

# Fundamentos de ciencia de datos con R

Gema Fernández-Avilés y José-María Montero

2023-05-22



# Índice general

<b>Prefacio</b>	<b>5</b>
¡Hola mundo! . . . . .	5
¿Por qué este libro? . . . . .	6
¿A quién va dirigido? . . . . .	7
El paquete CDR . . . . .	8
¿Por qué <b>R</b> ? . . . . .	8
Agradecimientos . . . . .	9
<b>1. R para ciencia de datos</b>	<b>11</b>
1.1. Introducción . . . . .	11
1.2. La sesión de <b>R</b> . . . . .	12
1.3. Instalación de <b>R</b> . . . . .	13
1.4. Trabajar con proyectos de RStudio . . . . .	14
1.5. Tratamiento de datos con <b>R</b> . . . . .	14
1.6. Organización de datos con el <i>tidyverse</i> . . . . .	19



# Prefacio

## ¡Hola mundo!

El siglo XXI está siendo testigo de grandes cambios vertiginosos en el contexto social y tecnológico, entre otros. Los tiempos han cambiado, la sociedad se ha globalizado y “exige” respuestas inmediatas a problemas muy complejos. Vivimos en el mundo de la **información**, de los **datos**, o mejor, de las **bases de datos masivas**, y los ciudadanos y, sobre todo, las empresas y los gobiernos, dirigen su mirada hacia el mundo científico para que les ayude a “**oír las historias**” que cuentan esos datos acerca de la realidad de la que han sido extraídos. Y dado su enorme volumen y sofisticación (en el nuevo mundo las imágenes y los textos, por ejemplo, también son datos), exigen algoritmos de nueva generación en el campo del *machine learning*, o incluso del *deep learning*, para “oír las historias” que cuentan. No parecen mirar al “antiguo” investigador científico, sino al “nuevo” *científico de datos*.

Ello, inevitablemente, se traduce en la necesidad de profesionales con una gran capacidad de adaptación a este nuevo paradigma: los científicos de datos, también llamados por algunos los “nuevos hombres del Renacimiento”, para lo cual las Universidades y demás instituciones educativas especializada se apresuran a incluir el grado de Ciencia de Datos en su oferta educativa y a ofrecer seminarios de software estadístico de acceso abierto para sus estudiantes de primeros cursos.

Con la emergencia de la nueva sociedad, en la que el manejo de la ingente cantidad de información que genera se hace absolutamente necesario para circular por ella, la **Ciencia de Datos** ha venido para quedarse. Sin embargo, el mundo de la Ciencia de Datos es cualquier cosa menos sencillo. En él, cualquier ayuda, cualquier guía es bienvenida. Por ello, es muy recomendable que la persona que se quiera introducir en él, sea con fines de investigación o con fines profesionales, se agarre de la mano de un guía especializado que le lleve, de una manera amena, comprensible y eficiente, desde el planteamiento de su problema y la captura de la información necesaria para poderle dar una solución, hasta la redacción de las conclusiones finales que ha obtenido con los modernos informes reproducibles colaborativos. Y como en la parte central de ese camino tendrá que luchar con grandes gigantes (en la actualidad denominados técnicas estadísticas y algoritmos), el guía tendrá que explicarle, de manera sencilla y amena, en qué consiste la lucha (las técnicas y los algoritmos) y cómo llegar a la victoria lo más rápido posible, enseñándole a moverse por el mundo del software estadístico, en nuestro caso **R**, que le permitirá realizar los cálculos necesarios para vencer al problema planteado a una velocidad vertiginosa.

En resumen, la información masiva y el moderno tratamiento estadístico de la misma son la “mano invisible” que gobierna la sociedad del siglo XXI, y este manual pretende ser el guía anteriormente mencionado que le llevará de la mano cuando quiera caminar por ella.

## ¿Por qué este libro?

Lo dicho anteriormente ya justifica por sí solo la aparición de este manual. Afortunadamente, no es el primero en la materia, pues son ya bastantes los materiales de calidad publicados sobre Ciencia de Datos. Sin embargo, quizás, éste pueda ser considerado el más completo. Y ello por varias razones.

