar mejor el comportamiento.

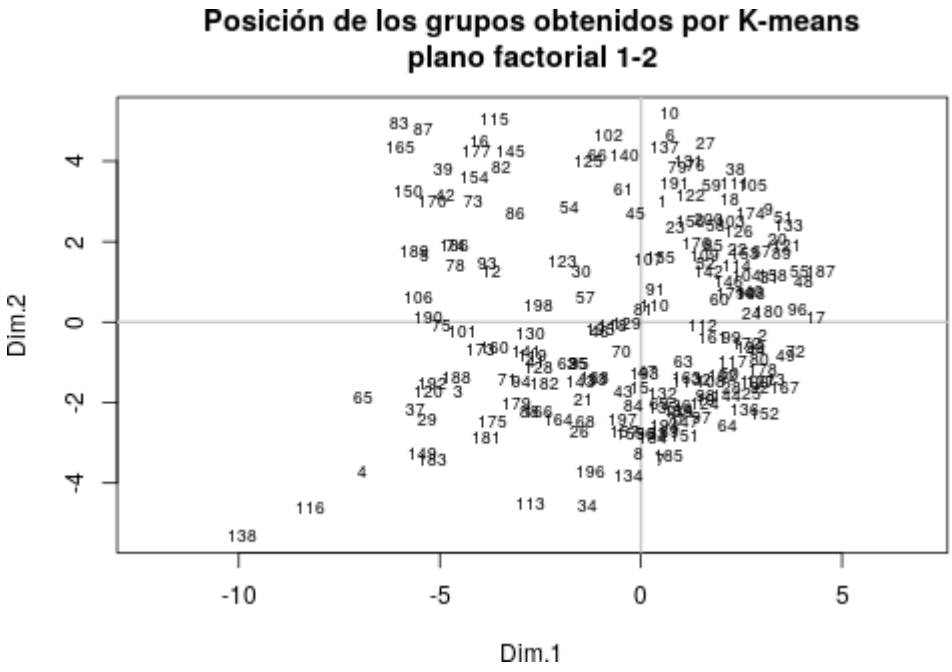
12.4.5. K–means

```
clasifKm <- kmeans( pcaRaster$ind$coord, nKlases, iter.max = 5000 )
```

Obtenemos en primer lugar la partición en 200. que nos proporcionará una primera variable, la encontramos en `clasifKm$cluster`.

Cada grupo queda definido espacialmente en el plano factorial por su centroide `clasifKm$center`. Podemos ver la ubicación de los centroides en el plano facotorial 1–2.

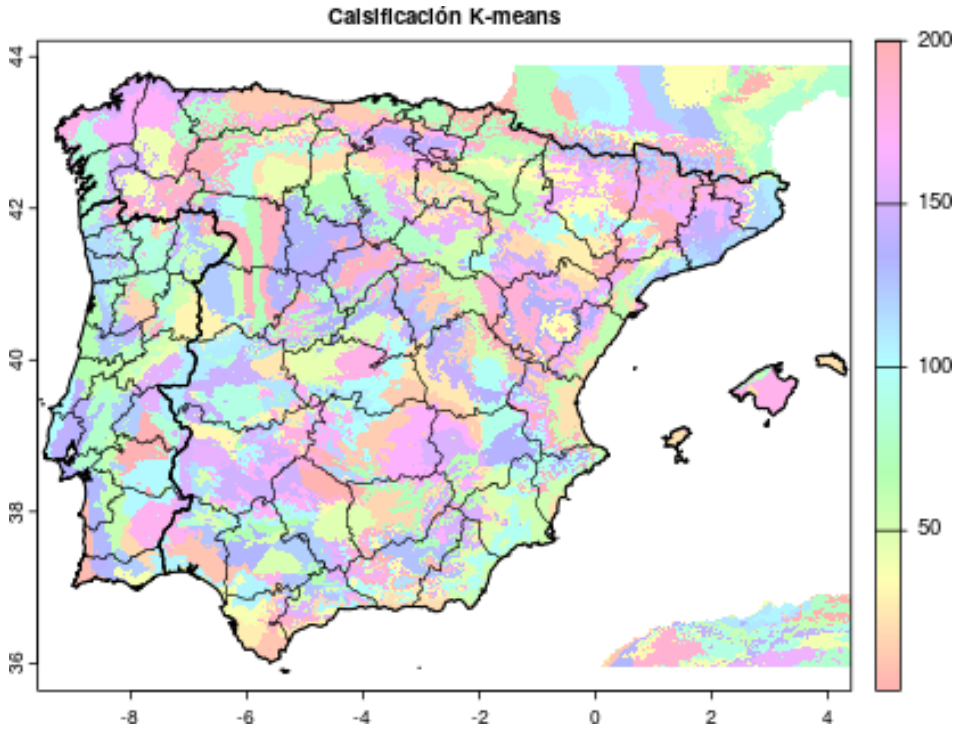
```
plot( clasifKm$centers[  , 1:2 ], type = "n", asp = 1,
   main = "Posición de los grupos obtenidos por K-means\nplano factorial 1-2"
abline( h =0, v=0, col = "grey80", lwd = 2 )
text( clasifKm$centers[  , 1:2 ], labels = rownames( clasifKm$centers ),
     cex = 0.7 )
```



Espacialmente, hacemos como en el caso anterior y generamos la capa a partir de la clase a la que pertenecen los píxeles, los grupos obtenidos presentan la siguiente distribución.

```
rasterKm <- raster[[ 1 ]]
names( rasterKm ) <- "rasterKm"
clasesKm <- rep( NA, ncell( rasterKm ) )
clasesKm[ as.integer( names( clasifKm$cluster ) ) ] <- clasifKm$cluster
rasterKm[] <- clasesKm
```

