Técnicas de Análisis de Datos Económicos con R

2024-09-25

Índice general

Autores:

Miguel Ángel Tarancón Morán. Catedrático de Economía Aplicada. Universidad de Castilla - La Mancha.

Prefacio

Este libro recoge las diversas prácticas que se han ido desarrollando a lo largo de multitud de cursos en varias asignaturas de grado y máster relacionadas con el análisis de datos, especialmente de tipo económico, en la Facultad de Derecho y Ciencias Sociales de Ciudad Real.

Gracias a tantas y tantas personas y compañeros/as que hacen posible la construcción y mejora del libro.

Capítulo 1

Introducción.

1.1 Llámalo Estadística.

Todo el mundo habla de las estadísticas. Constantemente se hace referencia a estas en los medios de comunicación, todo está medido y estructurado por estos entes que convierten la realidad en una amalgama de números. Y más aún en el campo del comportamiento humano, es decir, lo que conocemos como *Ciencias Sociales*. A diario nos llegan las estadísticas sobre la intención de voto cuando hay unas elecciones, del crecimiento de la economía en términos del *PIB*, del comportamiento de los precios medido mediante el concepto de *inflación*...

El secreto de la relevancia que les damos a las estadísticas subyace en que, de partida, suponen una forma sintética y objetiva de representar la realidad que nos rodea, de manera que podemos abarcar el conocimiento de tal realidad de un modo más o menos plausible. Y esta representación de la realidad es a priori *objetiva* porque las estadísticas se elaboran siguiendo unas metodologías que se apoyan en un lenguaje universal: las matemáticas.

Sí. El lenguaje matemático es un lenguaje que pueden entender todas las personas, tengan la procedencia que tengan, y sean de la condición que sean. Si necesitas comunicarte con *casi* cualquier persona del mundo, habla en inglés. Si necesitas comunicarte con cualquier persona del mundo, hazlo mediante las matemáticas, aunque sean matemáticas más o menos elementales.

Por ello, las estadísticas se expresan en lenguaje matemático.

Pero lo que comúnmente entendemos como *estadísticas* no son más que unos resultados, unos outputs de la Estadística. La Estadística en realidad es algo mucho más complejo. Es una Ciencia. Las estadísticas son construidas usando el método estadístico; pero la Estadística se utiliza para muchas más cosas que para publicar estadísticas.

1.1.1 Concepto de Estadística.

El término "Estadística" proviene de la palabra latina *status*, "el Estado", y fue acuñado por Achenwall a mediados del siglo XVIII con el significado de "recogida, procesamiento y utilización de datos por parte del Estado".

Sin embargo, tal y como se entiende hoy en día, es decir, en el sentido de ***Ciencia Estadística***, surgió como resultado de la integración de dos disciplinas: la *Aritmética Política*, en ese sentido de la cuantificación del Estado; y del *Cálculo de Probabilidades*, que nace en el siglo XVII como Teoría Matemática de los juegos de azar y que podríamos asociar al sentido de *Estadística Matemática*.

Ciñéndonos pues a este último sentido, a lo largo de la Historia se han dado múltiples definiciones de Estadística. Fisher propone una definición quizá demasiado generalista al decir que la Ciencia Estadística es

esencialmente una rama de las matemáticas aplicada a los datos observados. Una reflexión que puede ayudar a delimitar la definición de la Ciencia Estadística es la que realiza (Peña 1983) cuando realiza la siguiente reflexión:

> *"La Estadística como disciplina científica ocupa un lugar muy singular en el conjunto de las ciencias. La Física, la Medicina o la Sociología tienen un área sustantiva de conocimiento y cuando utilizan modelos matemáticos, los subordinan al objeto principal de hacer avanzar el conocimiento en su parcela de estudio de la realidad. El objetivo de la Matemática, en contraposición, es ampliar la concepción y generalidad de sus propias herramientas analíticas, con absoluta independencia de la posible relación entre los entes matemáticos abstractos y los fenómenos reales. La Estadística participa de esos dos objetivos, aunque con rasgos muy peculiares. Su campo de estudio son los fenómenos aleatorios que están presentes, en mayor o menor medida, en toda actividad humana de adquisición de conocimiento empírico."*

En este mismo sentido, (Martín-Pliego 2004) apunta:

> *"La Estadística, por tanto, se configura como la tecnología del método científico que proporciona instrumentos para la toma de decisiones cuando éstas se adoptan en ambiente de incertidumbre, siempre que esa incertidumbre pueda ser medida en términos de probabilidad. Por ello, la Estadística se preocupa de los métodos de recogida y descripción de datos, así como de generar técnicas para el análisis de esta información."*

En definitiva, la Estadística reúne tanto la concepción derivada de la *Aritmética Política*, entendida como recopilación sistemática de datos cara a la descripción de la realidad ("hacer" *estadísticas*); como la concepción *probabilística*, entendida como la modelización de dicha realidad cuando está inscrita en un ambiente de incertidumbre, con el objeto de acotar dicha incertidumbre y servir de ayuda en la toma decisiones (*representar matemáticamente el comportamiento de fenómenos sujetos a incertidumbre*, cuando contamos con **datos** que caracterizan a esos fenómenos).

1.1.2 El método estadístico.

En cuanto al método seguido por la Ciencia Estadística, prima el r**azonamiento *inductivo:*** las hipótesis que se plantean en la investigación implican propiedades observables en un conjunto de casos, cuyo análisis lleva a formular hipótesis más generales, aplicables ya a un conjunto mayor de casos. El método estadístico consiste, en definitiva, en sistematizar y organizar este procedimiento de aprendizaje que parte de lo particular para llegar a lo general.

En la aplicación del método estadístico podemos diferenciar una serie de **etapas básicas** que se exponen a continuación:

a) **Planteamiento del problema.** Consiste en definir el objeto de la investigación, (¿qué quiero obtener? ¿a dónde quiero llegar?), para lo cual debemos precisar la **población** de referencia y determinar las características que debemos observar y cómo serán recogidas. El resultado de esta fase es un sistema de **características** de interés observadas en un subconjunto de la población representativo de esta, al que llamamos **muestra**. Estas características se llamarán variables si están en escala métrica; o atributos, variables cualitativas o factores si están en escala no-métrica (nominal u ordinal). Las variables toman valores para cada elemento o caso de la muestra. Los atributos adoptan una categoría o nivel para cada uno de los casos que integran la muestra. Según los objetivos planteados en la investigación, el tamaño de la muestra, tipo de características, etc., se podrá hacer una primera selección de los posibles tipos de técnicas y modelos estadíticos a aplicar.

b) **Recogida y preparación de la información muestral**. Los datos, que son los valores (en caso de trabajar con variables) o categorías o niveles (en el caso de trabajar con atributos o factores) que adoptan los distintos casos que constituyen la muestra en relación con las características de interés de la población; han de ser obtenidos de las fuentes disponibles. Estas fuentes pueden ser primarias, cuando somos los propios

investigadores los que generamos los datos (a través de la observación o la realización de encuestas), o secundarias, cuando estos datos ya han sido generados y/o recopilados por otros investigadores o instituciones. En cualquier caso, la muestra debe ser lo suficientemente amplia como para extraer conclusiones válidas para toda la población, y los datos deben ser de calidad, pues son la materia prima con la que trabajamos. Para ello, un requisito importante es que las fuentes de datos sean fiables.

c) **Depuración de los datos**. Antes de utilizar los datos muestrales conviene aplicar un análisis descriptivo que permitirá detectar posibles inconsistencias en los datos identificando los valores anómalos, posibles errores, etc. En esta fase es clave tanto identificar las carencias de datos existentes (datos faltantes o *missing data*), como identificar aquellos elementos de la muestra que no representan bien a la población, puesto que presentan comportamientos extraños en alguna o algunas de las variables o atributos en estudio (casos atípicos u *outliers*).

d) **Aplicación de técnicas o modelos estadísticos** para obtener resultados generalizables al conjunto de la población. Una vez se tienen claros los objetivos de la investigación y las características de la información muestral de la que se dispone (datos), y se han depurado convenientemente los datos, será el momento de plantear qué técnica o modelo estadístico aplicar. Aquí podemos distinguir, a su vez, distintas subetapas.

- Por un lado, la aplicación correcta de ciertas técnicas o modelos de naturaleza inferencial, requiere del **cumplimiento por parte de los datos de ciertos patrones de comportamiento** (por ejemplo, el cumplimiento por parte de las variables de un comportamiento acorde con una Ley Normal). Así, deberán aplicarse una serie de pruebas para comprobar hasta qué punto los datos de partida cumplen con estos patrones.
- Tras superar el punto anterior, podrá aplicarse la técnica o modelo a los datos para obtener los resultados que contribuyan a cubrir los objetivos de la investigación (usualmente, esta etapa se corresponde con la de **estimación** del modelo estadístico aplicado).
- Por último, los resultados deben ser sometidos a una subetapa de **validación y contraste**, en la que se valora hasta qué punto los resultados representan el comportamiento real de los casos estudiados (estudio de la bondad del modelo), y el grado de aptitud técnica del modelo, en el sentido de si el modelo estimado cumple con los requisitos que garantizan la calidad de los resultados (por ejemplo, si se cumplen ciertas hipótesis básicas que garanticen que los coeficientes estimados del modelo gozan de las mejores propiedades estadísticas, como insesgadez, eficiencia y consistencia).
- En esta etapa, además, se intentará simplificar el modelo, es decir, conseguir un modelo tan sencillo como sea posible, sin más parámetros de los necesariosy, que represente la realidad sin mucha pérdida de calidad con respecto a otro modelo más complejo, o sea, ciñéndose al ***principio de parsimonia*** de la modelización.

e) **Crítica y diagnosis del modelo**. Si una vez culminada la fase anterior se considera que el modelo es válido y técnicamente correcto, podrá ser adoptado para ayudar a la toma de decisiones, mediante análisis estructural, realización de previsiones o planteamiento de simulaciones. En caso contrario, si el modelo no se considera válido y/o correcto, deberemos reformular dicho modelo repitiendo las etapas anteriores hasta obtener un modelo que represente la realidad en estudio más adecuado.

En definitiva, el método estadístico sigue el método científico en cuanto a que tiene unas etapas bien delimitadas en las que se trata el **conocimiento *a priori*** (teoría) para obtener un **conocimiento *a posteriori*,** lo que pasa a engrosar el cuerpo de la Ciencia.

Es relevante destacar cómo, a su vez, el método científico, al ser aplicado al resto de ciencias, y a la propia Ciencia Estadística, recurre al método estadístico en su ejecución. Así, por ejemplo, en la etapa de recogida de evidencias observables (datos), a fin de verificar las consecuencias o hipótesis que se desprenden de una teoría previa, la Estadística interviene tanto a partir de la *Teoría de Muestras* como del *Diseño de Experimentos* para garantizar la validez y coherencia de los datos. En una fase posterior del método científico, se pasaría a verificar la nueva teoría que se desprende de las hipótesis articuladas a partir de la teoría preexistente. Nuevamente aquí interviene la Estadística como herramienta auxiliar, mediante la *modelización inferencial*. Además, en

todo el proceso, que abarca tanto la observación de la realidad como a la generalización de los resultados como modo de confirmar una nueva teoría, aparece la incertidumbre en mediciones y resultados, por lo que el papel de la Estadística como procedimiento para la medición de dicha incertidumbre es indispensable.

De lo dicho se desprende una característica que hace de la Estadística una ciencia singular: su carácter de ***ciencia instrumental*** que auxilia al resto de ciencias en el desarrollo de sus cuerpos de conocimiento. De ahí que la Estadística es aplicada en la totalidad de las ciencias, bien sean naturales, jurídicas o sociales, y en todos los campos del saber, desde las áreas más técnicas hasta en las propias humanidades. Es decir, la Estadística es una herramienta fundamental en todo el proceso de adquisición de conocimientos a través de datos empíricos y, desde este punto de vista, podemos referirnos a la afirmación de (Mood 1963):

> *"La Estadística es la tecnología del método científico".*

Esta extensión de la Ciencia Estadística como ciencia auxiliar de otras ciencias, junto con su crecimiento y madurez metodológica, ha permitido el nacimiento de áreas con un cuerpo de conocimiento específico que pueden ser consideradas, a su vez, como entidades con la categoría de ciencia, como pueden ser la Psicometría, la Estadística Económica y la Econometría[1]. Así, a continuación, nos centraremos en la Estadística Económica, rama que ha ocupado un papel primordial en el desarrollo de la propia Ciencia Estadística desde el principio de sus orígenes.

[1] En nuestra opinión, no existe una delimitación clara entre *Estadística Económica* y *Econometría*, siendo la diferencia en todo caso un matiz dependiente de las técnicas y el enfoque empleado al enfrentarse a un determinado estudio. Quizá ambas disciplinas pudieran englobarse en otra disciplina más general que podría ser llamada 'Economía Cuantitativa'. Véase en relación con este respecto (Hernández-Alonso 2000).

1.1.3 Economía y Estadística.

La aplicación del método estadístico a la Economía puede entenderse como el proceso de representación de los sistemas económicos, constituidos por los distintos agentes que operan en las economías, y las relaciones que los ligan. La Economía suele especificar dichas relaciones dándoles forma de teorías económicas. No obstante, las teorías económicas con frecuencia son demasiado imprecisas a la hora de plantear modelos económicos verificables. Como Paul Samuelson apunta ((Samuelson 2006)):

> "*Solo en una muy pequeña parte de las obras de Economía teóricas o aplicadas se ha tratado la derivación de los teoremas significativos operacionalmente. En parte, por lo menos, tal situación se debe a los malos preconceptos metodológicos, según los cuales, las leyes económicas deducidas de los supuestos a priori poseen rigor y validez, independientemente de cualquier conducta humana real… De hecho, las obras de economía rebosan de malas generalizaciones.*"

La aplicación de los instrumentos estadísticos, y en concreto del Método Estadístico, permite dotar a la Teoría Económica del grado de concreción necesario para verificar en los sistemas reales el cumplimiento y la validez de dichas teorías. Este proceso de representación de sistemas reales puede llegar a tal grado de especificación que se puedan cuantificar las consecuencias en los cambios provocados en los elementos y relaciones del sistema ((Intriligator 1996), capítulo II). Sin embargo, por muy alto que sea el nivel de especificación del modelo que representa la realidad económica, este deberá llevar implícito cierta carga de abstracción de la realidad a la que representa, para poder ser abarcable. La realidad económica, el sistema económico, supera necesariamente en complejidad a cualquier modelo propuesto por la Teoría Económica, ya que el sistema económico depende, en última instancia, de fenómenos inmersos en cierto grado de incertidumbre; lo que es atribuible, a su vez, a su vinculación con el comportamiento humano. De este hecho se deduce la necesidad de incluir en la modelización de la realidad económica elementos estocásticos, lo que origina una visión no determinista, sino probabilista de la realidad económica.

Como señala (Martín-Pliego 2004), parte del conjunto de técnicas estadísticas aplicadas a la investigación económica es común a otras ciencias, mientras que otra parte es específica de este tipo de investigación, fruto

de una evolución de la aplicación de la disciplina en el tratamiento de los temas económicos. Entre estas metodologías específicas se encuentran el estudio de las series temporales económicas, de la distribución de la renta, la construcción y análisis de números índices, la modelización regional, el análisis input-output e intersectorial, las técnicas demográficas e incluso, en nuestra opinión, la propia Econometría.

1.2 ¿Qué es R y cómo nos ayuda a analizar datos desde el punto de vista estadístico?

En los apartados anteriores hemos partido del concepto de Estadística como ciencia instrumental hasta llegar a la Estadística Económica, como aquel cuerpo de la Ciencia Económica que se sirve de las herramientas que ofrece la Estadística para profundizar en el conocimiento de la realidad económica.

Pero claro, lo interesante de esto es llevarlo a la práctica. Se necesita un *soporte de hardware y software* para poder aplicar las técnicas estadísticas a los datos económicos, con el objetivo de crear conocimiento a partir de dichos datos. Este conocimeinto se traducirá en una reducción de la incertidumbre que inevitablemente viene aparejada a los fenómenos económicos, lo que redundará en una mejor toma de decisiones.

En los últimos tiempos se ha producido una evolución de hardware sin precedentes, lo que ha dado soporte al desarrollo de un potente software dedicado al análisis de datos (todo tipo de datos, no solamente económicos). Este software permite a cualquier investigador aplicar las últimas técnicas de análisis estadístico a cualquier masa de datos, lo que ha supuesto una verdadera revolución. A su vez, esta realidad se ha retroalimentado, de modo que se ha producido un constante avance en el desarrollo de técnicas y tecnologías de análisis de datos cada vez más complejas. Así, podemos hablar de técnicas de aprendizaje automático o *machine learning* (supervisado, no-supervisado o reforzado) o, más recientemente, de modelos de análisis basados en la *inteligencia artificial*.

En este caldo de cultivo, en el que se dispone de grandes masas de datos, de hardware capaz de procesarlas, y de técnicas capaces de extraer información de las mismas, se ha desarrollado un software cada vez más potente que une todos estos elementos para modelizar la realidad. Este software se concreta en aplicaciones y plataformas diversas: SPSS, Stata, SAS… Y también lenguajes de programación orientados al análisis estadístico y matemático, como pueden ser Python, Matlab, Julia o… R.

Sí. R no es solo una aplicación al uso. Es todo un lenguaje de programación, orientado principalmente a la analítica de datos, sobre todo desde una perspectiva estadística. R es un proyecto de GNU, por lo que los usuarios son libres de modificarlo y extenderlo. R se distribuye como software libre bajo la licencia GNU y es multiplataforma, lo que ha facilitado su difusión y la existencia de una comunidad muy activa de ususarios y desarrolladores.

1.3 Instalación de R y R-Studio.

Como ya se ha mencionado, R es un software o lenguaje de uso y difusión gratuitos, bajo licencia GNU. El modo de instalar R es sencillo: basta con ir a la web *CRAN* (*Comprehensive R Archive Network*) y descargar la última versión disponible en el sistema operativo del que se sea usuario (en este manual, Microsoft® Windows®). Se ejecutará el archivo descargado, y se completará la instalación.

Una limitación de R es la interfaz o IDE (entorno de desarrollo integrado) que incorpora. Es decir, el "software" con el que se interactúa con el lenguaje R. Esta IDE es muy poco amigable. Para superar esta limitación, existen IDEs alternativas, entre las que destaca RStudio, desarrollada por Posit® Software. Esta IDE es gratuita. De nuevo, simplemente tendremos que ir a la web deRStudio y descargar e instalar la versión gratuita.

1.4 R y RStudio. Comienzo: Proyectos.

Tras instalar R y su IDE RStudio, podremos comenzar a trabajar. Para ello, abriremos RStudio pulsando en el icono correspondiente. Aparecerá la siguiente ventana:

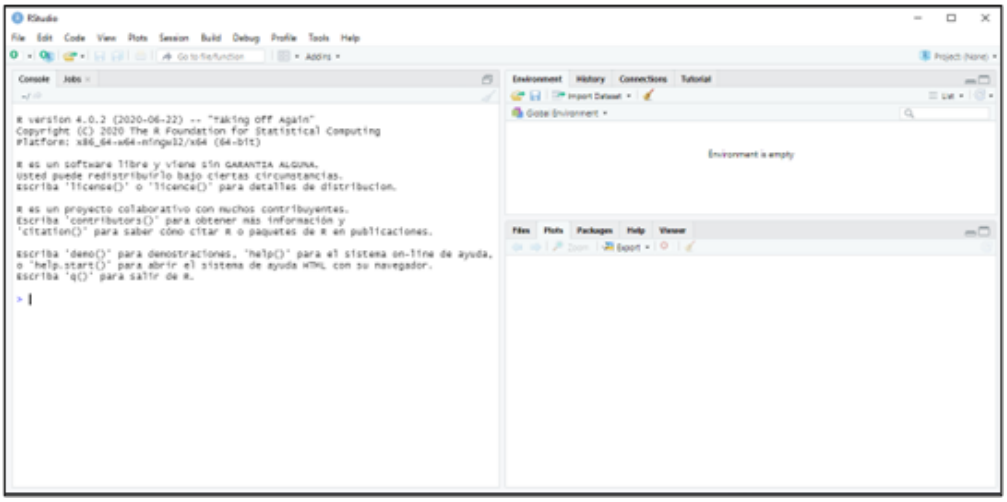


Figura 1.1: IDE de RStudio.

La parte izquierda de la ventana es la **consola**. La consola es la sección de RStudio donde podemos manejar R mediante la introducción de código. Por ejemplo, podemos escribir `2+2` después del cursor (signo ">"), y pulsar Enter. La propia consola nos devolverá el valor 4:

```
2+2
```

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

De todos modos, la forma más eficiente de trabajar es mediante "proyectos" y "scripts".

Un **proyecto** básicamente viene asociado a la carpeta donde R trabajará, buscando los datos que sean sus "inputs", y, en su caso, enviando sus resultados u "outputs". Dicho de otro modo, es una carpeta más de nuestro sistema de carpetas o directorios; pero a la que dotamos de una característica especial: ser un proyecto de R. Si abrimos desde RStudio el proyecto, estaremos diciendo a R que, por defecto, preferentemente busque todos los archivos e inputs (datos, etc.) que necesite en esa carpeta de proyecto; y que, en su caso, guarde en tal carpeta los outputs que genere.

Para crear un nuevo proyecto, seguiremos la instrucción `File → New Project`, luego se nos preguntará si se crea el proyecto en una nueva carpeta o en una ya existente. Vamos a crearlo, por ejemplo, en el disco extraíble D, carpeta R, subcarpeta "explora", que ya está creada. Nos saldrá una ventana para buscar la carpeta y, cuando la encontremos, pulsaremos `Open` y `Create Project`. Ya tendremos creado nuestro proyecto. Si nos vamos al explorador de Windows®, y buscamos la carpeta "explora", encontraremos que en tal carpeta aparece un archivo de nombre "explora", con un icono de un cubo con una "R". Ese archivo lo que está haciendo es actuar como un "faro" que le dice a R que, cuando trabajemos en el proyecto "explora", todos los archivos de datos necesarios estarán en esa carpeta (también llamada "explora", porque el proyecto adopta el nombre de la carpeta donde lo localizamos). Y que, si nuestro trabajo aporta algún fichero de "output", también se depositará en esa carpeta del proyecto.

En futuras sesiones, si queremos trabajar en el mismo proyecto, en lugar de seguir la ruta `File → New Project`, tendremos que hacer `File → Open Project`.

1.5 Scripts.

En cuanto a los **scripts**, son programas o rutinas donde varias instrucciones se ejecutan secuencialmente. Para crear un script, se seguirá la ruta `File → New File → R Script`. Y si el script lo guardamos, ¿dónde lo hará? Pues en la carpeta "explora", que es la del proyecto en el que estamos trabajando.

Informáticamente, un script es simplemente un archivo de texto plano. Se puede modificar con cualquier editor de texto. Afortunadamente, para no estar entrando y saliendo de R-Studio, esta interfaz incorpora un **editor** de scripts, lo cual es muy cómodo.

Vemos cómo ahora, a la izquierda de RStudio, ha aparecido, en la parte superior, una nueva ventana, pasando la consola a ocupar la parte inferior. Es la ventana del "editor":

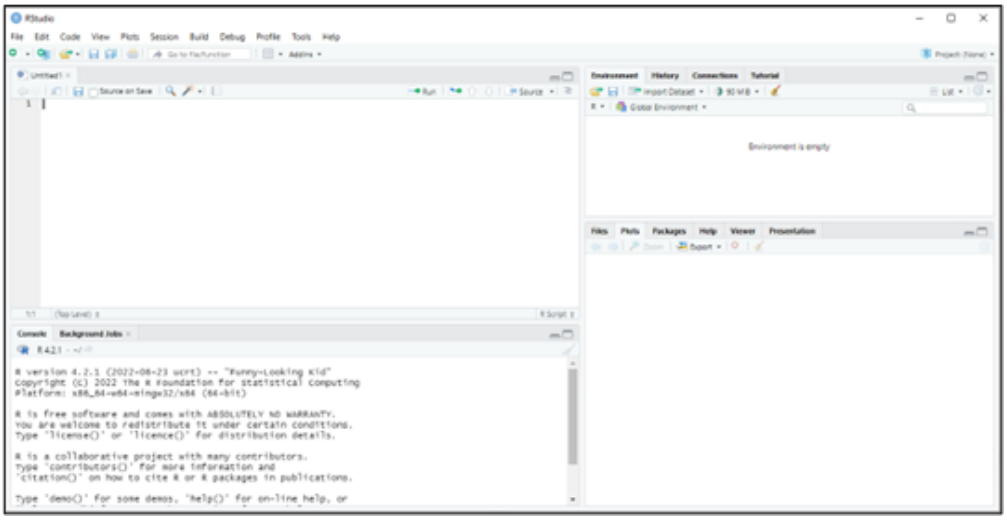


Figura 1.2: El editor de Scripts de RStudio.

Igual que con los proyectos, podemos crear desde RStudio un script nuevo, o abrir uno preexistente; y modificarlo, ejecutarlo, o volverlo a guardar.

Vamos a comenzar a escribir nuestro script. Si queremos hacer un comentario que no ejecute ninguna instrucción, éste irá precedido del símbolo almohadilla o *hashtag* "#". Luego, vamos a ordenar a R que haga la operación de suma: 2+2. Escribimos, por tanto, en el editor:

```
#Ejemplo de Script
2+2  #este script hace una simple suma.
```

Si pulsamos `Control + Mayúsculas + ENTER` o al desplegable de `Source → Source with Echo`, se ejecutará el script (para ejecutar solo la línea donde está el cursor, pulsaremos `Control + ENTER` o el botón de `Run`; y para ejecutar varias líneas, hemos de sombrearlas y pulsar `Control + ENTER` o el botón de `Run`). En la consola aparecerá:

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

Podemos guardar el script con `File → Save As…` ¿Dónde se guardará por defecto? Pues en la carpeta "explora", que es la de nuestro proyecto. Una vez nuestro script ya tiene nombre, podemos ir guardándolo de vez en cuando pulsando simplemente en el botón del "disquete" del editor. Vamos a llamarlo, por ejemplo, "explorando". Si vamos, en el explorador de Windows®, a nuestra carpeta de proyecto, veremos que hay un archivo de texto llamado "explorando" con extensión ".R" (explorando.R). Este script lo podremos ejecutar

cuantas veces queramos sin tener que escribir nada, o reescribirlo si vemos que no funciona o que necesitamos hacer modificaciones. Esa es la ventaja de trabajar con scripts.

Para recuperar un script en una nueva sesión de trabajo simplemente tenemos que seguir las instrucciones `File → Open File…` y seleccionarlo.

1.6 Funciones.

R trabaja con datos y funciones, principalmente. Pero, ¿qué es una **función**?

Una función es un conjunto o **sistema de instrucciones** que convierten unos datos de entrada o **inputs** en otros datos de salida, resultados, u **outputs**. Una función puede ser muy sencilla o ser verdaderamente compleja. Por otro lado, no todas las funciones están integradas en "paquetes"; sino que el usuario puede crear sus propias funciones (por ejemplo, escribiéndolas en un script) y ejecutarlas.

Las partes básicas de una función son:

- **Entradas, inputs o argumentos:** son las diversas informaciones necesarias para realizar el procedimiento de la función. Los argumentos pueden ser introducidos por el usuario, o pueden venir dados por defecto, lo que quiere decir que, si el usuario no dota de valor a un argumento, este tomará automáticamente un valor prestablecido.
- **Cuerpo:** está formado por un conjunto de instrucciones que transforman los *inputs* o entradas en los *outputs* o salidas. Si el cuerpo de la función está formado por varias instrucciones, éstas deben escribirse entre llaves `{ }`.
- **Salidas:** son los resultados u *output* de la función. Si una función ofrece como salida varios tipos de *objetos,* estos objetos suelen ser almacenados en una estructura de almacenaje de *lista.*

Como ejemplo, vamos a integrar en nuestro script una función, llamada "suma". Esta función requerirá de dos entradas o argumentos (dos números cualesquiera), y ofrecerá, como resultado, salida u output; la suma de tales entradas. El código es:

```
suma <- function(x, y) {
  resultado <- x + y
  return(resultado)
}
```

Ahora, una vez ejecutado el código anterior; si queremos sumar, por ejemplo, los números 12 y 16, solo tendremos que teclear en la consola, o escribir en el script y hacer run, a la línea:

```
suma(x=12, y=16)
```

```
## [1] 28
```

1.7 Paquetes (packages).

R es un lenguaje de programación en torno al cual se ha desarrollado una cantidad casi inimaginable de recursos: funciones, bases de datos, utilidades… Tal es la cantidad de recursos, que no sería operativo abrir R (directamente, o a través de una IDE, como RStudio) y tener inmediatamente todos esos recursos activos y preparados para ser utilizados. Además, R debería ser actualizado de un modo casi constante.

Por todo ello, todos los recursos disponibles están organizados mediante "paquetes" ("packages" en inglés). Un **paquete** es una colección de funciones y/o un conjunto de datos desarrollados por la comunidad de R. Estos incrementan el potencial de R ampliando sus capacidades básicas, o añadiendo otras nuevas.

De hecho, cuando abrimos R, algunos de estos paquetes, que se han instalado junto al propio lenguaje, se activan. Pero solo algunos. Un ejemplo es el paquete `{base}` o el paquete `{stats}` (R Core Team 2024).

La mayor parte de los paquetes disponibles no forman parte, por "defecto", en la misma instalación de R. Se encuentran en diversos servidores llamados repositorios. El más importante, es CRAN, que es el "repositorio oficial" y que alberga más de 10.000 paquetes. Pero existen otros repositorios, a destacar, por ejemplo, GitHub.

Para **instalar** un paquete en nuestra máquina que esté albergado en CRAN, un modo sencillo es, dentro de R-Studio, pulsar en la ventana inferior / izquierda sobre la pestaña "Packages", y sobre el botón "Install". Emergerá entonces una ventana donde hay un campo para escribir el nombre del paquete (al comenzar a escribirlo, el propio R-Studio te sugerirá los paquetes disponibles). Esto equivale a usar (bien directamente en la consola, o bien como línea de código insertada en un script) la instrucción `install.packages()`, con el nombre del paquete entre comillas (si son varios, pues irán separados por comas.

Una vez se tiene instalado el paquete, ya no habrá que volver a instalarlo para utilizarlo; sino **activarlo**. De hecho, todos los paquetes que no se encuentran por defecto en la propia instalación de R, deben ser activados para poder usar sus funcionalidades y/o datos. Para hacerlo, se debe utilizar la instrucción `library()`, y el nombre del paquete dentro del paréntesis.

Del nombre de esta instrucción surge la confusión común de tomar como sinónimos las palabras "paquete" y "librería" en el entorno de R. Si nos referimos a estas colecciones de funcionalidades y/o datos; lo correcto es "paquete", ya que "librería" tiene más que ver con la organización informática de un software.

1.8 Help! (sistema de ayuda).

A veces podemos albergar dudas sobre la correcta utilización de las funcionalidades y herramientas que nos proporciona un paquete. Hay varias fuentes de ayuda para intentar encontrar respuesta a las cuestiones que se nos plantean.

Una opción, para obtener información general sobre un paquete, es utilizar la función `help()`, con el argumento "package". Por ejemplo:

```
help(package="base")
```

Observaremos como en la ventana inferior / izquierda de R-Studio nos saldrá la información correspondiente. De hecho, en tal ventana existe una pestaña **"Help"** para obtener la ayuda sin teclear código.

Además, cada función puede ser consultada individualmente mediante `help("nombre de la función")` o `help(function, package = "package")` si el paquete no ha sido cargado. Estas instrucciones nos mostrarán la descripción de la función y sus argumentos acompañados de ejemplos de utilización. Por ejemplo:

```
help("rm", package="base")
```

La instrucción anterior nos aporta la documentación sobre la función `rm()` del paquete `{base}` de R (nota: este paquete se activa por defecto al abrir R o R-Studio; por lo que el segundo argumento, con el nombre del paquete que contiene la instrucción no es necesario).

Otra opción para mostrar información de ayuda es la exploración de las "viñetas" (vignettes). Las **viñetas** son documentos que muestran de un modo más detallado las funcionalidades de un paquete. La información de las viñetas de un paquete están disponibles en el archivo "documentation". Puede obtenerse una lista de las viñetas de nuestros paquetes instalados con la función `browseVignettes()`. Si solo queremos consultar las viñetas de un paquete concreto pasaremos como argumento a la función el nombre del mismo:

`browseVignettes(package = "packagename")`. En ambos casos, una ventana del navegador se abrirá para que podamos fácilmente explorar el documento.

Si optamos por permanecer en la consola, la instrucción `vignette()` nos mostrará una lista de viñetas, `vignette(package = "packagename")` las viñetas incluidas en el paquete, y una vez identificada la viñeta de interés podremos consultarla mediante `vignette("vignettename")`.

Capítulo 2

Almacenando y manipulando datos.

2.1 Objetos. Datos.

Como vimos en el capítulo 1, tras ejecutar un sencillo script (o al escribir instrucciones directamente desde la consola), R es interactivo: responde a las entradas que recibe. Las entradas o **expresiones** pueden ser, básicamente:

- Expresiones aritméticas.
- Expresiones lógicas.
- Llamadas a funciones.
- Asignaciones.

Las expresiones realizan acciones sobre **objetos** de R. Los objetos en R son entes que tienen ciertas características, *metadatos*, llamados atributos. No todos los objetos tienen los mismos atributos y, ni tan siquiera, todos los objetos tienen atributos que los caractericen.

Los objetos más importantes en R son ciertas estructuras o *contenedores* diseñados para almacenar elementos:

- Vectores.
- Matrices.
- Listas.
- Data frames.
- Factores.

Los elementos almacenados en los objetos se dividen en *clases*. Entre las diferentes clases, destacan las clases referidas a **datos**, que pueden ser de diferentes *modos*: logical (verdadero/falso), numeric (números) o character (cadena de texto). El modo numeric puede ser, a la vez, de tipo integer (número entero) o double (número real). En el caso de logical y carácter, modo y tipo coinciden.

Vamos a profundizar un poco en algunas de estos contenedores de datos. Vamos a suponer que trabajamos en el proyecto que creamos en el capítulo anterior (proyecto "explora"), y que vamos a editar el script que también creamos en tal capítulo (script "explorando.R", que se encontrará ubicado en la carpeta del proyecto "explora").

2.1.1 Vectores.

Los **vectores**, son conjuntos de elementos **de la misma clase**. Vamos a definir por ejemplo el vector x = (1,3,5,8). Para ello, vamos a escribir en nuestro script:

```
x <- c(1,3,5,8)
```

Ejecutamos la línea (situando el cursor en algún lugar de ella, dentro del script; y pulsando a la vez las teclas `Control + Enter` o pinchando con el ratón en el botón `Run` del editor). Ya tenemos nuestro primer objeto de tipo *vector* en memoria. Por cierto, lo que hemos hecho es una **asignación**, que se escribe con una flecha creada mediante los signos "<" y "-". Hemos asignado a un vector llamado "x" los elementos 1, 3, 5 y 8.

Para ver el vector simplemente escribimos en la consola (o en el script) el nombre del vector, "x". El resultado será:

```
## [1] 1 3 5 8
```

Además, si miramos en la ventana superior-derecha de R-Studio, veremos que en el ***Global Environment*** se muestra nuestro vector y que, además, se nos informa de que tiene *modo numérico*. El *Global Environment* nos informa de los objetos que R tiene en memoria:

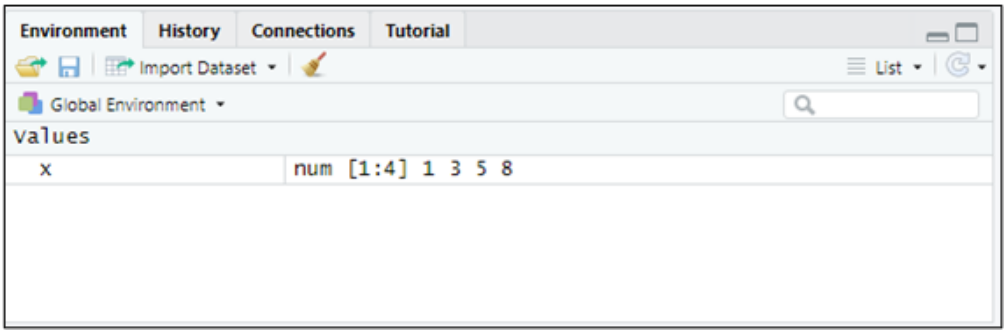


Figura 2.1: Nuestro vector en memoria.