La primera es su **completitud**: este manual lleva de la mano al lector desde el planteamiento del problema hasta el informe que contiene la solución al mismo; o desde no saber qué hacer con la información de la que dispone, hasta ser capaz de transformar tales bases de datos masivas, y casi imposibles de manejar, en respuestas a problemas fundamentales de una empresa, institución o cualquier agente social.

La segunda es su **amplitud temática**:

- (I) Parte de las dos primeras preguntas que un neófito se puede hacer sobre esta temática: ¿qué es eso de la Ciencia de Datos que está en boca de todos? Y, ¿qué diablos es **R** y cómo funciona?
- (II) Enseña cómo moverse en la jungla del *Big Data* y de los “nuevos” tipos de datos, siempre bajo el paraguas de la ética de los datos y del buen gobierno de dichos datos.
- (III) Muestra al lector cómo obtener conocimiento de la oscuridad del enorme banco de información a su disposición, que no sabe cómo abordar ni manejar.
- (IV) No deja a nadie atrás, y de forma previa al contenido central del manual (las técnicas de Ciencia de Datos), incluye unas breves, pero magníficas, secciones sobre los rudimentos de la probabilidad, la inferencia estadística y el muestreo, para aquéllos no familiarizados con estas cuestiones.
- (V) Aborda una treintena de técnicas de Ciencia de Datos en el ámbito de la modelización, análisis de datos cualitativos, discriminación, *machine learning* supervisado y no supervisado, con especial incidencia en las tareas de clasificación y clusterización -así como, en el caso no supervisado, de reducción de la dimensionalidad, escalamiento multidimensional y análisis de correspondencias-, *deep learning*, análisis de datos textuales y de redes, y, finalmente, ciencia de datos espaciales (desde las perspectivas de la geoestadística, la econometría espacial y los procesos de punto).
- (VI) Hace especial hincapié en la reproducibilidad en tiempo real (o no) entre los distintos miembros de un equipo (sea universitario, empresarial, o del tipo que sea) y en la difusión de los resultados obtenidos, enseñando al lector cómo generar informes reproducibles mediante RMarkdown y documentos Quarto o en otros modernos formatos.
- (VII) Dedica un capítulo a la creación de aplicaciones web interactivas (con Shiny).

## Índice general

7

- (viii) Para aquéllos con pasión por la codificación, y que quieran compartir código y colaborar con otros desarrolladores, este manual aborda la gestión rápida y eficaz de proyectos (del tamaño que sean) mediante Git, un sistema de control de versiones distribuido, gratuito y de código abierto, y GitHub, un servicio de alojamiento de repositorios Git del cual, aquellos no familiarizados con la cuestión de la codificación, o con aversión a ella, podrán tomar el código que necesitan.
- (ix) Muestra al lector los primeros pasos para iniciarse en el geoprocесamiento en la nube.
- (x) Y, finalmente, aborda más de una docena de casos de uso (en medicina, periodismo, economía, criminología, marketing, moda, demanda de electricidad, cambio climático, reconocimiento de patrones en la forma de tuitear...) que ilustran la puesta en práctica de todos los conocimientos anteriormente adquiridos.

La cuarta razón es que todo lo que el lector aprende en este manual lo puede reproducir y poner en práctica inmediatamente con **R**, puesto que el manual está trufado de *chunks* (o trozos de código **R**) que no tiene más que cortar y pegar para reproducir los ejemplos que se muestran en el libro, cuyos datos están en el paquete CDR; o utilizar dichas *chunks* para abordar el problema que le ocupa con los datos que tenga a su disposición. Una buena razón, sin duda. Por consiguiente, el manual es una buena combinación “teoría-práctica-software” que permite abordar cualquier problema que el científico de datos se plante en cualquier disciplina o situación empresarial, médica, periodística...

La quinta es su **variedad de perspectivas**. Son **más de 40 los participantes** en este manual. Algunos de ellos, prestigiosos profesores universitarios; otros, destacados miembros de instituciones públicas; otros, CEOs de empresas en la órbita de la ciencia de datos; otros, *big names* del mundo de **R** software... El manual es, sin duda, un magnífico ejemplo de colaboración Universidad-Empresa para buscar soluciones a los problemas de las sociedades modernas.