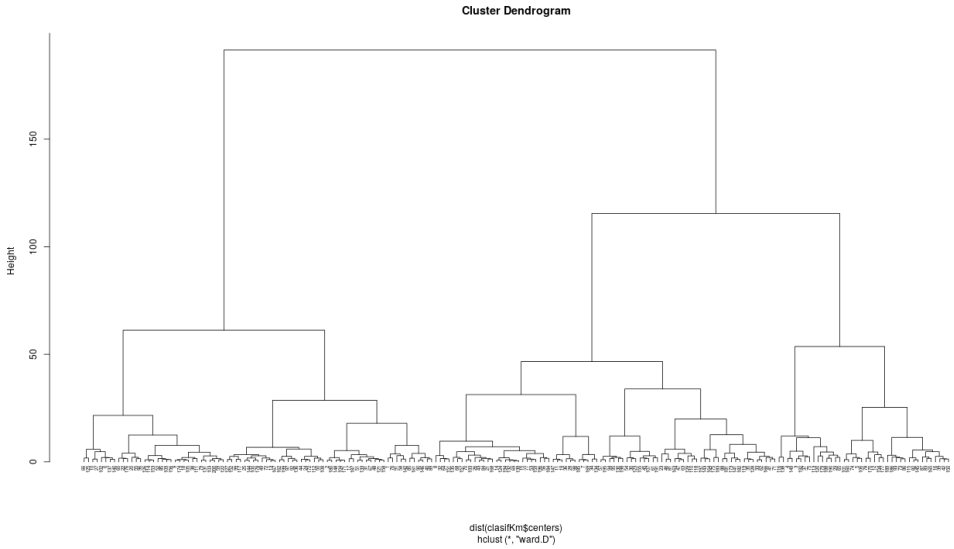
```
colKm <- rainbow( nKlases, alpha = 0.3 )
plot( rasterKm, main = "Calsificación K-means", col = colKm )
lines( vect( "datos/shp/gadm40_ESP_2.shp" ), lwd = 0.5 )
lines( vect( "datos/shp/gadm40_PRT_1.shp" ), lwd = 0.5 )
lines( vect( "datos/shp/gadm40_ESP_0.shp" ), lwd = 1.4 )
lines( vect( "datos/shp/gadm40_PRT_0.shp" ), lwd = 1.4 )
```



12.4.6. Clasificación jeraráquica

Dado el gran numero de clases con el que hemos realizado la partición, vamos a una segunda clasificación que nos permita una mejor interpretación de las clases obtenidas. La clasificación de las clases se obtiene creando una clasificación jerárquica y posteriormente cortando el dendrograma para conseguir las clases integradas.

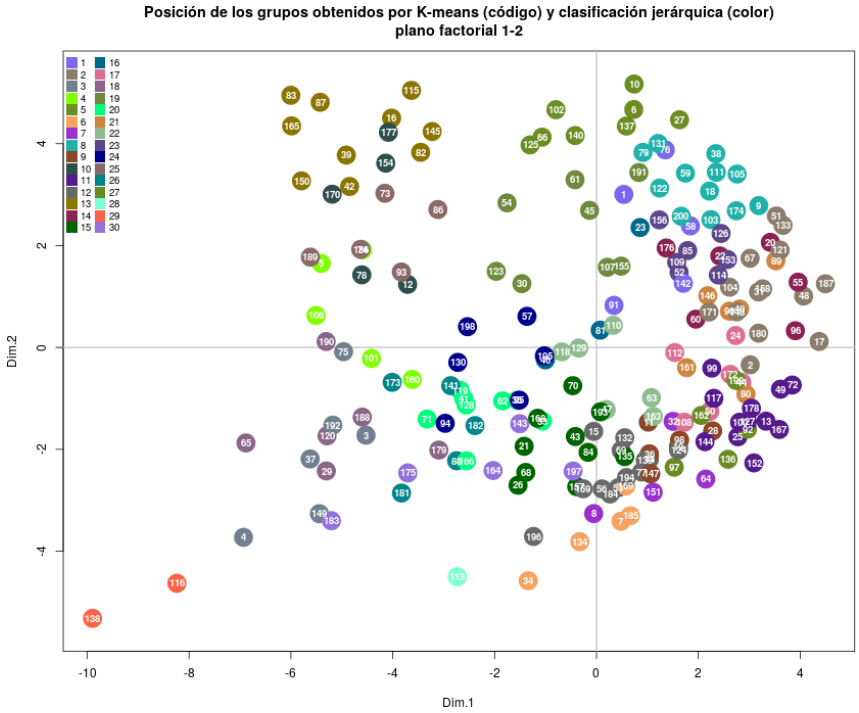
```
clasifHc <- hclust( dist( clasifKm$centers ), method = "ward.D"  )
plot( clasifHc, hang = -1, cex = 0.5 )
```