Si queremos obtener un vector de números consecutivos del 2 al 6, basta con ejecutar en la "consola" (o escribir y ejecutar en el script):

```
y <- c(2:6)
```

Al escribir el nombre del vector "y" en la "consola" obtendremos:

```
y
```

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

Si queremos saber la longitud de un vector, usaremos la *función* `length()`. Por ejemplo, `length(y)` nos devolverá el valor 5. Escribamos en el script y ejecutemos:

```
length(y)
```

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

Un vector puede incluir, además de números, caracteres o grupos de caracteres alfanuméricos; siempre entrecomillados (lo fundamental es que sean elementos de la misma clase). Por ejemplo, el vector "genero" (¡no pongamos tildes o podemos tener problemas!). Así, si ejecutamos estas dos líneas de código:

```
genero<-c("Mujer","Hombre")
genero
```

Se habrá creado el vector "genero":

```
## [1] "Mujer"  "Hombre"
```

Podemos obtener la clase de los elementos almacenados en nuestro vector con la función `class()`:

```
class(genero)
```

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

Si falta un dato en un vector, habrá que escribir "NA" (not available). Por ejemplo, si falta el tercer dato de este vector "z", este vector se escribirá como:

```
Z <- c(1,2,NA,2,8)
```

Para **seleccionar un elemento** concreto de un vector, indicaremos entre corchetes la posición en la que se encuentra. Por ejemplo, refiriéndonos al vector "x", para obtener el valor de su tercer elemento, haremos:

```
x[3]
```

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

Si queremos que se nos muestre los elementos del vector x del 2º al 4º:

```
x[2:4]
```

```
## [1] 3 5 8
```

Por último, si queremos sacar en pantalla los elementos 1º y 4º, tendremos que incluir una "c" seguida de un paréntesis que recoja el orden de los elementos que queremos seleccionar:

```
x[c(1,4)]
```

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

2.1.2 Matrices.

Las **matrices**, internamente en R, son vectores; pero con dos atributos adicionales: número de filas y número de columnas. Se definen mediante la función matrix(). Por ejemplo, para definir la matriz "a":

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

Tendremos que escribir:

```
a <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3)
a
```

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

El número de filas de la matriz (y por tanto, el número de columnas) se fija con el argumento `nrow =` . También podríamos fijar el número de columnas, con `ncol =` .

Como vemos, por defecto, R va "cortando" el vector por columnas (si lo preferimos, lo puede hacer también por filas, añadiendo a la función `matrix()` el argumento `by row = true`; pero, en nuestro ejemplo, obtendríamos la matriz traspuesta a la que queremos almacenar).

Las dimensiones (número de filas y de columnas) de la matriz pueden obtenerse mediante la función `dim()`:

```
dim(a)
```

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

3 filas y 3 columnas.

Si queremos seleccionar elementos concretos de una matriz, lo haremos utilizando corchetes para indicar filas y columnas. Hemos de tener en cuenta que, trabajando con matrices, siempre tenemos

$$rangodefilas, rangodecolumnas$$

Si se deja en blanco el espacio entre el corchete inicial y la coma, esto querrá decir que consideramos todas las filas. Y si no insertamos nada entre la coma y el corchete de cierre, esto significará que consideramos todas las columnas. A continuación tenemos varios ejemplos de código, con el resultado obtenido en la consola:

```
a[2,3]
```

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

```
a[1:2,2:3]
```

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    5    8
```

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

```
##      [,1] [,2]
## [1,]    1    7
## [2,]    2    8
## [3,]    3    9
```

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

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    3    6    9
```

Tanto para vectores como para matrices, funcionan las operaciones suma y diferencia sin más complicaciones. En el caso del producto, sin embargo, hay que tener en cuenta que, por ejemplo, a*a devuelve la multiplicación elemento a elemento, es decir:

```
a*a
```

Devuelve la multiplicación elemento a elemento (en este caso, el cuadrado de cada número, al multiplicar la matriz a por sí misma):

```
##      [,1] [,2] [,3]
## [1,]    1   16   49
## [2,]    4   25   64
## [3,]    9   36   81
```

Para hacer el verdadero producto matricial deberá introducirse:

```
a%*%a
```

```
##      [,1] [,2] [,3]
## [1,]   30   66  102
## [2,]   36   81  126
## [3,]   42   96  150
```

2.1.3 Data frames.

Un ***data frame*** es un objeto que almacena datos organizados mediante la clase `data.frame`. Esta organización consiste en que, por filas, se disponen los diferentes casos o sujetos; mientras que por columnas se posicionan las variables. Así:

- Es similar a una matriz en el sentido de que tiene dos dimensiones. Podemos acceder a sus elementos con corchetes, tenemos nombres de filas y columnas, y podemos operar con ellas.
- Cada columna tiene un nombre, de manera que podemos acceder a una columna concreta con el símbolo `$`. Todas las columnas (variables) son vectores con la misma longitud.
- Cada columna puede ser un vector numérico, factor, de tipo carácter o lógico.

Por ejemplo, vamos a crear el *data frame* **"datos"**, con tres variables: "peso", "altura", y "color de ojos", llamadas "Peso", "Altura" y "Cl.ojos", respectivamente; para 3 individuos o casos. Una opción es crear primero las tres variables como vectores, y luego crear el *data frame* mediante la función `dataframe()`:

```
Peso<-c(68,75,88)
Altura<-c(1.6,1.8,1.9)
Cl.ojos<-c("azules","marrones","marrones")
datos<-data.frame(Peso,Altura,Cl.ojos)
```

Si ahora ejecutamos una línea con el nombre de nuestro *data frame*, lo obtendremos como resultado en la consola:

```
datos
```

```
##   Peso Altura  Cl.ojos
## 1   68    1.6   azules
## 2   75    1.8 marrones
## 3   88    1.9 marrones
```

Para obtener los nombres de las variables (es decir, el nombre de cada columna) teclearemos la función:

```
names(datos)
```

Obteniéndose:

```
## [1] "Peso"    "Altura"  "Cl.ojos"
```

Para obtener solo los datos de la columna (variable) color de ojos teclearemos datos$Cl.ojos:

```
datos$Cl.ojos
```

```
## [1] "azules"   "marrones" "marrones"
```

Y para obtener los datos de peso: datos$Peso:

```
datos$Peso
```

```
## [1] 68 75 88
```

Para saber el número de filas y de columnas de una hoja de datos utilizaremos las funciones `nrow()` y `ncol()`:

```
nrow(datos)
```

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

```
ncol(datos)
```

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

Para seleccionar elementos de un data frame, se pueden seguir las mismas reglas que para la selección de elementos de una matriz (con el número de cada fila, que es cada individuo; y el número de cada columna, que es cada variable. Para elegir una variable, no obstante, ya hemos visto que es posible usar su nombre; aunque precedido del nombre del data frame y el signo `$`. Por ejemplo, si ejecutamos:

```
datos[,2]
```

```
## [1] 1.6 1.8 1.9
```

```
datos$Altura
```

```
## [1] 1.6 1.8 1.9
```

Obtenemos el mismo resultado.

2.2 Importando datos.

Lo más frecuente es que no tecleemos los datos, como hemos hecho hasta ahora; sino que los importemos a R desde algún contenedor externo (archivo de texto, hoja de cálculo, base de datos…). Nosotros vamos a importar nuestros datos desde Microsoft® Excel®. Vamos a cerrar el script que hemos estado construyendo en los apartados anteriores (para conservarlo hay que guardarlo antes), aunque vamos a seguir trabajando en el mismo proyecto (que habíamos llamado "explora"). Iremos a la carpeta del proyecto y guardaremos en ella los dos archivos de esta práctica (obtén el enlace a los archivos en la sección final del capítulo):

- Un archivo de Microsoft® Excel® llamado "eolica_20.xlsx"
- Un script con las instrucciones que vamos a mostrar a continuación, y que se llama "explora_eolica.R"

Si abrimos el archivo de Microsoft® Excel® comprobaremos que se compone de tres hojas. La primera muestra el criterio de búsqueda de casos en la un aviso sobre el uso exclusivo que se debe dar a los datos incorporados; la segunda recoge la descripción de las variables consideradas; y la tercera (hoja "Top 20") guarda los datos que debemos **importar** desde R-Studio. Estos datos se corresponden con diferentes variables económico-financieras de 20 empresas productoras de electricidad mediante generación eólica. Luego vamos a cerrar el archivo de Microsoft® Excel® y volveremos a R-Studio. Vamos a abrir nuestro script "explora_eolica.R" con `File → Open File…` Este script contiene el programa que vamos a ir ejecutando en la práctica. La primera línea / instrucción en los scripts suele ser:

```
rm(list = ls())
```

La instrucción tiene como objeto limpiar el *Environment* (memoria) de objetos de anteriores sesiones de trabajo. Para importar los datos localizados en el archivo de Excel "eolica_20.xlsx" el código es:

```
# DATOS
library(readxl)
eolica_20 <- read_excel("eolica_20.xlsx", sheet = "Top 20")
```

¡Atención! Si nunca se ha utilizado el paquete `{readxl}` (que contiene el código necesario para importar datos de un archivo de Microsoft® Excel®), cuando la intentemos activar con la función `library()` nos dará un error o nos dirá que previamente hay que importarla. En ese caso, iremos a la ventana inferior-derecha y pulsaremos la pestaña `Packages`, pulsaremos en `Install`, y emergerá una ventana donde dejaremos el "repositorio" que viene por defecto y, en el campo "Packages", escribiremos el nombre del "paquete" que contiene la librería que nos hace falta (normalmente coincide con el nombre de la propia librería, en nuestro caso `{readxl}`. Una vez descargado el "paquete", podremos ejecutar el código anterior sin problemas.

Otra cuestión importante a tener en cuenta es que, en la hoja de cálculo del ejemplo, los "valores perdidos" o *missing values* (celdas en las que no hay datos), venían en blanco. Pero, en ocasiones, pueden contener algún tipo de anotación, como por ejemplo, "n.d." (no disponible). En tal caso, deberá incluirse un argumento más que informe de estas celdas que, sin estar en blanco, no tienen dato:

```
eolica_20 <- read_excel("eolica_20.xlsx", sheet = "Top 20", na = c("n.d."))
```

Volviendo a nuestro ejemplo, podemos observar cómo en el *Environment* ya aparece un objeto. Este objeto es una estructura de datos tipo *data frame*, se llama "eolica_20" y contiene 11 columnas, una por cada una de las variables almacenadas en el archivo de Microsoft® Excel®. De estas variables, tres son de tipo cualitativo, formadas por cadenas de caracteres: el nombre de la empresa, "NOMBRE"; y el nombre del grupo empresarial matriz al que pertenece, "MATRIZ". Puede explorarse el contenido del *data frame* y los principales estadísticos con la función `summary()`:

```
summary (eolica_20)
```

```
##     NOMBRE               RES              ACTIVO             FPIOS
##  Length:20          Min.   : -5662   Min.   :  109024   Min.   : -77533
##  Class :character   1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615
##  Mode  :character   Median :  7388   Median :  271636   Median :  77740
##                     Mean   : 50754   Mean   : 1183599   Mean   : 563678
##                     3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345
##                     Max.   :727548   Max.   :13492812   Max.   :6904824
##                     NA's   :1
##
##      RENECO             RENFIN            LIQUIDEZ          MARGEN
##  Min.   :-2.8130   Min.   :-359.773   Min.   :0.0780   Min.   :-302.03
##  1st Qu.: 0.8765   1st Qu.:   1.664   1st Qu.:0.7342   1st Qu.:  12.39
##  Median : 3.6150   Median :  10.812   Median :1.2345   Median :  21.42
##  Mean   : 2.9399   Mean   :  -3.450   Mean   :1.4200   Mean   :  16.40
##  3rd Qu.: 4.7735   3rd Qu.:  25.312   3rd Qu.:1.5615   3rd Qu.:  38.56
##  Max.   : 8.5860   Max.   :  52.261   Max.   :5.3300   Max.   : 208.36
##  NA's   :1
##
##    SOLVENCIA          APALANCA           MATRIZ
##  Min.   :-40.74   Min.   :-6265.50   Length:20
##  1st Qu.: 11.26   1st Qu.:   16.13   Class :character
##  Median : 23.68   Median :  145.93   Mode  :character
##  Mean   : 32.68   Mean   :  -17.17
##  3rd Qu.: 52.62   3rd Qu.:  504.74
##  Max.   : 99.08   Max.   : 1019.62
```

Veremos cómo aparecen 11 variables con algunos estadísticos básicos.

R ha considerado la primera columna como una variable de tipo cualitativo (atributo). En realidad no es una variable, sino el nombre de los individuos o casos. Para evitar que R tome los nombres de los casos como una variable, podemos redefinir nuestro *data frame* diciéndole que considere esa primera columna como los *nombres de los individuos o filas*:

```
eolica_20 <- data.frame(eolica_20, row.names = 1)
```

En la línea anterior hemos asignado al data frame "eolica_20" los propios datos de "eolica_20"; pero indicando que la primera columna de datos no es una variable; sino el nombre de los casos. Si hacemos ahora el `summary()`:

```
##       RES              ACTIVO             FPIOS              RENECO
##  Min.   : -5662   Min.   :  109024   Min.   : -77533   Min.   :-2.8130
```

```
##  1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615   1st Qu.: 0.8765
##  Median :  7388   Median :  271636   Median :  77740   Median : 3.6150
##  Mean   : 50754   Mean   : 1183599   Mean   : 563678   Mean   : 2.9399
##  3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345   3rd Qu.: 4.7735
##  Max.   :727548   Max.   :13492812   Max.   :6904824   Max.   : 8.5860
##  NA's   :1                                             NA's   :1
##
##      RENFIN            LIQUIDEZ          MARGEN          SOLVENCIA
##  Min.   :-359.773   Min.   :0.0780   Min.   :-302.03   Min.   :-40.74
##  1st Qu.:   1.664   1st Qu.:0.7342   1st Qu.:  12.39   1st Qu.: 11.26
##  Median :  10.812   Median :1.2345   Median :  21.42   Median : 23.68
##  Mean   :  -3.450   Mean   :1.4200   Mean   :  16.40   Mean   : 32.68
##  3rd Qu.:  25.312   3rd Qu.:1.5615   3rd Qu.:  38.56   3rd Qu.: 52.62
##  Max.   :  52.261   Max.   :5.3300   Max.   : 208.36   Max.   : 99.08
##
##     APALANCA           MATRIZ
##  Min.   :-6265.50   Length:20
##  1st Qu.:   16.13   Class :character
##  Median :  145.93   Mode  :character
##  Mean   :  -17.17
##  3rd Qu.:  504.74
##  Max.   : 1019.62
```

Vemos que ya no aparece "NOMBRE" como variable, y en el *Environment* ya aparece el data frame "eolica_20" con 20 observaciones (casos), pero con 10 variables (una menos).

Antes de seguir con la manipulación de nuestros datos, es preciso decir que existen otros muchos formatos de datos que pueden ser importados. Por ejemplo, con el paquete `{readr}` se pueden importar datos de archivos de texto de tipo tabular (por ejemplo, archivos *.csv). Con el paquete `{haven}` se pueden capturar los datos almacenados en archivos de SPSS® (.sav), Stata® (.dta), SAS® (.sas7bdat), etc. Finamente, se pueden capturar datos almacenados en páginas web (archivos en formato JSON o XML, o en tablas HTML)) o en bases de datos gestionadas mediante diversos sistemas (SQLite, MySQL, MariaDB, PostgreSQL, Oracle®).

2.3 {dplyr}.

2.3.1 El Tidyverse. Cargando {dplyr}.

El ***Tidyverse*** es un conjunto de paquetes / librerías con una filosofía común, como es el uso de ciertas estructuras gramaticales, que facilitan muchas de las tareas y análisis que podrían hacerse con el lenguaje R estándar. Una buena obra para profundizar en el Tidyverse es Wickham and Grolemund (2017).

Uno de esos paquetes es `{dplyr}`, que proporciona una gramática más sencilla que la del lenguaje R convencional para **manipular** los objetos de estructuras de datos conocidos como ***data frames***.

Los data frames, como ya sabemos, son estructuras en las que se almacenan datos de modo que, por columnas, se disponen las variables del análisis; y por filas los casos que conforman la muestra / población.

Vamos a suponer que trabajamos dentro del proyecto que hemos creado previamente, de nombre "explora" (ver capítulo 1). Dentro de la carpeta del proyecto guardaremos el script llamado "explora_dplyr.R" y el archivo de Microsoft® Excel® llamado "eolica_20.xlsx". Si abrimos este último archivo, comprobaremos que se compone de tres hojas. La hoja "Top 20" contiene los datos a importar desde R-Studio. Estos datos se corresponden con diferentes variables económico-financieras de 20 empresas productoras de electricidad mediante generación eólica.

Luego cerraremos el archivo de Microsoft® Excel®, "eolica_20.xlsx", y volveremos a R-Studio. Después, abriremos nuestro script "explora_dplyr.R" con `File → Open File…` Este script contiene el programa que vamos a ir ejecutando en la práctica.

La primera línea / instrucción en los scripts suele ser:

```
rm(list = ls())
```

La instrucción tiene como objeto limpiar el *Environment* (memoria) de objetos de anteriores sesiones de trabajo.

Para importar los datos que hay en la hoja "Top 20" del archivo de Microsoft® Excel® llamado "eolica_20.xlsx", ejecutaremos el código:

```
# DATOS

library(readxl)
eolica_20 <- read_excel("eolica_20.xlsx", sheet = "Top 20")
```

Podemos observar como en el *Environment* ya aparece un objeto. Este objeto es una estructura de datos tipo *data frame*, se llama "eolica_20" y contiene 11 columnas. R ha considerado la primera columna como una variable de tipo cualitativo. En realidad, la primera columna no es una variable, sino que está formada por el nombre (identificador) de los diferentes casos u observaciones. Para evitar que R tome los nombres de los casos como una variable más, podemos redefinir nuestro *data frame* diciéndole que tome esa primera columna como los *nombres de los individuos*:

```
eolica_20 <- data.frame(eolica_20, row.names = 1)
```

En la línea anterior hemos asignado al *data frame* "eolica_20" los propios datos de "eolica_20"; pero indicando que la primera columna de datos no es una variable; sino el nombre de los casos.

A continuación, cargaremos el paquete `{dplyr}`. Si nunca antes se ha utilizado este paquete, cuando lo intentemos activar con la función `library()` nos dará un error o nos dirá que previamente hay que importarlo. En ese caso, iremos a la ventana inferior-derecha y pulsaremos la pestaña "Packages", pulsaremos en `Install`, y emergerá una ventana donde dejaremos el "repositorio" que viene por defecto y, en el campo `Packages`, escribiremos el nombre del "paquete" (en nuestro caso `{dplyr}`). Una vez descargado el "paquete", podremos ejecutar el código sin problemas:

```
#Cargando dplyr

library (dplyr)
```

Para entender mejor la **sintaxis** que siguen las funciones o instrucciones a las que da acceso `{dplyr}`, hay que tener en cuenta lo siguiente:

- El primer argumento que tiene una función de `{dplyr}` es el *data frame* con el que se va a trabajar.
- Los otros argumentos describen qué hay que hacer con el *data frame* especificado en el primer argumento. Es posible referirse a las columnas (variables) del *data frame* con su nombre, **sin utilizar el operador `$`**.
- El valor de retorno es un **nuevo** *data frame*.

En los siguientes subapartados practicaremos con algunas de las principales funciones que aporta `{dplyr}`.

2.3.2 Seleccionando columnas de un data frame.

La función clave de `{dplyr}` para seleccionar una o varias columnas (variables) de un *data frame* es la función `select()`.

Así, vamos a imaginar por ejemplo que queremos eliminar de nuestro *data frame* la variable (de tipo "carácter") MATRIZ. Podremos ejecutar la asignación:

```
#Seleccionando variables

eolica_20 <-select(eolica_20, -MATRIZ)
summary (eolica_20)
```

```
##       RES              ACTIVO             FPIOS
##  Min.   : -5662   Min.   :  109024   Min.   : -77533
##  1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615
##  Median :  7388   Median :  271636   Median :  77740
##  Mean   : 50754   Mean   : 1183599   Mean   : 563678
##  3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345
##  Max.   :727548   Max.   :13492812   Max.   :6904824
##  NA's   :1
##
##      RENECO            RENFIN            LIQUIDEZ
##  Min.   :-2.8130   Min.   :-359.773   Min.   :0.0780
##  1st Qu.: 0.8765   1st Qu.:   1.664   1st Qu.:0.7342
##  Median : 3.6150   Median :  10.812   Median :1.2345
##  Mean   : 2.9399   Mean   :  -3.450   Mean   :1.4200
##  3rd Qu.: 4.7735   3rd Qu.:  25.312   3rd Qu.:1.5615
##  Max.   : 8.5860   Max.   :  52.261   Max.   :5.3300
##  NA's   :1
##
##      MARGEN          SOLVENCIA         APALANCA
##  Min.   :-302.03   Min.   :-40.74   Min.   :-6265.50
##  1st Qu.:  12.39   1st Qu.: 11.26   1st Qu.:   16.13
##  Median :  21.42   Median : 23.68   Median :  145.93
##  Mean   :  16.40   Mean   : 32.68   Mean   :  -17.17
##  3rd Qu.:  38.56   3rd Qu.: 52.62   3rd Qu.:  504.74
##  Max.   : 208.36   Max.   : 99.08   Max.   : 1019.62
```

Podemos verificar que, en el *Environment,* el *data frame* ha pasado a tener una variable menos (9), ya que hemos eliminado la variable MATRIZ. Es decir, con el guión "-" se pueden eliminar directamente variables de un *data frame.*

Ahora, suponemos que queremos visualizar las variables del *data frame* "eolica_20": ACTIVO, FPIOS, LIQUIDEZ, MARGEN, SOLVENCIA y APALANCA (es decir, todas las variables menos RES, RENECO, RENFIN). Para ello, ejecutaremos el código:

```
select(eolica_20, ACTIVO, FPIOS, LIQUIDEZ, MARGEN, SOLVENCIA, APALANCA)
```

```
##                                  ACTIVO       FPIOS LIQUIDEZ   MARGEN
## Holding De Negocios De GAS SL.  13492812.0 6904824.000    1.020   91.152
## Global Power Generation SA.      2002458.0 1740487.000    2.006   22.403
## Naturgy Renovables SLU           1956869.0  318475.000    1.263   20.442
## EDP Renovables España SLU        1275939.0  726783.000    1.596   47.193
```

```
## Corporacion Acciona Eolica SL     864606.0  136064.000    0.788   20.091
## Saeta Yield SA.                   796886.4  665319.556    2.687   16.258
## Elawan Energy SL.                 443467.0  186302.006    0.595  208.357
## Olivento SL                       381207.0   58340.998    0.771   16.629
## Parque Eolico La Boga SL.         303904.4   29316.797    1.407    1.001
## Naturgy Wind, S.L.                273542.0   28418.000    1.364   39.575
## Viesgo Renovables SL.             269730.0  177707.000    0.272   11.818
## Al-Andalus Wind Power SL          249853.8   21466.121    1.550   12.582
## Innogy Spain SA.                  230338.5   85447.212    1.416  -18.025
## Guzman Energia SL                 190287.0  -77532.698    0.078  -19.193
## Acciona Eolica Del Levante SL     188354.0   21769.000    2.855   27.520
## Biovent Energia SA                183899.0   70033.000    1.206   22.792
## Esquilvent SL                     157630.6   48769.130    5.330   39.476
## Eolica La Janda SL                153429.4   25206.748    1.184   38.256
## Parque Eolico Santa Catalina SL   147742.5   -1664.755    0.388   31.780
## WPD Wind Investment SL.           109023.8  108023.826    0.624 -302.027
##
##                                 SOLVENCIA  APALANCA
## Holding De Negocios De GAS SL.     51.174    91.964
## Global Power Generation SA.        86.917     1.044
## Naturgy Renovables SLU             16.274   494.729
## EDP Renovables España SLU          56.960    67.028
## Corporacion Acciona Eolica SL      15.737   422.263
## Saeta Yield SA.                    83.489    17.067
## Elawan Energy SL.                  42.010   123.771
## Olivento SL                        15.304   534.761
## Parque Eolico La Boga SL.           9.646   921.591
## Naturgy Wind, S.L.                 10.388   824.537
## Viesgo Renovables SL.              65.883    13.330
## Al-Andalus Wind Power SL            8.591  1019.616
## Innogy Spain SA.                   37.096   150.688
## Guzman Energia SL                 -40.745  -343.542
## Acciona Eolica Del Levante SL      11.557   743.754
## Biovent Energia SA                 38.082   141.163
## Esquilvent SL                      30.938   218.275
## Eolica La Janda SL                 16.428   480.122
## Parque Eolico Santa Catalina SL    -1.126 -6265.496
## WPD Wind Investment SL.            99.082     0.000
```

Como no hemos asignado el resultado de la función a ningún "nombre", R simplemente saca el resultado en pantalla; pero no guarda ningún objeto en el *Environment*. Si asignamos un `select()` a un "nombre", se creará un *data frame* con ese nombre, y las variables seleccionadas:

```
eolica_20A <-select(eolica_20, ACTIVO, FPIOS, LIQUIDEZ, MARGEN, SOLVENCIA, APALANCA)
summary (eolica_20A)
```

```
##       RES              ACTIVO              FPIOS
##  Min.   : -5662   Min.   :  109024   Min.   : -77533
##  1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615
##  Median :  7388   Median :  271636   Median :  77740
##  Mean   : 50754   Mean   : 1183599   Mean   : 563678
##  3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345
##  Max.   :727548   Max.   :13492812   Max.   :6904824
##  NA's   :1
```

```
##
##      RENECO            RENFIN            LIQUIDEZ
##  Min.   :-2.8130   Min.   :-359.773   Min.   :0.0780
##  1st Qu.: 0.8765   1st Qu.:   1.664   1st Qu.:0.7342
##  Median : 3.6150   Median :  10.812   Median :1.2345
##  Mean   : 2.9399   Mean   :  -3.450   Mean   :1.4200
##  3rd Qu.: 4.7735   3rd Qu.:  25.312   3rd Qu.:1.5615
##  Max.   : 8.5860   Max.   :  52.261   Max.   :5.3300
##  NA's   :1
##
##      MARGEN          SOLVENCIA         APALANCA
##  Min.   :-302.03   Min.   :-40.74   Min.   :-6265.50
##  1st Qu.:  12.39   1st Qu.: 11.26   1st Qu.:   16.13
##  Median :  21.42   Median : 23.68   Median :  145.93
##  Mean   :  16.40   Mean   : 32.68   Mean   :  -17.17
##  3rd Qu.:  38.56   3rd Qu.: 52.62   3rd Qu.:  504.74
##  Max.   : 208.36   Max.   : 99.08   Max.   : 1019.62
```

Podemos comprobar en el *Environment* cómo hay otro objeto *data frame* llamado "eolica_20A", con 6 variables (y los mismos 20 casos). Este *data frame* lo podríamos haber creado, también, eliminando del *data frame* original ("eolica_20"), las variables que nos sobran:

```
eolica_20A <-select(eolica_20, -RES, -RENECO, -RENFIN)
```

Más aún, si nos fijamos bien, los nombres de todas las variables que hemos excluido empiezan por "RE", a diferencia de las incluidas. Podríamos haber hecho también:

```
eolica_20A <-select(eolica_20, -(starts_with("RE")))
```

Y de nuevo obtendríamos el mismo resultado. El argumento `starts_with()` permite seleccionar variables cuyos nombres comienzan por cierta cadena de caracteres. También se puede hacer mismo con los caracteres finales (`ends_with()`) o contenidos en alguna posición del nombre (`contains()`).

Otra posibilidad que tenemos es hacer una copia de un *data frame* rápidamente con el argumento `everything()`. Por ejemplo:

```
eolica_20_replica <-select(eolica_20, everything())
```

Se ha creado el *date frame* "eolica_20_replica" que es una copia exacta de "eolica_20".

2.3.3 Seleccionando casos de un *data frame.*

Además de seleccionar variables, con `{dplyr}` también se pueden seleccionar casos que cumplan ciertas condiciones. La función para realizar este cometido es `filter()`. Por ejemplo, si queremos seleccionar las empresas eólicas con un resultado (variable RES) mayor o igual a 50.000 y presentarlas en pantalla, la instrucción será:

```
filter(eolica_20, RES >= 50000)
```

```
##                                  RES   ACTIVO   FPIOS RENECO RENFIN
## Holding De Negocios De GAS SL. 727548 13492812 6904824  5.264 10.287
## EDP Renovables España SLU       67033  1275939  726783  6.458 11.338
```

```
##
##                                LIQUIDEZ MARGEN SOLVENCIA APALANCA
## Holding De Negocios De GAS SL.    1.020 91.152    51.174   91.964
## EDP Renovables España SLU         1.596 47.193    56.960   67.028
```

Se pueden incluir varias condiciones en un mismo filtro. Por ejemplo, vamos a construir un nuevo *data frame* llamado "eolica_20B" con las empresas que posean un resultado mayor o igual a 50000 y una rentabilidad económica (variable RENECO) inferior al 6%:

```
eolica_20B <-filter(eolica_20, RES >= 50000 & RENECO < 6)
eolica_20B
```

```
##                                   RES   ACTIVO   FPIOS
## Holding De Negocios De GAS SL. 727548 13492812 6904824
##
##                                RENECO RENFIN LIQUIDEZ
## Holding De Negocios De GAS SL.  5.264 10.287     1.02
##
##                                MARGEN SOLVENCIA APALANCA
## Holding De Negocios De GAS SL. 91.152    51.174   91.964
```

En el *Environment* aparecerá el *data frame* "eolica_9B" con solo un caso: la empresa que cumple con ambas condiciones, introducidas mediante el operador lógico relacional "&", que es el equivalente a la conjunción "y" o, dicho de otro modo, la intersección. Otro operador lógico relacional muy utilizado es la barra vertical "|", que es el equivalente a la conjunción "o", es decir, la unión.

Los filtros más usuales son >, <, >=, <=, == (igual, ojo, con dos símbolos de igualdad seguidos) y != (no igual).

2.3.4 Ordenando casos de un *data frame*.

Además de seleccionar determinados casos u observaciones (filas) de un *data frame*, con las funciones de `{dplyr}` también se pueden ordenar estos casos a partir de los valores de ciertas variables (columnas). La función a utilizar es `arrange()`. Esta función, por defecto, ordena los casos de modo **ascendente**. Por ejemplo:

```
arrange(eolica_20, RENECO)
```

```
##                                     RES     ACTIVO       FPIOS
## Guzman Energia SL                -5661.463   190287.0  -77532.698
## Innogy Spain SA.                 -5268.573   230338.5   85447.212
## WPD Wind Investment SL.           -850.068   109023.8  108023.826
## Parque Eolico La Boga SL.           11.940   303904.4   29316.797
## Saeta Yield SA.                   2084.476   796886.4  665319.556
## Global Power Generation SA.      39995.000  2002458.0 1740487.000
## Naturgy Renovables SLU           42737.000  1956869.0  318475.000
## Al-Andalus Wind Power SL          4403.214   249853.8   21466.121
## Olivento SL                       7388.175   381207.0   58340.998
## Elawan Energy SL.                12818.975   443467.0  186302.006
## Naturgy Wind, S.L.                8500.000   273542.0   28418.000
## Parque Eolico Santa Catalina SL   3645.278   147742.5   -1664.755
## Biovent Energia SA                      NA   183899.0   70033.000
```

```
## Corporacion Acciona Eolica SL    29592.000   864606.0  136064.000
## Acciona Eolica Del Levante SL     6853.000   188354.0   21769.000
## Holding De Negocios De GAS SL.  727548.000 13492812.0 6904824.000
## EDP Renovables España SLU        67033.000  1275939.0  726783.000
## Esquilvent SL                     9010.214   157630.6   48769.130
## Eolica La Janda SL                9880.091   153429.4   25206.748
## Viesgo Renovables SL.             4609.000   269730.0  177707.000
##
##                                 RENECO   RENFIN LIQUIDEZ
## Guzman Energia SL               -2.813    6.904    0.078
## Innogy Spain SA.                -2.708   -7.302    1.416
## WPD Wind Investment SL.         -1.040   -1.049    0.624
## Parque Eolico La Boga SL.        0.162    1.684    1.407
## Saeta Yield SA.                  0.360    0.432    2.687
## Global Power Generation SA.      1.393    1.603    2.006
## Naturgy Renovables SLU           1.959   12.043    1.263
## Al-Andalus Wind Power SL         2.349   27.350    1.550
## Olivento SL                      2.553   16.684    0.771
## Elawan Energy SL.                3.615    8.605    0.595
## Naturgy Wind, S.L.               3.949   38.018    1.364
## Parque Eolico Santa Catalina SL  4.053 -359.773    0.388
## Biovent Energia SA               4.551   11.952    1.206
## Corporacion Acciona Eolica SL    4.562   28.990    0.788
## Acciona Eolica Del Levante SL    4.985   43.139    2.855
## Holding De Negocios De GAS SL.   5.264   10.287    1.020
## EDP Renovables España SLU        6.458   11.338    1.596
## Esquilvent SL                    7.621   24.633    5.330
## Eolica La Janda SL               8.586   52.261    1.184
## Viesgo Renovables SL.               NA    3.200    0.272
##
##                                  MARGEN SOLVENCIA  APALANCA
## Guzman Energia SL               -19.193   -40.745  -343.542
## Innogy Spain SA.                -18.025    37.096   150.688
## WPD Wind Investment SL.        -302.027    99.082     0.000
## Parque Eolico La Boga SL.         1.001     9.646   921.591
## Saeta Yield SA.                  16.258    83.489    17.067
## Global Power Generation SA.      22.403    86.917     1.044
## Naturgy Renovables SLU           20.442    16.274   494.729
## Al-Andalus Wind Power SL         12.582     8.591  1019.616
## Olivento SL                      16.629    15.304   534.761
## Elawan Energy SL.               208.357    42.010   123.771
## Naturgy Wind, S.L.               39.575    10.388   824.537
## Parque Eolico Santa Catalina SL  31.780    -1.126 -6265.496
## Biovent Energia SA               22.792    38.082   141.163
## Corporacion Acciona Eolica SL    20.091    15.737   422.263
## Acciona Eolica Del Levante SL    27.520    11.557   743.754
## Holding De Negocios De GAS SL.   91.152    51.174    91.964
## EDP Renovables España SLU        47.193    56.960    67.028
## Esquilvent SL                    39.476    30.938   218.275
## Eolica La Janda SL               38.256    16.428   480.122
## Viesgo Renovables SL.            11.818    65.883    13.330
```

En cambio, para ordenar de modo descendente, hay que utilizar el argumento `desc()`:

```r
arrange(eolica_20, desc(RENECO))
```

```
##                                       RES     ACTIVO       FPIOS
## Eolica La Janda SL                9880.091   153429.4   25206.748
## Esquilvent SL                     9010.214   157630.6   48769.130
## EDP Renovables España SLU        67033.000  1275939.0  726783.000
## Holding De Negocios De GAS SL.  727548.000 13492812.0 6904824.000
## Acciona Eolica Del Levante SL     6853.000   188354.0   21769.000
## Corporacion Acciona Eolica SL    29592.000   864606.0  136064.000
## Biovent Energia SA                     NA   183899.0   70033.000
## Parque Eolico Santa Catalina SL   3645.278   147742.5   -1664.755
## Naturgy Wind, S.L.                8500.000   273542.0   28418.000
## Elawan Energy SL.                12818.975   443467.0  186302.006
## Olivento SL                       7388.175   381207.0   58340.998
## Al-Andalus Wind Power SL          4403.214   249853.8   21466.121
## Naturgy Renovables SLU           42737.000  1956869.0  318475.000
## Global Power Generation SA.      39995.000  2002458.0 1740487.000
## Saeta Yield SA.                   2084.476   796886.4  665319.556
## Parque Eolico La Boga SL.           11.940   303904.4   29316.797
## WPD Wind Investment SL.           -850.068   109023.8  108023.826
## Innogy Spain SA.                 -5268.573   230338.5   85447.212
## Guzman Energia SL                -5661.463   190287.0  -77532.698
## Viesgo Renovables SL.             4609.000   269730.0  177707.000
##
##                                  RENECO   RENFIN LIQUIDEZ
## Eolica La Janda SL                8.586   52.261    1.184
## Esquilvent SL                     7.621   24.633    5.330
## EDP Renovables España SLU         6.458   11.338    1.596
## Holding De Negocios De GAS SL.    5.264   10.287    1.020
## Acciona Eolica Del Levante SL     4.985   43.139    2.855
## Corporacion Acciona Eolica SL     4.562   28.990    0.788
## Biovent Energia SA                4.551   11.952    1.206
## Parque Eolico Santa Catalina SL   4.053 -359.773    0.388
## Naturgy Wind, S.L.                3.949   38.018    1.364
## Elawan Energy SL.                 3.615    8.605    0.595
## Olivento SL                       2.553   16.684    0.771
## Al-Andalus Wind Power SL          2.349   27.350    1.550
## Naturgy Renovables SLU            1.959   12.043    1.263
## Global Power Generation SA.       1.393    1.603    2.006
## Saeta Yield SA.                   0.360    0.432    2.687
## Parque Eolico La Boga SL.         0.162    1.684    1.407
## WPD Wind Investment SL.          -1.040   -1.049    0.624
## Innogy Spain SA.                 -2.708   -7.302    1.416
## Guzman Energia SL                -2.813    6.904    0.078
## Viesgo Renovables SL.                NA    3.200    0.272
##
##                                   MARGEN SOLVENCIA  APALANCA
## Eolica La Janda SL                38.256    16.428   480.122
## Esquilvent SL                     39.476    30.938   218.275
## EDP Renovables España SLU         47.193    56.960    67.028
## Holding De Negocios De GAS SL.    91.152    51.174    91.964
## Acciona Eolica Del Levante SL     27.520    11.557   743.754
## Corporacion Acciona Eolica SL     20.091    15.737   422.263
```

```
## Biovent Energia SA                22.792    38.082   141.163
## Parque Eolico Santa Catalina SL   31.780    -1.126 -6265.496
## Naturgy Wind, S.L.                39.575    10.388   824.537
## Elawan Energy SL.                208.357    42.010   123.771
## Olivento SL                       16.629    15.304   534.761
## Al-Andalus Wind Power SL          12.582     8.591  1019.616
## Naturgy Renovables SLU            20.442    16.274   494.729
## Global Power Generation SA.       22.403    86.917     1.044
## Saeta Yield SA.                   16.258    83.489    17.067
## Parque Eolico La Boga SL.          1.001     9.646   921.591
## WPD Wind Investment SL.         -302.027    99.082     0.000
## Innogy Spain SA.                 -18.025    37.096   150.688
## Guzman Energia SL                -19.193   -40.745  -343.542
## Viesgo Renovables SL.             11.818    65.883    13.330
```