## ¿A quién va dirigido?

*Fundamentos de ciencia de datos con R* está dirigido a todos aquellos que desean desarrollar las habilidades necesarias para abordar proyectos complejos de Ciencia de Datos y “pensar con datos” (como lo acuñó Diane Lambert, de Google). El deseo de resolver problemas utilizando datos es su piedra angular. Por tanto, como se avanzó anteriormente, este manual no deja a nadie atrás, y lo único que requiere es “el deseo de resolver problemas utilizando datos”. No excluye ninguna disciplina, no excluye a las personas que no tengan un elevado nivel de análisis estadístico de datos, no excluye a nadie. Se ha procurado una combinación de rigor y sencillez, y de teoría y práctica, todo ello con sus correspondientes códigos en **R**, que satisfaga tanto a los más exigentes como a los principiantes.

También está destinado a todos aquellos que quieran sustituir la navegación por la web (la búsqueda del video, publicación de blog o tutorial *online* que solucione su problema –frustración tras frustración por la falta de consistencia, rigor e integridad de dichos materiales, así como por su sesgo hacia paquetes singulares para la implementación de las cuestiones que tratan–), por

una “**biblia de la ciencia de datos**” rigurosa pero sencilla, práctica y de aplicación inmediata sin ser ni un experto estadístico ni un experto informático.

Pero si a alguien está destinado especialmente, es a la comunidad hispano hablante. Este manual es un guiño a dicha comunidad, para que tenga a su disposición, en su lengua nativa, uno de los mejores manuales de Ciencia de Datos de la actualidad.

## El paquete CDR



El paquete **CDR** contiene la mayoría de conjuntos de datos utilizados en este libro que no están disponibles en otros paquetes. Para instalarlo use la función `install_github()` del paquete `remotes`.

```
# este comando sólo necesita ser ejecutado una vez
# si el paquete remotes no está instalado, descomentar para instalarlo

# install.packages("remotes")
remotes::install_github("cdr-book/CDR")
```

La lista de todos los conjuntos de datos puede obtenerse haciendo `data()`.

```
library('CDR')
data(package = "CDR")
```

Este paquete ayudará al lector a reproducir todos los ejemplos del libro. De acuerdo con las mejores prácticas en **R**, el paquete **CDR** sólo contiene los datos utilizados en el libro.

## ¿Por qué R?

**R** es un lenguaje de código abierto para computación estadística que se ha consolidado entre la comunidad científica internacional, en las últimas dos décadas, como una herramienta de primer

## Índice general

9

nivel, consolidándose como líder permanente en el ámbito de la implementación de metodologías estadísticas para el análisis de datos. La utilidad de **R** para la Ciencia de Datos deriva de un fantástico ecosistema de paquetes (activo y en crecimiento), así como de un buen elenco de otros excelentes recursos: libros, manuales, *blogs*, foros y *chats* interactivos en las redes sociales, y una gran comunidad dispuesta a colaborar, a orientar y a resolver diferentes cuestiones relacionadas con **R**.

Por otra parte, **R** es el lenguaje estadístico y de análisis de datos más utilizado en la mayoría de los entornos académicos y, cómo no, por una larga lista de importantes empresas, entre las que se cuentan Facebook (análisis de patrones de comportamientos relacionado con actualizaciones de estado e imágenes de perfil), Google (para la efectividad de la publicidad y la previsión económica), Twitter (visualización de datos y agrupación semántica), Microsoft (adquirió la empresa Revolution R), Uber (análisis estadístico), Airbnb (ciencia de datos), IBM (se unió al grupo del consorcio R), New York Times (visualización)...

La comunidad **R** también es particularmente generosa e inclusiva, y hay grupos increíbles, como *R-Ladies* y *Minority R Users*, diseñados para ayudar a garantizar que todos aprendan y usen las capacidades de **R**.

## Agradecimientos

No queremos dar por finalizado este prefacio sin agradecer a los 44 autores participantes en esta obra su esfuerzo por condensar, en no más de 20 páginas, la teoría, práctica y tratamiento informático de la parte de la Ciencia de Datos que les fue encargada. Y no sólo eso; el “más difícil todavía” fue que debían dirigirse a un abanico de potenciales lectores tan grande como personas haya con “el deseo de resolver problemas utilizando datos”. Era misión imposible. Sin embargo, a la vista del resultado, ha sido misión cumplida. El esfuerzo mereció la pena.

Además, nos gustaría agradecer el apoyo incondicional recibido por (en orden alfabético): Itzcoatl Bueno, Ismael Caballero, Emilio L. Cano, Diego Henangómez, Ricardo Pérez, Manuel Vargas y Jorge Velasco.