Para asignar cada grupo de la partición a su nuevo grupo según esta clasificación cortamos el dendrográma para 30 grupos y a continuación generamos la nueva capa siguiendo el procedimiento anterior. También podemos representar de nuevo los grupos obtenidos por *K–means* en el plano factorial como hicimos anteriormente, identificando los nuevos grupos por colores.

```
clasesHc <- cutree( clasifHc, nGrupos )
rasterHc <- rasterKm
rasterHc[] <- clasesHc[ rasterKm[] ]
names( rasterHc ) <- "rasterHc"
colHc <- rainbow( nGrupos, alpha = 0.3 )

set.seed( 2 )
paleta <- sample( readLines( "datos/coloresSelFondoClaro.dat" ), nGrupos )
plot( clasifKm$centers[  , 1:2 ], type = "n", asp = 1,
     main = "Posición de los grupos obtenidos por K-means (código) y"
            "clasificación jerárquica (color)"\n
            plano factorial 1-2" )
abline( h =0, v=0, col = "grey80", lwd = 2 )
points(clasifKm$centers[,1:2],cex= 5,pch= 20, col= paleta[clasesHc])
```

```
text( clasifKm$centers[  , 1:2 ], labels = rownames( clasifKm$centers ),
            cex = 0.8, col = "white", font = 2  )

legend( "topleft", legend = ( 1:nGrupos ), col =  paleta[ 1:nGrupos ],
        pt.cex = 2, box.col = "transparent", cex = 0.8,
        ncol = 2,  pch = 15, bg = "transparent" )
```

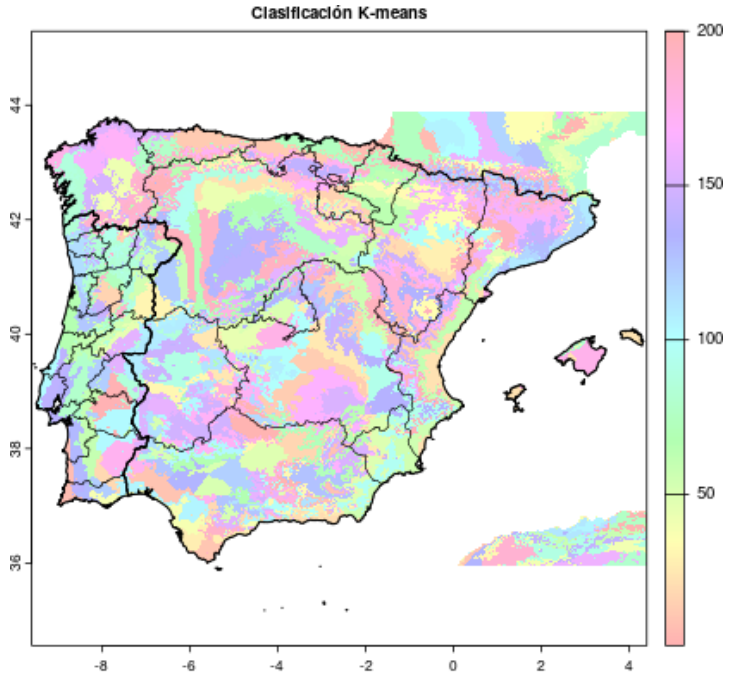


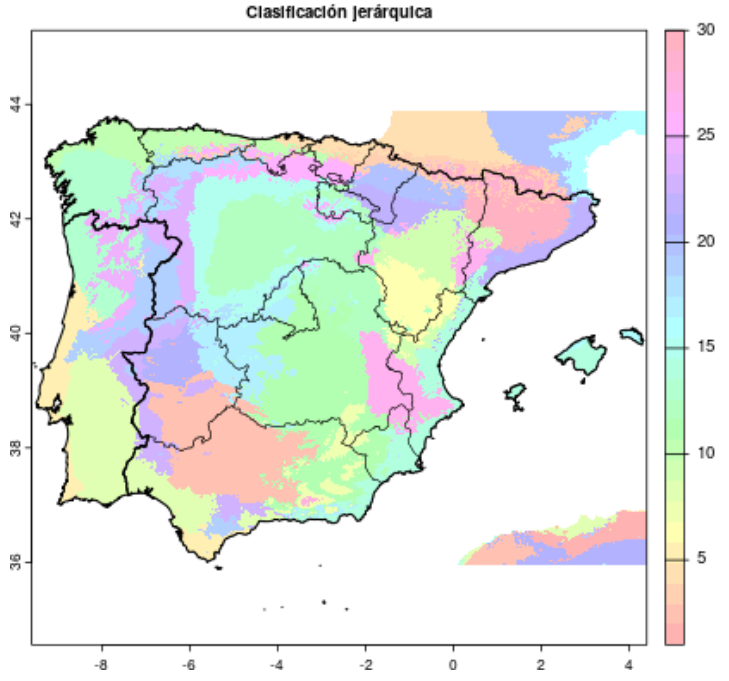
Para comparar los resultados de ambas clasificaciones tenemos los siguiente mapas.

```
par( mfrow = 2:1 )
plot( rasterKm, col = colKm, main = "Clasificación K-means" )
lines( vect( "datos/shp/gadm40_ESP_1.shp" ), lwd = 0.5 )
```

```
lines( vect( "datos/shp/gadm40_PRT_1.shp" ), lwd = 0.5 )
lines( vect( "datos/shp/gadm40_ESP_0.shp" ), lwd = 1.4 )
lines( vect( "datos/shp/gadm40_PRT_0.shp" ), lwd = 1.4 )
```

```
plot( rasterHc, col = colHc, main = "Clasificación jerárquica" )
lines( vect( "datos/shp/gadm40_ESP_1.shp" ), lwd = 0.5 )
lines( vect( "datos/shp/gadm40_ESP_0.shp" ), lwd = 1.4 )
lines( vect( "datos/shp/gadm40_PRT_0.shp" ), lwd = 1.4 )
```


Clasificación K-means
200
150
100
50
44
42
40
38
36
-8
-6
-4
-2
0
2
4


Clasificación jerárquica
30
25
20
15
10
5
44
42
40
38
36
-8
-6
-4
-2
0
2
4

12.4.7. Visión local

Para visualizar una región dada con el máximo detalle, seguiremos el procedimiento relativamente simple de recortar la información en función de los límites de la unidad que nos interesa.