En el supuesto de que, por ejemplo, hubiera varias empresas con la misma rentabilidad económica (RENECO), podría añadirse otro criterio de ordenación con otra variable, que afectaría a tales empresas para deshacer el "empate" en rentabilidad económica. Por ejemplo, para ordenar de modo ascendente por rentabilidad y, en caso de que haya rentabilidades iguales, por liquidez (variable LIQUIDEZ), se ejecutaría:

```
arrange(eolica_20, RENECO, LIQUIDEZ)
```

```
##                                     RES     ACTIVO       FPIOS
## Guzman Energia SL                -5661.463   190287.0  -77532.698
## Innogy Spain SA.                 -5268.573   230338.5   85447.212
## WPD Wind Investment SL.           -850.068   109023.8  108023.826
## Parque Eolico La Boga SL.           11.940   303904.4   29316.797
## Saeta Yield SA.                   2084.476   796886.4  665319.556
## Global Power Generation SA.      39995.000  2002458.0 1740487.000
## Naturgy Renovables SLU           42737.000  1956869.0  318475.000
## Al-Andalus Wind Power SL          4403.214   249853.8   21466.121
## Olivento SL                       7388.175   381207.0   58340.998
## Elawan Energy SL.                12818.975   443467.0  186302.006
## Naturgy Wind, S.L.                8500.000   273542.0   28418.000
## Parque Eolico Santa Catalina SL   3645.278   147742.5   -1664.755
## Biovent Energia SA                      NA   183899.0   70033.000
## Corporacion Acciona Eolica SL    29592.000   864606.0  136064.000
## Acciona Eolica Del Levante SL     6853.000   188354.0   21769.000
## Holding De Negocios De GAS SL.  727548.000 13492812.0 6904824.000
## EDP Renovables España SLU        67033.000  1275939.0  726783.000
## Esquilvent SL                     9010.214   157630.6   48769.130
## Eolica La Janda SL                9880.091   153429.4   25206.748
## Viesgo Renovables SL.             4609.000   269730.0  177707.000
##
##                                  RENECO   RENFIN LIQUIDEZ
## Guzman Energia SL                -2.813    6.904    0.078
## Innogy Spain SA.                 -2.708   -7.302    1.416
## WPD Wind Investment SL.          -1.040   -1.049    0.624
## Parque Eolico La Boga SL.         0.162    1.684    1.407
## Saeta Yield SA.                   0.360    0.432    2.687
## Global Power Generation SA.       1.393    1.603    2.006
## Naturgy Renovables SLU            1.959   12.043    1.263
## Al-Andalus Wind Power SL          2.349   27.350    1.550
## Olivento SL                       2.553   16.684    0.771
```

```
## Elawan Energy SL.                3.615    8.605    0.595
## Naturgy Wind, S.L.               3.949   38.018    1.364
## Parque Eolico Santa Catalina SL  4.053 -359.773    0.388
## Biovent Energia SA               4.551   11.952    1.206
## Corporacion Acciona Eolica SL    4.562   28.990    0.788
## Acciona Eolica Del Levante SL    4.985   43.139    2.855
## Holding De Negocios De GAS SL.   5.264   10.287    1.020
## EDP Renovables España SLU        6.458   11.338    1.596
## Esquilvent SL                    7.621   24.633    5.330
## Eolica La Janda SL               8.586   52.261    1.184
## Viesgo Renovables SL.               NA    3.200    0.272
##
##                                   MARGEN SOLVENCIA  APALANCA
## Guzman Energia SL                -19.193   -40.745  -343.542
## Innogy Spain SA.                 -18.025    37.096   150.688
## WPD Wind Investment SL.         -302.027    99.082     0.000
## Parque Eolico La Boga SL.          1.001     9.646   921.591
## Saeta Yield SA.                   16.258    83.489    17.067
## Global Power Generation SA.       22.403    86.917     1.044
## Naturgy Renovables SLU            20.442    16.274   494.729
## Al-Andalus Wind Power SL          12.582     8.591  1019.616
## Olivento SL                       16.629    15.304   534.761
## Elawan Energy SL.                208.357    42.010   123.771
## Naturgy Wind, S.L.                39.575    10.388   824.537
## Parque Eolico Santa Catalina SL   31.780    -1.126 -6265.496
## Biovent Energia SA                22.792    38.082   141.163
## Corporacion Acciona Eolica SL     20.091    15.737   422.263
## Acciona Eolica Del Levante SL     27.520    11.557   743.754
## Holding De Negocios De GAS SL.    91.152    51.174    91.964
## EDP Renovables España SLU         47.193    56.960    67.028
## Esquilvent SL                     39.476    30.938   218.275
## Eolica La Janda SL                38.256    16.428   480.122
## Viesgo Renovables SL.             11.818    65.883    13.330
```

Obviamente, en este ejemplo concreto el resultado es el mismo que se obtuvo con `arrange(eolica_20, RENECO)`, puesto que no hay rentabilidades iguales entre las 20 empresas de la muestra.

2.3.5 Cambiando el nombre de las variables de un *data frame*.

`{dplyr}` cuenta con una función que cambia fácilmente el nombre de una variable o columna de un *data frame*: la función `rename()`. Por ejemplo, si queremos cambiar el nombre de la variable SOLVENCIA por SOLVE, simplemente ejecutaremos:

```
#Renombrando variables
eolica_20 <- rename(eolica_20, SOLVE = SOLVENCIA)
```

Podemos comprobar en el *Environment*, despegando el objeto "eolica_20", cómo ya no aparece la variable SOLVENCIA; pero sí SOLVE en su lugar (obviamente, con los mismos datos). Es necesario tener en cuenta que en el **lado izquierdo** de la igualdad hay que poner el **nuevo nombre**, y en la derecha el antiguo. Además, en el mismo `rename()` se pueden cambiar los nombres de **varias variables**, separando las igualdades correspondientes con comas.

2.3.6 Añadiendo variables como transformación de otras variables en un *data frame*.

El paquete `{dplyr}` también permite añadir a un *data frame* variables que son el resultado de **someter a otras variables a diversas transformaciones**. La función para realizar este cometido es `mutate()`.

Así, por ejemplo, imaginemos que necesitamos calcular una variable como el cociente entre los resultados obtenidos y el activo. A esta nueva variable la denominaremos RATIO. El código será:

```
# Añadiendo variables como transformacion de otras variables
eolica_20 <- mutate (eolica_20, RATIO = RES / ACTIVO)
summary(eolica_20)
```

```
##       RES              ACTIVO             FPIOS              RENECO
##  Min.   : -5662   Min.   :  109024   Min.   : -77533   Min.   :-2.8130
##  1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615   1st Qu.: 0.8765
##  Median :  7388   Median :  271636   Median :  77740   Median : 3.6150
##  Mean   : 50754   Mean   : 1183599   Mean   : 563678   Mean   : 2.9399
##  3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345   3rd Qu.: 4.7735
##  Max.   :727548   Max.   :13492812   Max.   :6904824   Max.   : 8.5860
##  NA's   :1                                             NA's   :1
##
##      RENFIN            LIQUIDEZ          MARGEN            SOLVE
##  Min.   :-359.773   Min.   :0.0780   Min.   :-302.03   Min.   :-40.74
##  1st Qu.:   1.664   1st Qu.:0.7342   1st Qu.:  12.39   1st Qu.: 11.26
##  Median :  10.812   Median :1.2345   Median :  21.42   Median : 23.68
##  Mean   :  -3.450   Mean   :1.4200   Mean   :  16.40   Mean   : 32.68
##  3rd Qu.:  25.312   3rd Qu.:1.5615   3rd Qu.:  38.56   3rd Qu.: 52.62
##  Max.   :  52.261   Max.   :5.3300   Max.   : 208.36   Max.   : 99.08
##
##     APALANCA            RATIO
##  Min.   :-6265.50   Min.   :-0.029752
##  1st Qu.:   16.13   1st Qu.: 0.009852
##  Median :  145.93   Median : 0.021840
##  Mean   :  -17.17   Mean   : 0.022180
##  3rd Qu.:  504.74   3rd Qu.: 0.035305
##  Max.   : 1019.62   Max.   : 0.064395
##                     NA's   :1
```

En la transformación de variables mediante la función `mutate()`, se pueden utilizar **funciones integradas en otros paquetes** de R. Por ejemplo, si queremos calcular la variable ACTIVOS_ACUM como la variable que recoge los activos acumulados de las empresas, comenzando por la empresa con menor activo, podríamos utilizar la función `cumsum()` del paquete `{base}`, y hacer:

```
eolica_20 <- arrange(eolica_20, ACTIVO)
eolica_20 <- mutate (eolica_20, ACTIVOS_ACUM = cumsum(ACTIVO))
select(eolica_20, ACTIVO, ACTIVOS_ACUM)
```

```
##                                   ACTIVO ACTIVOS_ACUM
## WPD Wind Investment SL.           109023.8     109023.8
## Parque Eolico Santa Catalina SL   147742.5     256766.3
## Eolica La Janda SL                153429.4     410195.8
## Esquilvent SL                     157630.6     567826.4
```

```
## Biovent Energia SA                183899.0     751725.4
## Acciona Eolica Del Levante SL     188354.0     940079.4
## Guzman Energia SL                 190287.0    1130366.4
## Innogy Spain SA.                  230338.5    1360704.9
## Al-Andalus Wind Power SL          249853.8    1610558.7
## Viesgo Renovables SL.             269730.0    1880288.7
## Naturgy Wind, S.L.                273542.0    2153830.7
## Parque Eolico La Boga SL.         303904.4    2457735.1
## Olivento SL                       381207.0    2838942.0
## Elawan Energy SL.                 443467.0    3282409.0
## Saeta Yield SA.                   796886.4    4079295.4
## Corporacion Acciona Eolica SL     864606.0    4943901.4
## EDP Renovables España SLU        1275939.0    6219840.4
## Naturgy Renovables SLU           1956869.0    8176709.4
## Global Power Generation SA.      2002458.0   10179167.4
## Holding De Negocios De GAS SL.  13492812.0   23671979.4
```

Podemos verificar cómo se ha integrado en el *data frame* la variable ACTIVOS_ACUM.

Un último ejemplo de adición de una variable que es transformación de otras. En este caso, crearemos la variable TAM (tamaño), que es **categórica** (los datos son conjuntos de carcteres). Esta variable toma valor "G" para las empresas con un valor de la variable ACTIVO mayor que 1.000.000, y "P" para las que tengan un valor en la variable ACTIVO menor o igual a 1.000.000. Para calcular automáticamente esta nueva variable categórica, utilizaremos la función de `{base} cut()`. De este modo, haremos:

```
eolica_20 <- mutate(eolica_20, TAM = cut(ACTIVO,
  breaks = c(-Inf, 1000000, Inf), labels = c("P", "G")))
select(eolica_20, ACTIVO, TAM)
```

```
##                                   ACTIVO TAM
## WPD Wind Investment SL.           109023.8   P
## Parque Eolico Santa Catalina SL   147742.5   P
## Eolica La Janda SL                153429.4   P
## Esquilvent SL                     157630.6   P
## Biovent Energia SA                183899.0   P
## Acciona Eolica Del Levante SL     188354.0   P
## Guzman Energia SL                 190287.0   P
## Innogy Spain SA.                  230338.5   P
## Al-Andalus Wind Power SL          249853.8   P
## Viesgo Renovables SL.             269730.0   P
## Naturgy Wind, S.L.                273542.0   P
## Parque Eolico La Boga SL.         303904.4   P
## Olivento SL                       381207.0   P
## Elawan Energy SL.                 443467.0   P
## Saeta Yield SA.                   796886.4   P
## Corporacion Acciona Eolica SL     864606.0   P
## EDP Renovables España SLU        1275939.0   G
## Naturgy Renovables SLU           1956869.0   G
## Global Power Generation SA.      2002458.0   G
## Holding De Negocios De GAS SL.  13492812.0   G
```

Podemos advertir cómo la función `cut()`, que incluimos dentro de nuestra función de `{dplyr} mutate()`, tiene, a su vez, varios argumentos: la variable numérica de referencia (ACTIVO); el argumento **"breaks"**, en el que decimos los intervalos en que quedarán divididos los casos (uno, de menos infinito a 1.000.000; y

otro de 1.000.000 a más infinito), y **"labels"**, que es el valor que tomará la variable creada (TAM) según el intervalo en el que se sitúe cada caso de la muestra.

Cabe destacar que podíamos haber escrito el código para crear la variable TAM de un modo más elegante y cómodo, utilizando el operador **"*pipe*"** (%>%). Este operador permite concatenar una serie de instrucciones:

```
eolica_20 <- eolica_20 %>% mutate(TAM = cut(ACTIVO,
  breaks = c(-Inf, 1000000, Inf), labels = c("P", "G")))
select(eolica_20, ACTIVO, TAM)
```

```
##                                     ACTIVO TAM
## WPD Wind Investment SL.           109023.8   P
## Parque Eolico Santa Catalina SL   147742.5   P
## Eolica La Janda SL                153429.4   P
## Esquilvent SL                     157630.6   P
## Biovent Energia SA                183899.0   P
## Acciona Eolica Del Levante SL     188354.0   P
## Guzman Energia SL                 190287.0   P
## Innogy Spain SA.                  230338.5   P
## Al-Andalus Wind Power SL          249853.8   P
## Viesgo Renovables SL.             269730.0   P
## Naturgy Wind, S.L.                273542.0   P
## Parque Eolico La Boga SL.         303904.4   P
## Olivento SL                       381207.0   P
## Elawan Energy SL.                 443467.0   P
## Saeta Yield SA.                   796886.4   P
## Corporacion Acciona Eolica SL     864606.0   P
## EDP Renovables España SLU        1275939.0   G
## Naturgy Renovables SLU           1956869.0   G
## Global Power Generation SA.      2002458.0   G
## Holding De Negocios De GAS SL.  13492812.0   G
```

Podríamos interpretar la línea de código así: "asigna al *data frame*"eolica_20" sus propios datos, después (%>%) crea la variable TAM con la función cut() y añádela a "eolica_20".

2.3.7 Extrayendo y sintetizando información de las variables de un *data frame*.

Otra posibilidad que permite `{dplyr}` es extraer y sintetizar la información de las variables contenidas en un *data frame*. Para ello, nos ayudaremos de la función `summarise()`. Como ejemplo, calculemos la *rentabilidad financiera* media de las 20 empresas:

```
#Extrayendo información de las variables de un data frame
summarise(eolica_20, RENFIN_media = mean(RENFIN))
```

```
##   RENFIN_media
## 1     -3.45005
```

A veces, es de gran utilidad combinar `summarise()` con `group_by()`, que extrae la información por grupos definidos por una de las variables. Para ilustrarlo, vamos a utilizar la variable recién creada TAM, para hacer dos grupos de empresas: las de menor ("P") y las de mayor ("G") volumen de activo; tras lo cual calcularemos la media de las rentabilidades para cada grupo:

```
eolica_20 %>%  group_by(TAM) %>% summarise(RENFIN_media = mean(RENFIN))
```

```
## # A tibble: 2 x 2
##   TAM   RENFIN_media
##   <fct>        <dbl>
## 1 P            -6.52
## 2 G             8.82
```

Hemos utilizado el operador *pipe* (%>%) para concatenar diferentes instrucciones de `{dplyr}`: primero agrupar casos, y luego calcular las medias de cada grupo. Es decir, en este caso se podría "traducir" la línea de código como: "Toma el *data frame*"eolica_9", divide los casos en grupos según el valor de la variable TAM, y para cada grupo calcula la media de la variable RENFIN".

2.4 Exportando datos.

Antes de concluir el capítulo, vamos a tratar brevemente el aspecto de la exportación de datos.

R cuenta con un **formato propio de datos**, que se traduce en archivos de extensión "RData", y que puede incluir cualquier objeto de R. Como ejemplo, en el siguiente *script*, llamado "explora_exporta.R" (obtener aquí), vamos a importar los datos del archivo de Microsoft (R) Excel (R) "eolica_20.xlsx", y el *data frame* donde almacenemos los datos vamos a exportarlo como el archivo de datos de R "eolica_20.RData". Posteriormente, borraremos el *data frame* del *Environment* y recuperaremos los datos cargando ese archivo "eolica_20.RData". Por supuesto, seguimos trabajando, como en todo el capítulo, en el proyecto "explora".

Tras abrir el *script* "explora_exporta.R", las primeras líneas de código que veremos serán las que ya hemos estudiado para borrar el contenido del *Environment*, importar los datos de la hoja "Top 20" del archivo "eolica_20.xlsx" (situado en nuestra carpeta de proyecto), y tratar la variable "NOMBRE" para transformarla en el conjunto de nombres de las filas:

```
# Exportando datos de empresas eolicas (disculpad la falta de tildes)

rm(list = ls())

# DATOS

library(readxl)
eolica_20 <- read_excel("eolica_20.xlsx", sheet = "Top 20")
eolica_20 <- data.frame(eolica_20, row.names = 1)
summary(eolica_20)
```

```
##       RES              ACTIVO             FPIOS              RENECO       
##  Min.   : -5662   Min.   :  109024   Min.   : -77533   Min.   :-2.8130  
##  1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615   1st Qu.: 0.8765  
##  Median :  7388   Median :  271636   Median :  77740   Median : 3.6150  
##  Mean   : 50754   Mean   : 1183599   Mean   : 563678   Mean   : 2.9399  
##  3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345   3rd Qu.: 4.7735  
##  Max.   :727548   Max.   :13492812   Max.   :6904824   Max.   : 8.5860  
##  NA's   :1                                             NA's   :1        
##
##      RENFIN             LIQUIDEZ          MARGEN          SOLVENCIA      
##  Min.   :-359.773   Min.   :0.0780   Min.   :-302.03   Min.   :-40.74  
##  1st Qu.:   1.664   1st Qu.:0.7342   1st Qu.:  12.39   1st Qu.: 11.26  
```

```
##  Median :  10.812   Median :1.2345   Median :  21.42   Median : 23.68
##  Mean   :  -3.450   Mean   :1.4200   Mean   :  16.40   Mean   : 32.68
##  3rd Qu.:  25.312   3rd Qu.:1.5615   3rd Qu.:  38.56   3rd Qu.: 52.62
##  Max.   :  52.261   Max.   :5.3300   Max.   : 208.36   Max.   : 99.08
##
##     APALANCA           MATRIZ
##  Min.   :-6265.50   Length:20
##  1st Qu.:   16.13   Class :character
##  Median :  145.93   Mode  :character
##  Mean   :  -17.17
##  3rd Qu.:  504.74
##  Max.   : 1019.62
```

Posteriormente, se exportará el data frame "eolica_20" al archivo de formato R, "eolica_20.RData", mediante la función `save`:

```
# Exportando data frame a formato R (.RData)

save(eolica_20, file = "eolica_20.RData")
```

Puede comprobarse cómo se ha generado el archivo correspondiente en la carpeta de proyecto. Para comprobar que la exportación es correcta, vamos a borrar del *Environment* el *data frame* "eolica_20". Después, cargaremos el archivo "eolica_20.RData". Como resultado, podremos comprobar que tenemos un nuevo *data frame* "eolica_20" que es exactamente igual al que teníamos al principio:

```
# Borrando el data frame eolica_20

rm(eolica_20)

# Importando el archivo .RData con los mismos datos

load("eolica_20.RData")
summary (eolica_20)
```

```
##       RES              ACTIVO             FPIOS             RENECO
##  Min.   : -5662   Min.   :  109024   Min.   : -77533   Min.   :-2.8130
##  1st Qu.:  2865   1st Qu.:  187240   1st Qu.:  27615   1st Qu.: 0.8765
##  Median :  7388   Median :  271636   Median :  77740   Median : 3.6150
##  Mean   : 50754   Mean   : 1183599   Mean   : 563678   Mean   : 2.9399
##  3rd Qu.: 21206   3rd Qu.:  813816   3rd Qu.: 219345   3rd Qu.: 4.7735
##  Max.   :727548   Max.   :13492812   Max.   :6904824   Max.   : 8.5860
##  NA's   :1                                             NA's   :1
##
##      RENFIN            LIQUIDEZ          MARGEN          SOLVENCIA
##  Min.   :-359.773   Min.   :0.0780   Min.   :-302.03   Min.   :-40.74
##  1st Qu.:   1.664   1st Qu.:0.7342   1st Qu.:  12.39   1st Qu.: 11.26
##  Median :  10.812   Median :1.2345   Median :  21.42   Median : 23.68
##  Mean   :  -3.450   Mean   :1.4200   Mean   :  16.40   Mean   : 32.68
##  3rd Qu.:  25.312   3rd Qu.:1.5615   3rd Qu.:  38.56   3rd Qu.: 52.62
##  Max.   :  52.261   Max.   :5.3300   Max.   : 208.36   Max.   : 99.08
##
##     APALANCA           MATRIZ
##  Min.   :-6265.50   Length:20
```

```
##  1st Qu.:   16.13   Class :character
##  Median :  145.93   Mode  :character
##  Mean   :  -17.17
##  3rd Qu.:  504.74
##  Max.   : 1019.62
```

Por supuesto, hay más formatos en los que se pueden exportar datos desde R. Por ejemplo, **a un archivo de Microsoft (R) Excel (R)**. Un modo de hacerlo es haciendo uso de la función `write_xlsx()` del paquete {writexl}. Para que en la hoja de cálculo resultante se incluyan los nombres de las filas (empresas eólicas), hemos tenido previamente que crear un vector con el nombre de estas (vector "NOMBRE"), mediante la función `row.names()`, y unir ese vector al *data frame* "eolica_20", a modo de primera columna, creando un nuevo finalmente un *data frame* llamado "eolica_20n", para lo que se ha utilizado la función `cbind()`, que permite **pegar columnas de datos** que tengan un mismo número de filas.

Como resultado de todo el código, se ha obtenido el archivo de Microsoft (R) Excel (R) "eolica_20_new.xlsx":

```
# Exportando el data frame eolica_20 a Microsoft Excel

library(writexl)
NOMBRE <- row.names(eolica_20)
eolica_20n <- cbind(NOMBRE, eolica_20)
write_xlsx(eolica_20n, path = "eolica_20_new.xlsx")

#Fin del script
```

2.5 Materiales para realizar las prácticas del capítulo.

En esta sección se muestran los links de acceso a los diferentes materiales (*scripts*, datos…) necesarios para llevar a cabo los contenidos prácticos del capítulo.

Datos (en formato Microsoft (R) Excel (R)):

- eolica_20.xlsx (obtener aquí)

Scripts:

- explora_eolica.R (obtener aquí)
- explora_dplyr.R (obtener aquí)
- explora_exporta.R (obtener aquí)

Capítulo 3

Gráficos.

3.1 Tidyverse para gráficos: ggplot2.

R, en su instalación básica, cuenta con funciones destinadas a crear gráficos y, de este modo, visualizar nuestros datos a fin de generar información y extraer conclusiones de un modo sencillo. No obstante, estas funciones, a veces, se quedan "cortas", o requieren de un complejo y/o extenso código. Esta es la razón por la que en el *Tidyverse* se incluyó un paquete específico destinado a la construcción de gráficos de un modo flexible y amigable. Recordemos que el *Tidyverse* es un conjunto de paquetes con una filosofía común, como es el uso de ciertas estructuras gramaticales, que facilitan muchas de las tareas y análisis que podrían hacerse con el lenguaje R estándar.

Este paquete destinado a la producción de gráficos es `{ggplot2}`, que proporciona unas herramientas muy flexibles para visualizar conjuntos de datos. A continuación, se expondrán los fundamentos de la sintaxis de `{ggplot2}` y se indicará cómo construir algunos de los gráficos más habituales.

Para ilustrar la creación de gráficos, vamos a suponer que trabajamos dentro de un proyecto que hemos creado previamente, de nombre "explora". Dentro de la carpeta del proyecto guardaremos el *script* llamado "explora_ggplot2.R" y el archivo de *Microsoft® Excel®* llamado "eolica_100.xlsx".

Si abrimos este último archivo, comprobaremos que se compone de tres hojas. La primera muestra un aviso sobre el uso de los datos; la segunda recoge la descripción de las variables consideradas; y la tercera (hoja "Datos") guarda los datos que debemos importar desde R-Studio. Estos datos se corresponden con diferentes variables económico-financieras de las 100 empresas productoras de electricidad mediante generación eólica con mayor volumen de activo total.

Tras abrir el *script* "explora_ggplot2.R" e el editor de R-Studio, observaremos que la primera línea / instrucción es:

```
rm(list = ls())
```

La instrucción tiene como objeto limpiar el *Environment* (memoria) de objetos de anteriores sesiones de trabajo.

Para importar los datos que hay en la hoja "Datos" del archivo de Microsoft® Excel® llamado "eolica_100.xlsx", ejecutaremos el código:

```
library(readxl)
eolica_100 <- read_excel("eolica_100.xlsx", sheet = "Datos")
summary (eolica_100)
```

```
##     NOMBRE               RES               ACTIVO            FPIOS
##  Length:100         Min.   : -5661.5   Min.   :   24944   Min.   : -77533
##  Class :character   1st Qu.:   670.2   1st Qu.:   34437   1st Qu.:   2305
##  Mode  :character   Median :  2114.7   Median :   46896   Median :  11936
##                     Mean   : 11477.3   Mean   :  274756   Mean   : 123743
##                     3rd Qu.:  3951.2   3rd Qu.:   85542   3rd Qu.:  28292
##                     Max.   :727548.0   Max.   :13492812   Max.   :6904824
##
##      RENECO           RENFIN            LIQUIDEZ           MARGEN
##  Min.   :-3.446   Min.   :-359.773   Min.   :  0.0140   Min.   :-2248.157
##  1st Qu.: 1.421   1st Qu.:   2.556   1st Qu.:  0.6567   1st Qu.:   12.126
##  Median : 4.144   Median :  15.326   Median :  1.0650   Median :   26.618
##  Mean   : 5.294   Mean   :  17.243   Mean   :  2.7214   Mean   :    3.583
##  3rd Qu.: 7.904   3rd Qu.:  31.307   3rd Qu.:  1.6078   3rd Qu.:   39.580
##  Max.   :35.262   Max.   : 588.190   Max.   :128.4330   Max.   :  400.899
##
##    SOLVENCIA         APALANCA          MATRIZ            DIMENSION
##  Min.   :-40.74   Min.   :-8254.11   Length:100         Length:100
##  1st Qu.:  4.71   1st Qu.:   16.13   Class :character   Class :character
##  Median : 16.65   Median :  161.97   Mode  :character   Mode  :character
##  Mean   : 27.57   Mean   :  345.03
##  3rd Qu.: 45.59   3rd Qu.:  623.13
##  Max.   : 99.08   Max.   :12244.35
```

Podemos observar cómo, en el *Environment,* ya aparece un *data frame* que se llama "eolica_100", y contiene 12 columnas. R ha considerado la primera columna como una variable de tipo cualitativo o atributo. En realidad, esa columna no es una variable, sino que está formada por los nombres de los diferentes casos u observaciones (filas). Para evitar que R tome la columna de los nombres de los casos como una variable más, podemos redefinir nuestro *data frame* diciéndole que considere esa primera columna como el conjunto de los *nombres de los individuos o casos*:

```
eolica_100 <- data.frame(eolica_100, row.names = 1)
```

En la línea anterior, hemos asignado al *data frame* "eolica_100" los propios datos de "eolica_100"; pero indicando que la primera columna no es una variable; sino que contiene el nombre de los casos. Si hacemos ahora un `summary()`:

```
summary (eolica_100)
```

```
##       RES               ACTIVO            FPIOS             RENECO
##  Min.   : -5661.5   Min.   :   24944   Min.   : -77533   Min.   :-3.446
##  1st Qu.:   670.2   1st Qu.:   34437   1st Qu.:   2305   1st Qu.: 1.421
##  Median :  2114.7   Median :   46896   Median :  11936   Median : 4.144
##  Mean   : 11477.3   Mean   :  274756   Mean   : 123743   Mean   : 5.294
##  3rd Qu.:  3951.2   3rd Qu.:   85542   3rd Qu.:  28292   3rd Qu.: 7.904
##  Max.   :727548.0   Max.   :13492812   Max.   :6904824   Max.   :35.262
##
##      RENFIN            LIQUIDEZ           MARGEN            SOLVENCIA
##  Min.   :-359.773   Min.   :  0.0140   Min.   :-2248.157   Min.   :-40.74
##  1st Qu.:   2.556   1st Qu.:  0.6567   1st Qu.:   12.126   1st Qu.:  4.71
##  Median :  15.326   Median :  1.0650   Median :   26.618   Median : 16.65
##  Mean   :  17.243   Mean   :  2.7214   Mean   :    3.583   Mean   : 27.57
##  3rd Qu.:  31.307   3rd Qu.:  1.6078   3rd Qu.:   39.580   3rd Qu.: 45.59
```

```
##  Max.   : 588.190   Max.   :128.4330   Max.   :  400.899   Max.   : 99.08
##
##     APALANCA           MATRIZ           DIMENSION
##  Min.   :-8254.11   Length:100         Length:100
##  1st Qu.:   16.13   Class :character   Class :character
##  Median :  161.97   Mode  :character   Mode  :character
##  Mean   :  345.03
##  3rd Qu.:  623.13
##  Max.   :12244.35
```

Comprobamos que ya no aparece NOMBRE como variable y que, en el *Environment*, se recoge el *data frame* "eolica_100" con 100 casos y con 11 variables.

3.2 Gráficos de una variable: histogramas, gráficos de densidad, gráficos de caja o *boxplots*.

A continuación, cargaremos el paquete `{ggplot2}`. Si nunca antes se ha utilizado, cuando lo intentemos activar con la función `library()` nos dará un error, advirtiendo que previamente hay que instalarlo. En ese caso, iremos a la ventana inferior-derecha de R-Studio y pulsaremos en la pestaña `Packages`, luego en `Install`, y emergerá una ventana donde dejaremos el "repositorio" que viene por defecto y, en el campo `Packages`, escribiremos el nombre del "paquete" (en nuestro caso *ggplot2*). Una vez descargado el "paquete", podremos ejecutar el código sin problemas:

```
#Cargando ggplot2
library (ggplot2)
```

La primera instrucción para crear un gráfico con el paquete `{ggplot2}` es `ggplot()`. A continuación, entre paréntesis, se deberán aportar una serie de argumentos o informaciones. Estas informaciones irán definiendo el gráfico en mayor o menor detalle.

En realidad, lo que se hace es definir el conjunto de datos a representar (que suelen estar contenidos en un *data frame*, o en varios), y a partir de ellos se van añadiendo capas gráficas o "***geoms***", que son caracterizadas con ciertos atributos estéticos *("*aesthetics"*, o* "aes").

3.2.1 Histograma.

Uno de los gráficos indispensables para tener una idea de la distribución de frecuencias que siguen los casos (en nuestro ejemplo, las empresas eólicas) con relación a una variable métrica es el **histograma**. Vamos a construir un histograma para la variable de rentabilidad económica, RENECO. El código será:

```
#Histograma
ggplot(data = eolica_100, map = aes(x = RENECO)) +
       geom_histogram()
```

Como acabamos de decir, en primer lugar viene el comando `ggplot()`, seguido de unos paréntesis que recogen ciertas informaciones:

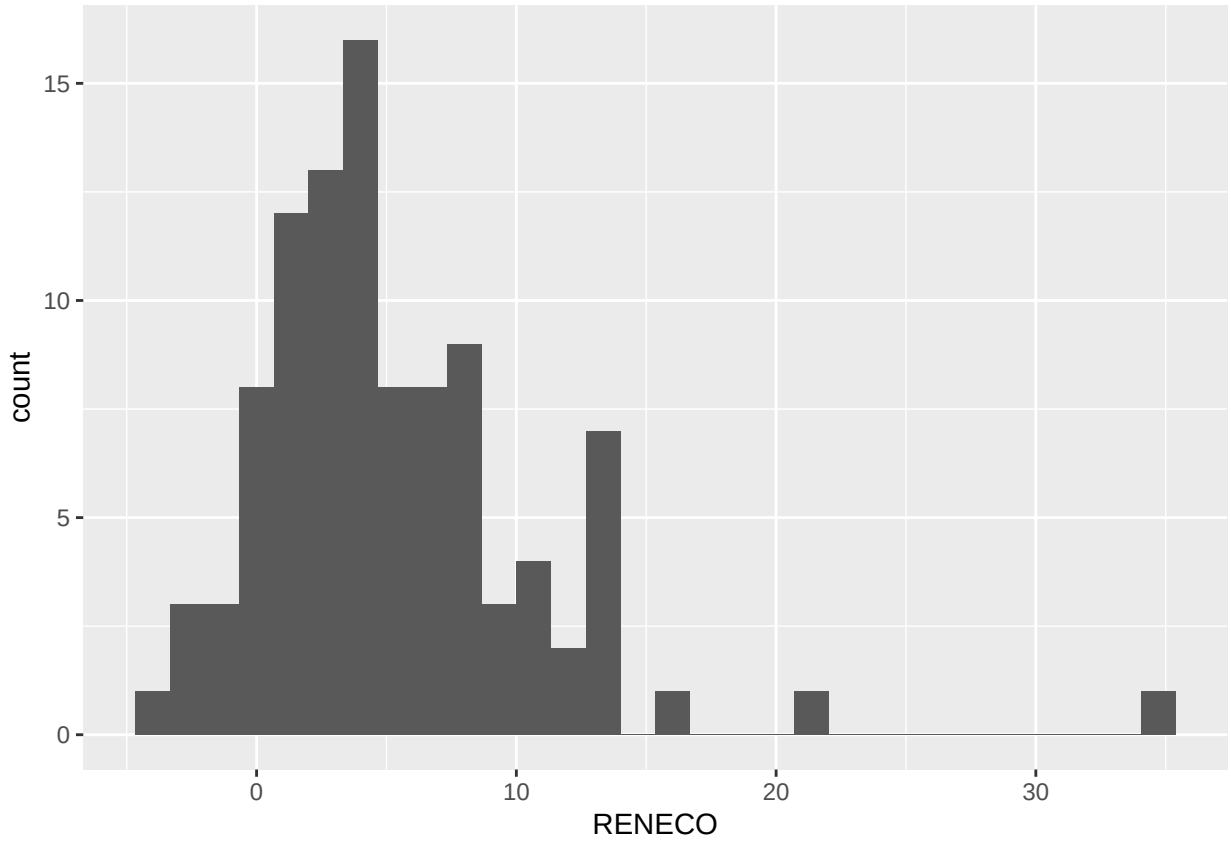
- **"data =**", seguido de la fuente que almacena los datos a graficar (en nuestro caso, el *data frame* "eolica_100").

- **"map ="**, o "mapeo", que define los aspectos del gráfico que dependen del valor de alguna o algunas variables. Siempre que alguna característica del gráfico no sea "fija", sino que dependa de los valores que toma una variable, tal variable deberá ir indicada dentro de un *elemento estético* (*aes*). En el código de ejemplo, el elemento *aes* sirve para indicar que las coordenadas del eje x que toman los casos a representar, dependen de los valores de la variable RENECO.