También queremos poner de manifiesto que la edición de este texto ha sido financiada por diversos entes de la Universidad de Castilla-La Mancha. En su mayor parte, por el **Máster en Data Science y Business Analytics (con R software)** (a través de la orgánica: 02040M0280), pero también por la Facultad de Ciencias Jurídicas y Sociales de Toledo (a través de su contrato programa: orgánica 00440710), el Departamento de Economía Aplicada I (mediante sus fondos departamentales, DEAI 00421I126) y el Grupo de Investigación Economía Aplicada y Métodos Cuantitativos (que ha dedicado parte de sus fondos a la edición de esta obra, orgánica 01110G3044-2023-GRIN-34336).

A todos, eternamente agradecidos por ayudarnos en este reto de transformar la oscuridad en conocimiento, de convertir en una ciencia y en un arte la difícil tarea de sacar valor de los datos, el petróleo del futuro. Quizás en este momento no seamos conscientes de que hemos puesto nuestro granito de arena a la ciencia que, a buen seguro, juegue uno de los papeles más importantes de este siglo, caracterizado por el predominio de la información. Una ciencia, la Ciencia de Datos, que combina el análisis estadístico de datos, la algoritmia y el conocimiento del

negocio para sacar valor del bien más abundante de la sociedad en la que vivimos: la información. Una disciplina cuyo dominio caracteriza a los científicos de datos (también denominados los nuevos personajes del Renacimiento), profesión que ya fue calificada hace más de veinte años en la *Harvard Business Review* y en *The New York Times*, entre otros, como la “más sexy del siglo XXI”.

### Nota

Este manual está publicado por [McGraw Hill](#). Las copias físicas están disponibles en [McGraw Hill](#). La versión *online* se puede leer de forma gratuita en <https://cdr-book.github.io/> y tiene la [licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional](#).

Si tiene algún comentario o sugerencia, no dude en contactar con los editores y los autores. ¡Gracias!

# Capítulo 1

## R para ciencia de datos

*Emilio L. Cano*

Universidad Rey Juan Carlos

### 1.1. Introducción

El análisis estadístico de datos es una tarea fundamental en la transformación digital de las empresas y organizaciones. Siempre ha estado ahí, pero en la actualidad la disponibilidad de datos, la cantidad de los mismos, y la velocidad con la que se requieren resultados, está haciendo necesario el capacitar a los profesionales para su análisis con nuevas herramientas. Nuevas tendencias (muchas veces malinterpretadas) como Inteligencia Artificial, *Big Data*, Industria 4.0, *Internet of Things* (IoT), o *Data Science*, aumentan el interés por parte de las empresas, los profesionales y los investigadores en estas técnicas.

El tratamiento de datos y su análisis requiere el uso de software avanzado. Aunque algunas tareas como, por ejemplo, mecanizar y almacenar datos, se pueden realizar eficazmente con programas de hoja de cálculo como Excel, se debería utilizar software especializado para el análisis de datos. Existen distintos paquetes estadísticos comerciales, como SPSS, Statgraphics, Stata, SAS, JMP o Minitab. En los últimos años se ha abierto camino como alternativa el [software estadístico y lenguaje de programación R](#) ([R Core Team, 2021](#)). Hay otras alternativas que, en su mayoría, o son parciales, referidas a un ámbito concreto, o son más lenguajes de programación que software estadístico, como Python. **R** es software libre, pero su gratuidad sólo es una de sus ventajas, como se verá a lo largo del libro. Su gran inconveniente es la curva de aprendizaje: no es tan fácil de aprender y usar como un software de ventanas, ya que el uso de **R** se basa en expresiones que hay que ejecutar desde *scripts* (archivos de código).

**R** es un sistema para **computación estadística**: software de **análisis de datos y lenguaje de programación**. Ha sido ampliamente utilizado en investigación y docencia, y actualmente también en las empresas y organismos públicos. Es la evolución del trabajo de los laboratorios Bell con el lenguaje S ([Venables and Ripley, 2002](#)), llevado al mundo del software libre por Ross

Ihaka y Robert Gentleman en los años 90 ([Ihaka and Gentleman, 1996](#)). La versión R 1.0.0 se publicó el 29 de febrero de 2000.