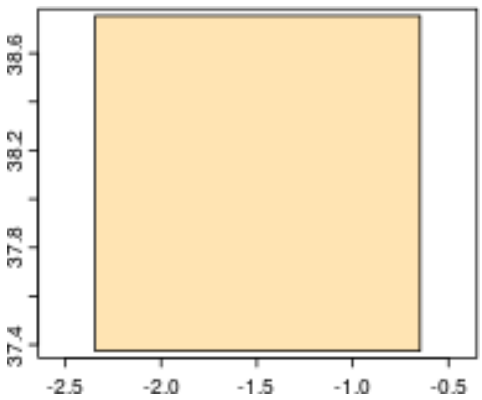
Es importante tener en cuenta que a la vez que recortamos espacialmente la información también debemos recortar las leyendas, ya que la clasificación de una subregión no tiene tantas clases como las que presenta la pi.

Supongamos que queremos trabaja con la Región de Murcia, que presenta el código 52 en el nivel administrativo 2 de GADM.

12.4.8. Recortando

Determinamos la extensión de la región —incluyendo los límites de la Región de Murcia— y recortamos la información de las distintas capas vectoriales para obtener solo la presente en la región,

```
murLimV <- ( vect( "datos/shp/gadm40_ESP_2.shp" ) )[ c( 52 ) ]
prov <- ext( murLimV )

murMunV <- crop( vect( "datos/shp/gadm40_ESP_4.shp" ), murLimV )
murComV <- crop( vect( "datos/shp/gadm40_ESP_3.shp" ), murLimV )

plot( prov, col = "orange", alpha = 0.3 )
```

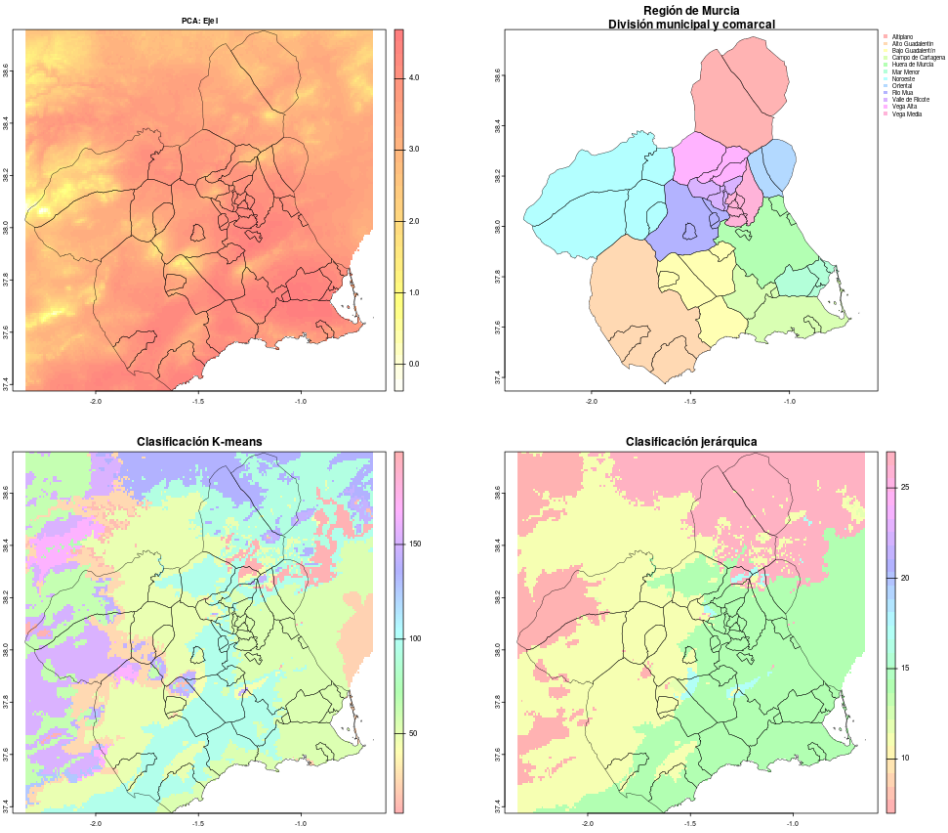


También recortamos las capas ráster convenientemente a la región de trabajo.

```
murPCA <- crop( rasterPCA, prov  )
murKm  <- crop( rasterKm,  prov )
murHc  <- crop( rasterHc,  prov )
```

Preparamos una visualización comparativa de las capas disponibles.

```
par( mfrow = c( 2, 2 ) )
# Generamos una paleta de color adecuada a cada clasificación
colKm <- rainbow( max( murKm[], na.rm = TRUE ), alpha = 0.3 )
colHc <- rainbow( max( murHc[], na.rm = TRUE ), alpha = 0.3 )

plot( murPCA, main = "PCA: Eje I", col = heat.colors( 30, rev = TRUE,
alpha = 0.5 ) )
lines( murMunV, lwd = 0.5 )

plot( murComV, "NAME_3",
      border = "transparent", plg = list( cex = 0.6 ),
      col = rainbow( nrow(murComV ) ), alpha = 0.3,
      main = "Región de Murcia\nDivisión municipal y comarcal" )
lines( murMunV, lwd = 0.5 )

plot( murKm, main = "Clasificación K-means", col = colKm, cex.main = 1.2 )
lines( murMunV, lwd = 0.5 )

plot( murHc, main = "Clasificación jerárquica", col = colHc, cex.main = 1.2  )
lines( murMunV, lwd = 0.5 )
```