Para indicar que las siguientes líneas continúan con el código del gráfico, se añade al final de esta línea el símbolo "+".

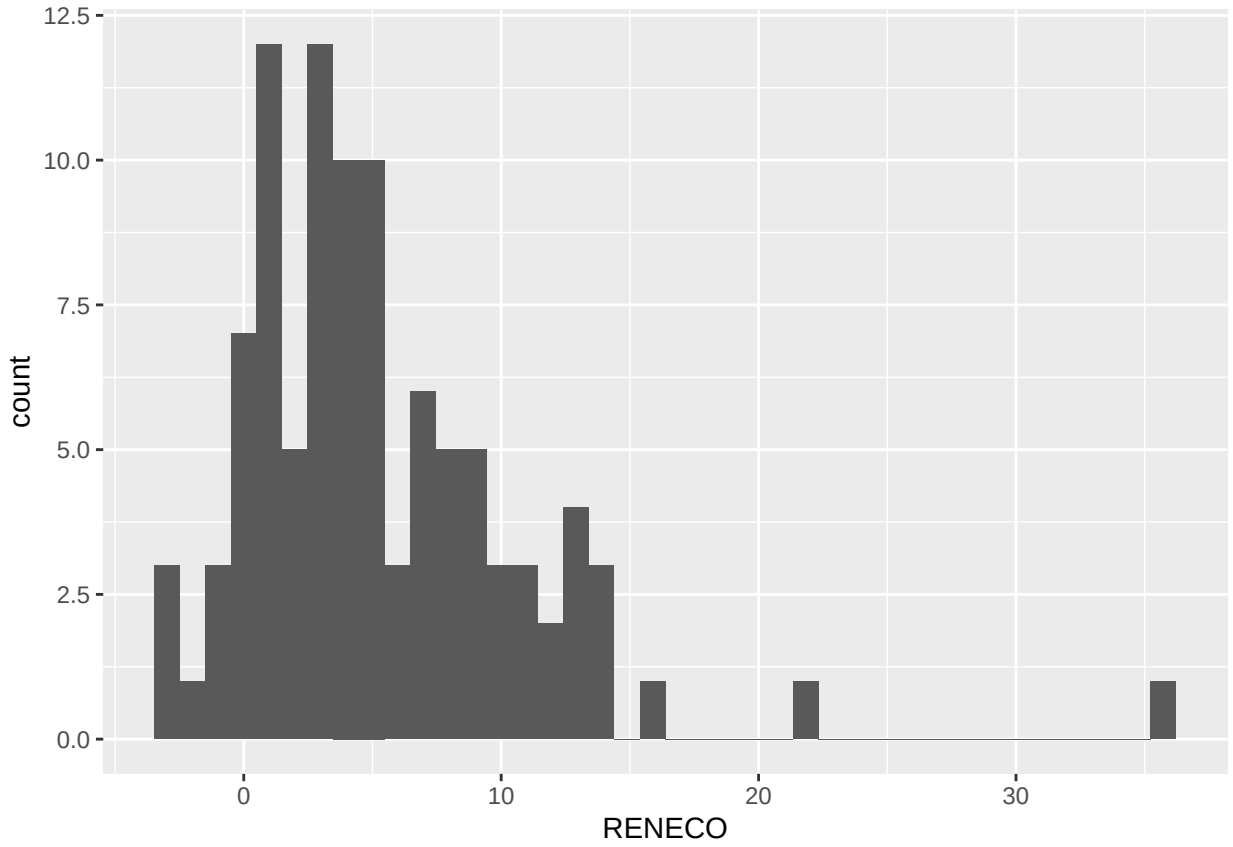
En la segunda línea, se establece el tipo de gráfico que se va a realizar, mediante la inclusión de un elemento ***geom***. Para decir que lo que queremos construir es un histograma, el elemento *geom* será `geom_histogram()`.

El resultado del código anterior es el siguiente gráfico:



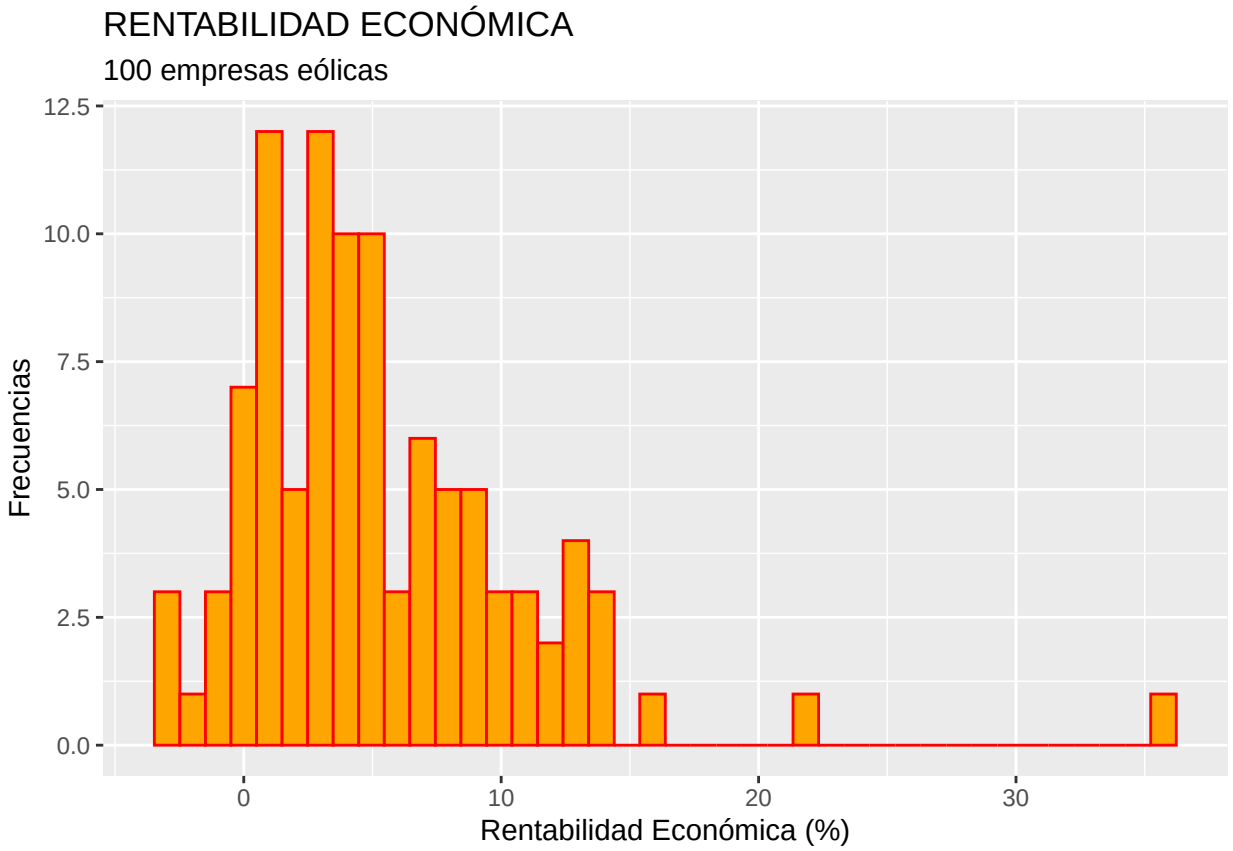
Por supuesto, `{ggplot2}` permite personalizar y refinar la apariencia del gráfico. Uno de los aspectos que nos puede interesar modificar es el número de intervalos en los que queda dividido el rango de valores que puede tomar la variable ("grosor" de las barras), o *bins*. Por defecto, el número es 30. Para incrementar este número de barras a 40, por ejemplo, añadiremos en la línea del *geom* el argumento **"bins ="**:

```
#Histograma
ggplot(data = eolica_100, map = aes(x = RENECO)) +
       geom_histogram(bins = 40)
```



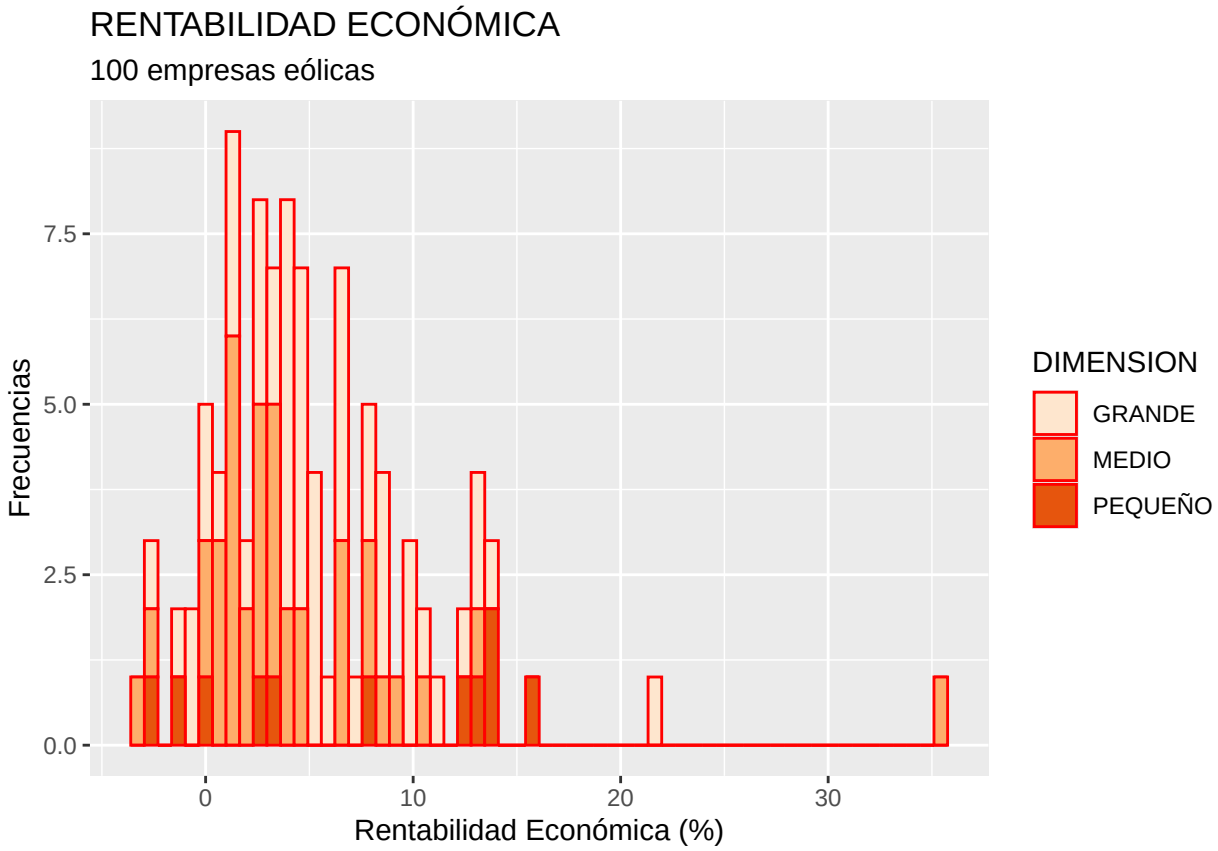
A continuación, vamos a modificar el color de las barras. Para el borde de estas, se utiliza el argumento **"colour ="**; y, para el relleno, **"fill ="**. Además, vamos a mejorar la presentación del gráfico añadiéndole un título y un subtítulo, y unas etiquetas en los ejes. Hay que prestar atención a los signos **"+"** incluidos para que R entienda que el código de la siguiente línea pertenece al mismo gráfico que estamos diseñando:

```
ggplot(data = eolica_100, map = aes(x = RENECO)) +
  geom_histogram(bins = 40, colour = "red", fill = "orange") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas")+
  xlab("Rentabilidad Económica (%)") +
  ylab("Frecuencias")
```



Puede ser que nos interese diferenciar los casos según grupos preestablecidos. Por ejemplo, entre las variables de nuestro *data frame* "eolica_100", existe una variable categórica, atributo o factor, denominada DIMENSION, que clasifica a las 100 empresas en "GRANDE", "MEDIO" o "PEQUEÑO" atendiendo al tamaño del grupo empresarial al que pertenecen (medido en número de empresas). Lo que vamos a hacer es crear, en el mismo gráfico, un histograma de la rentabilidad económica, pero para cada categoría de DIMENSION. Para ello, habrá que incluir esta variable categórica en el "mapeo", en concreto mediante el argumento "fill =". **Es necesario hacerlo dentro del elemento *aes***, ya que el resultado (color de grupo de empresas) depende del valor que toma la variable DIMENSION para cada caso o empresa:

```
ggplot(data = eolica_100, map = aes(x = RENECO, fill = DIMENSION)) +
  geom_histogram(bins = 60, colour = "red") +
  scale_fill_brewer(palette = "Oranges") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas")+
  xlab("Rentabilidad Económica (%)") +
  ylab("Frecuencias")
```

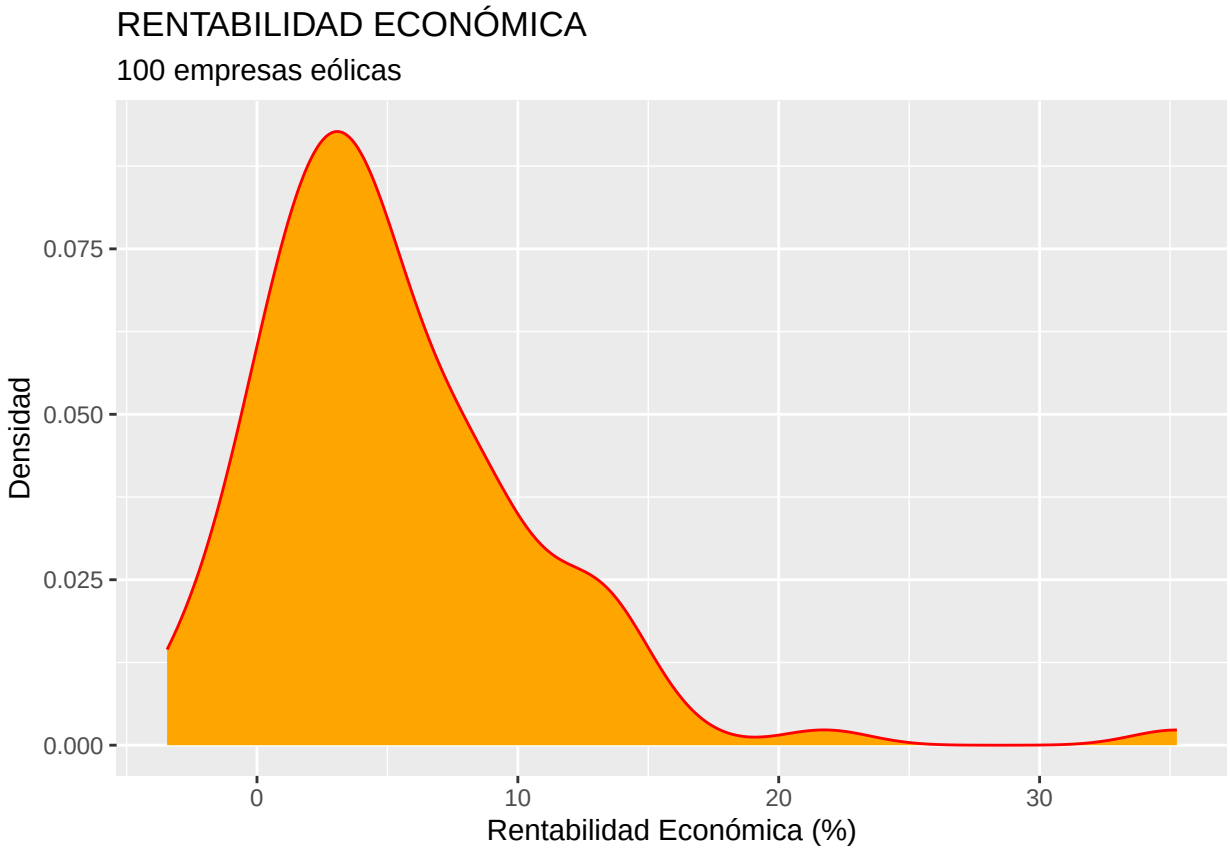


Como puede comprobarse, se superponen los tres histogramas, con tres colores diferentes, dependiendo de la dimensión considerada. Además, aparece, al lado derecho del gráfico, una leyenda que detalla qué color se asocia a cada uno de los grupos de empresas. La función `scale_fill_brewer()` nos permite personalizar la paleta de colores a utilizar (para ver las paletas disponibles, podemos consultar esta sección de (Wickham 2021).

3.2.2 Gráfico de densidad.

Un gráfico parecido al histograma es el de **densidad**. Un gráfico de densidad estima la función de probabilidad empírica de la variable representada. En realidad, podemos considerarlo como un histograma "suavizado". Probemos a ejecutar este código:

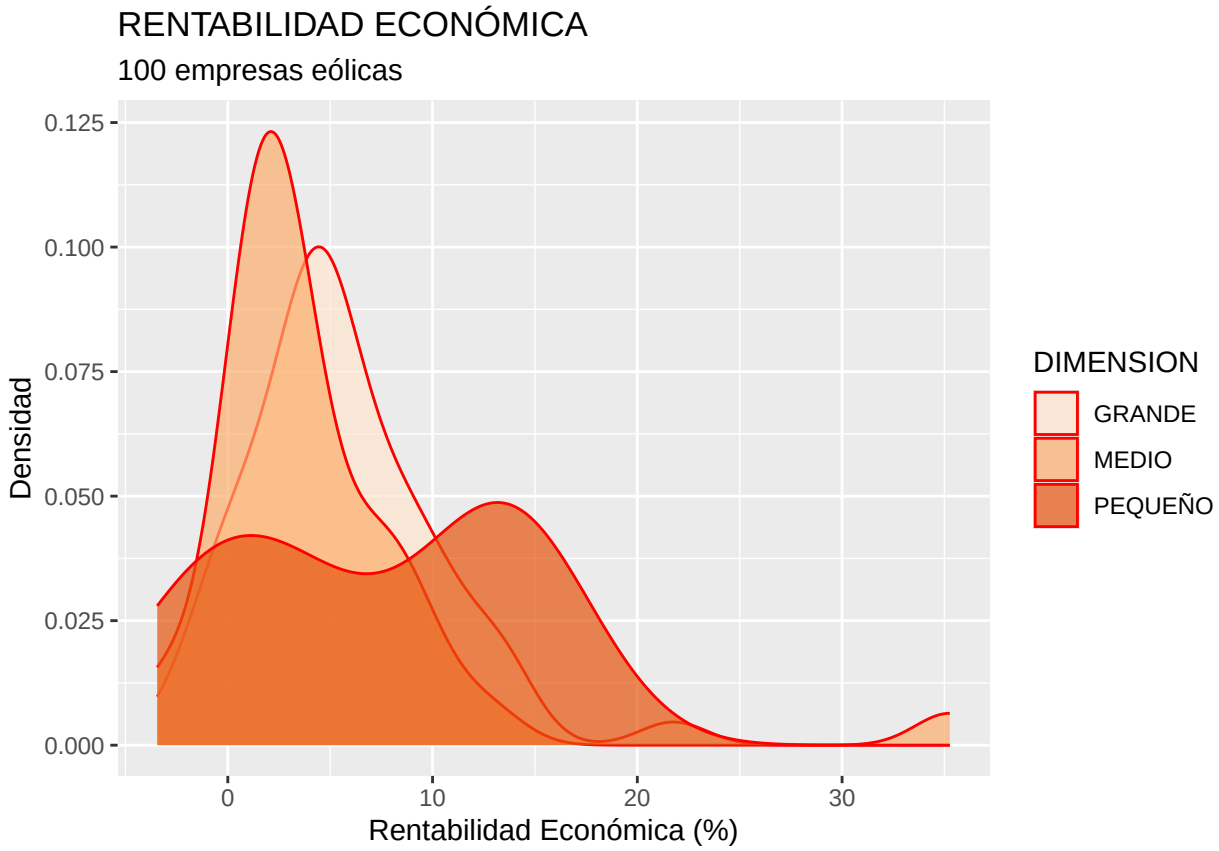
```
ggplot(data = eolica_100, map = aes(x = RENECO)) +
  geom_density(colour = "red", fill = "orange") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas")+
  xlab("Rentabilidad Económica (%)") +
  ylab("Densidad")
```



En el código se observa la utilización del tipo de gráfico `geom_density()`. Además, desaparece el número de intervalos o *bins*, y se puede dotar a la función de densidad estimada de un color en su borde (`colour=`), y de un color de relleno (`fill=`).

Como en casos anteriores, se puede crear una función de densidad estimada para cada grupo de empresas, eliminando las características, `colour=` y `fill=` del bloque del *geom*, y añadiéndolas en el "mapeo", dentro del `aes()`:

```
ggplot(data = eolica_100, map = aes(x = RENECO, fill = DIMENSION)) +
  geom_density(colour = "red", alpha = 0.70, ) +
  scale_fill_brewer(palette = "Oranges") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas")+
  xlab("Rentabilidad Económica (%)") +
  ylab("Densidad")
```



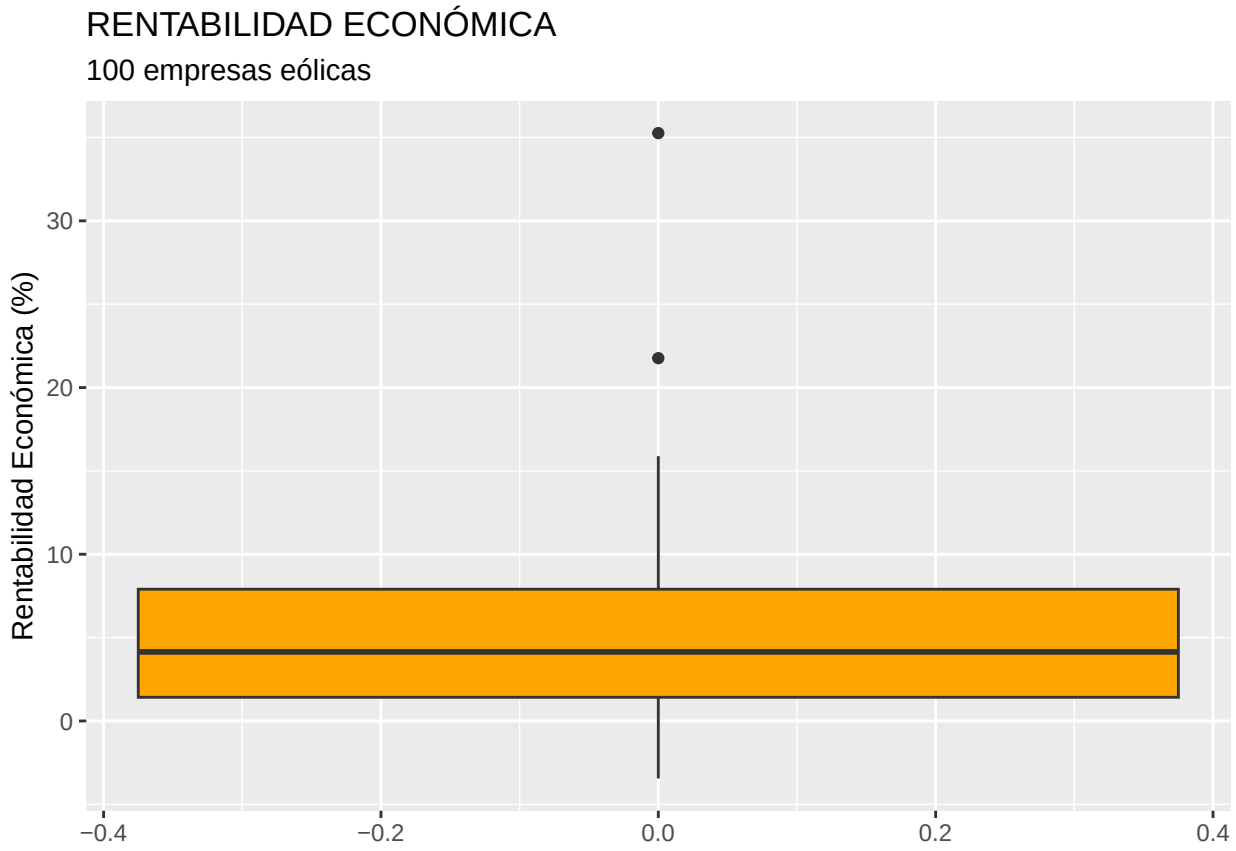
En efecto, el argumento `fill=` ha pasado a integrarse, en el "mapeo", dentro de un elemento `aes`, ya que el color de relleno va a variar dependiendo del grupo de pertenencia de la empresa (variable DIMENSION). Por otro lado, en el `geom` se ha añadido el argumento `alpha=`. Esta información consiste en un número de 0 a 1 que gradúa el grado de transparencia / opacidad de los rellenos de las figuras (en este caso las funciones de densidad estimadas) incluidas en los gráficos.

3.2.3 Gráfico de caja o *Box-Plot.*

Un tipo muy interesante de gráfico es el de "caja" (***box-plot***), que informa de la dispersión de una variable. Fijémonos en el siguiente código:

```
ggplot(data = eolica_100, map = (aes(y = RENECO))) +
  geom_boxplot(fill= "orange") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas") +
  ylab("Rentabilidad Económica (%)")
```

Puede observarse cómo en el "mapeo" se fija la variable que va a determinar las coordenadas del eje "y". Como es una variable, hay que incluirla en el "mapeo" mediante una característica `aes`. El `geom` o tipo de gráfico es `geom_boxplot()`, y en este caso no le hemos añadido ninguna característica específica. Las últimas líneas configuran los títulos del gráfico y del eje "y". El resultado de ejecutar el código es el siguiente gráfico:



El gráfico se caracteriza por una "caja" (rectángulo) central. Esta caja está limitada por el primer y tercer cuartil, luego recoge el 50% de los casos con una rentabilidad económica superior al 25% de los casos con menor rentabilidad, y por debajo del 25% de los casos con la rentabilidad más alta. Así, la altura de la caja es la diferencia entre los cuartiles tercero y primero, que es lo que se denomina "rango intercuartílico" (*IQR* por las siglas en inglés). La caja, a su vez, está dividida en dos zonas por una línea horizontal, que es la mediana de la distribución: la rentabilidad económica que divide a los casos en dos grupos con el mismo número de casos, uno con los casos de mayor rentabilidad, y otro con los casos de menor rentabilidad.

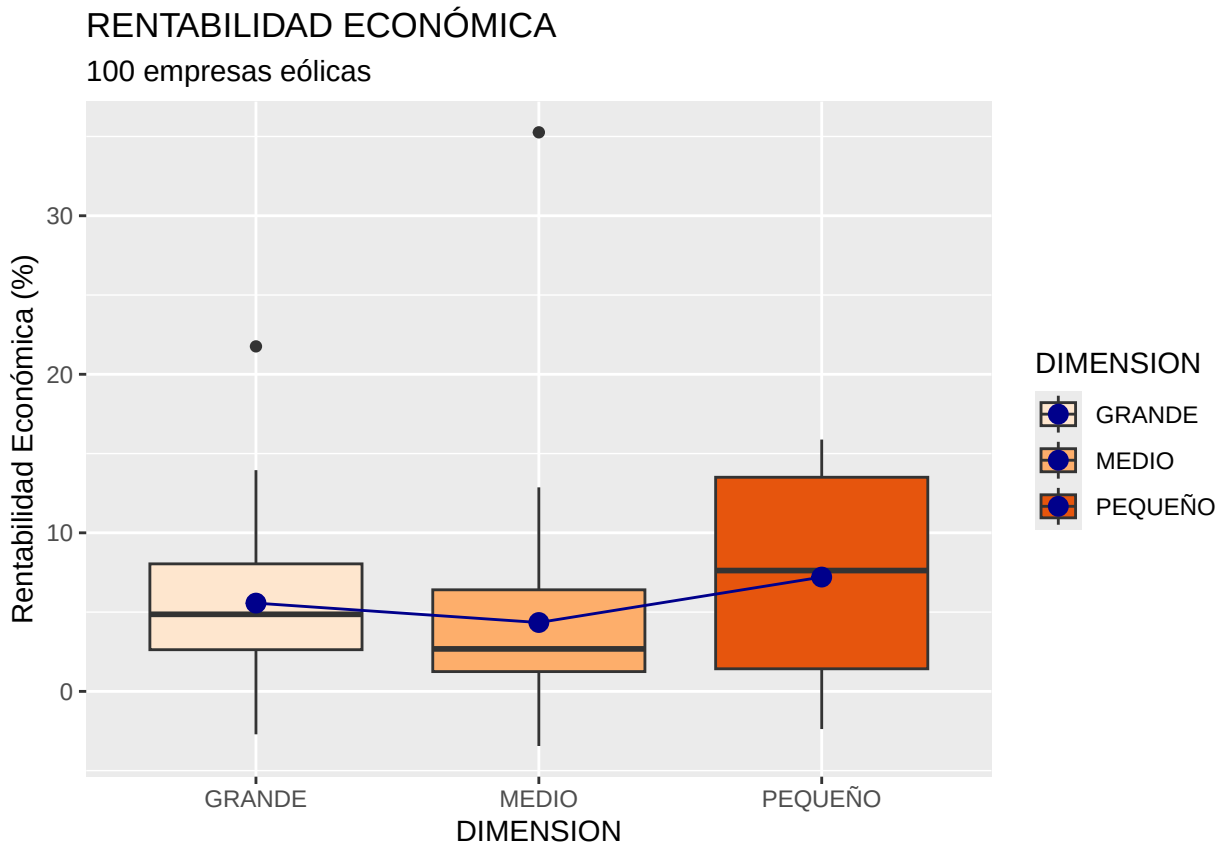
Por encima y por debajo de la caja se disponen dos segmentos (llamados "bigotes"). Estos "bigotes" recogen los casos con valores en la variable inferiores al primer cuartil (comenzando por la base de la caja, hacia abajo), o superiores al tercer cuartil (comenzando por el techo de la caja, hacia arriba); y que están a menos de **1.5 veces** la altura de la caja. Los casos con valores de rentabilidad inferiores al primer cuartil (por abajo) y superiores al tercero (por arriba), que están alejados de la caja en más de 1.5 veces la altura de esta, se indican con puntos, y se corresponden con los casos conocidos como **casos atípicos** o ***outliers***. La identificación de los *outliers* es una fase muy importante a la hora de aplicar algunas técnicas estadísticas.

En esta práctica, comprobamos cómo, en el caso de la rentabilidad económica (RENECO), existen dos *outliers*, es decir, dos casos que presentan sendas rentabilidades anormalmente elevadas (más de un 20%).

Vamos a refinar el *box-plot* anterior. Por ejemplo, quizá nos pueda interesar crear un *box-plot* para cada grupo de empresas, según el tamaño del grupo empresarial de pertenencia (atributo DIMENSION). Esto lo conseguiremos con el código:

```
ggplot(data = eolica_100, map = (aes(x = DIMENSION, y = RENECO, fill = DIMENSION))) +
  geom_boxplot() +
  stat_summary(fun = "mean",
               geom = "point",
               size = 3,
```

```
              col = "darkblue") +
  stat_summary(fun = "mean",
               geom = "line",
               col = "darkblue",
               map = (aes(group = TRUE))) +
  scale_fill_brewer(palette = "Oranges") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas") +
  ylab("Rentabilidad Económica (%)")
```



Para construir una caja por categoría de la variable cualitativa o atributo DIMENSION, se ha incluido, en el "mapeo" de la primera línea, el eje x con la variable tal variable DIMENSION. Como, además, queremos que cada caja sea de un color diferente, hemos hecho que los colores de estas dependan de la variable DIMENSION; añadiendo en el `aes()` del "mapeo" la característica `fill=` (que se refiere al color de relleno de las cajas).

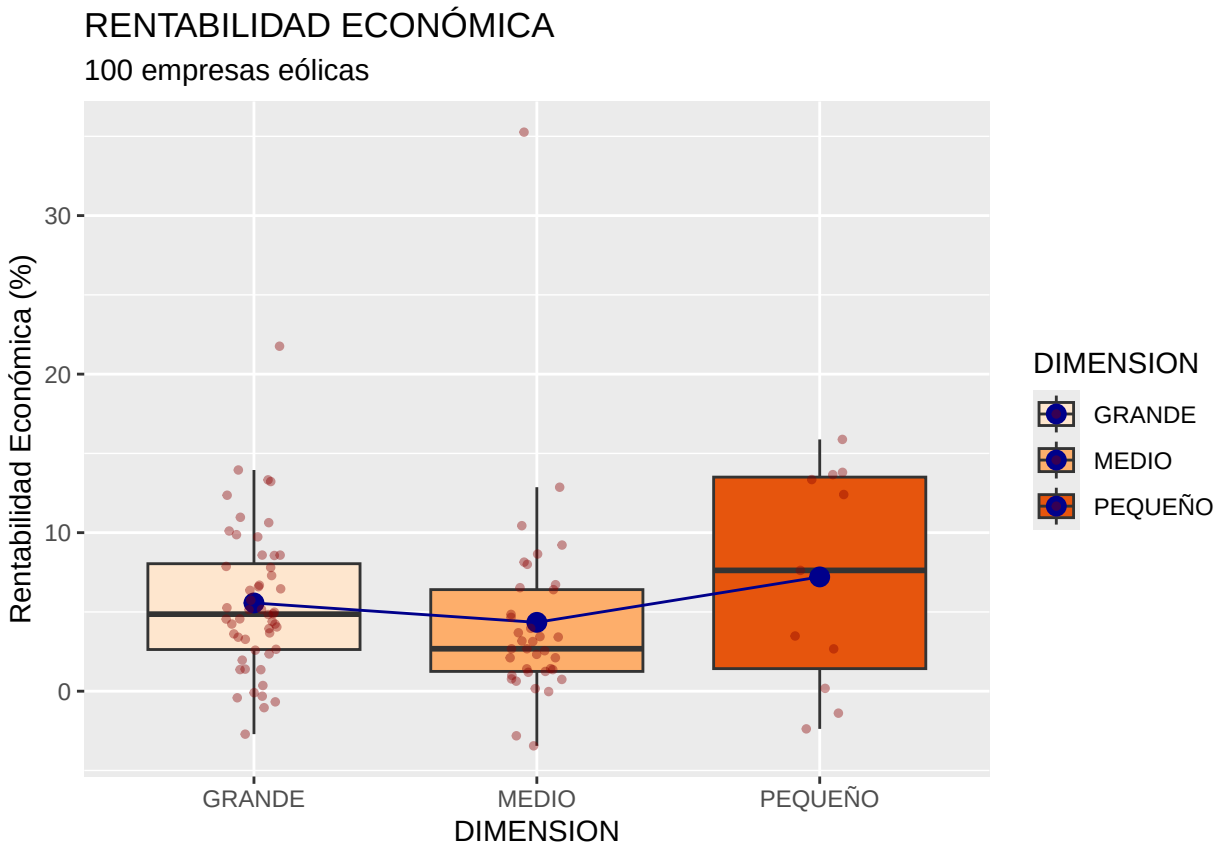
Dentro de las cajas, se representan las medianas de la variable estudiada (RENECO), para cada grupo de empresas. Pero, a veces, es muy útil representar también otra medida de posición central como es la **media**. En los gráficos diseñados con `{ggplot2}` se pueden añadir medidas estadísticas mediante elementos `stat_summary()`, algo parecido al `summarise()` de `{dplyr}`. Estos elementos contienen una serie de informaciones, como son `fun =` (la función, medida, o estadístico a representar), un `geom` (tipo de gráfico), y otros elementos opcionales. En el ejemplo, el primer bloque de `stat_summary()` consigue puntear, para cada grupo de empresas, la media de RENECO en dicho grupo, en color azul oscuro. Para comparar mejor estas medias, se ha procedido a unir los puntos con unos segmentos o líneas de color azul oscuro, lo que se consigue con el segundo bloque de `stat_summary()`. La última línea de ese bloque, `map = (aes(group = TRUE))`, obliga a que las líneas vayan de una media a otra de los grupos (de punto azul oscuro a punto azul oscuro).