Uno de los aspectos más espectaculares de **R** es la cantidad de **paquetes** disponibles. Un paquete (*package*) de **R** es un componente con funcionalidad adicional que se puede instalar en el sistema para ser utilizado por **R**. En el momento de compilar este libro, el número de paquetes disponibles en el repositorio oficial es de 19549.

Una vez conocido el mundo de **R**, se plantea la siguiente pregunta: ¿y por qué utilizar **R**? Es imposible dar un único motivo. A continuación se enumeran algunos de ellos:

- Es Free and Open Source Software (FOSS). Gratis y libre. En inglés se suele decir *free as in free beer, and free as in free speech*.
- Tiene una amplia comunidad de usuarios que proporciona recursos.
- Es multiplataforma.
- Se usa cada vez en más empresas e instituciones.
- Es posible obtener soporte comercial, por ejemplo a través de Posit Software PBC<sup>1</sup>.
- Se ha alcanzado una masa crítica de usuarios que lo hace confiable.
- Es extensible (desde pequeñas funciones, hasta paquetes).
- Se puede implementar la innovación inmediatamente. En software comercial hay que esperar a nuevas versiones, en el mejor de los casos.
- Posee características de “investigación reproducible”. En el Cap. ?? se tratará qué implica este enfoque. En contextos distintos a la investigación, se puede hablar de informes reproducibles y trazabilidad del análisis.

Por otra parte, el uso de **R** en las empresas está **creciendo exponencialmente** debido, principalmente, a la necesidad de analizar y visualizar datos con herramientas potentes para explotar todo su potencial. Grandes empresas de todos los sectores llevan tiempo utilizándolo, si bien la popularización del software y su conocimiento entre los nuevos titulados está facilitando que empresas de todo tipo y tamaño aprovechen esta herramienta en su estrategia digital. Así, además de la **visualización y presentación efectiva** de los datos, equipos bien formados pueden descubrir relaciones entre variables clave, realizar **predicciones**, tomar mejores decisiones o **mejorar sus procesos** gracias al análisis avanzado de datos más allá de la hoja de cálculo.

## 1.2. La sesión de R

**R** es una aplicación de análisis estadístico y representación gráfica de datos, y además un lenguaje de programación. **R** es **interactivo**, en el sentido de que responde a través de un “intérprete” a las **entradas** que recibe a través de la **consola**.

La interfaz de usuario de **R** (R GUI, *Graphical User Interface*) cumple las funciones básicas para interactuar con **R**, pero es muy pobre a la hora de trabajar con ella. En su lugar, es más conveniente utilizar el entorno de desarrollo [RStudio Desktop](#) (o su versión en la nube

---

<sup>1</sup><https://posit.co>, antes RStudio PBC.

1.3. Instalación de **R**

13

<https://posit.cloud/>), que es como un “envoltorio” del sistema **R** con más funcionalidades y ayudas, pero manteniendo el mismo nivel de interacción: consola y *scripts*<sup>2</sup>. Al igual que **R**, RStudio es una aplicación de software libre, pero, en este caso, desarrollada y mantenida por la compañía privada Posit PBC.

Una cosa muy importante en **R** es que las expresiones son **sensibles a mayúsculas**, y por tanto los objetos **datos** y **Datos** son distintos.

### 1.3. Instalación de R

Durante todo el libro se utiliza la interfaz RStudio. Pero, como se avanzó anteriormente, RStudio es solo un “envoltorio” de **R**, por lo que previamente hay que tener instalado en el ordenador el sistema “base” de **R**. **R** está disponible para sistemas Windows, MacOS y Linux. Por cuestiones de espacio, no se incluyen detalles en este libro, pero la instalación es sencilla siguiendo las instrucciones en sus correspondientes websites:

1. Instalación de **R**: <http://www.r-project.org>
2. Instalación de RStudio: <https://posit.co>

Para completar la instalación de **R**, se muestra cómo instalar<sup>3</sup> los paquetes del **tidyverse**<sup>4</sup> mediante expresiones en la consola o *script* con la función `install.packages()`:

```
install.packages(pkgs = "tidyverse")
```

Una vez instalado el paquete, se cargará con la instrucción `library("nombre_paquete")` en la sesión de **R** donde se quiera utilizar.

```
library("tidyverse")
```

A veces resulta útil usar directamente la función que se va a utilizar en vez de cargar todo el paquete. Esto se hace con el operador `::`, es decir, `nombre_paquete::funcion()`. La siguiente expresión serviría para usar la función `select()` del paquete `dplyr` sin cargar el paquete entero.

```
dplyr::select()
```

---

<sup>2</sup>Lo importante es seguir un estilo consistente en cuanto a nombres de objetos, espacios en blanco y el uso de delimitadores y tabulación en el *script*. Véase por ejemplo la [guía de estilo de Hadley Wickam \(Wickham, 2015\)](#).