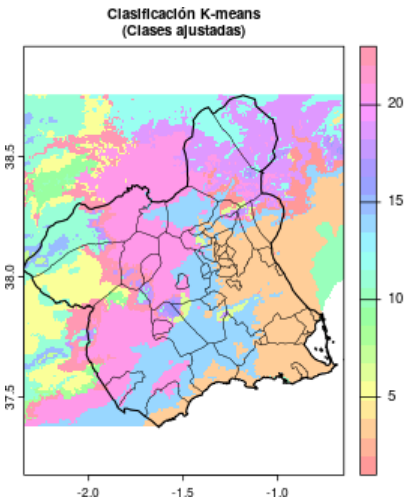


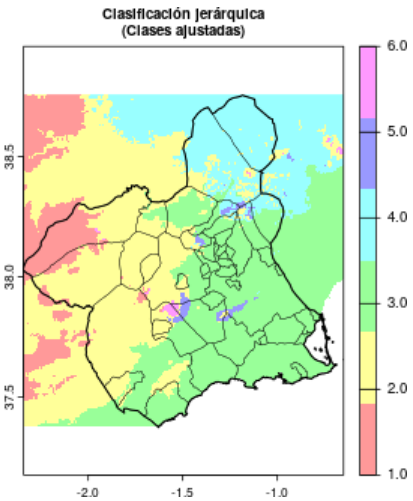
12.4.9. Reajustando la leyenda

Los resultados anteriores muestran una aparente falta de detalle, esto se debe al la proximidad de color entre clases similares. El número de clases obtenidas por *K–means* presentes en la región es de 23, mientras, que en el caso de la clasificación jerárquica es de 6.

```
par( mfrow = c( 1, 2 ) )
valores <- unique( murKm[ !is.na( murKm ) ] )
murKmS <- subst( murKm, valores, order( valores ) )
colKm <- rainbow( max( murKmS[], na.rm = TRUE ) )

plot( murKmS, col = colKm, alpha = 0.4, main =
     "Clasificación K-means\n(Clases ajustadas)" )
```

```
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )

valores <- unique( murHc[ !is.na( murHc ) ] )
valores <- valores[ !is.na( valores ) ]
murHcS <- subst( murHc, valores, order( valores ) )
colHc <- rainbow( max( murHcS[], na.rm = TRUE ) )

plot( murHcS, col = colHc, alpha = 0.4, main =
    "Clasificación jerárquica\n(Clases ajustadas)"  )
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )
```

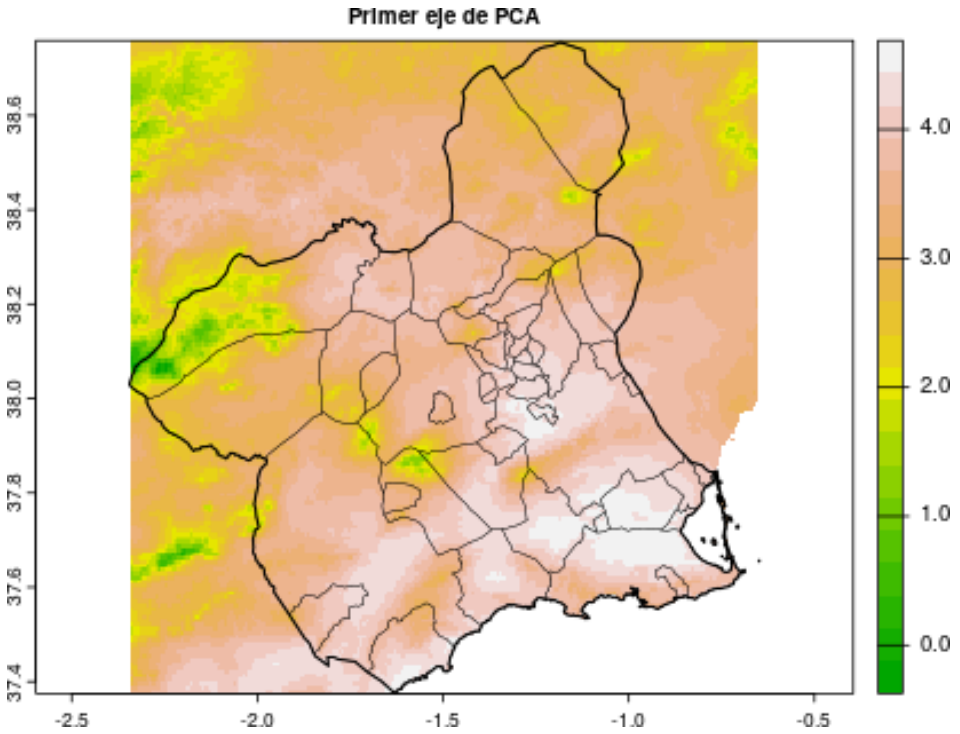




12.4.10. Comparando ejes y clases

Utilizamos la capa correspondiente al primer eje del PCA como fondo para ayudar a la interpretación de las clasificación en el región.