Como última extensión, se ha considerado que, a veces, es conveniente tener en cuenta la posición de cada caso individual dentro del gráfico. Una opción es utilizar una capa o bloque `geom_jitter()`. Con este `geom`

se dispondrán, para cada grupo, los valores individuales de la variable RENECO; y para que estos, en su caso, no se solapen, se situarán un poco más a la izquierda o a la derecha, de modo aleatorio. Como los *outliers* son ya casos individuales, para que no se dupliquen con los provenientes del "jitter", se indicará en el `geom_boxplot()` que, en ese bloque gráfico, no se señalen los *outliers*. Esto se conseguirá con el argumento `outlier.shape = NA`. El código, en definitiva, será:

```
ggplot(data = eolica_100, map = (aes(x = DIMENSION, y = RENECO, fill = DIMENSION))) +
  geom_boxplot(outlier.shape = NA) +
  stat_summary(fun = "mean",
               geom = "point",
               size = 3,
               col = "darkblue") +
  stat_summary(fun = "mean",
               geom = "line",
               col = "darkblue",
               map = (aes(group = TRUE))) +
    geom_jitter(width = 0.1,
              size = 1,
              col = "darkred",
              alpha = 0.40) +

  scale_fill_brewer(palette = "Oranges") +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas") +
  ylab("Rentabilidad Económica (%)")
```



Como puede observarse, el `geom_jitter()` proporciona, en cada caja, la nube de casos (empresas) individuales, en cuanto a la rentabilidad económica (incluidos los *outliers*). Las características de estos puntos

(amplitud del desplazamiento lateral "aleatorio", tamaño, color, opacidad) se controlan con diversos argumentos (`width=`, `size=`, `col=`, `alpha=`).

3.3 Gráficos de dos variables.

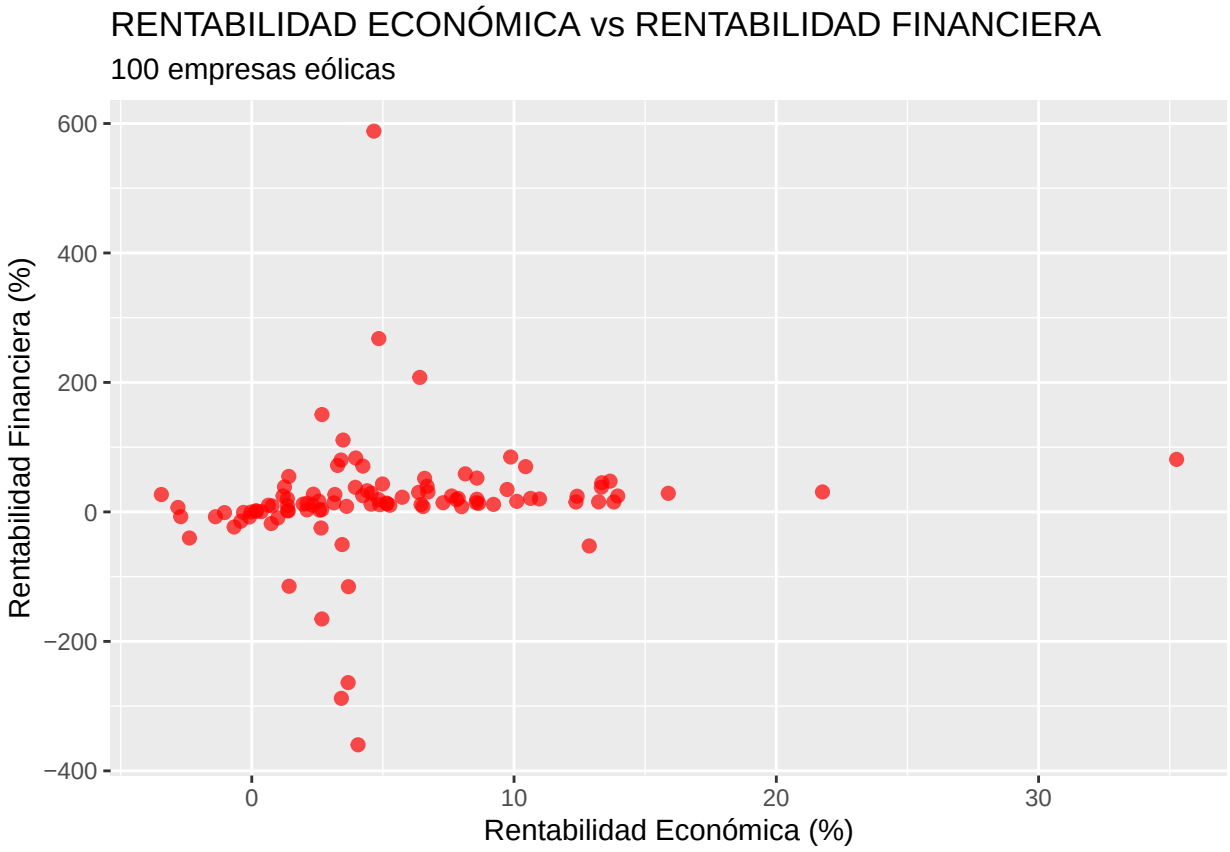
3.3.1 Gráfico de dispersión o *scatterplot*.

Pasamos ahora a comentar un tipo de gráfico muy común cuando trabajamos con dos variables métricas: los **gráficos de dispersión** (o ***scatterplots***). En este tipo de gráficos, cada variable ocupa un eje (x o y), y los puntos internos al gráfico representan los diversos casos u observaciones.

Como ejemplo, vamos a crear un gráfico de dispersión que represente las empresas eólicas en función de su rentabilidad económica (RENECO) y de su rentabilidad financiera (RENFIN). El código es el siguiente:

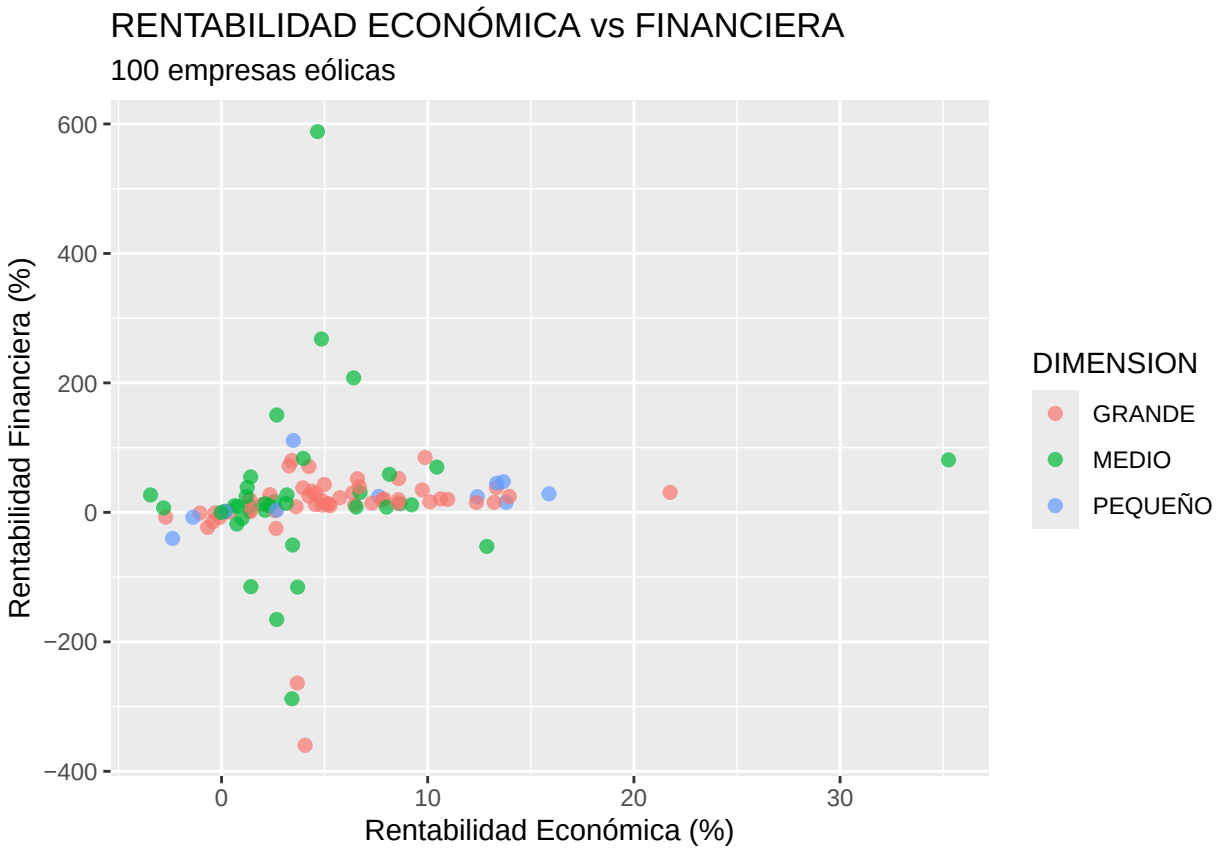
```
ggplot(data = eolica_100, map = (aes(x = RENECO, y = RENFIN))) +
  geom_point(color = "red", size = 2, alpha = 0.7) +
  ggtitle("RENTABILIDAD ECONÓMICA vs RENTABILIDAD FINANCIERA", subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Rentabilidad Financiera (%)")
```

El resultado es el siguiente gráfico:



Vamos a refinar el gráfico algo más. En primer lugar, puede ser interesante **distinguir entre los tipos de empresas**, según el tamaño del grupo empresarial al que pertenecen (variable DIMENSION). Para ello, podemos poner el color de los puntos en el "mapeo", en función de la variable DIMENSION:

```
ggplot(data = eolica_100, map = (aes(x = RENECO,
                                     y = RENFIN,
                                     col = DIMENSION))) +
  geom_point(size = 2, alpha = 0.7) +
  ggtitle("RENTABILIDAD ECONÓMICA vs FINANCIERA",
          subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Rentabilidad Financiera (%)")
```



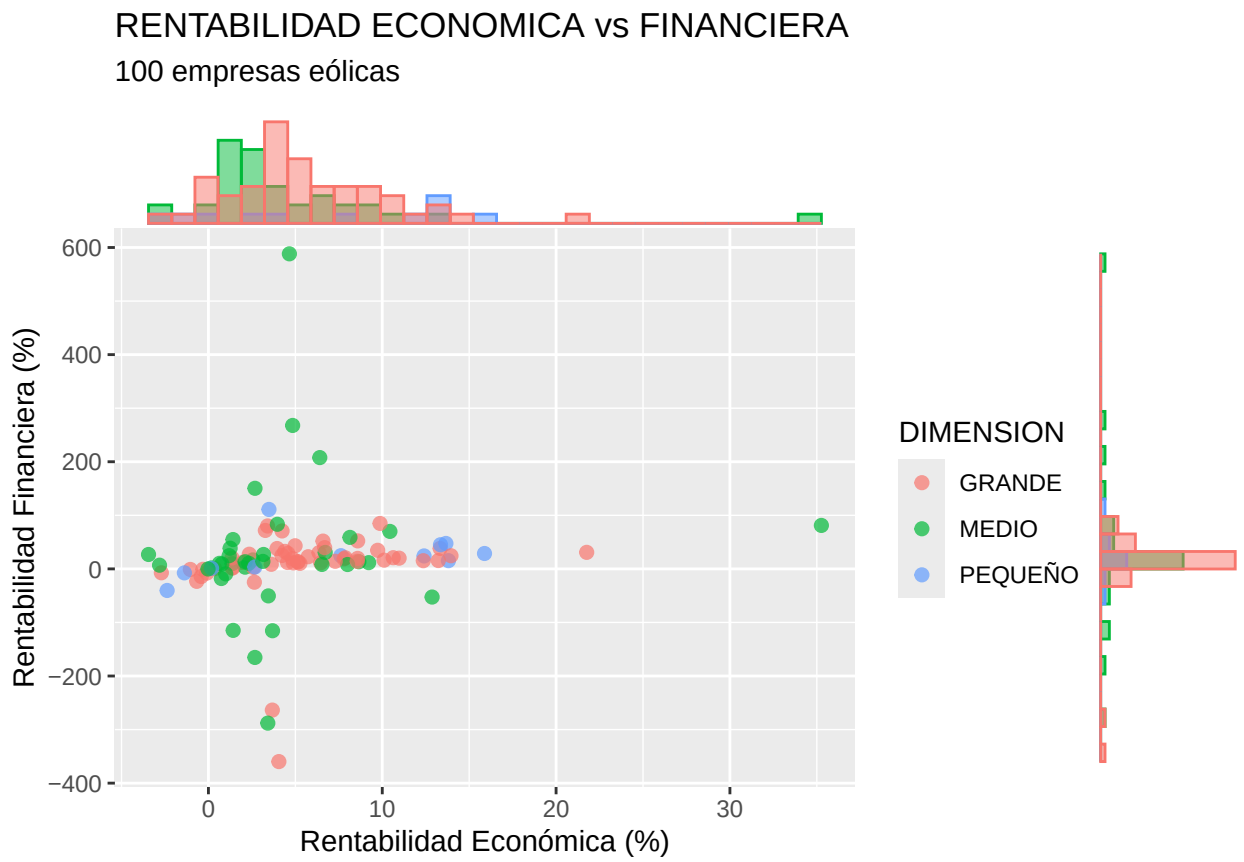
En los dos gráficos anteriores pueden observarse puntos (casos) candidatos a ser *outliers* para cada una de las dos variables analizadas. En el caso de RENECO, ya se pudo advertir esta circunstancia al construir los *boxplots*.

Por otro lado, podría ser interesante complementar el gráfico con información sobre las dos variables por separado, es decir, con **información sobre las distribuciones marginales**. Existe un paquete complementario a `{ggplot2}`, llamado `{ggExtra}`, que puede ayudar fácilmente a este cometido. Para ello, hemos de activar dicho paquete con `library()` (si no ha sido previamente instalado, habrá que hacerlo con anterioridad). El segundo paso consistirá en **asignar** nuestro *scatterplot*, diseñado con la función `ggplot()`, a un objeto con el nombre que queramos, por ejemplo, "scatter_plus". Luego, ese objeto, que contiene nuestro gráfico, entrará como argumento en la función de `{ggExtra}` llamada `ggMarginal()`, como se muestra en el siguiente código:

```
library ("ggExtra")
scatter_plus <- ggplot(data = eolica_100, map = (aes(x = RENECO,
                                                     y = RENFIN,
                                                     col = DIMENSION))) +
  geom_point(size = 2, alpha = 0.7) +
```

```
  ggtitle("RENTABILIDAD ECONÓMICA vs FINANCIERA",
          subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Rentabilidad Financiera (%)")
ggMarginal(scatter_plus, type = "histogram", groupColour = T,
           groupFill = T, position = "identity", alpha = 0.5)
```

Con el código anterior, apreciamos cómo se añaden los histogramas de cada variable, RENECO y RENFIN, en los márgenes del gráfico:

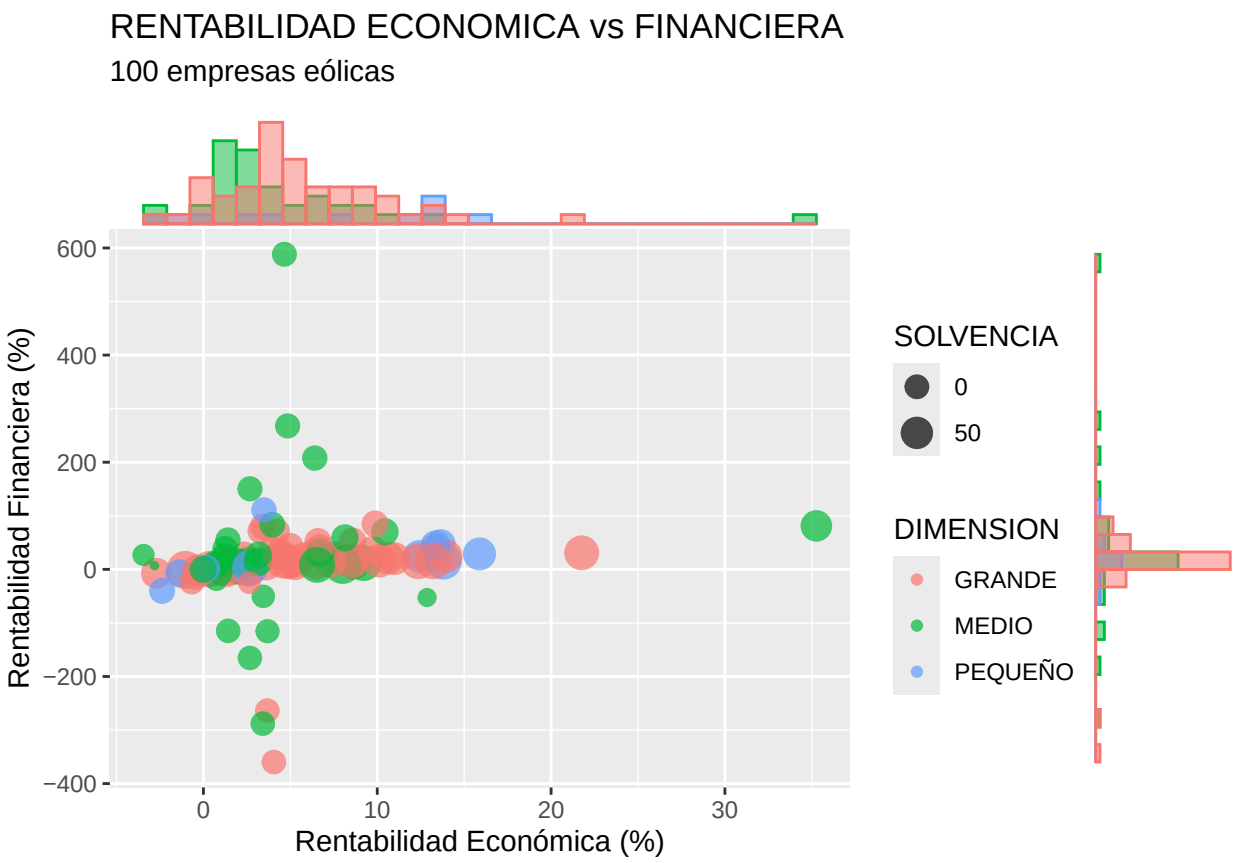


Conviene apuntar que el argumento position = "identity" hace que las barras del histograma estén perfectamente alineadas con los datos del gráfico de dispersión, sin ningún tipo de desplazamiento.

Adicionalmente, los diámetros de los puntos de los diversos casos podrían contener también información, haciéndolos proporcionales a una tercera variable. Por ejemplo, podrían ser proporcionales al nivel de solvencia (variable SOLVENCIA). Para ello, ejecutaríamos el código:

```
scatter_plus <- ggplot(data = eolica_100, map = (aes(x = RENECO,
                                                     y = RENFIN,
                                                     col = DIMENSION,
                                                     size = SOLVENCIA))) +
  geom_point(alpha = 0.7) +
  ggtitle("RENTABILIDAD ECONÓMICA vs FINANCIERA",
          subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Rentabilidad Financiera (%)")
```

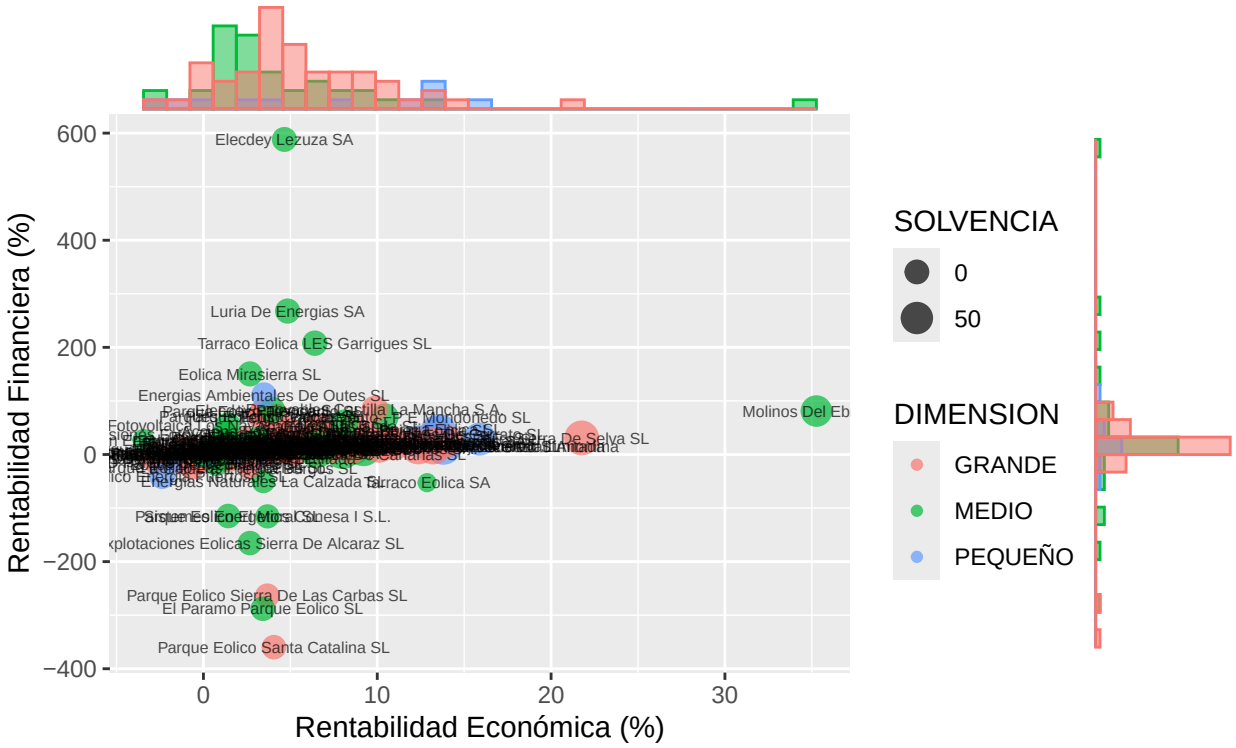
```
ggMarginal(scatter_plus, type = "histogram", groupColour = T,
           groupFill = T, position = "identity", alpha = 0.5)
```



En el código anterior, puede comprobarse que la característica `size =` sube del bloque de `geom` al "mapeo" (incluido en el `aes`), debido a que el diámetro de cada punto ya no va a ser un parámetro fijo, sino que va a depender de la magnitud de la variable SOLVENCIA.

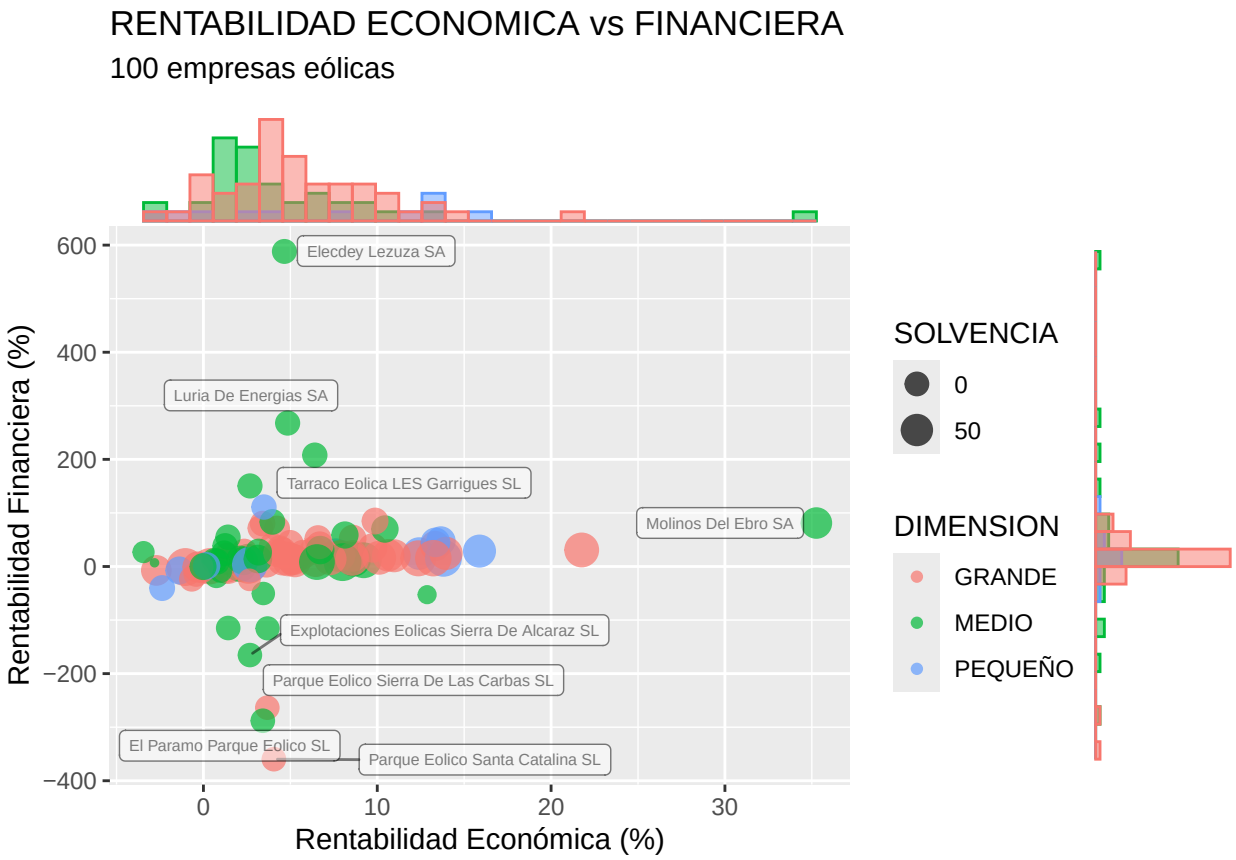
Finalmente, podría ser útil, en algunos gráficos, añadir una etiqueta (*label*) a cada punto, para **identificar el caso concreto** al que representa. Si bien en esta práctica, el elevado número de casos y el extenso nombre de las empresas hacen poco claro el uso de estas etiquetas, vamos a añadirlas por motivos pedagógicos. Para ello, se añadirá un bloque `geom` llamado `geom_text()`, con una información `label =` que se hace depender de valores que cambian (en concreto, el nombre de los casos, es decir, de las filas del *data frame*), por lo que tendrá que integrarse en una característica `aes`:

```
scatter_plus <- ggplot(data = eolica_100, map = (aes(x = RENECO,
                                                     y = RENFIN,
                                                     col = DIMENSION,
                                                     size = SOLVENCIA))) +
 geom_point(alpha = 0.7) +
  geom_text(aes(label=row.names(eolica_100)), size=2, color="black", alpha = 0.7) +
  ggtitle("RENTABILIDAD ECONÓMICA vs FINANCIERA",
          subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Rentabilidad Financiera (%)")
ggMarginal(scatter_plus, type = "histogram", groupColour = T,
           groupFill = T, position = "identity", alpha = 0.5)
```



Las etiquetas de los casos pueden refinarse algo más mediante la función `geom_label_repel()`, disponible al cargar el paquete `{ggrepel}`:

```
library(ggrepel)
scatter_plus <- ggplot(data = eolica_100, map = (aes(x = RENECO,
                                                     y = RENFIN,
                                                     col = DIMENSION,
                                                     size = SOLVENCIA))) +
  geom_point(alpha = 0.7) +
  geom_label_repel(aes(label = row.names(eolica_100)),
                   size = 2,
                   color = "black",
                   alpha = 0.5) +
  ggtitle("RENTABILIDAD ECONÓMICA vs FINANCIERA",
          subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Rentabilidad Financiera (%)")
ggMarginal(scatter_plus, type = "histogram", groupColour = T,
           groupFill = T,
           position = "identity", alpha = 0.5)
```



La ventaja de este gráfico, como se puede apreciar, es que se omiten las etiquetas superpuestas, si bien existe el riesgo de que se omitan una gran cantidad de estas.

3.4 Materiales para realizar las prácticas del capítulo.

En esta sección se muestran los links de acceso a los diferentes materiales (scripts, datos…) necesarios para llevar a cabo los contenidos prácticos del capítulo.

Datos (en formato Microsoft (R) Excel (R)):

- eolica_100.xlsx (obtener aquí)

Scripts:

- explora_ggplot2.R (obtener aquí)

Capítulo 4

Estadística descriptiva.

La Estadística Descriptiva es la parte de la Ciencia Estadística que se ocupa de la recopilación de datos, su depuración, y la caracterización mediante dichos datos de un conjunto de casos o individuos.

Los datos se organizan en variables y/o atributos.

Las **variables** son características de los casos o individuos en estudio que se plasman en valores que están expresados en **escala métrica**. Los **atributos** son características de los casos o individuos en estudio que se concretan en diversas categorías (si el atributo tiene **escala nominal**) o niveles (si el atributo tiene **escala ordinal**). Los atributos se denominan también variables cualitativas o factores.

Centrándonos en las variables (características que afectan a un grupo de casos o individuos, y que se concretan en valores que poseen una escala métrica), podemos plantearnos el estudio de una única variable sin tener en cuenta la existencia de otras variables que caracterizan al mismo grupo de casos o individuos. En tal caso estaremos planteando un **análisis estadístico univariante**. Si nuestro análisis se centra en cómo dos variables caracterizan al mismo conjunto de individuos o casos, y la posible relación entre ambas, estaremos planteando un análisis bivariante. Generalizando, si estudiamos cómo un grupo de variables caracterizan de modo conjunto a un mismo grupo de casos o individuos, estaremos planteando un **análisis estadístico multivariante**.

4.1 Análisis univariante.

En el análisis estadístico univariante, estudiamos cómo una única característica (nos centraremos en una variable, aunque también puede tratarse de un atributo) afecta a un grupo de casos, individuos o elementos. Por ejemplo, la variable podría ser el salario percibido por un grupo de individuos que podría ser el conjunto de trabajadores en nómina en una empresa. Otro ejemplo podría ser el de la (variable) rentabilidad económica obtenida por un grupo de empresas pertenecientes a un determinado sector económico.

El conjunto de pares formado por cada valor que puede tomar la variable en estudio (o categoría o nivel, en el caso de un atributo) y el número de casos que toman tal valor se denomina **distribución de frecuencias** de la variable.

¿Cómo podemos estudiar el modo en que afecta una variable, de modo global, a un grupo de casos? Mediante el cálculo de una serie de **medidas**. Las medidas son instrumentos matemáticos que extraen y sintetizan la información contenida en una distribución de frecuencias.

Hay diferentes tipos de medidas, principalmente las de **posición**, **dispersión** y **forma**.

Antes de profundizar en las principales medidas, su significado y su obtención; vamos a indicar cómo se pueden presentar en R, mediante la creación de tablas, los datos referentes a un grupo de individuos y las variables o atributos que los caracterizan, y las distribuciones de frecuencias univariantes.

4.2 Representando datos y distribuciones de frecuencias en tablas con R.

Para aprender a representar los datos referentes a las variables y atributos que caracterizan a un grupo de casos o individuos, y las distribuciones de frecuencias univariantes, vamos a suponer que trabajamos dentro de un proyecto que hemos creado previamente, de nombre "explora". Dentro de la carpeta del proyecto guardaremos el *script* llamado "explora_trabajadores.R" y el archivo de *Microsoft® Excel®* llamado "trabajadores.xlsx".

Las primeras líneas del script se refieren, como ya hemos visto en otras secciones del libro, a la limpieza de la memoria o environment, eliminando objetos que se hayan podido crear con anterioridad, y en la importación de los datos que hay en la hoja "Datos" de "trabajadores.xlsx". Estos datos se almacenan en el *data frame* "datos", y consisten en el registro del salario (variable SALARIO, expresada en cientos de euros), el nivel de estudios (atributo NESTUDIOS) y departamento al que se pertenece (atributo DEP), correspondientes a los 49 trabajadores de una determinada empresa:

```
# Script para la construcción de tablas de datos
# y trabajo con distribucionesde frecuencias univariantes.
#

rm(list = ls())

## DATOS

# Importando

library(readxl)
datos <- read_excel("trabajadores.xlsx", sheet = "Datos")
```

Como sabemos, R interpreta el nombre de los trabajadores como una variable más, en lugar de como la identificador de cada "fila" o caso. Para corregir esto, y hacer saber a R que la primera columna no es una variable, sino el nombre de cada fila (caso, en este caso trabajador), añadimos la línea:

```
datos <- data.frame(datos, row.names = 1)
```

Posteriormente, podremos comprobar que "datos" contiene el valor del salario, el nivel de estudios y la categoría de departamento para cada uno de los 49 trabajadores de la empresa:

```
summary (datos)
```

```
##     SALARIO       NESTUDIOS            DEP
##  Min.   : 8.00   Length:49          Length:49
##  1st Qu.:12.00   Class :character   Class :character
##  Median :15.00   Mode  :character   Mode  :character
##  Mean   :16.04
##  3rd Qu.:20.00
##  Max.   :30.00
```

Sabemos que, simplemente escribiendo el nombre del *data frame*, aparecerán en la consola los datos almacenados en él. No obstante, esta presentación no es muy elegante para presentar los datos. Vamos a presentar tales datos de un modo más amigable, mediante la **confección de una "tabla"**.

Un paquete de R muy popular para generar tablas de datos es `{knitr}`. Este paquete contiene la función `kable()`, que permite generar tablas en varios formatos y con diversas características que pueden ser personalizadas (como el título de la tabla). Si queremos personalizar más aún la apariencia de nuestras tablas,

podemos usar las facilidades del paquete `{kableExtra}`, que complementa las posibilidades que ofrece la función `kable()` de `{knitr}`.

Para hacer una tabla con nuestros casos y variables, es decir, para escribir nuestro *data frame* "datos" de un modo más elegante, primero activaremos los paquetes anteriores, y añadiremos una línea donde diremos el formato de la tabla a generar (en nuestro ejemplo, formato .html), todo con el siguiente código:

```
# Tabla de datos

library (knitr)
library (kableExtra)
```

Después, generaremos nuestra tabla con los datos contenidos en el *data frame* "datos". El código para generar la tabla, y el resultado, es el siguiente:

```
  library("magick")
  datos %>%
        kable(caption = "Trabajadores asalariados de la empresa",
              "latex",
              col.names = c("Trabajador", "Salario", "Nivel de estudios",
                            "Departamento"))  %>%
#  kable_styling(full_width = F, bootstrap_options = "striped", "bordered",
#                "condensed", position = "center", font_size = 11) %>%
  row_spec(0, bold= T, align = "c") %>%
  row_spec(1:(nrow(datos)), bold= F, align = "c") #%>%
  #as_image()
```

Trabajadores asalariados de la empresa

Trabajador

Salario

Nivel de estudios

Departamento

Andrés

8

Básicos

Almacén

Ángela

8

Medios

Almacén

Miguel

9

Básicos

Almacén

Luis

9

Básicos

Almacén

María

9

Medios

Gestión Interna

Lourdes

10

Medios

Gestión Interna

Carlos

10

Medios

Gestión Interna

Adriana

10

Medios

Gestión Interna

Ricardo

10

Medios

Gestión Proveedores

Juan José

10

Medios

Gestión Proveedores

Daniel

10

Medios

Gestión Proveedores

Pedro

12

Básicos

Almacén

Ana

12

Básicos

Almacén

Isabel

12

Básicos

Almacén

Manuel

12

Medios

Gestión Proveedores

Isidro

12

Medios

Gestión Proveedores

Carla

12

Medios

Gestión Proveedores

Fernando

12

Universitarios

Gestión Clientes

Belén

12

Universitarios

Marketing

Margarita

15

Básicos

Almacén

Andrea

15

Básicos

Almacén

Pablo

15

Medios

Gestión Interna

Celia

15

Medios

Gestión Interna

Alba

15

Medios

Gestión Interna

Nicolás

15

Medios

Gestión Proveedores

David

15

Medios

Gestión Proveedores

Elena

15

Medios

Gestión Proveedores

Victoria

15

Medios

Gestión Proveedores

Antonio

15

Universitarios

Gestión Clientes

Tomás

15

Universitarios

Gestión Clientes

Bartolomé

20

Universitarios

Almacén

Irene

20

Universitarios

Gestión Proveedores

Guadalupe

20

Universitarios

Gestión Proveedores

Ignacio

20

Universitarios

Gestión Proveedores

Ernesto

20

Universitarios

Marketing

Abel

20

Universitarios

Gestión Clientes

Nieves

20

Universitarios

Gestión Clientes

Carolina

20

Universitarios

Gestión Clientes

Luisa

22

Universitarios

Gestión Clientes

Alberto

22

Universitarios

Marketing

Paula

22

Universitarios

Almacén

Sergio

22

Universitarios

Gestión Proveedores

Estrella

22

Universitarios

Gestión Interna

Alicia

22

Universitarios

Coordinación

Marta

25

Universitarios

Dirección

Alfonso

25

Universitarios

Dirección

Mar

25

Universitarios

Dirección

Martín

25

Universitarios

Dirección

Blanca

30

Universitarios

Dirección

Primero llamamos al *data frame* a partir de cuyos datos vamos a generar la tabla, "datos". Con el operador *pipe* `%>%`, ligamos los datos del *data frame* al diseño la tabla realizado con la función `kable()` de `{knitr}`. `kable()` tiene diversos argumentos, entre los que destacan:

- `caption =`: Este argumento informa del título de la tabla.
- `col.names =`: Este argumento, opcional, fija el nombre para las columnas de la tabla, si no queremos que aparezcan los nombres "por defecto", que son los nombres de cada columna en el propio *data frame*.