<sup>3</sup>Una vez instalado un paquete no hay que volver a instalarlo.

<sup>4</sup>El **tidyverse** es un conjunto de paquetes que se irán describiendo a medida que se utilicen, especialmente en la Sec. 1.6.

## 1.4. Trabajar con proyectos de RStudio

La manera más eficiente de trabajar con **R**, es mediante **proyectos** de RStudio. Esto permite abstraerse de los detalles de la sesión de **R** (espacio de trabajo, directorio de trabajo, *environment*), ya que al abrir un proyecto estará todo preparado para seguir el trabajo donde se dejó, o empezar de cero si se acaba de crear. Para crear un proyecto de RStudio, se despliega el menú de proyectos a la derecha en la barra de herramientas y se selecciona “New Project...”. También se puede hacer en el menú “File/New Project...”.

Es aconsejable crear siempre una estructura de carpetas que permita tener todo organizado desde el principio, porque al final los proyectos crecen. La estructura perfecta no existe, y depende del proyecto particular. Las siguientes carpetas pueden ser útiles en un amplio abanico de proyectos, y las tres primeras se pueden usar prácticamente en cualquier proyecto:

- **data**: en esta carpeta se tienen los archivos de datos, tanto aquellos orígenes de datos que se quieran importar, como los que se puedan guardar desde un *script*.
- **R**: para los *scripts*. Es posible que solamente haya un *script* en nuestro proyecto, pero si hubiera más se pueden guardar en esta carpeta.
- **inform**: aquí se pueden guardar los archivos Quarto o R Markdown que se utilicen para generar informes o presentaciones.
- **img**: si en nuestro proyecto se utilizan imágenes de cualquier tipo, es una buena idea tenerlas en una carpeta aparte.
- **test**: si se quieren separar los *scripts* que se utilicen para pruebas y no se quieren mezclar con los “buenos” en la carpeta **R**.
- **aux, tmp, util, notas, doc, ...**: este tipo de carpetas vienen bien cuando hay información que está relacionada o es útil para un proyecto, pero el archivo no es del proyecto de análisis de datos en sí. Por ejemplo, unas especificaciones de un producto o servicio, un artículo científico, fotografías de una fábrica, comunicaciones con clientes, etc.
- **ejercicios, practicas, ...**: si nuestro proyecto forma parte de una asignatura, curso, o similar.

Un aspecto importante cuando se trabaja en proyectos colaborativos es el control de versiones. Este tema se aborda en el Cap. ??.

## 1.5. Tratamiento de datos con R

En este apartado se van a empezar a utilizar expresiones de **R**. Las expresiones se escribirán en *scripts*, que pueden contener “comentarios” (texto que no se ejecutará) utilizando el símbolo “almohadilla” (#). Muchas de las expresiones que se usan son llamadas a funciones<sup>5</sup>. La ayuda de cualquier función se puede obtener en la consola usando la expresión `?function`, donde `function` es el nombre de la función u objeto del que se quiere obtener ayuda.

<sup>5</sup>Por motivos de espacio, no se incluyen mayores explicaciones de las mismas, pero se anima al lector a explorar la ayuda de cada una de ellas para comprender mejor su funcionamiento.

### 1.5.1. Estructuras y tipos de datos

Las estructuras y tipos de datos más frecuentes con las que se trabaja en **R** son las que se detallan a continuación.