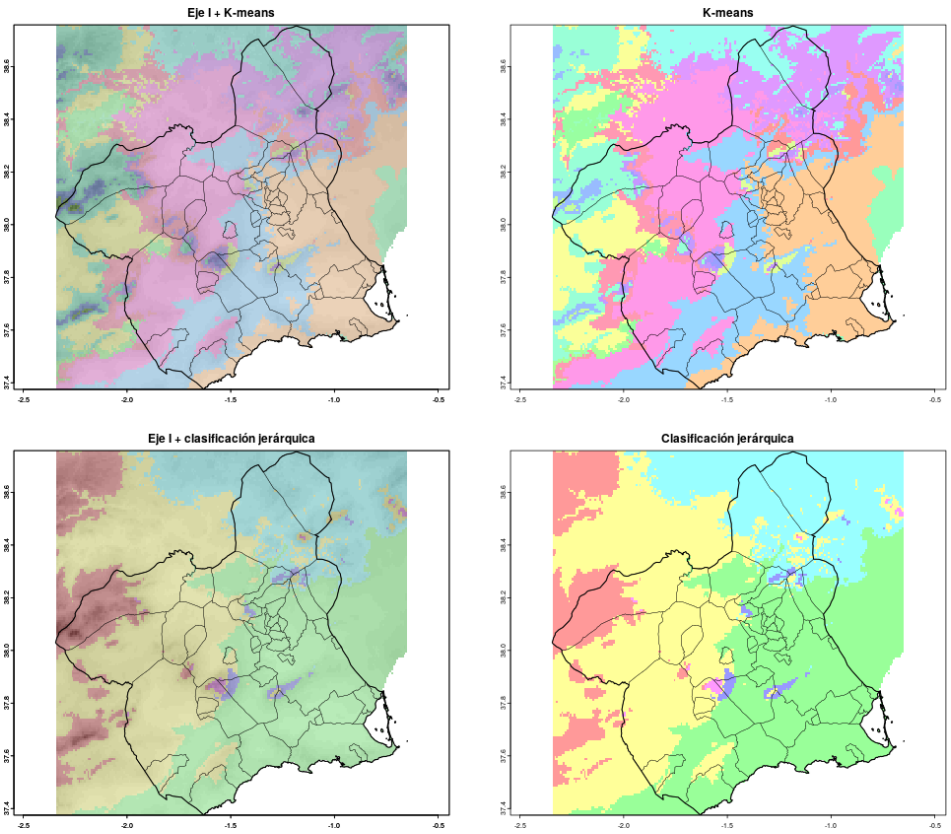
```
plot( murPCA, col = terrain.colors( 20 ), main = "Primer eje de PCA" )
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )
```



La comparación gráficamente es la siguiente.

```
par( mfrow = c( 2, 2 ) )
plot( murPCA, col = grey.colors( 20 ), legend = FALSE )
title( "Eje I + K-means" )
plot( murKmS, add = TRUE, alpha = 0.2, legend = FALSE, col = colKm )
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )

plot( murKmS, legend = FALSE, col = colKm, alpha = 0.4 )
title( "K-means" )
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )

plot( murPCA, col = grey.colors( 20 ), legend = FALSE )
title( "Eje I + clasificación jerárquica" )
plot( murHcS, add = TRUE, alpha = 0.2, legend = FALSE, col = colHc )
```

```
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )

plot( murHcS, legend = FALSE, col = colHc, alpha = 0.4 )
title( "Clasificación jerárquica" )
lines( murMunV, lwd = 0.4 )
lines( murLimV, lwd = 1.4 )
```



12.5. Resumen

- Se habla de datos multivariantes cuando los objetos de estudio son descritos por un número alto de variables. Formalmente dos o más variables ya se podría considerar multivariante.
- Las técnicas multivariantes permiten simplificar y ordenar los datos eliminando redundancia o bien agruparlos según características similares. Esto facilita la exploración e interpretación de la información contenida en los datos.
- Las técnicas multivariantes de clasificación, consisten en clasificar o agrupar a los individuos según características homogéneas. En la clasificación jerárquica, los individuos son agregados (o desagregados) de forma progresiva, según su similitud. En la clasificación iterativa o no jerárquica, se realiza una partición de los individuos en grupos homogéneos.
- En las técnicas multivariantes de ordenación se busca explicar la mayor cantidad de información posible, con la menor cantidad de variables posibles. Para ello se crean nuevas variables, combinación lineal de las originales, eliminando información redundante y asumiendo una mínima pérdida de información. Los datos son "ordenados" a lo largo de estos nuevos ejes que son las nuevas variables.
- El PCA es una técnica de ordenación empleada cuando existe correlación entre las variables y éstas son continuas. El CA se emplea con el mismo propósito para variables cualitativas. En este caso segundo caso no se habla de correlación de variables sino de dependencia o independencia entre ellas.
- En PCA, las nuevas variables (factores o componentes), son ortogonales entre sí, su correlación es cero. Es importante conocer la proporción de varianza total explicada por cada factor, para determinar cuántos factores serán necesarios en análisis posteriores.
- En CA, el número de dimensiones (factores, componentes), será siempre una menos que el número de categorías de la variable con menos categorías. Las inercias asociadas a cada eje reflejan la proporción de variación explicada por uno.

Bibliografía

[1] Stephen McDaniel. Powerful dashboard frameworks for R shiny apps in 2021, 2021.

[2] Yihui Xie. Dynamic Documents with R and knitr. Chapman & Hall/CRC, 2013.

[3] Christopher Gandrud. Reproducible Research with R and RStudio. Chapman & Hall/CRC Press, 2013.

[4] Ángel Carmona, Francesc y Berihuete. Curso de R básico, 2022.

[5] Joseph Adler. R in a Nutshell. O'Reilly Media, 2nd edition, 2012.

[6] Emmanuel Paradis. R para Principiantes. Paradis, Emmanuel, 2003.

[7] Robert Kabacoff. R in Action. Manning Publications, 1st edition, 2011.

[8] Función de densidad de probabilidad - Wikipedia, la enciclopedia libre.

[9] T. D. V. Swinscow and MJ Campbell. Statistics at Square One. BMJ Publishing Group, 10th edition, 2002.