Luego, con el operador *pipe* `%>%` informamos de que vamos a completar o personalizar el diseño de esta tabla con otras funciones complementarias del paquete `{kableExtra}`. En primer lugar, utilizamos la función `kable_styling()`, que aporta algunas características adicionales a la tabla, según sus argumentos:

- `full_width =` : este argumento ha de tener un valor lógico, y se refiere a si deseamos que la tabla ocupe todo el ancho del documento (TRUE) o solo lo necesario (FALSE).
- `bootstrap_options =` : este argumento es de tipo alfanumérico, y sirve para fijar ciertas características estéticas complementarias. "striped" se refiere a que las filas aparezcan sombreadas de modo alternativo, "bordered" se refiere a que cada fila quede delimitada por unas finas líneas en la parte superior y en la inferior, "condensed" significa que la tabla tendrá un aspecto más compacto.
- `position =` : este argumento se utiliza para situar la tabla centrada, a la izquierda del párrafo, o a la derecha.
- `font_size =` : este argumento numérico se refiere al tamaño de los caracteres, lo cuál es importante a la hora de que una tabla "quepa" en un documento de deteminada anchura.

Por último, hacemos uso dos veces de la función `row_spec()` del paquete `{kableExtra}`. Esta función sirve para personalizar algo más las filas concretas de la tabla que consideremos. El encabezado se identifica como la fila "0". En el ejemplo, se ha utilizado esta función dos veces: una para el encabezado (el primer argumento de la función nos informa de las filas a las que se refiere, en esta ocasión la fila 0), y otra para el resto de filas (desde la fila 1 hasta la que contiene al último caso individuo, la fila con posición `nrow(datos)`). Los otros argumentos definen si se quiere que los caracteres aparezcan en negrita (`bold =`) y cómo deben estar alineados los elementos, dentro de las columnas (`align =`).

A veces, puede ocurrir que solo nos interese estudiar una variable (columna del data frame). Además, esm posible que el conjunto de casos sea muy numeroso, y que, adicionalmente, algunos de los valores de la variable que queremos estudiar estén repetidos para varios casos. Cuando esto ocurre, una opción interesante es, en lugar de representar en una tabla todos nuestros datos, **representar la distribución de frecuencias** de la variable (o atributo) que nos interesa. Es lo que vamos a hacer a continuación, tomando como variable a analizar la variable SALARIO.

Lo primero a tener en cuenta es que, en las distribuciones de frecuencias, **los valores de la variable suelen disponerse de menor a mayor**. Para ello, previamente vamos a ordenar las filas del *data frame* "datos" según el valor que toma, en el caso correspondiente, la variable SALARIO y, si existen casos con el mismo valor de SALARIO, los ordenaremos por orden alfabético del nombre del caso (nombre del trabajador, o de la fila del *data frame*). Para realizar este reordenamiento de casos (filas) del *data frame* de un modo sencillo, vamos a utilizar la función `arrange()` del paquete `{deplyr}`:

```
# Colocar los datos

library(dplyr)
datos <- datos %>% arrange(SALARIO, row.names(datos))
```

Una vez que los casos están ordenados en el data frame de menor a mayor valor de SALARIO, calcularemos, para cada valor del SALARIO, el número de casos que lo poseen, es decir, la frecuencia absoluta de cada valor de la variable SALARIO. Para ello, vamos a crear un objeto denominado "conteo" que va a ser de clase "tabla" de la variable SALARIO. Todo ello lo realizamos mediante la función `table()`, como se muestra a continuación:

```r
conteo <- table(datos$SALARIO)
conteo
```

```
##
##  8  9 10 12 15 20 22 25 30
##  2  3  6  8 11  8  6  4  1
```

Al mostrar en la consola el objeto "tabla" conteo, vemos cómo se compone de dos filas de datos. La primera se corresponde con los valores que toma la variable SALARIO en los distintos casos, y la segunda es el número de casos (**frecuencia absoluta**) que toma cada valor. Es decir, el objeto "tabla" es la distribución de frecuencias de la variable SALARIO.

Vamos a convertir este objeto "tabla" en un *data frame*, llamado "conteo_df", con el objeto de poder representar de un modo más elegante la distribución de frecuencias. Para ello, ejecutaremos el código:

```r
# Convertir el resultado a un data frame para una mejor visualización

conteo_df <- as.data.frame(conteo)
conteo_df
```

```
##   Var1 Freq
## 1    8    2
## 2    9    3
## 3   10    6
## 4   12    8
## 5   15   11
## 6   20    8
## 7   22    6
## 8   25    4
## 9   30    1
```

Al mostrar en la consola el data frame "conteo_df", observamos que consta de dos columnas o variables. "Var1" recoge los valores que toma la variable SALARIO en el grupo de casos, y "Freq" es el conjunto de frecuencias absolutas de los diferentes valores. Para que se entienda mejor qué es cada columna, las renombraremos:

```r
# Renombrar las columnas para mayor claridad

colnames(conteo_df) <- c("Valor", "Frecuencia")
```

A continuación, vamos a calcular el resto de frecuencias que suelen calcularse para una variable. La **frecuencia total**, N, que es la suma de todas las frecuencias absolutas, es decir, el número total de casos, se puede calcular fácilmente como:

```r
# Calcular y guardar la frecuencia total

N <- sum(conteo_df$Frecuencia)
```

La serie de frecuencias absolutas acumuladas se calcularán del siguiente modo:

```r
# Calcular frecuencias absolutas acumuladas

conteo_df$Frecuencia_acum <- cumsum(conteo_df$Frecuencia)
```

Como sabemos, la última frecuencia absoluta acumulada debe coincidir con la frecuencia total. Por último, calcularemos las frecuencias relativas, que son las frecuencias absolutas divididas por la frecuencia total, y recogen la proporción de casos correspondientes al valor de la variable:

```r
# Calcular frecuencias relativas

conteo_df$Frecuencia_R <- conteo_df$Frecuencia / N

# Calcular frecuencias relativas acumuladas

conteo_df$Frecuencia_R_acum <- cumsum(conteo_df$Frecuencia_R)
```

La suma de las frecuencias relativas es siempre 1 (el 100% de los casos). Además, la última frecuencia relativa acumulada siempre es, igualmente, 1.

Ahora vamos a construir una tabla que recoja la distribución de frecuencias de la variable SALARIO (con los diversos tipos de frecuencias). Para ello, simplemente hemos de aplicar al data frame "conteo_df" las funciones `kable()` del paquete `{knitr}`, y el resto de funciones auxiliares del paquete `{kableExtra}`:

```r
conteo_df %>%
  kable(caption = "Distribución de frecuencias de los salarios de la empresa",
        col.names = c("x(i) = Salario", "Frecuencia absoluta n(i)",
                      "Frecuencia absoluta acum. N(i)", "Frecuencia relativa f(i)",
                      "Frecuencia relativa acum. F(i)"),
                      format.args = list(decimal.mark = ".", digits = 2)) # %>%
```

Tabla 4.1: Distribución de frecuencias de los salarios de la empresa

x(i) = Salario	Frecuencia absoluta n(i)	Frecuencia absoluta acum. N(i)	Frecuencia relativa f(i)	Frecuencia relativa acum. F(i)
8	2	2	0.041	0.041
9	3	5	0.061	0.102
10	6	11	0.122	0.224
12	8	19	0.163	0.388
15	11	30	0.224	0.612
20	8	38	0.163	0.776
22	6	44	0.122	0.898
25	4	48	0.082	0.980
30	1	49	0.020	1.000

```r
  #kable_styling(full_width = F, bootstrap_options = "striped", "bordered", #"condensed", position = "c
 # row_spec(0, bold= T, align = "c") %>%
  #row_spec(1:(nrow(conteo_df)), bold= F, align = "c")
```

Hemos de advertir que en la función `kable()` se ha insertado un nuevo argumento, `format.args =` , que es una "lista" que controla aspectos de formato como si los decimales se indican con un punto o una coma, o el número de decimales a mostrar en la tabla.

Hay ocasiones en las que la cantidad de valores diferentes que toma la variable analizada para los diferentes casos es muy elevado. Esto puede deberse, por ejemplo, a que el número de casos es muy elevado, o a que la variable es de naturaleza continua, y puede tomar una gran variedad de posibles valores (incluso infinitos). En estos casos, un modo de representar la distribución de frecuencias de la variable en una tabla de dimensión reducida es **agrupando los valores en intervalos**. Esto es lo que vamos a hacer ahora con la variable SALARIO.

La primera tarea a realizar será formar los intervalos. Para ello podemos usar la función `cut()`, que permite decir el número de intervalos (de la misma amplitud) en que queremos dividir el intervalo que va desde el menor valor de la distribución (menor salario) al mayor valor (mayor salario). Por ejemplo, si deseamos agrupar los valores en 4 intervalos, el código será:

```
# Distribución de frecuencias agrupadas en intervalos
# del salario de los trabajadores de la empresa.

# Crear los intervalos
datos$intervalos <- cut(datos$SALARIO, breaks = 4, include.lowest = TRUE)
```

El resultado del código anterior es una nueva columna en el *data frame* "datos", llamada "intervalos", que informa, para cada caso, cuál de los 4 intervalos calculados lo contiene. El argumento lógico `include.lowest =` se especifica para indicar que el intervalo inferior es cerrados por la izquierda. Lo usual es que, salvo este, el resto sean abiertos, es decir, que los casos que toman como valor de la variable un extremo de intervalo se contabilicen dentro del intervalo donde ese valor es el extremo superior.

La columna "intervalos" es de la clase "factor". Precisamente, los posibles "niveles" de ese factor son los 4 intervalos que se han creado con `cut()`:

```
# Obtener los niveles de los intervalos

levels(datos$intervalos)
```

```
## [1] "[7.98,13.5]" "(13.5,19]"   "(19,24.5]"   "(24.5,30]"
```

Es preciso advertir que el vector "limites" contiene elementos de clase caracter (aunque contengan cifras, ya que también contienen corchetes, paréntesis y comas).

Las siguientes líneas de código son similares a las que vimos en el caso de distribuciones de frecuencias no agrupadas: se creará un objeto "tabla" para contabilizar el número de casos que pertenecen a cada intervalo (frecuencias absolutas), se transformará este objeto en un *data frame* para poder trabajar de un modo más fácil, y se cambiarán el nombre de las dos columnas para que se entienda mejor:

```
# Contar las frecuencias de cada intervalo

conteo_intervalos <- table(datos$intervalos)

# Convertir el resultado a un data frame para una mejor visualización

conteo_intervalos_df <- as.data.frame(conteo_intervalos)

# Renombrar las columnas para mayor claridad

colnames(conteo_intervalos_df) <- c("Intervalo", "Frecuencia")
```

Con todo lo anterior, se obtiene un *data frame* denominado "conteo_intervalos_df", que contiene dos columnas: la columna "Intervalo", con los 4 intervalos calculados, y la columna "Frecuencia", con el número de casos que tienen un salario incluido dentro de cada intervalo salarial.

Antes de proceder a diseñar la tabla de presentación de la distribución de frecuencia con `kable()`, vamos a obtener, para incluir en la tabla, otras informaciones que suelen ser presentadas junto a las frecuencias absolutas de cada intervalo.

Una de estas informaciones es lo que denominamos "**marca de clase**" de un intervalo. La marca de clase de un intervalo de valores es simplemente el punto medio de dicho intervalo.

La obtención en nuestro ejemplo de las marcas de clase puede resultar algo compleja, ya que hemos de recordar que los intervalos, tal y como están almacenados, son los niveles de una variable de clase "factor":

```
marca_clase <- sapply(strsplit(as.character(conteo_intervalos_df$Intervalo), ",|\\[|\\(|\\]"), function(
  mean(as.numeric(x[2:3]))
})
```

Explicaremos detenidamente el código anterior:

1. **`conteo_intervalos_df$Intervalo`**: Aquí se está accediendo a la columna "Intervalo" del *data frame* "conteo_intervalos_df".
2. **`as.character(conteo_intervalos_df$Intervalo)`**: convierte los valores de la columna "Intervalo" a caracteres (*strings*). Esto es necesario porque la función `strsplit()` trabaja con cadenas de texto.
3. **`strsplit(as.character(conteo_intervalos_df$Intervalo), ",|\\[|\\(|\\]")`**: `strsplit()` divide cada cadena de texto en partes usando los delimitadores especificados. En este caso, se están utilizando como delimitadores las comas ",", los corchetes "[" y "]", y el paréntesis de apertura "(". El resultado es una lista de vectores de caracteres, donde cada vector contiene las partes de la cadena original que estaban separadas por los delimitadores.
4. **`sapply(..., function(x) { ... })`**: `sapply()` aplica una función a cada elemento de una lista y simplifica el resultado a un vector o matriz. Por otro lado, la función anónima `function(x) { ... }` se aplica a cada vector resultante de `strsplit()`.
5. **`function(x) { mean(as.numeric(x[2:3])) }`**: Esta es la función anónima que se aplica a cada vector "x". Después, **`x[2:3]`** selecciona el segundo y tercer elemento del vector "x". Estos elementos corresponden con los límites del intervalo. **`as.numeric(x[2:3])`** convierte estos elementos a números. **`mean(as.numeric(x[2:3]))`** calcula la media de estos dos números, que representa el punto medio del intervalo.
6. **`marca_clase <- ...`**: Finalmente, el resultado de `sapply()` se asigna al vector"marca_clase", que contendrá los puntos medios de los 4 intervalos.

El resto de código integra el vector "marca_clase" en el *data frame* "conteo_intervalo_df" como una variable más, reodena con la función select() del paquete {dplyr} el orden de las columnas del *data frame*, calcula el resto de frecuencias (absoluta acumulada, relativa, relativa acumulada), y diseña la tabla de presentación de la distribución de frecuencias de los salarios de los trabajadores de la empresa; pero agrupada en 4 intervalos de valores.

```
# Agregar la columna "marca_clase" al data frame

conteo_intervalos_df$marca_clase <- marca_clase

#Cambiar el orden de las columnas en el data frame con dplyr

conteo_intervalos_df <- conteo_intervalos_df %>% select(Intervalo, marca_clase, Frecuencia)

# Calcular y guardar la frecuencia total
```

```r
N_agre <- sum(conteo_intervalos_df$Frecuencia)

# Calcular frecuencias absolutas acumuladas

conteo_intervalos_df$Frecuencia_acum <- cumsum(conteo_intervalos_df$Frecuencia)

# Calcular frecuencias relativas

conteo_intervalos_df$Frecuencia_R <- conteo_intervalos_df$Frecuencia / N_agre

# Calcular frecuencias relativas acumuladas

conteo_intervalos_df$Frecuencia_R_acum <- cumsum(conteo_intervalos_df$Frecuencia_R)

# Mostrar el resultado

conteo_intervalos_df %>%
  kable(caption = "Distribución de frecuencias agrupadas en intervalos de los salarios de la empresa",
        col.names = c("Intervalo salarial", "Marca de clase x(i)", "Frecuencia absoluta n(i)",
                      "Frecuencia absoluta acum. N(i)", "Frecuencia relativa f(i)",
                      "Frecuencia relativa acum. F(i)"),
        format.args = list(decimal.mark = ".", digits = 2)) #%>%
```

Tabla 4.2: Distribución de frecuencias agrupadas en intervalos de los salarios de la empresa

Intervalo salarial	Marca de clase x(i)	Frecuencia absoluta n(i)	Frecuencia absoluta acum. N(i)	Frecuencia relativa f(i)	Frecuencia relativa acum. F(i)
[7.98,13.5]	11	19	19	0.39	0.39
(13.5,19]	16	11	30	0.22	0.61
(19,24.5]	22	14	44	0.29	0.90
(24.5,30]	27	5	49	0.10	1.00

```r
#  kable_styling(full_width = F, bootstrap_options = "striped", "bordered", #"condensed", position = "c
#  row_spec(0, bold= T, align = "c") %>%
#  row_spec(1:(nrow(conteo_intervalos_df)), bold= F, align = "c")
```

4.3 Medidas de posición.

Las medidas de posición son instrumentos matemáticos que pretenden, mediante un único valor o muy pocos valores, **caracterizar de modo global** la distribución de frecuencias de una variable determinada.

Las medidas de posición se pueden clasificar en medidas de posición central, y en medidas de posición no central (principalmente, los llamados cuantiles).

Las principales **medidas de posición central** son: la media, la mediana y la moda. Dentro de la media, podemos distinguir la media aritmética, la geométrica y la armónica. De ellas, nos centraremos en la más común: la media aritmética.

La **media aritmética** de la distribución de frecuencias de una variable X se calcula como:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{h} x_i n_i$$

Hemos de tener en cuenta en la fórmula anterior que N es la frecuencia total, y h es el número de valores diferentes que toma la variable.

Si las frecuencias de todos los valores de la variable son 1 (distribución de frecuencias unitarias), lógicamente la media pasará a ser:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

En R, la función para obtener la media de una variable es `mean()`. Así, para obtener el salario medio de la variable SALARIO de los trabajadoes de la empresa, ejecutaremos el código:

```
## MEDIDAS

# Media aritmética.

media <- mean(datos$SALARIO)
media
```

```
## [1] 16.04082
```

Como podemos observar, el salario medio de los trabajadores de la empresa, recogido en el valor "media", es de 16.04 cientos de euros, es decir, 1604 euros.

¿Qué significado tiene la media aritmética? La media aritmética es el "centro de gravedad" de la distribución, el punto de equilibrio, en el sentido de que, si todos los trabajadores ganaran el salario medio, no habría diferencias salariales aun cuando la "masa" salarial invertida por la empresa permanecería invariable. Es decir, la media aritmética supone un reparto igualitario de la masa total de la variable.

Entre sus ventajas destaca el que, para variables (escala métrica) es siempre calculable y única. Como inconvenientes, que pierde su representatividad ante la existencia de casos atípicos o *outliers*, y que no se puede calcular en el caso de trabajar con atributos, variables cualitativas o factores (escalas nominal u ordinal).

La **mediana** es el valor que se corresponde con el caso o casos que dividen a la distribución en dos grupos con el mismo número de casos (frecuencias), siempre teniendo en cuenta que, previamente, la distribución ha sido ordenada según los valores de la variable en estudio, de menor a mayor. Si la distribución tiene frecuencia total par, los casos "frontera" entre los dos grupos en que queda dividida la distribución son dos, por lo que, si estos casos asumen valores diferentes en la variable estudiada, podría ocurrir que hubiera dos medianas diferentes. En tal caso, se suele tomar, como convenio, el promedio de de ambos valores para tener una única mediana.

En R, la función para obtener la mediana de una variable es `median()`. De este modo, para obtener el salario mediano de la variable SALARIO de los trabajadoes de la empresa, ejecutaremos el código:

```
# Mediana

mediana <- median(datos$SALARIO)
mediana
```

```
## [1] 15
```

En el ejemplo de la variable SALARIO, la mediana es 15. Es decir, 15 es el salario percibido por el caso 25, que es el trabajador que divide la distribución de frecuencias en dos grupos de 24 trabajadores: 24 que ganan un salario menor o igual que el caso 25 (menos o igual que 15 cientos de euros), y otros 24 trabajadores que ganan más o lo mismo que el caso en la posición 25 (o sea, más o iagual que 15 cientos de euros).

Como ventaja de la mediana contamos con que no es sensible a la existencia de casos atípicos o outliers, y que se puede calcular en el caso de atributos o factores en escala ordinal. Como desventajas tenemos que no tiene por qué ser única, y que no tiene en cuenta la totalidad de los valores de la distribución.

Con **la moda** hacemos referencia al valor (o valores) que posee (o poseen) una mayorv frecuencia absoluta.

En R, la moda se calcula mediante la función `Mode()` del paquete `{DescTools}`, que habremos de activar con `library()` (e instalar previamente, si aún no tenemos instalado este paquete):

```
# Moda

library(DescTools)
moda <- Mode(datos$SALARIO)
moda
```

```
## [1] 15
## attr(,"freq")
## [1] 11
```

Como podemos apreciar, la moda de la distribución es 15 (un salario de 1500 euros), que aparece en la distribución en 11 ocasiones (la frecuencia absoluta de ese salario es 11).

La moda puede ser calculada en atributos o factores en escala nominal. Como inconveniente principal tenemos que no tiene por qué ser un valor único (existen distribuciones multimodales).

Existen otras medidas que son de posición no central, principalmente lo que llamamos **cuantiles**. La naturaleza de loc cuantiles es fácil de comprender si los consideramos una generalización de la mediana. Ya sabemos que, ordenados los valores (y por tanto, los casos que toman dichos valores) de una distribución de frecuencias de una variable de menor a mayor, la mediana es el valor (o valores, porque pueden existir dos medianas, aunque vamos a suponer que solo hay una)) de la variable correspondiente al caso que divide a la distribución en dos grupos con el mismo número de frecuencias. Pues bien, si en lugar de dividir a la distribución de frecuencias en dos grupos con el mismo número de elementos, la dividimos en 4 grupos, estaremos hablando de tres valores correspondientes a los casos que delimitan a esos cuatro grupos. Estos valores serán los **cuartiles** de la distribución.

Si queremos dividir la distribución de 9 valores de la variable que toman los casos "frontera" que separan a estos 10 grupos. Esos valores serán los **deciles**. Y si queremos dividir la distribución de frecuencias en 100 grupos con el mismo número de casos o individuos, estaríamos hablando de 99 valores de la variable que toman los casos "frontera" que separan a estos 100 grupos. Esos valores serán los **percentiles**.

En R, la función para calcular los diferentes cuantiles es `quantile()`. Para calcular, por ejemplo, los cuartiles de la variable SALARIO, procederemos así:

```
# Calcular los cuartiles
cuartiles <- quantile(datos$SALARIO, probs = c(0.25, 0.5, 0.75))
cuartiles
```

```
## 25% 50% 75%
##  12  15  20
```

El argumento `probs =` informa de la proporción de los casos que han de quedar por detrás (con valores menores o iguales) de cada uno de los casos que hacen de "frontera" entre los grupos. En el caso de los cuartiles, estos son 0.25, 0.5 (este cuartil es, a su vez, la mediana de la distribución) y 0.75. Vemos cómo los cuartiles son 12, 15 y 20.

4.4 Medidas de dispersión o variabilidad.

Las medidas de dispersión cuantifican **lo cerca o lejos que, en general, los valores asumidos por los casos de una distribución de frecuencias se hallan respecto a una medida de posición central**. Si la medida de dispersión toma un valor muy elevado, querrá decir que la medida de posición central no representa bien a la distribución de frecuencias, ya que, en general, los casos toman valores alejados de dicha medida.

La medida de posición central a la que suelen hacer referencia las medidas de dispersión es la media aritmética.

Existen múltiples medidas de dispersión, que principalmente se dividen en **medidas absolutas** (que se expresan en ciertas unidades, como por ejemplo euros, o euros al cuadrado) y **medidas relativas** (que carecen de unidades y siven, por tanto, para comparar la dispersión entre distribuciones de frecuencias expresadas en distintas unidades).

La medida de dispersión absoluta más utilizada es la **varianza**, cuya fórmula es:

$$S^2 = \frac{1}{N}\sum_{i=1}^{h}(x_i - \overline{x})^2 n_i$$

Hemos de tener en cuenta en la fórmula anterior que N es la frecuencia total, y h es el número de valores diferentes que toma la variable.

Si las frecuencias de todos los valores de la variable son 1 (distribución de frecuencias unitarias), lógicamente la varianza pasará a ser:

$$S^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2$$

En realidad, la varianza es el promedio de las diferencias que existen entre los valores que toma la variable y la media aritmética de esta, diferencias que son elevadas al cuadrado para evitar la compensación entre diferencias por los signos.

Una limitación de la varianza viene referida a que, debido al exponente del paréntesis, puede tomar valores muy elevados. Para evitar el inconveniente, una medida alternativa es la **desviación típica**, que queda definida como la raíz cuadrada positiva de la varianza:

$$S = +\sqrt{S^2}$$

Otra media de dispersión muy utilizada, sobre todo en *Econometría*, es la *varianza insesgada* o **cuasivarianza**, cuya fórmula es:

$$\overline{S}^2 = \frac{1}{N-1}\sum_{i=1}^{h}(x_i - \overline{x})^2 n_i$$

Como siempre, si las frecuencias de todos los valores de la variable son 1 (distribución de frecuencias unitarias), lógicamente la cuasivarianza pasará a ser:

$$\overline{S}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2$$

En cuanto a una medida de dispersión relativa, cabe nombrar al **coeficiente de variación de Pearson**, definido como el cociente entre la desviación típica y la media aritmética (en valor absoluto):

$$V = \frac{S}{|\overline{x}|}$$

El coeficiente de variación informa del número de medias aritméticas que "caben" en la desviación típica de una distribución de frecuencias. A mayor coeficiente, mayor dispersión y menor representatividad de la media aritmética con respecto a la distribución. Además, pueden compararse coeficientes de distribuciones expresadas en unidades diferentes (medida relativa).

A continuación, vamos a calcular varianza, desviación típica, cuasivarianza, y coeficiente de variación en R. Para ello, hemos de tener en cuenta que la función `var()` de R, en realidad, calcula la cuasivarianza. Para obtener la varianza, pues, hemos de realizar una corrección (en realidad, para un número de casos muy grande, ambas medidas prácticamente coinciden):

```
# Varianza

varianza <- var(datos$SALARIO)*(N-1)/N # recordar que la frecuencia total N ya fue calculada
varianza
```

```
## [1] 30.48813
```

```
# Desviación típica

desv <- varianza ^ (1/2)
desv
```

```
## [1] 5.521606
```

```
# Cuasivarianza

cuasivarianza <- var (datos$SALARIO)
cuasivarianza
```

```
## [1] 31.1233
```

```
# Coeficiente de variación

cvariacion <- desv / abs(media)
cvariacion
```

```
## [1] 0.3442222
```

4.5 Medidas de forma.

Las medidas de forma cuantifican el grado de deformación vertical y horizontal de la representación gráfica de una distribución de frecuencias. Son de dos tipos: medidas de asimetría y medidas de apuntamiento o curtosis.

Las **medidas de asimetría** miden el grado de deformación vertical con respecto a un "eje de simetría", que es aquel que pasa por el valor medio de la distribución. Si suponemos que la distribución es unimodal y campaniforme, tendremos los casos que se muestran en la figura:

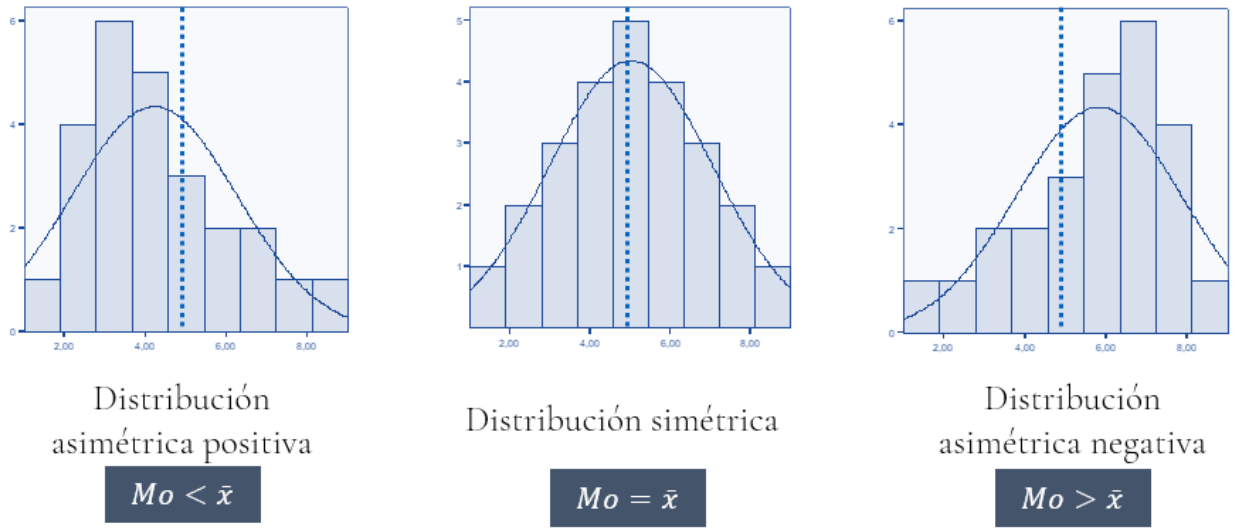


Figura 4.1: Tipos de asimetría

El tipo y grado de asimetría se puede obtener mediante el **coeficiente de asimetría de Fisher**. Este coeficiente toma valor negativo si la distribución es *asimétrica negativa* (mayores frecuencias a la derecha de la media), valor positivo si la distribución es *asimétrica positiva* (mayores frecuencias a la izquierda de la media), y se acerca a 0 en caso de que la distribución seaaproximadamente *simétrica*, aunque pueden darse casos de distribuciones no simétricas con coeficiente 0. En R, se puede obtener el coeficiente de asimetría mediante la función `skewness()` del paquete `{moments}`:

```
# Coeficiente de asimetría de Fisher

library(moments)
asimetria <- skewness(datos$SALARIO)
asimetria
```

```
## [1] 0.413764
```

El valor obtenido para la variable SALARIO de nuestra distribución de frecuencias de los trabajadores de la empresa es positivo, lo que indica asimetría positiva: las mayores frecuencias se localizan a la derecha de la media. Esto se puede comprobar fácilmente construyendo el histograma de la variable SALARIO y trazando una línea vertical que pase por la media salarial. Para ello usamos el paquete `{ggplot2}`:

```
library (ggplot2)
ggplot(data = datos, map = aes(x = SALARIO)) +
  geom_histogram(bins = 7, colour = "red", fill = "orange") +
  geom_vline(aes(xintercept = mean(SALARIO)), colour = "blue", linetype = "dashed", size = 1) +
  ggtitle("SALARIO MENSUAL", subtitle = "trabajadores de la empresa XXX")+
  xlab("Salario (cientos de euros)") +
  ylab("Frecuencias")
```

Como comprobamos, la distribución es claramente asimétrica positiva.

En cuanto a las **medidas de curtosis**, miden el grado de deformación horizontal con respecto a una distribución "tipo", la distribución normal. Suponemos previamente que la distribución de frecuencias estudiada es campaniforme, unimodal y simétrica (o con ligera asimetría). Pueden darse los casos que se muestran en la figura:

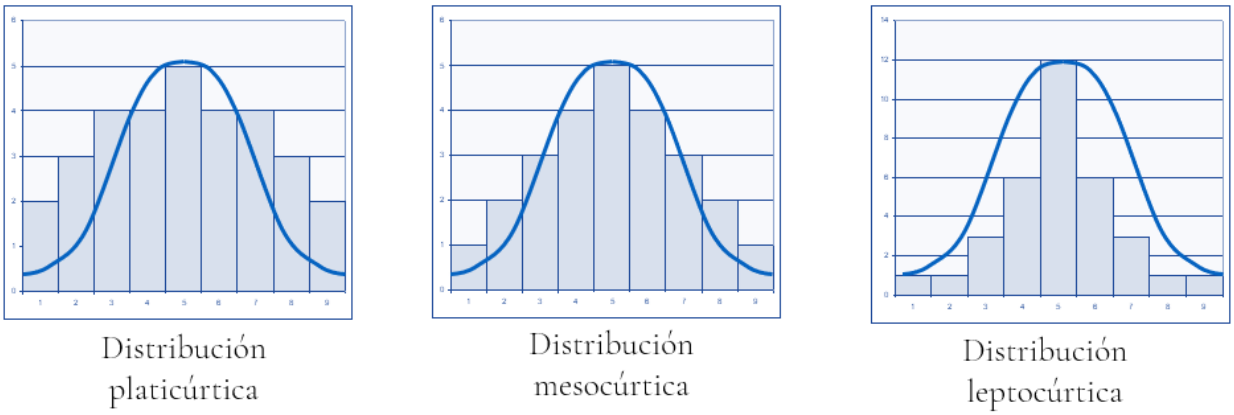


Figura 4.2: Tipos de distribución según el apuntamiento o curtosis

El tipo y grado de apuntamiento o curtosis se puede obtener mediante el **coeficiente de apuntamiento de Fisher**. Este coeficiente toma valor negativo si la distribución es *platicúrtica* (más aplastada que la distribución normal), valor positivo si la distribución es *leptocúrtica* (más apuntada que la distribución normal), y se acerca a 0 en caso de que la distribución sea aproximadamente igual de apuntada que la distribución normal.

En R, se puede obtener el coeficiente de asimetría mediante la función `kurtosis()` del paquete `{moments}`. Hay que tener en cuenta que para que esta versión coincida con lo dicho anteriormente, al valor calculado hay que restarle el valor "3":

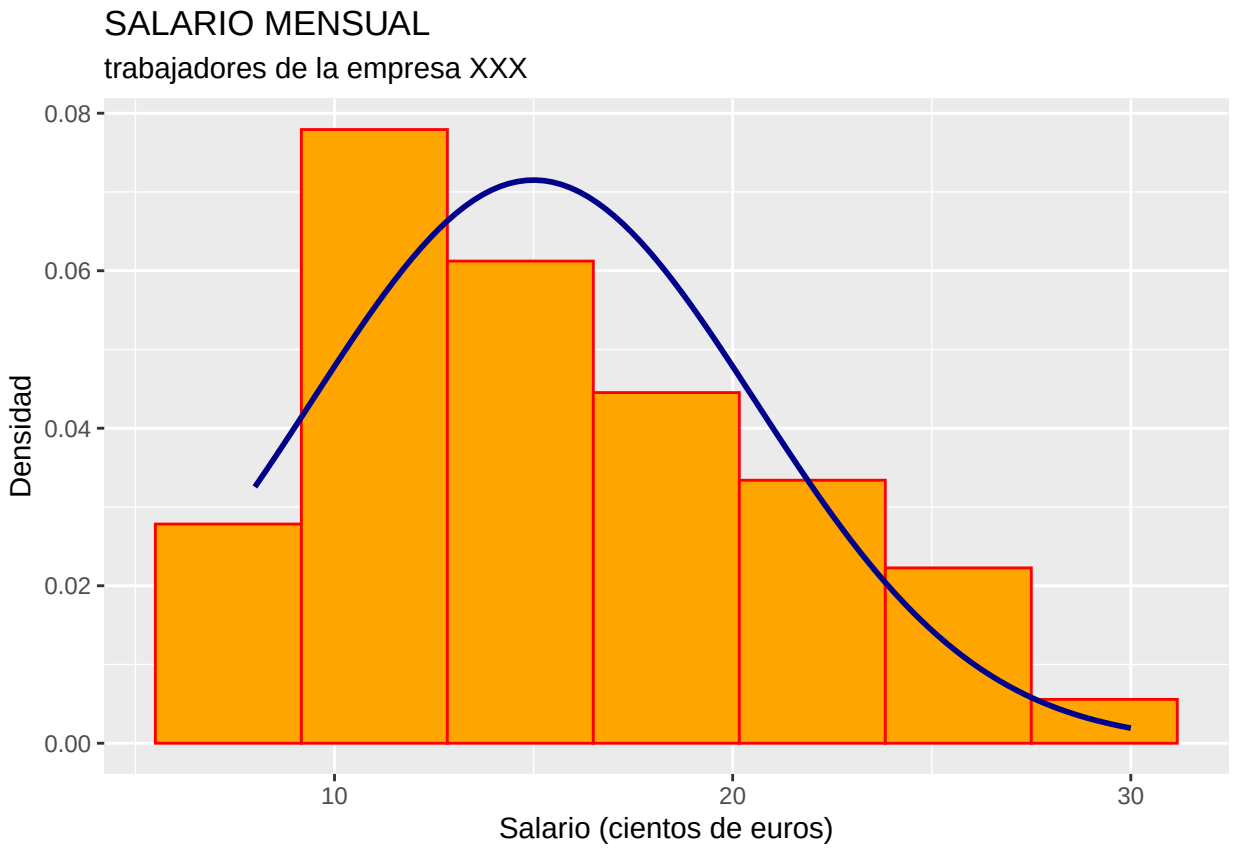
```
# Coeficiente de apuntamiento o curtosis de Fisher

curtosis <- kurtosis(datos$SALARIO) - 3
curtosis
```

```
## [1] -3.823296
```

Se aprecia como el coeficiente (corregido) es menor que 0, por lo que la distribución de frecuencias de la variable SALARIO es platicúrtica (más "aplastada" que la distribución de frecuencias normal. También hay que tener en cuenta que debemos tener precaución en la aplicación del coeficiente, ya que vimos con anterioridad que la distribución no es aproximadamente simétrica. Gráficamente, podemos comprobar lo anterior representando el histograma y una curva normal que posea la misma moda (que ya calculamos anteriormente). Para que sean comparables, debemos transformar el eje "y" del gráfico, pasando de "frecuencias" a "densidad", para lo cuál se incluye en el `geom_histogram()` el argumento `aes(y = ..density..)`:

```
ggplot(data = datos, map = aes(x = SALARIO)) +
  geom_histogram(bins = 7, colour = "red", fill = "orange", aes(y = ..density..)) +
  stat_function(fun = dnorm, args = list(mean = moda, sd = sd(datos$SALARIO)), colour = "darkblue", size
  ggtitle("SALARIO MENSUAL", subtitle = "trabajadores de la empresa XXX")+
  xlab("Salario (cientos de euros)") +
  ylab("Densidad")
```



4.6 Materiales para realizar las prácticas del capítulo.

En esta sección se muestran los links de acceso a los diferentes materiales (*scripts*, datos…) necesarios para llevar a cabo los contenidos prácticos del capítulo.