**Tablas de datos.** Son colecciones de variables numéricas y/o atributos organizadas en columnas, en las que cada fila se corresponde con algún elemento en el que se han observado las características que representan las variables. La forma más común es el **data.frame**. Cada columna del **data.frame** es, en realidad, otra estructura de datos, en concreto, un **vector**. Un ejemplo de **data.frame** es el conjunto de datos **tempmin\_data** del paquete **CDR** que se analiza en el Cap. ?? y del que se muestran a continuación las primeras tres filas con la función **head()**.

```
library("CDR")
head(tempmin_data, 3)
#>      fecha indicativo tmin longitud latitud
#> 1 2021-01-06      4358X -4.7 -5.880556 38.95556
#> 2 2021-01-06      4220X -7.0 -4.616389 39.08861
#> 3 2021-01-06      6106X  4.7 -4.748333 37.02944
```

Un **data.frame** es un objeto de datos en dos dimensiones, en el que las filas son la dimensión 1 y las columnas la dimensión 2. Los datos se pueden “extraer” de un **data.frame** por filas, por columnas o por celdas. Para extraer una de las variables del **data.frame** se suele utilizar el operador **\$** después del nombre del **data.frame**, y a continuación el nombre de la variable.

El operador **<-** asigna al “símbolo” que hay a su izquierda el resultado de la expresión que hay a su derecha, y lo guarda con ese nombre en el espacio de trabajo<sup>6</sup>. Por ejemplo, la siguiente expresión extrae todas las filas de la columna **tmin** o, dicho de otra forma, el vector con todas las temperaturas mínimas registradas y lo guarda en el objeto **temp\_min**.

```
temp_min <- tempmin_data$tmin
```

**Vectores y matrices.** Ya se ha visto que una columna de una tabla de datos es un vector. También se pueden crear vectores con la función **c()** y los elementos del vector separados por comas. Una matriz es un vector organizado en filas y columnas. A modo de ejemplo, la primera de las siguientes expresiones crea un vector llamado **nombres** con dos cadenas de texto, y la segunda crea una matriz numérica llamada **coordenadas** a partir de las columnas 4 y 5 del conjunto de datos **tempmin\_data**. Nótese que la extracción de valores de un conjunto de datos o de una matriz se puede realizar también por sus índices de filas y columnas entre corchetes separados por una coma. En este caso se extraen todas las filas (pues no se especifica ninguna en la dimensión 1) de las columnas 4 y 5.

```
nombres <- c("longitud", "latitud")
coordenadas <- as.matrix(tempmin_data[, 4:5])
```

---

<sup>6</sup>También se puede utilizar el símbolo igual (**=**) para realizar asignaciones. No obstante, en el marco de este libro se recomienda el uso del operador específico **<-**.

**Factor.** Es un tipo especial de vector para representar variables categóricas (también denominadas atributos o factores). En general, una variable categórica suele tomar un número reducido de valores diferentes (categorías), identificados con etiquetas (*labels*) y que se llaman **niveles** del factor (*levels*). Un ejemplo es el dataset `dp_entr` del paquete CDR que se analiza en el Cap. ???. La columna `ind_pro11` es un indicador que toma los valores S y N, mientras que `des_nivel_edu` toma tres posibles valores.

```
dp_entr[1:5, c(1, 17)]
#>     ind_pro11 des_nivel_edu
#> 1      S      MEDIO
#> 497    N      MEDIO
#> 265    N      BASICO
#> 534    N      MEDIO
#> 415    N      BASICO
levels(dp_entr$des_nivel_edu)
#> [1] "ALTO"   "BASICO" "MEDIO"
```

**Listas.** Son estructuras de datos que contienen una colección de elementos indexados que, además, pueden tener un nombre. Pueden ser heterogéneas, en el sentido de que cada elemento de la lista puede ser de cualquier tipo.

A modo de ejemplo, se muestran los nombres del objeto `tempmax_data` del paquete CDR, que contiene 6 elementos de distintas clases.

```
names(tempmax_data)
#> [1] "ESP"           "ESP_utm"        "grd_sf"         "grd_sp"
#> [5] "temp_max_utm_sf" "temp_max_utm_sp"
```

**Fechas.** Son un tipo de datos especial que algunas veces provoca problemas al compartir datos entre programas. El conjunto de datos `tempmin_data` contiene la columna `fecha`, que puede convertirse de manera inmediata a tipo fecha (`Date`) porque viene en un formato estándar (véase la ayuda de `strptime` para especificar otros formatos). El paquete `lubridate` del *tidyverse* contiene funciones para hacer más fácil el trabajo con fechas.

```
tempmin_data$fecha<- as.Date(tempmin_data$fecha)
class(tempmin_data$fecha)
#> [1] "Date"
```