[10] Martin Krzywinski and Naomi Altman. Points of Significance: Error bars. Nature methods, 10(10):921–922, 2013.

[11] Florence Nightingale - Wikipedia, la enciclopedia libre.

[12] Harvey Motulsky. Intuitive Biostatistics. Oxford University Press, Inc, January 1995.

[13] Martin Krzywinski and Naomi Altman. Points of significance: Importance of

being uncertain. Nature methods, 10(9):809–810, 2013.

[14] Harvey Motulsky. Analyzing Data With GraphPad Prism. GraphPad Software, mar 1999.

[15] Wikipedia. Distribución normal- Wikipedia, la enciclopedia libre, 2022. [Internet; descargado 4-marzo-2022].

[16] Teorema del límite central- Wikipedia, la enciclopedia libre.

[17] JF.J Braón-López. Bioestadística. Teorema del límite central - YouTube.

[18] J. Martin Bland. An introducction to medical statistics. Oxford Medical Publications, 2000.

[19] John Verzani. Using R for introductory statistics. Chapman and Hall/CRC, 2004.

[20] M. G Bulmer. Principles of Statistics. Dover Books on Mathematics, 1979.

[21] Stan Brown. Measures of Shape: Skewness and Kurtosis.

[22] Morton B. Brownab and Alan B. Forsythec. Robust Tests for the Equality of Variances. Journal of the American Statistical Association, Volume 69, Issue 346, 1974(96):364–367, 1974.

[23] Antonio Valera Espín and Julio Sánchez Meca. Pruebas de significación y magnitud del efecto: Reflexiones y propuestas.

[24] Andy Field, Jeremy Miles, and Zoe Field. Discovering Statistics Using R. Sage Publications Ltd, 1st edition edition, 2012.

[25] Francisco Javier Barón López. Apuntes y vídeos de Bideoestadística.

[26] Juan Vilar. 4. Chequeo y validación del modelo con un factor. Independencia de los errores.

[27] Russ Clay. Open Science Colaboration Blog: Confidence Intervals for Effect Sizes from Noncentral Distributions.

[28] Peter Watson. Rules of thumb on magnitudes of effect sizes.

[29] Richard Hall. Between-SubjectsOne-Way ANOVA Description.

[30] Bertolt. Obtaining the same ANOVA results in R as in SPSS - the difficulties with Type II and Type III sums of squares .

[31] Repeated Measures ANOVA.

[32] Jonathan Baron. Notes on the use of R for psychology experiments and questionnaires.

[33] Winston Chang. Cookbook for R. ANOVA.

[34] Karen Grace-Martin. Why ANOVA and Linear Regression are the Same Analysis.

[35] John M Quick. R Tutorial Series ANOVA pairwise comparison methods.

[36] Dennis D. Wackerly, William Mendenhall, and Richard L. Scheaffer. Mathematical Statistics with Applications. Cengage Learning, 7th edition, 2008.

[37] Harvey Motulsky. Choosing a statistical test (material de Intuitive Biostatistics), 2012.

[38] Jean Dickinson Gibbons and Subhabrata Chakraborti. Nonparametric Statistical Inference. Marcel Dekker, 4th ed edition, 2003.

[39] Patrick Giraudoux. pgirmess & pgirbric: Miscellaneous functions for data handling and analysis in ecology, 2014.

[40] Alan Agresti. Categorical Data Analysis. Wiley-Interscience, 2nd ed edition, 2002.

[41] PennState University. STAT 504 - Analysis of Discrete Data, 2014.

[42] Carolyn J. Anderson. Applied Categorical Data Analysis, 2014.

[43] Jessica DeVries. About Chi Squares, 2007.

[44] University of Nebraska–Lincoln. Bivariate Statistics Hand-Computation Cache: Cochran's Q Test, 2007.

[45] Koji Yatani. Statistical Methods for HCI Research, 2014.

[46] Wikipedia. G-test — Wikipedia, The Free Encyclopedia, 2014.

[47] John H. McDonald. Handbook of Biological Statistics, 2014.

[48] Jaume Llopis Pérez. La estadística: una orqueta hecha instrumento, 2014.

[49] José Gabriel Palomo Sánchez. Regresión lineal simple, 2011.

[50] Denise Ferrari and Tiffany Head. Regression in R. Part I: Simple Linear Regression, 2010.

[51] Samprit Chattefuee and Ali S. Hadi. Linear Models with R. John Wiley & Sons, 4th edition edition, 2006.

[52] SPSS. Análisis de regresión lineal: El procedimiento Regresión lineal, 2007.

[53] Julian J. Faraway. Linear Models with R. Taylor & Francis e-Library, 1st edition edition, 2009.

[54] Robert Kabacoff. Creating a figure arrangement with fine control, 2014.

[55] SDSU's Statistical Consulting Group (SCG). Multiple Linear Regression (R), 2013.

[56] Joaquín González-Revaldería, Juan Manuel Paz Fernández, Belén Prieto García, and Josep M. Queraltó. Curso de Estadística para el laboratorio Clínico. Módulo 3: Regresión logística, 2007.

[57] UCLA: Statistical Consulting Group. R Data Analysis Examples: Logit Regression, 2014.

[58] UCLA: Statistical Consulting Group. FAQ: How do I interpret odds ratios in logistic regression?, 2014.

[59] Stephane Champely. pwr: Basic Functions for Power Analysis, 2020. R package version 1.3-0.

[60] Jacob Cohen. Statistical power analysis for the behavioral science (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[61] Antonio Maurandi-López, Carlos Balsalobre-Rodríguez, and Laura del Río-Alonso. Fundamentos estadísticos para investigación. Introducción a R. BUBOK Publishing S.L., March 2013.