Datos (en formato Microsoft (R) Excel (R)):

- trabajadores.xlsx (obtener aquí)

Scripts:

- explora_trabajadores.R (obtener aquí)

Capítulo 5

Análisis previo de datos.

5.1 Introducción.

Antes de la aplicación de técnicas complejas que permitan extraer de los datos conclusiones relevantes, conviene aplicar a tales datos unos **pasos iniciales** y técnicas básicas destinadas a conseguir dos objetivos:

- **Preparar nuestros datos** para que puedan ser procesados correctamente sin provocar distorsiones en los resultados.
- Obtener una **visión inicial de la información** que esconden nuestros datos, fundamentalmente en cuanto a las medidas básicas que caracterizan la distribución de frecuencias de las variables integradas en nuestros datos, así como, en el caso de contar con más de una variable, de las relaciones que existen entre ellas.

Además, es preciso tener en cuenta que, usualmente, es conveniente que estos rasgos iniciales que caracterizan a nuestra muestra o población sean plasmados de un modo **visualmente amigable**, claro y conciso.

En esta práctica, por medio de un ejemplo basado en información económico-financiera de una muestra constituida por 100 empresas dedicadas a la producción de electricidad mediante tecnología eólica, se mostrarán una serie de buenas prácticas y análisis básicos útiles a la hora de preparar y analizar inicialmente nuestro conjunto de datos.

Vamos a suponer que trabajamos dentro de un **proyecto** que hemos creado previamente, de nombre "explora". Dentro de la carpeta del proyecto guardaremos el *script* llamado "explora_describe.R", y el archivo de Microsoft® Excel® llamado "eolica_100_mv.xlsx". Si abrimos este último archivo, comprobaremos que se compone de tres hojas. La primera muestra un mensaje sobre el uso de los datos, la segunda recoge la descripción de las variables consideradas, y la tercera (hoja "Datos") guarda los datos que debemos importar. Estos datos se corresponden con diferentes variables económico-financieras de las 100 empresas productoras de electricidad mediante generación eólica con mayor volumen de activo.

Luego cerraremos el archivo de Microsoft® Excel®, "eolica_100.xlsx", y volveremos a RStudio. Después, abriremos nuestro *script* "explora_describe.R" con `File → Open File…` Este *script* contiene el programa que vamos a ir ejecutando en la práctica.

La primera línea / instrucción en el *script* es:

```
rm(list = ls())
```

La instrucción tiene como objeto limpiar el *Environment* (memoria) de objetos de anteriores sesiones de trabajo. Para importar los datos que hay en la hoja "Datos" del archivo de Microsoft® Excel® llamado "eolica_100_mv.xlsx", ejecutaremos el código:

```
library(readxl)
eolica_100 <- read_excel("eolica_100_mv.xlsx", sheet = "Datos")
summary (eolica_100)
```

```
##     NOMBRE               RES               ACTIVO             FPIOS
##  Length:100         Min.   : -5661.5   Min.   :   24944   Min.   : -77533
##  Class :character   1st Qu.:   669.5   1st Qu.:   34547   1st Qu.:   2305
##  Mode  :character   Median :  2084.5   Median :   46950   Median :  11936
##                     Mean   : 11529.8   Mean   :  277270   Mean   : 123743
##                     3rd Qu.:  3806.7   3rd Qu.:   85610   3rd Qu.:  28292
##                     Max.   :727548.0   Max.   :13492812   Max.   :6904824
##                     NA's   :1          NA's   :1
##
##      RENECO           RENFIN            LIQUIDEZ          ENDEUDA
##  Min.   :-2.813   Min.   :-359.773   Min.   :  0.0140   Min.   :  0.917
##  1st Qu.: 1.558   1st Qu.:   2.556   1st Qu.:  0.6567   1st Qu.: 50.852
##  Median : 4.236   Median :  15.326   Median :  1.0650   Median : 83.346
##  Mean   : 5.416   Mean   :  17.243   Mean   :  2.7214   Mean   : 72.227
##  3rd Qu.: 7.970   3rd Qu.:  31.307   3rd Qu.:  1.6078   3rd Qu.: 95.388
##  Max.   :35.262   Max.   : 588.190   Max.   :128.4330   Max.   :140.745
##  NA's   :2                                             NA's   :2
##
##      MARGEN            SOLVENCIA         APALANCA           MATRIZ
##  Min.   :-2248.157   Min.   :-40.74   Min.   :-8254.11   Length:100
##  1st Qu.:   12.316   1st Qu.:  4.71   1st Qu.:   16.13   Class :character
##  Median :   26.618   Median : 16.65   Median :  161.97   Mode  :character
##  Mean   :    3.228   Mean   : 27.57   Mean   :  345.03
##  3rd Qu.:   39.590   3rd Qu.: 45.59   3rd Qu.:  623.13
##  Max.   :  400.899   Max.   : 99.08   Max.   :12244.35
##  NA's   :2
##
##   DIMENSION
##  Length:100
##  Class :character
##  Mode  :character
```

R ha considerado la primera columna como una variable de tipo cualitativo, atributo, o factor. En realidad, esta columna no es una variable, sino que está formada por los nombres de los diferentes casos u observaciones. Para evitar que R tome la columna de los nombres de los casos como una variable más, podemos redefinir nuestro *data frame* diciéndole que tome esa primera columna como el conjunto de los *nombres* de los casos:

```
eolica_100 <- data.frame(eolica_100, row.names = 1)
summary (eolica_100)
```

```
##       RES               ACTIVO             FPIOS             RENECO
##  Min.   : -5661.5   Min.   :   24944   Min.   : -77533   Min.   :-2.813
##  1st Qu.:   669.5   1st Qu.:   34547   1st Qu.:   2305   1st Qu.: 1.558
##  Median :  2084.5   Median :   46950   Median :  11936   Median : 4.236
##  Mean   : 11529.8   Mean   :  277270   Mean   : 123743   Mean   : 5.416
##  3rd Qu.:  3806.7   3rd Qu.:   85610   3rd Qu.:  28292   3rd Qu.: 7.970
##  Max.   :727548.0   Max.   :13492812   Max.   :6904824   Max.   :35.262
##  NA's   :1          NA's   :1                            NA's   :2
```

```
##
##      RENFIN            LIQUIDEZ          ENDEUDA           MARGEN
##  Min.   :-359.773   Min.   :  0.0140   Min.   :  0.917   Min.   :-2248.157
##  1st Qu.:   2.556   1st Qu.:  0.6567   1st Qu.: 50.852   1st Qu.:   12.316
##  Median :  15.326   Median :  1.0650   Median : 83.346   Median :   26.618
##  Mean   :  17.243   Mean   :  2.7214   Mean   : 72.227   Mean   :    3.228
##  3rd Qu.:  31.307   3rd Qu.:  1.6078   3rd Qu.: 95.388   3rd Qu.:   39.590
##  Max.   : 588.190   Max.   :128.4330   Max.   :140.745   Max.   :  400.899
##                                        NA's   :2         NA's   :2
##
##    SOLVENCIA         APALANCA           MATRIZ           DIMENSION
##  Min.   :-40.74   Min.   :-8254.11   Length:100         Length:100
##  1st Qu.:  4.71   1st Qu.:   16.13   Class :character   Class :character
##  Median : 16.65   Median :  161.97   Mode  :character   Mode  :character
##  Mean   : 27.57   Mean   :  345.03
##  3rd Qu.: 45.59   3rd Qu.:  623.13
##  Max.   : 99.08   Max.   :12244.35
```

Observaremos que ya no aparece NOMBRE, puesto que la columna correspondiente ya no es considerada como una variable.

5.2 Análisis de una variable.

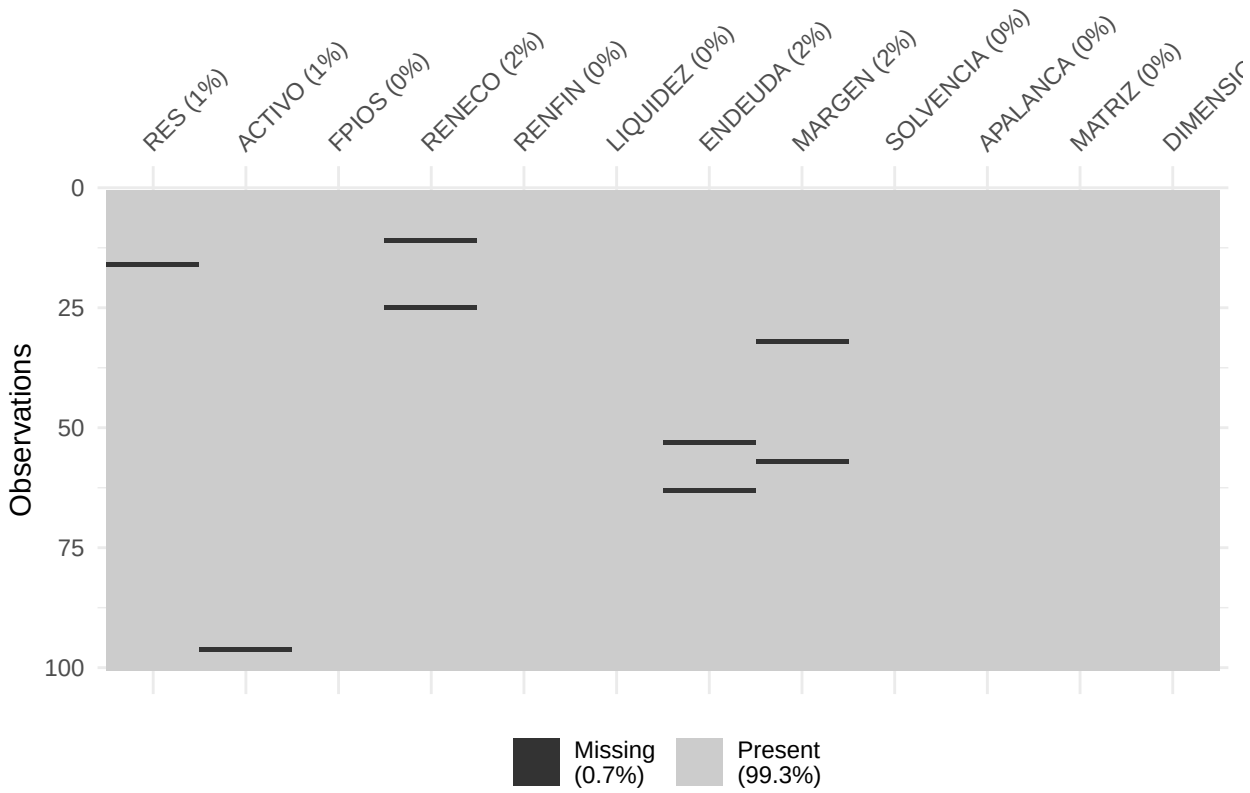
5.2.1 Buscando *missing values* y *outliers*.

Vamos a suponer que la variable que queremos estudiar es la variable *Rentabilidad Económica* (RENECO).

La primera acción que debe realizarse es comprobar que todos los casos (empresas) tienen su correspondiente dato o valor para la variable (RENECO), es decir, que no existen **valores perdidos o *missing values***.

Para tener una idea general, se puede utilizar la función `vis_miss()` del paquete `{visdat}`, que nos localizará gráficamente los *missing values* de las diferentes variables, y calculará el porcentaje de casos que supone, con respecto al total de observaciones:

```
library(visdat)
vis_miss(eolica_100)
```



Puede observarse cómo, en el caso concreto de la variable RENECO, un 2% de los casos no tienen dato (es decir, 2 casos de los 100). Para localizar los casos concretos, puede recurrirse a utilizar las herramientas de manejo de *data frames* del paquete `{dplyr}`. En concreto, realizaremos una copia del *data frame* original, "eolica_100", a la que llamaremos "muestra", que es con la que trabajaremos (para mantener la integridad del *data frame* original); y filtraremos los casos para detectar aquellos que carecen de valor en la variable RENECO:

```
library (dplyr)
muestra<- select(eolica_100, everything())
muestra %>% filter(is.na(RENECO)) %>% select(RENECO)
```

La función `is.na()` comprueba si, en la posición correspondiente a una fila o caso, para la variable escrita en el argumento; hay o no un dato o valor. Como resultado se obtienen dos empresas, para las que se puede comprobar que no hay valor para la variable RENECO:

```
##                         RENECO
## Viesgo Renovables SL.       NA
## Sargon Energias SLU         NA
```

Ante la existencia de *missing values*, se puede actuar de varios modos. Por ejemplo, se puede intentar obtener por otro canal de información el conjunto de valores de RENECO que no están disponibles, o recurrir a alguna estimación para los mismos y asignarlos. En caso de que esto sea difícil, se puede optar, simplemente, por eliminar estos casos, en especial cuando representan un porcentaje muy reducido respecto al total de casos. En nuestro ejemplo, vamos a suponer que hemos optado por esta última vía, al no conseguir unos valores más o menos verosímiles de RENECO para las empresas de las que se carece de dato. Esta **eliminación de casos** se podrá realizar mediante el código:

```
muestra <- muestra %>% filter(! is.na(RENECO))
```

El operador `!` significa "no".

Podemos comprobar cómo en el *Global Environment* aparece el *data frame* "muestra" con dos casos menos (98):

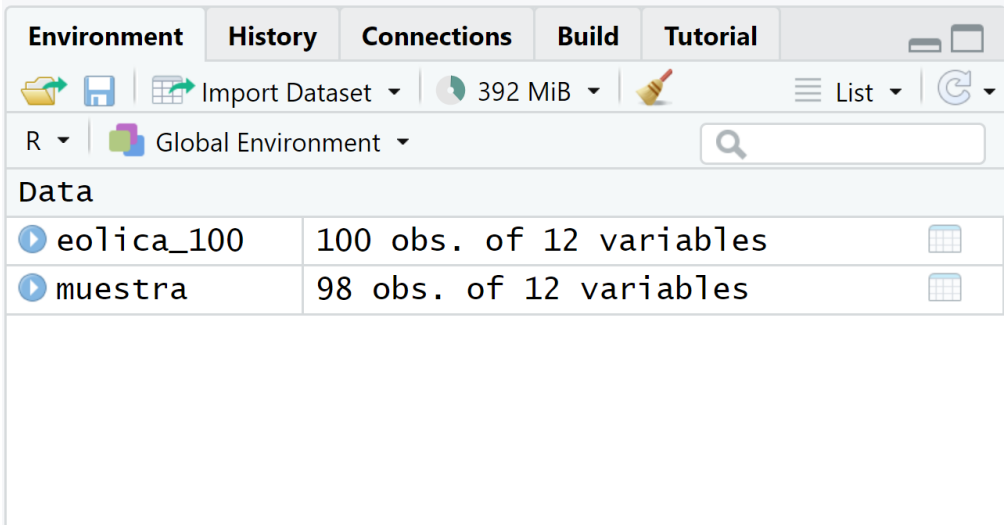
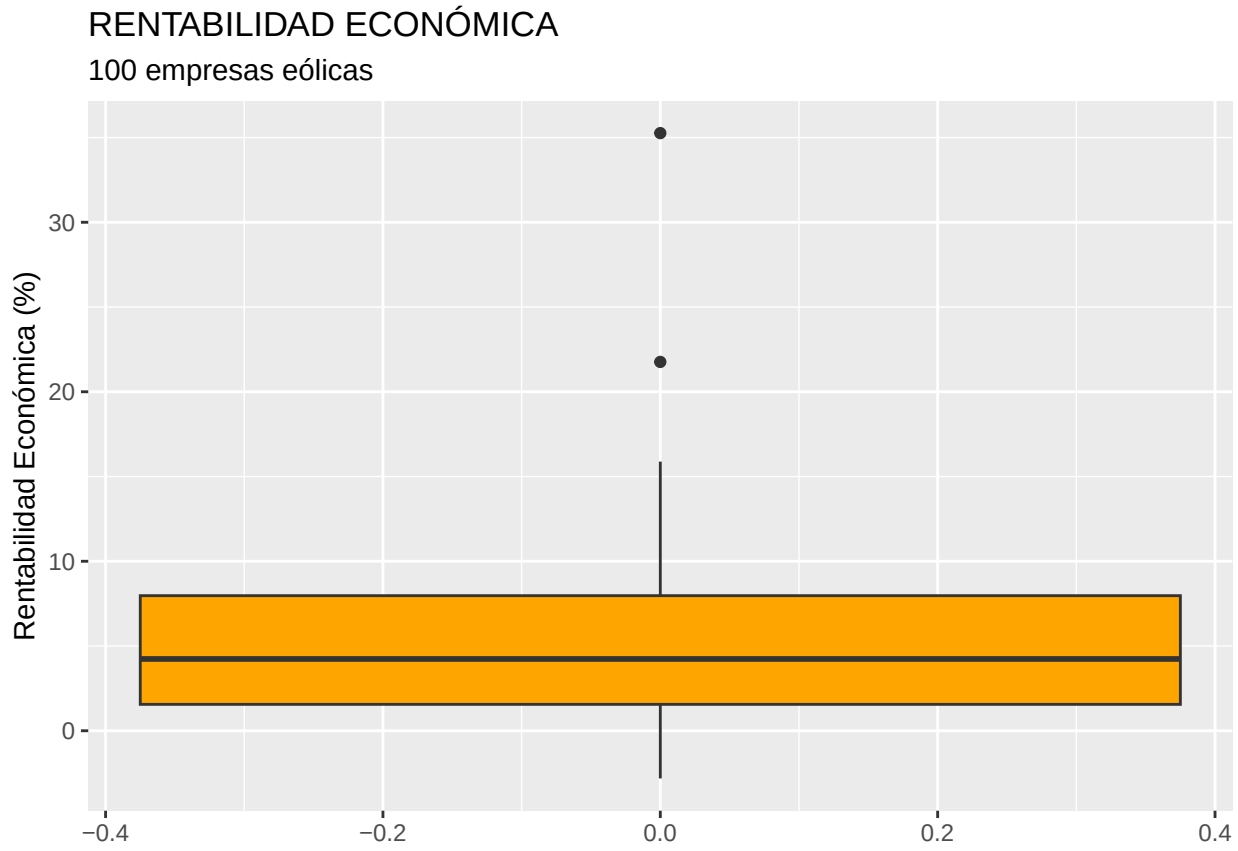


Figura 5.1: Global Environment.

Una vez tratados los casos con valores perdidos o *missing values*, **conviene detectar la posible presencia de *outliers*** o casos atípicos en la muestra, que pudieran desvirtuar los resultados derivados de ciertos análisis. Al trabajar con una sola variable métrica (la rentabilidad económica, RENECO), podemos intentar realizar esta tarea **representando gráficamente** la variable mediante un ***boxplot*** o *gráfico de caja*. Aplicaremos, por ejemplo, el código siguiente, que utiliza la gramática del paquete `{ggplot2}`:

```
library (ggplot2)
ggplot(data = muestra, map = (aes(y = RENECO))) +
    geom_boxplot(fill = "orange") +
    ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas") +
    ylab("Rentabilidad Económica (%)")
```

Obteniéndose el gráfico:



La "caja" contiene el 50% de los casos centrales (los que van del primer cuartil al tercero, cuya diferencia se llama *rango intercuartílico*), y contiene una línea horizontal que es la mediana (segundo cuartil). Por arriba sobresale un segmento que llega al valor de la variable más grande que no llega a ser atípico; y por debajo de la caja otro segmento que llega al valor de la variable más pequeño que no llega a ser atípico. Un valor atípico es el que se aleja más de 1.5 veces el rango intercuartílico (altura de la caja) del tercer cuartil, por arriba; o del primer cuartil, por abajo. Se registran mediante puntos.

En nuestro caso, el *boxplot* ratifica la existencia de dos casos atípicos. Para identificar esos dos casos concretos, podemos recurrir al paquete `{dplyr}`, y **establecer un filtro** con el siguiente código:

```
Q1 <- quantile (muestra$RENECO, c(0.25))
Q3 <- quantile (muestra$RENECO, c(0.75))
muestra %>%
        filter(RENECO > Q3 + 1.5*IQR(RENECO) | RENECO < Q1 - 1.5*IQR(RENECO)) %>%
        select(RENECO)
```

En el código anterior, las dos primeras filas calculan los cuartiles primero (Q1) y tercero (Q3) mediante la función `quantile()`. Es preciso tener en cuenta que esta función calcula los *percentiles*. Luego se filtran, mediante la función de `{dplyr}` `filter()`, los *outliers*, calculadoscomo aquellos casos con valores de RENECO mayores que Q3 más 1,5 veces el rango intercuartílico de la variable; o menores que Q1 menos 1,5 veces dicho rango intercuartílico. Para calcular el rango intercuartílico se recurre a la función `IQR()`. Finalmente, con `select()`, se muestran los casos en la consola de R-Studio:

```
##                      RENECO
## Molinos Del Ebro SA 35.262
## Sierra De Selva SL  21.761
```

Como ocurría con los *missing values*, el tratamiento de los *outliers* depende de la información que se tenga, existiendo varias alternativas (corrección del dato, estimación, etc.) Si no se tiene información fiable, y los *outliers* no representan una gran proporción respecto al total de casos, puede optarse por su eliminación de la muestra. En este ejemplo, efectivamente, **eliminaremos estas dos empresas con comportamiento atípico** en la rentabilidad económica (RENECO), a fin de que su presencia en la muestra **no distorsione los resultados en la aplicación posterior de ciertas técnicas** (por ejemplo, un ANOVA o un análisis de regresión). Podemos hacerlo creando un nuevo *data frame* a partir de "muestra"; pero sin esos dos casos. Ese nuevo *data frame* se llamará, por ejemplo, "**muestra_so**":

```
muestra_so <- muestra %>%
            filter(RENECO <= Q3 + 1.5*IQR(RENECO) & RENECO >= Q1 - 1.5*IQR(RENECO))
```

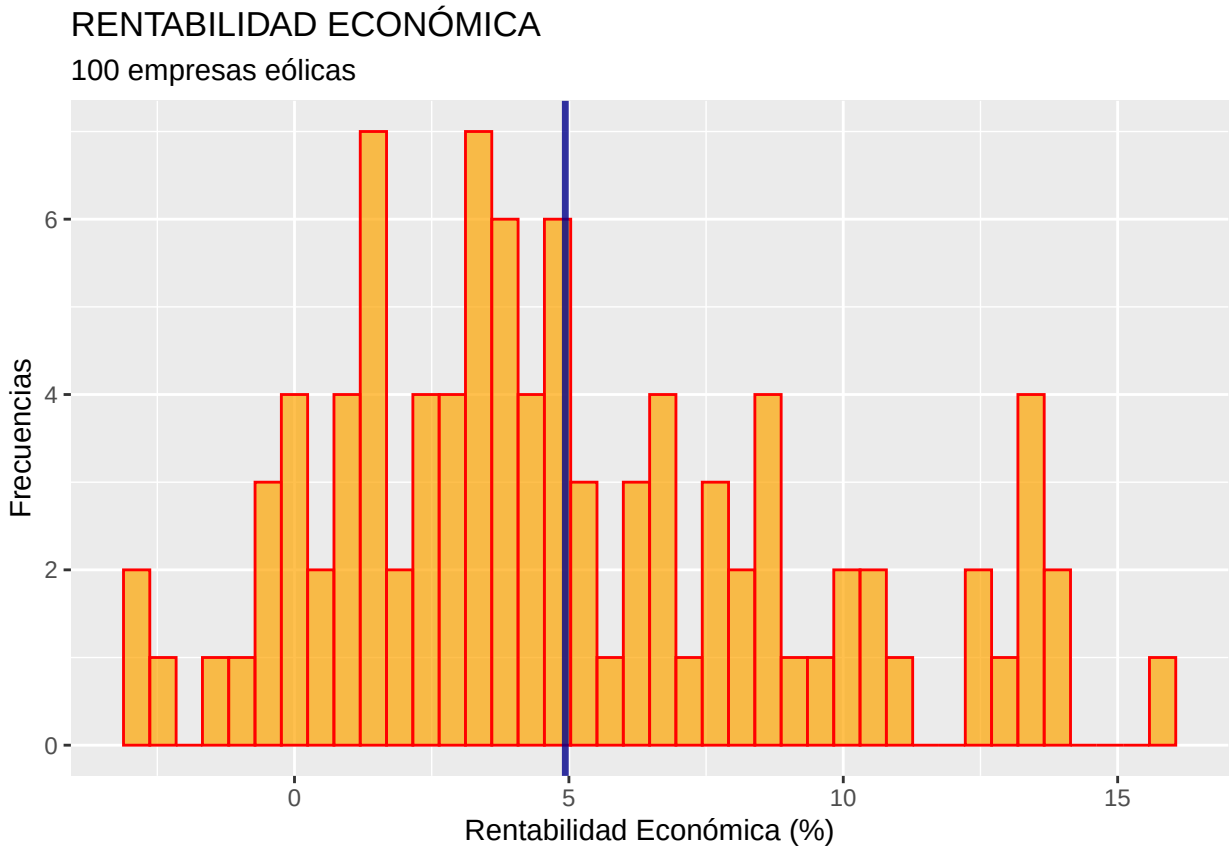
Es importante observar que, en el código de la función `filter()`, las desigualdades deben cambiar, así como el operador "|" por el operador "**&**". En el *Global Environment* podemos comprobar cómo el *data frame* "muestra_so" posee el mismo número de variables que el *data frame* "muestra"; pero con dos observaciones o casos menos (96).

5.2.2 Descripción de una variable.

Una vez se tiene preparada nuestra base de datos, con un tratamiento adecuado de los *missing values* y de los *outliers,* y **antes** de proceder a la aplicación de una técnica adecuada, según los objetivos perseguidos en el estudio; suelen presentarse una serie de **gráficos** básicos y **medidas** descriptivas que proporcionan una **idea inicial de la estructura** del sector para la variable o variables analizadas. Nos referimos a medidas y/o gráficos de posición, dispersión y forma (asimetría y curtosis).

El **análisis gráfico** suele dar una idea atractiva e intuitiva de la estructura de la distribución de frecuencias de nuestro conjunto de casos en relación con la variable a analizar. Un gráfico fundamental es el **histograma** de la variable estudiada. Para ello, utilizaremos la gramática del paquete `{ggplot2}`:

```
ggplot(data = muestra_so, map = aes(x = RENECO)) +
  geom_histogram(bins = 40,
                 colour = "red",
                 fill = "orange",
                 alpha = 0.7) +
  geom_vline(xintercept = mean(muestra_so$RENECO),
             color = "dark blue",
             size = 1.2,
             alpha = 0.8) +
  ggtitle("RENTABILIDAD ECONÓMICA", subtitle = "100 empresas eólicas")+
  xlab("Rentabilidad Económica (%)") +
  ylab("Frecuencias")
```



En el gráfico vemos de un modo claro la distribución de frecuencias en cuanto a la rentabilidad económica (RENECO). Se ha incorporado una línea vertical azul (mediante `geom_vline()`) para localizar la rentabilidad media. Entre otras cosas, se puede apreciar que la distribución de frecuencias es acampanada y *asimétrica positiva*.

Por otro lado, conviene tener conocimiento del valor de las principales **medidas descriptivas** (de posición, dispersión, forma) que caracterizan a la distribución de la variable a analizar. Una opción rápida y completa consiste en hacer uso de la función `descr()` del paquete `{summarytools}`. Así por ejemplo, el código:

```
library (summarytools)
descr(muestra_so$RENECO,
      stats = c("mean", "sd", "min", "q1", "med", "q3", "max", "iqr", "cv"),
      transpose = FALSE,
      style = "simple",
      justify = "center",
      headings = T)
```

Da lugar a la siguiente tabla:

Medidas descriptivas

RENECO

Mean

4.9350208

Std.Dev

4.3120451

Min

-2.8130000

Q1

1.4185000

Median

4.1440000

Q3

7.8380000

Max

15.8820000

IQR

6.4002500

CV

0.8737643

Estos valores se corresponden con el valor medio de RENECO, su desviación típica, valor mínimo, primer cuartil, mediana, tercer cuartil, valor máximo, rango intercuartílico, y *coeficiente de variación* o CV, que es una medida de dispersión relativa calculada como la desviación típica entre la media (número de "medias" que "caben" en una desviación típica. Estas medidas fueron explicadas con detenimiento en el capítulo 4.

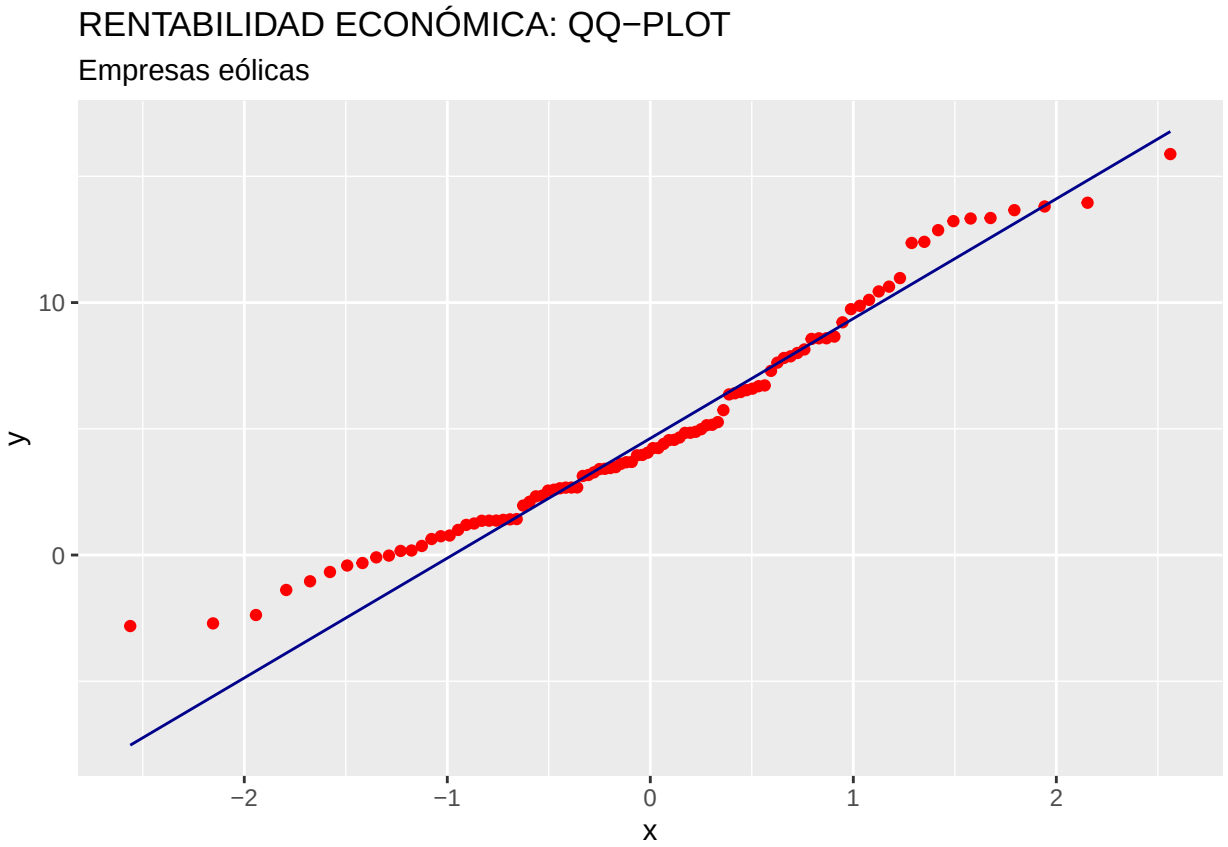
5.2.3 Normalidad.

En muchas técnicas multivariantes basadas en métodos inferenciales (por ejemplo, análisis de la varianza, o en la regresión lineal), se requiere que las variables sigan una **distribución normal**. Para comprobarlo, se puede a recurrir a análisis gráficos o a análisis formales, estos últimos basados en contrastar la hipótesis nula de normalidad.

Vamos a mostrar un **método gráfico** muy extendido. Comprobaremos la normalidad de la variable RENECO mediante un ***gráfico qq*** (cuantil-cuantil), que compara los cuantiles de nuestra muestra con los de una distribución normal teórica (con la misma media y desviación típica). Si los puntos se sitúan cercanos a la diagonal, entonces se asumirá un comportamiento (aproximadamente) normal. El código para realizar el gráfico con las herramientas del paquete `{ggplot2}` es:

```
ggplot(data = muestra_so, aes(sample = RENECO)) +
  stat_qq(colour = "red") +
  stat_qq_line(colour = "dark blue") +
  ggtitle("RENTABILIDAD ECONÓMICA: QQ-PLOT", subtitle = "Empresas eólicas")
```

Y el resultado:



A veces, es difícil obtener una conclusión sólida con el gráfico *qq*; aunque en el ejemplo se aprecia, sobre todo en los primeros puntos, una separación notable de estos con respecto a la línea, lo que induce a pensar en que podría **no** seguirse una distribución normal.

Si queremos ser más precisos, en lugar de un análisis gráfico se puede recurrir a realizar un análisis formal, basado en la realización de **contrastes de hipótesis**. Una prueba muy usual es la **prueba de normalidad de *Shapiro y Wilk***, que tiene un buen comportamiento en muestras relativamente reducidas. En esta prueba, la hipótesis nula equivale al supuesto de normalidad. Para un 5% de significación estadística, un p-valor superior a 0.05 implicará el no-rechazo de la hipótesis de normalidad. Para realizar la prueba, se ejecutará el código:

```
shapiro.test(x = muestra_so$RENECO)
```

El resultado obtenido en la consola:

```
##
##  Shapiro-Wilk normality test
##
## data:  muestra_so$RENECO
## W = 0.9605, p-value = 0.005523
```

Como el p-valor es (muy) inferior a 0.05, **se rechaza la hipótesis nula de normalidad en la distribución**, lo que implica que, para una significación estadística del 5%, admitimos que RENECO **no** sigue, para nuestra muestra, un comportamiento normal, como ya se anticipó con el gráfico *qq*.

5.3 Análisis de múltiples variables.

Son muchas las técnicas aplicadas al análisis de datos económicos basadas en una distribución de frecuencias multivariante. En este apartado nos centraremos en el caso de **variables métricas**, ya que al caso de atributos, variables categóricas o factores; le dedicaremos un tema en exclusiva. Algunas técnicas multivariantes son el análisis de componentes principales, el análisis de regresión, el análisis clúster…

Todas estas metodologías requieren, de nuevo, de una fase inicial que ponga a punto la base de datos y ofrezca una fotografía de cómo es la situación en cuanto a las variables en estudio. En este sentido, es conveniente aplicar, para cada variable por separado, algunos de los análisis gráficos básicos vistos anteriormente.

A estos análisis básicos hay que añadir, principalmente, algún estudio gráfico y alguna medida cuantitativa más, destinados fundamentalmente a comprobar el **grado de intensidad en la relación estadística** entre las variables implicadas. Antes de abordar esta última cuestión, trabajaremos con dos variables.

5.3.1 Caso de dos variables: localización de missing values y outliers, y representación gráfica.

En nuestro ejemplo, vamos a incorporar al análisis la variable ACTIVO (*volumen de activos de la empresa en miles de euros*). Partiremos, como ya hicimos en el caso univariante, de la detección de valores perdidos o *missing values*. Para no modificar el *data frame* original ("eolica_100"), trabajaremos con una copia, llamada "muestra2":

```
muestra2 <- select(eolica_100, everything())
muestra2 %>% filter(is.na(RENECO) | is.na(ACTIVO)) %>%
  select(RENECO, ACTIVO)
```

El operador | significa "o".