[62] Robert Kabacoff. Quick-R. Basic graphs, 2014.

[63] Roger D. Peng, Jeff Leek, and Brian Caffo. Quick-R. Basic graphs, 2014.

[64] Josh Peterson. Graph Gallery: a collection, 2013.

[65] Harley Wickham. ggplot2: elegant graphics for data analysis. Springer-Verlag New York, 2009.

[66] Alboukadel Kassambara. ggplot2: guide to create beautiful graphics in R. Alboukadel Kassambara, 2013.

[67] Winston Chang. R graphics cookbook: practical recipes for visualizing data. O'Reilly Media, 2018.

[68] Colin Fay, Sébastien Rochette, Vincent Guyader, and Cervan Girard. Engineering Production-Grade Shiny Apps. Chapman and Hall/CRC, 2021.

[69] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, et al. Welcome to the Tidyverse. Journal of open source software, 4(43):1686, 2019.

[70] Antonio Matas. Diseño del formato de escalas tipo Likert: un estado de la cuestión. Revista electrónica de investigación educativa, 20(1), 2018.

[71] Rensis Likert. A technique for the measurement of attitude. Archives of Psychology, 140:5–55, 1932.

[72] J Creswell. Educational research: planning conducting and evaluating quantitative and qualitative research. Pearson, 4nd edition, 2012.

[73] J. S. Uebersax. Likert scales: Dispelling the confusion, 2006.

[74] Nadler J., Weston R., and Voyles E. Stuck in the middle: the use and interpretation of mid-points in items on questionnaires. The Journal of General Psychology, 142(2):71–89, 2015.

[75] J. F. Morales. Las escalas de actitudes. UNED, 1981.

[76] George Darren. SPSS for Windows step by step: a simple study guide and reference. Pearson Education - India, 2011.

[77] J. F. Hair, W. C. Black, Babin B. J., and R. E. Anderson. Multivariate Data Analysis. Pearson Prentice Hall, 7th edition edition, 2009.

[78] Tomás Ventura-León, José Luis y Caycho-Rodríguez. El coeficiente Omega: un método alternativo para la estimación de la confiabilidad . Revista Latinoamericana de Ciencias Sociales, Niñez y Juventud, 2017.

[79] American Educational Research Association, American Psychological Association y National Council on Measurement in Education. Estándares para pruebas educativas y psicológicas (M. Lieve, Trans.). American Educational Research Association, 2018.

[80] Arias, A. and Sireci, S. G. Validez y Validación para Pruebas Educativas y Psicológicas . Revista Iberoamericana de Psicología, 14(1):11–22, 2021.

[81] Yves Rosseel. lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2):1–36, 2012.

[82] A Alexander Beaujean. Latent variable modeling using R: A step-by-step guide. Routledge, 2014.

[83] Paul Barrett. Structural Equation Modelling: Adjudging Model Fit. Personality and Individual Differences, 42:815–824, 05 2007.

[84] Leslie Hayduk, Greta Cummings, Kwame Boadu, Hannah Pazderka-Robinson, and Shelley Boulianne. Testing! testing! one, two, three – Testing the theory in structural equation models! Personality and Individual Differences, 42(5):841–850, 2007. Exported from https://app.dimensions.ai on 2019/03/20.

[85] David A Kenny. Structural equation modeling. Retrieved May, 27:2013, 2012.

[86] James H. Steiger. Structural Model Evaluation and Modification: An Interval Estimation Approach. Multivariate Behavioral Research, 25(2):173–180, 1990. PMID: 26794479.

[87] Pedro Morales Vallejo. Estadística aplicada a las Ciencias Sociales. Universidad Pontificia Comillas, 2008.

[88] Francisco Javier Ibáñez-López, Antonio Maurandi-López, and José Francisco

Castejón Mochón. Docencia práctica virtual y adquisición de competencias en la formación estadística de maestros durante el confinamiento sanitario. PNA, 16(2):99–113, 2022.

[89] Gavin Simpson. CRAN Task View: Analysis of Ecological and Environmental Data, 2022. Version 2014-07-25.

[90] MA Anderberg. Cluster analysis for applications. New York Acad Press, 1973.

Este libro de Análisis de Datos y Métodos Estadísticos con R (ADyMER) recopila parte del trabajo de muchos años que hemos realizado varios compañeros de la Universidad de Murcia (UMU) de distintas disciplinas, pero con el nexo común del entendimiento de R como *lingua franca* que conecta las ciencias y humanidades con el análisis de datos. ADyMER da forma y amplía el material de numerosos cursos y talleres que hemos impartido a distintos niveles y formatos, habiéndonos mezclado maestros con aprendices, profesores novatos y avezados, creando una filosofía especial y dando lugar a agrupaciones de divulgación del conocimiento de R en todas sus vertientes, especialmente la investigadora, docente y empresarial.

En ADyMER se tratan las particularidades de R, las definiciones formales y uso en R de la estadística descriptiva y contrastes de hipótesis (paramétricos y no paramétricos), los modelos de regresión (lineal y logística) y potencia estadística, detalles sobre la visualización de resultados mediante gráficas, tablas y aplicaciones web con R, la manipulación limpia (*tidy*) de datos, el análisis de cuestionarios mediante escalas Likert y análisis multivariante. Como puede ver el lector, es un paseo de aprendizaje de herramientas y librerías de R para el análisis de datos, permitiendo centrarse en la tarea que más necesaria le resulte sin dejar atrás conceptos básicos y generales del manejo de R.