Con el código anterior se obtiene en la consola:

```
##                              RENECO ACTIVO
## Viesgo Renovables SL.            NA 269730
## Sargon Energias SLU              NA  85745
## La Caldera Energia Burgos SL  2.643     NA
```

Como ya se discutió anteriormente, si comprobamos que los *missing values* no pueden ser obtenidos o estimados mediante alguna otra vía, y son relativamente pocas observaciones, se podría optar por eliminarlos. Vamos a suponer en este ejemplo, que ese es el caso:

```
muestra2 <- muestra2 %>% filter(! is.na(RENECO) & ! is.na(ACTIVO) )
```

El operador & significa "y".

El *data frame* "muestra2" contiene los mismos datos que "eolica_100", salvo los tres casos con *missing values* (97).

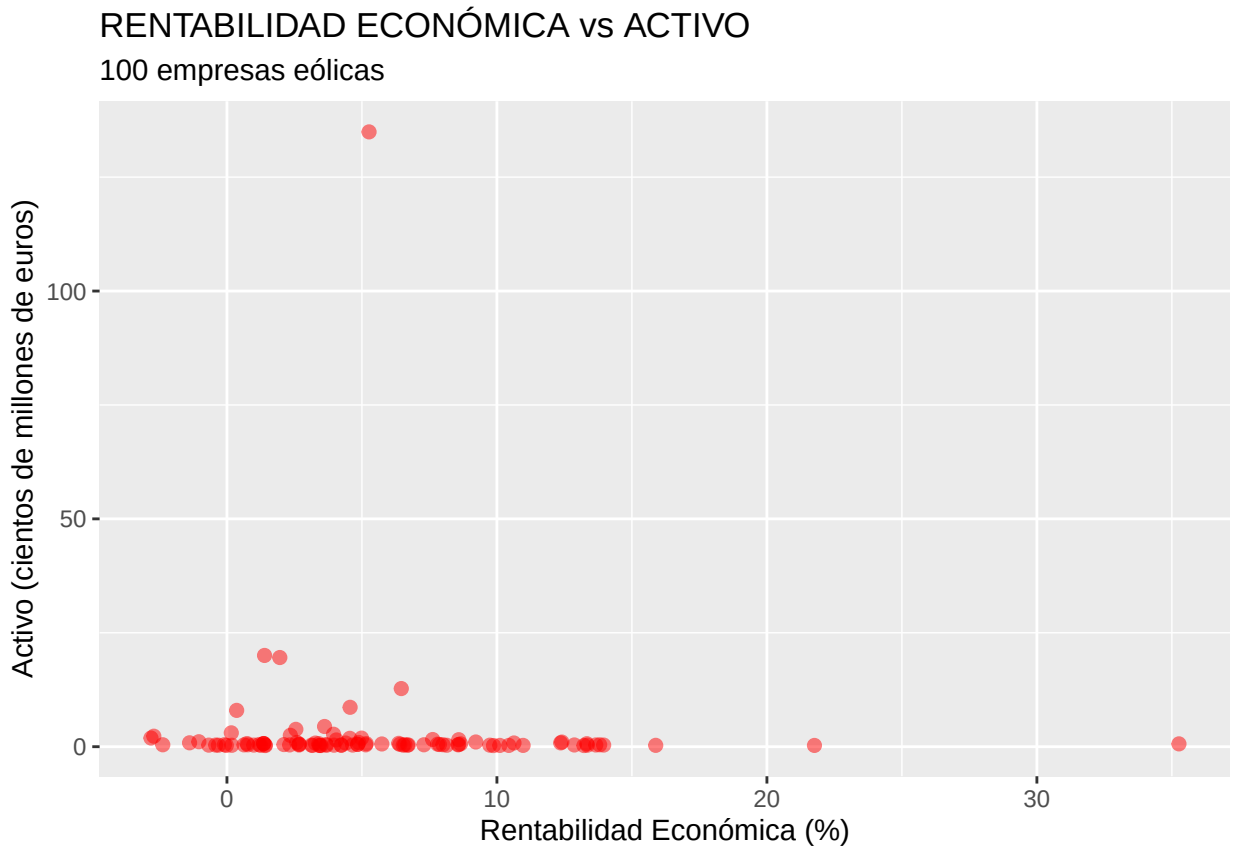
Se pasaría ahora a detectar los posibles *outliers*. Al trabajar con dos variables, una posibilidad cómoda para extraer una conclusión inicial es generar un **gráfico de dispersión**. Utilizando la gramática de ggplot2:

```
# Localizando outliers

dispersion <- ggplot(data = muestra2, map = (aes(x = RENECO,
                                                 y = ACTIVO/100000))) +
```

```
              geom_point(colour = "red", size = 2, alpha = 0.5) +
              ggtitle("RENTABILIDAD ECONÓMICA vs ACTIVO",
                      subtitle = "100 empresas eólicas") +
              xlab("Rentabilidad Económica (%)") +
              ylab("Activo (cientos de millones de euros)")
dispersion
```

Puede apreciarse cómo en el "mapeo" de las coordenadas, el ACTIVO aparece divido entre 100.000. Esto es simplemente para aminorar la escala del eje "y", y dotar de una escala más cómoda (visible) al gráfico. Eso ha hecho que en la etiqueta de este gráfico pongamos "cientos de milllones de euros", en lugar de "miles de euros", que son las unidades de la variable ACTIVO. Por otro lado, el gráfico se ha asignado, primeramente, al objeto "dispersion", al que se ha llamado con posterioridad.



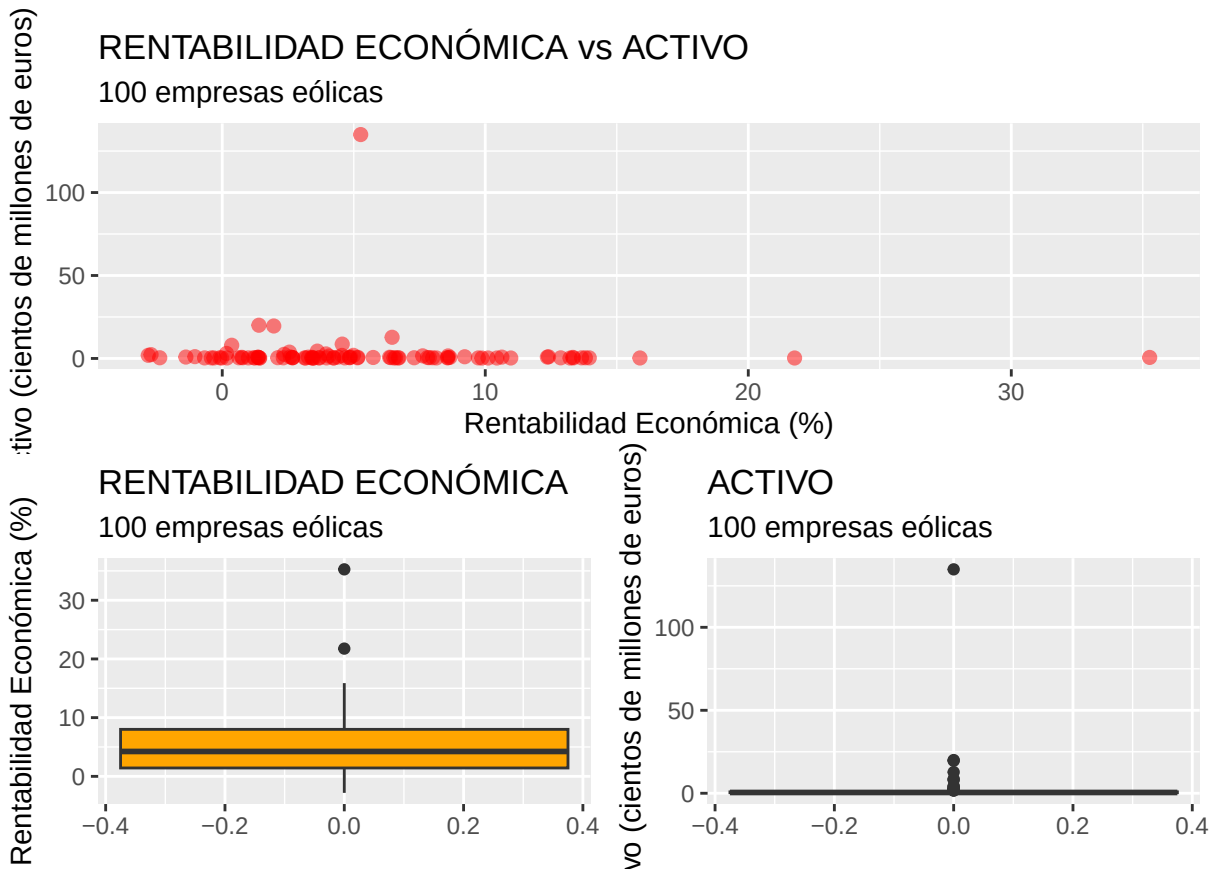
En el gráfico se distinguen varios puntos muy alejados del resto, que son candidatos a ser *outliers*.

Además, podríamos crear los gráficos de caja para cada una de las variables. Estos gráficos los vamos a asignar a dos objetos, "caja_RENECO" y "caja_ACTIVO". Posteriormente, con las facilidades del paquete `{patchwork}` se combinarán los tres gráficos, a fin de proporcionar una presentación más compacta. El código es el siguiente:

```
caja_RENECO <- ggplot(data = muestra2, map = (aes(y = RENECO))) +
               geom_boxplot(fill = "orange") +
               ggtitle("RENTABILIDAD ECONÓMICA",
                       subtitle = "100 empresas eólicas") +
               ylab("Rentabilidad Económica (%)")
```

```
caja_ACTIVO <- ggplot(data = muestra2, map = (aes(y = ACTIVO/100000))) +
               geom_boxplot(fill = "orange") +
               ggtitle("ACTIVO",
                       subtitle = "100 empresas eólicas") +
               ylab("Activo (cientos de millones de euros)")
library (patchwork)
dispersion / (caja_RENECO | caja_ACTIVO)
```

Y el resultado:



La última línea de código, mediante el paquete `{patchwork}`, maqueta la visualización de los tres gráficos. El operador `/` indica que los gráficos siguientes se dispondrán inmediatamente debajo; mientras que `|` indica que el gráfico siguiente se dispone al lado del anterior.

Lo más destacable es, en el caso de la variable ACTIVO, cómo el diagrama de caja confirma la existencia de *outliers*, con un caso muy claro.

Vamos a suponer que se decide eliminar los *outliers* de la muestra, a fin de evitar distorsiones en los resultados de la posterior aplicación de alguna técnica. El código para localizar los *outliers* será:

```
Q1_RENECO <- quantile (muestra2$RENECO, c(0.25))
Q3_RENECO <- quantile (muestra2$RENECO, c(0.75))
Q1_ACTIVO <- quantile (muestra2$ACTIVO, c(0.25))
Q3_ACTIVO <- quantile (muestra2$ACTIVO, c(0.75))

muestra2 %>%
  filter(RENECO > Q3_RENECO + 1.5*IQR(RENECO) |
```

```
           RENECO < Q1_RENECO - 1.5*IQR(RENECO) |
           ACTIVO > Q3_ACTIVO + 1.5*IQR(ACTIVO) |
           ACTIVO < Q1_ACTIVO - 1.5*IQR(ACTIVO)) %>%
  select(RENECO, ACTIVO)
```

Con lo que, en la consola, se listarán los siguientes casos:

```
##                                RENECO      ACTIVO
## Holding De Negocios De GAS SL.  5.264 13492812.00
## Global Power Generation SA.     1.393  2002458.00
## Naturgy Renovables SLU          1.959  1956869.00
## EDP Renovables España SLU       6.458  1275939.00
## Corporacion Acciona Eolica SL   4.562   864606.00
## Saeta Yield SA.                 0.360   796886.38
## Elawan Energy SL.               3.615   443467.00
## Olivento SL                     2.553   381206.98
## Parque Eolico La Boga SL.       0.162   303904.36
## Naturgy Wind, S.L.              3.949   273542.00
## Al-Andalus Wind Power SL        2.349   249853.83
## Innogy Spain SA.               -2.708   230338.51
## Guzman Energia SL              -2.813   190286.98
## Acciona Eolica Del Levante SL   4.985   188354.00
## Biovent Energia SA              4.551   183899.00
## Esquilvent SL                   7.621   157630.62
## Molinos Del Ebro SA            35.262    62114.37
## Sierra De Selva SL             21.761    27728.00
```

Son 18 casos. Vamos a eliminarlos de la muestra, pero creando un nuevo *data frame*, "muestra2_so", para conservar el anterior:

```
muestra2_so <- muestra2 %>%
  filter(RENECO <= Q3_RENECO + 1.5*IQR(RENECO) &
           RENECO >= Q1_RENECO - 1.5*IQR(RENECO) &
           ACTIVO <= Q3_ACTIVO + 1.5*IQR(ACTIVO) &
           ACTIVO >= Q1_ACTIVO - 1.5*IQR(ACTIVO))
```

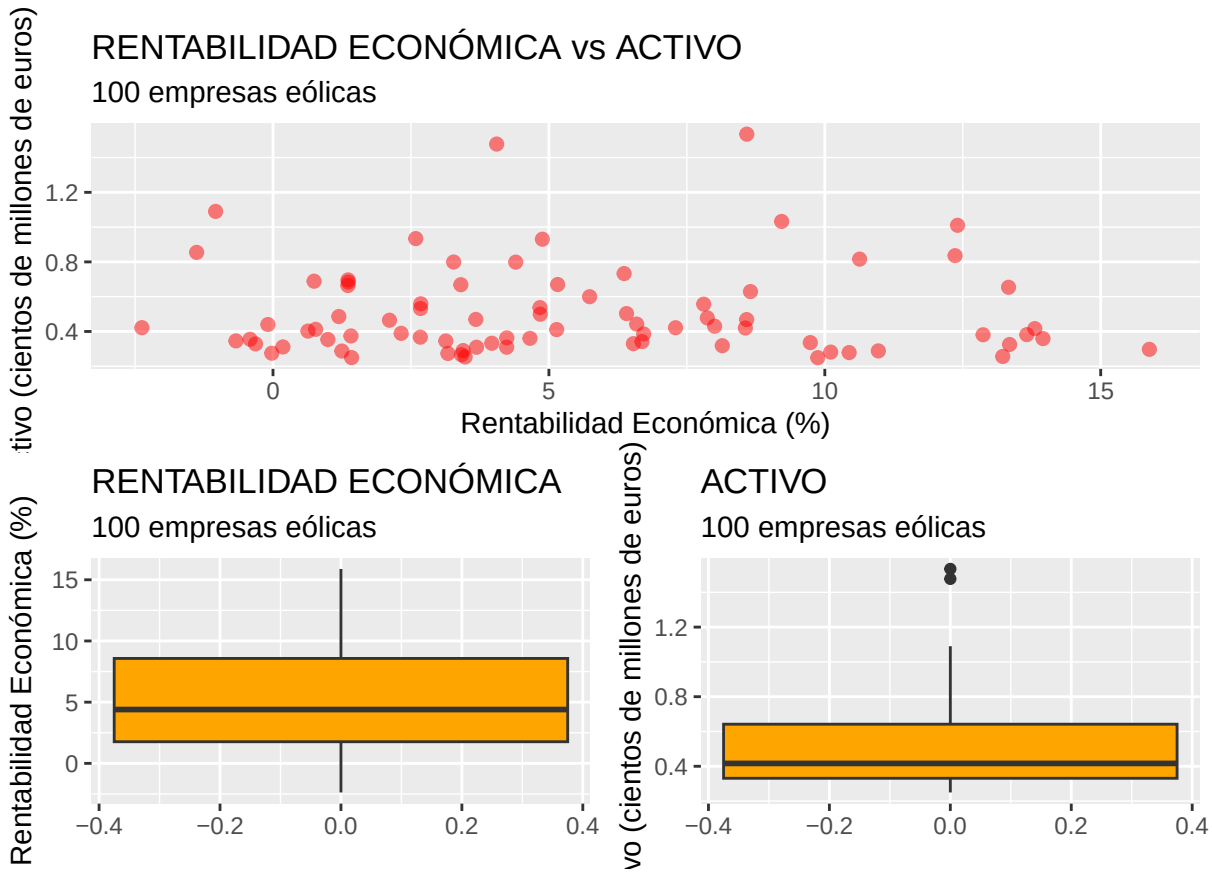
Si se construyen ahora los gráficos anteriores; pero con "muestra2_so", que ya no incluye *outliers*, se comprobará el cambio que experimentan tales gráficos:

```
dispersion_so <- ggplot(data = muestra2_so, map = (aes(x = RENECO,
                                                       y = ACTIVO/100000))) +
  geom_point(colour = "red", size = 2, alpha = 0.5) +
  ggtitle("RENTABILIDAD ECONÓMICA vs ACTIVO",
          subtitle = "100 empresas eólicas") +
  xlab("Rentabilidad Económica (%)") +
  ylab("Activo (cientos de millones de euros)")

caja_RENECO_so <- ggplot(data = muestra2_so, map = (aes(y = RENECO))) +
  geom_boxplot(fill = "orange") +
  ggtitle("RENTABILIDAD ECONÓMICA",
          subtitle = "100 empresas eólicas") +
  ylab("Rentabilidad Económica (%)")
```

```
caja_ACTIVO_so <- ggplot(data = muestra2_so, map = (aes(y = ACTIVO/100000))) +
  geom_boxplot(fill = "orange") +
  ggtitle("ACTIVO",
          subtitle = "100 empresas eólicas") +
  ylab("Activo (cientos de millones de euros)")

dispersion_so / (caja_RENECO_so | caja_ACTIVO_so)
```



Aunque vuelven a aparecer dos *outliers* en la variable ACTIVO, en el gráfico de dispersión se aprecia una nube de puntos más homogénea que antes de haber eliminado los casos u observaciones localizados con anterioridad.

5.3.2 Caso de múltiples variables.

Si las variables que entran en nuestro análisis son más de dos (suponemos que todas están en escala métrica), la detección y, en su caso, eliminación de *missing values* será un proceso semejante al de los casos anteriores.

Por ejemplo, vamos a imaginar que queremos realizar un análisis en el que tendremos en cuenta las variables RENECO (rentabilidad económica), ACTIVO (volumen de activos de la empresa), MARGEN (margen de beneficio) y RES (resultado del ejercicio).

Para detectar los *missing values* procederemos creando una copia del *data frame* original (para preservarlo), llamada "muestra3" y ejecutando el código:

```
muestra3 <- select(eolica_100, everything())
muestra3 %>% filter(is.na(RENECO) |
                      is.na(ACTIVO) |
                      is.na(MARGEN) |
                      is.na(RES)) %>%
  select(RENECO, ACTIVO, MARGEN, RES)
```

El listado de casos con *missing values* será:

```
##                               RENECO    ACTIVO   MARGEN       RES
## Viesgo Renovables SL.             NA 269730.00   11.818  4609.000
## Biovent Energia SA             4.551 183899.00   22.792        NA
## Sargon Energias SLU               NA  85745.00 -615.625 -2216.000
## Parc Eolic Sant Antoni SL      1.361  69654.00       NA   668.000
## Eolica La Brujula SA           7.295  42146.98       NA  2306.062
## La Caldera Energia Burgos SL   2.643        NA   14.448   511.304
```

Para eliminar estos casos (siempre que no se hayan podido obtener por otra vía o estimar) utilizaremos el código:

El *data frame* "muestra3" contiene los mismos datos que "eolica_100", salvo los 6 casos con *missing values* (94).

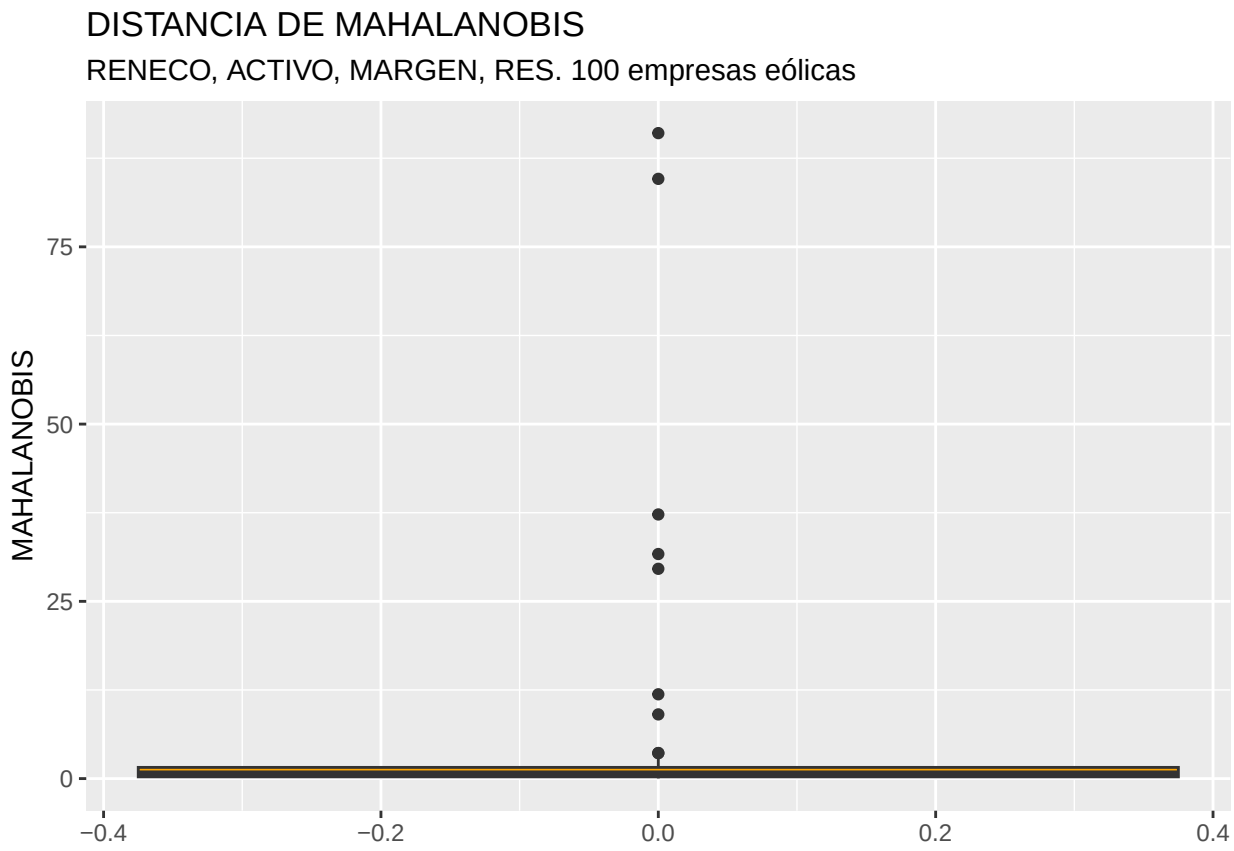
Para la detección de posibles *outliers*, al haber más de 2 variables, ya no puede utilizarse un gráfico de dispersión, porque implicaría más de dos ejes. Además, si las variables que entran en el análisis son numerosas, podría ser poco operativo estudiar las variables una a una. Una alternativa consiste en calcular la ***distancia de Mahalanobis*** de las variables del estudio, como **"resumen" del comportamiento de cada caso** en todas las variables del análisis. Así, primero vamos a calcular un vector con los valores de la *distancia de Mahalanobis* del conjunto de las 4 variables en cada uno de los casos (empresas eólicas). Este vector lo denominaremos, por ejemplo, MAHALANOBIS. Una vez calculado, lo añadiremos al *data frame* "muestra3" mediante la función de pegado de columnas, `cbind()`:

```
#Detectando y eliminando outliers.

muestra3.variables <- muestra3 %>% select(RENECO, ACTIVO, MARGEN, RES)
MAHALANOBIS <-mahalanobis(muestra3.variables,
                          center = colMeans(muestra3.variables),
                          cov = cov(muestra3.variables))
muestra3 <- cbind(muestra3, MAHALANOBIS)
```

Posteriormente, se puede construir el diagrama de caja de la variable MAHALANOBIS, como cualquier otra variable:

```
ggplot(data = muestra3, map = (aes(y = MAHALANOBIS))) +
    geom_boxplot(fill = "orange") +
    ggtitle("DISTANCIA DE MAHALANOBIS",
            subtitle = "RENECO, ACTIVO, MARGEN, RES. 100 empresas eólicas ") +
    ylab("MAHALANOBIS")
```



Para saber de qué casos concretos se trata, se podrá ejecutar el código:

```
Q1M <- quantile (muestra3$MAHALANOBIS, c(0.25))
Q3M <- quantile (muestra3$MAHALANOBIS, c(0.75))

muestra3 %>%
  filter(MAHALANOBIS > Q3M + 1.5*IQR(MAHALANOBIS) |
           MAHALANOBIS < Q1M - 1.5*IQR(MAHALANOBIS)) %>%
  select(MAHALANOBIS, RENECO, ACTIVO, MARGEN, RES)
```

En la consola se obtendrá el listado:

```
##                               MAHALANOBIS RENECO      ACTIVO
## Holding De Negocios De GAS SL.   91.041690  5.264 13492812.00
## Global Power Generation SA.      37.255573  1.393  2002458.00
## Naturgy Renovables SLU           31.675561  1.959  1956869.00
## Saeta Yield SA.                  11.891027  0.360   796886.38
## Molinos Del Ebro SA              29.589696 35.262    62114.37
## Tarraco Eolica SA                 3.600426 12.868    38102.00
## WPD Parque Eolico Navillas SL.   84.589929 -0.416    35511.45
## Brulles Eolica SL                 3.599069 15.882    29722.58
## Sierra De Selva SL                9.055155 21.761    27728.00
##
##                                  MARGEN         RES
## Holding De Negocios De GAS SL.    91.152 727548.0000
## Global Power Generation SA.       22.403  39995.0000
```

```
## Naturgy Renovables SLU            20.442  42737.0000
## Saeta Yield SA.                   16.258   2084.4760
## Molinos Del Ebro SA               41.821  17026.2569
## Tarraco Eolica SA                400.899   4953.0000
## WPD Parque Eolico Navillas SL. -2248.157   -110.9293
## Brulles Eolica SL                 47.227   3540.5693
## Sierra De Selva SL                47.045   4525.0000
```

Si se opta por eliminar estos casos cara al análisis posterior, se podrá crear un nuevo *data frame*, por ejemplo "muestra3_so", con el código siguiente:

```
muestra3_so <- muestra3 %>%
  filter(MAHALANOBIS <= Q3M + 1.5*IQR(MAHALANOBIS) &
           MAHALANOBIS >= Q1M - 1.5*IQR(MAHALANOBIS))
```

El *data frame* "muestra3_so" será una réplica de "muestra3", aunque sin incluir los casos detectados como atípicos o *outliers* (85 casos).

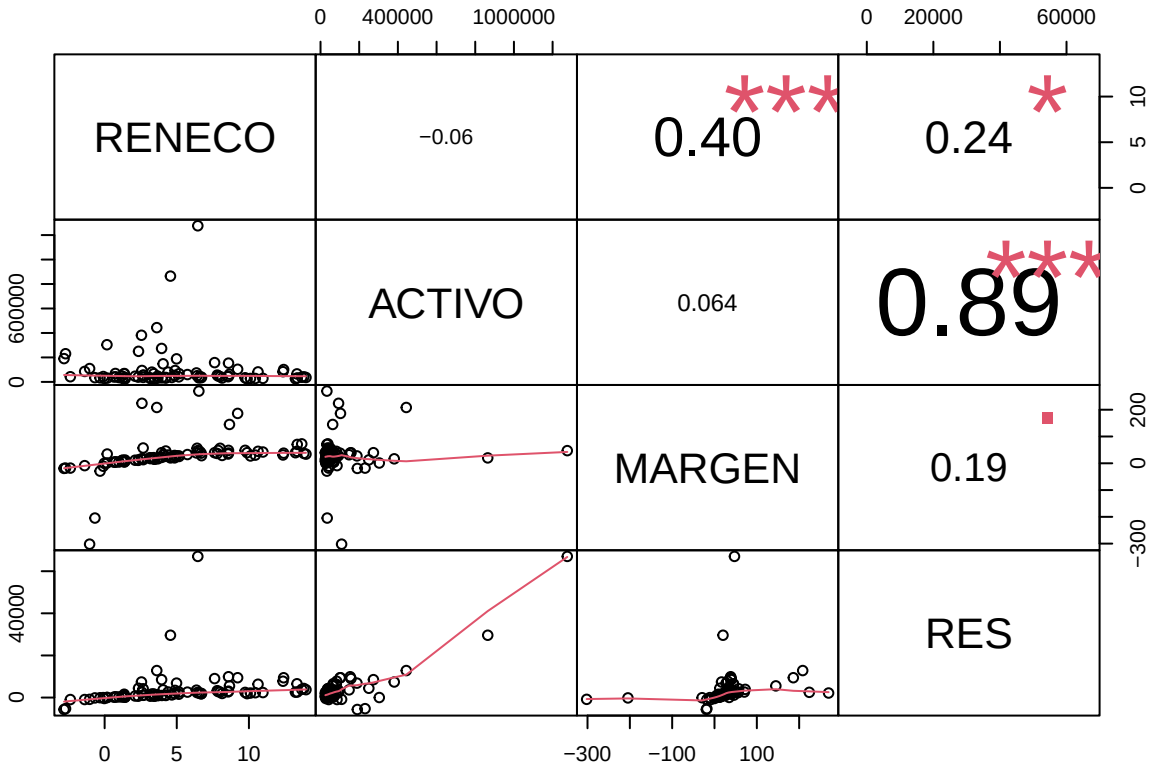
5.3.3 Correlación entre variables.

Cuando trabajamos con más de una variable, una característica muy importante viene dada por la intensidad con la que tales variables están relacionadas estadísticamente entre sí, es decir, el estudio de las correlaciones. Un modo atractivo y rápido de visualizar la **matriz de correlaciones** de las variables es a través de la función `chart.Correlation()` del paquete `{PerformanceAnalytics}`. Previamente, hemos creado el *data frame* "muestra3_so_variables" con sola las variables (métricas) del estudio:

```
# Correlaciones.

muestra3_so_variables <- muestra3_so %>%
  select(RENECO, ACTIVO, MARGEN, RES)

library(PerformanceAnalytics)
chart.Correlation(muestra3_so_variables, histogram = F, pch = 18)
```



Un coeficiente de correlación puede tomar un valor entre -1 (fuerte relación, en sentido opuesto) a 1 (fuerte relación, en el mismo sentido). Como puede apreciarse en el gráfico, las variables ACTIVO y RES mantienen una relación muy intensa y en sentido positivo. Entre MARGEN y RENECO existe también una relación de intensidad destacable. En cambio, ACTIVO y MARGEN; y RENECO y ACTIVO apenas están estadísticamente relacionadas.

5.4 Materiales para realizar las prácticas del capítulo.

En esta sección se muestran los links de acceso a los diferentes materiales (scripts, datos…) necesarios para llevar a cabo los contenidos prácticos del capítulo.

Datos (en formato Microsoft (R) Excel (R)):

- eolica_100_mv.xlsx (obtener aquí)

Scripts:

- explora_describe.R (obtener aquí)

Capítulo 6

Análisis de la varianza.

Capítulo 7

Componentes Principales.

Capítulo 8

Análisis Clúster.

8.1 Introducción.

El análisis de conglomerados o análisis clúster (AC) trata de clasificar individuos o casos asignándolos a grupos homogéneos, de manera que:

- Cada grupo, conglomerado o clúster contenga a **los casos más parecidos** entre sí, en términos de una serie de variables (**variables clasificadoras**).
- **Los grupos** contengan casos que, en general, **sean muy diferentes** a los casos del resto de grupos, de acuerdo con las variables consideradas.

En general, el proceso de determinación de los grupos, conglomerados o clústeres de casos es el siguiente:

- Se parte de un conjunto de **n** casos, y para cada uno de ellos se cuenta con el valor de **m** variables clasificadoras.
- Se establece una **medida de distancia** que cuantifica lo que dos casos se parecen, **considerando en conjunto** los valores que poseen para las variables clasificadoras.
- Se crean los grupos, conglomerados o clústeres con los casos que poseen entre sí una **menor distancia**. Existen dos **enfoques** principales a la hora de crear los grupos de casos a partir de las distancias observadas entre los casos: los *métodos jerárquicos* y los *métodos no-jerárquicos*.
- Finalmente, se **caracterizan** los grupos, conglomerados o clústeres obtenidos, y se comparan unos con otros para extraer conclusiones.

En lo que respecta a la medida de **distancia** entre los casos, la medida más habitual es la **distancia euclídea**. Así, la distancia euclídea entre dos caso, i e i', para las m variables clasificadoras x, será:

$$d(i,i') = \sqrt{\sum_{j=1}^{m}(x_{ij} - x_{i'j})^2}$$

Esta distancia es muy sensible a la escala de las variables clasificadoras. Para evitar este inconveniente, se trabaja con las variables previamente **tipificadas**.

8.2 Métodos de agrupación jerárquicos.

Como se acaba de comentar, existen dos enfoques fundamentales de realizar el análisis clúster, dependiendo de cómo son los métodos de agrupación de los casos (y grupos de casos): el enfoque de los métodos jerárquicos, y el enfoque que reúne a los métodos no-jerárquicos.

Ambos enfoques tienen sus ventajas e inconvenientes, y pueden adaptarse mejor a cada problema concreto. Es importante seleccionar un buen método de agrupación, puesto que pueden proporcionar soluciones muy diferentes entre sí.

En los **métodos jerárquicos,** se van formando sucesivamente grupos como agrupación de otros grupos precedentes, hasta llegar a un único grupo que recoge a todos los individuos; tomando el proceso una **estructura piramidal** (también existen métodos jerárquicos descendientes, que parten de un único grupo que contiene a todos los casos, para acabar el n grupos de un solo caso, aunque son menos frecuentes).

Estos métodos suelen aplicarse cuando hay un número reducido de casos. También, cuando nuestro objetivo pasa por crear **grupos que recojan a todos los casos**, más que definir simplemente tipologías más o menos homogéneas de casos (lo que se obtiene caracterizando los grupos obtenidos). Es decir, cuando se incluyen en el análisis a todos los individuos, incluidos los *outliers*. De hecho, estos métodos pueden emplearse, de por sí, como técnicas de localización de *outliers*. Por último, también se suelen emplearse cuando se desconoce a priori el número de grupos, conglomerados o clústeres a formar.

Entre los métodos jerárquicos de agrupación más extendidos, figuran los siguientes:

- **Método del vecino más cercano (single linkage):** la distancia que se considera entre grupos es la distancia entre sus elementos más próximos.
- **Método del vecino más lejano (complete linkage):** la distancia que se considera entre grupos es la distancia entre sus elementos más lejanos.
- **Método de Ward (Ward method):** se unen los grupos que dan lugar a otro grupo cuyos casos tienen una menor suma de los cuadrados de sus distancias respecto al centro de dicho grupo (menor varianza).
- **Otros métodos:** vinculación intergrupos (average linkage between groups), vinculación intragrupos (whithin group)…

De entre ellos, ¿cuál elegir?

La cuestión no es fácil de resolver, y no tiene por qué tener una única respuesta. Por otro lado, cada método proporciona soluciones que pueden variar mucho entre sí. Una estrategia puede pasar por probar con varios métodos y se seleccionar la solución que parezca más coherente desde el punto de vista teórico, y estable desde el punto de vista empírico.

En la práctica, uno de los métodos más utilizados es el **método de Ward**, porque proporciona grupos muy homogéneos, ya que se basa en la minimización de la varianza o dispersión de los elementos que componen cada grupo con respecto a su centro de gravedad o **centroide.** Precisamente, este método será aplicado en el ejemplo práctico que desarrollaremos en R a continuación.

Capítulo 9

Tablas de Contingencia.

Capítulo 10

Análisis de Correspondencias.

Bibliografía

Hernández-Alonso, J. 2000. *Economía Cuantitativa*. Síntesis.

Intriligator, R. G.; Hsiao, M. D.; Bodkin. 1996. *Econometric Models, Techniques and Applications*. Prentice-Hall.

Martín-Pliego, F. J. 2004. *Introducción a La Estadística Económica y Empresarial*. Thomson/Paraninfo.

Mood, F. A., A. M.; Graybill. 1963. *Introduction to the Theory of Statistics (2nd Edition)*. Mc Graw-Hill.

Peña, D. 1983. "La Influencia de Las Teorías de Newton y Darwin En El Nacimiento de La Estadística Matemática." *Estadística Española* 99: 103–4.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Samuelson, W. D., P. A.; Nordhaus. 2006. *Microeconomía (18ª Edición)*. Mc Graw Hill.

Wickham, Hadley. 2021. *Ggplot2: Elegant Graphics for Data Analysis, 3 Ed.* Springer-Verlag New York. https://ggplot2-book.org/.

Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. Paperback; O'Reilly Media. http://r4ds.had.co.nz/.