**Cadenas de texto.** Son estructuras de datos que aparecen en forma de vector de caracteres. La columna `indicativo` del conjunto de datos `tempmin_data` es un ejemplo de este tipo de datos. La ayuda de `?regexp` proporciona la información necesaria sobre cómo extraer texto con expresiones regulares, y la de `?paste` para aprender a unir cadenas de texto. El paquete `stringr` del *tidyverse* contiene funciones para facilitar el trabajo con cadenas de texto.

```
head(tempmin_data$indicativo)
#> [1] "4358X" "4220X" "6106X" "9698U" "4410X" "1331A"
```

### 1.5.2. Importación de datos

En el apartado anterior se han utilizado tablas de datos que están incluidas en un paquete de **R**. Pero lo habitual es que los datos se tengan que importar de fuentes externas, como ficheros. A continuación, se describen algunas de las formas de importar los tipos de ficheros más habituales<sup>7</sup>.

**Excel.** Sin duda una forma muy popular de organizar los datos en ficheros es mediante **hojas de cálculo como Microsoft Excel**. Hay varios paquetes con los que se puede trabajar con archivos de Excel. En este libro se utiliza el paquete `readxl` del *tidyverse*. Con la siguiente expresión se puede descargar un archivo Excel de ejemplo<sup>8</sup>.

```
download.file(url = "http://emilio.lcano.com/b/adr/p/datos/RRHH.xlsx",
              destfile = "data/RRHH.xlsx",
              mode = "wb")
```

Una vez el archivo está en el directorio de trabajo de la sesión de **R**, se puede importar su contenido al espacio de trabajo con la siguiente expresión:

```
rrhh <- readxl::read_excel("data/RRHH.xlsx")
```

**Texto.** Los **archivos de texto** son el formato más utilizado y conveniente para compartir datos. Es también muy común que el equipamiento o el software genere datos en formato de texto. Estos archivos suelen tener extensión `.csv` (*comma separated values*) o `.txt`, aunque pueden tener cualquier otra, o incluso no tener extensión. A modo de ejemplo, con la siguiente expresión se puede descargar un archivo csv.

```
download.file(url = "http://emilio.lcano.com/b/adr/p/datos/ejDatos.csv",
              destfile = "data/ejDatos.csv")
```

Si el archivo tiene extensión `.csv`, como el anterior, vendrá ya con una especificación muy concreta, pudiéndose usar directamente las funciones `read.csv()` o `read.csv2()` para tener la tabla de datos en el espacio de trabajo.

```
merma <- read.csv2("data/ejDatos.csv")
```

<sup>7</sup>Para poder reproducir los ejemplos, se debe tener una carpeta `data` en el directorio de trabajo.

<sup>8</sup>La función `download.file()` permite descargar cualquier archivo disponible en la `url` que se indique. Es obligatorio indicar el archivo de destino con el argumento `destfile`. Si el archivo no es de texto plano, se debe indicar `mode = "wb"`.

La función genérica de **R** para importar datos de texto es `read.table()`, que puede importar cualquier especificación cambiando los argumentos adecuados. Por ejemplo, la siguiente expresión tendría el mismo resultado que se ha obtenido con la función `read.csv2`<sup>9</sup>:

```
merma <- read.table(file = "data/ejDatos.csv",
                     header = TRUE,
                     sep = ";",
                     dec = ",",
                     fileEncoding = "utf-8")
```

Para saber cómo importar datos desde sistemas gestores de bases de datos véase el Cap. ??.

Hay infinidad de otras fuentes de las que se pueden importar datos a **R**. Por ejemplo, el paquete `rvest`, que forma parte del *tidyverse*, se puede utilizar para obtener datos de páginas web y otras fuentes de Internet, lo que se suele llamar *web scraping*. Por ejemplo, supóngase que se quiere importar la tabla con los datos de comunidades y ciudades autónomas españolas del enlace [https://www.ine.es/daco/daco42/codmun/cod\\_ccaa\\_provincia.htm](https://www.ine.es/daco/daco42/codmun/cod_ccaa_provincia.htm). Las siguientes expresiones importan esta tabla al conjunto de datos `ccaa_ine`.

```
library("rvest")
url <- "https://www.ine.es/daco/daco42/codmun/cod_ccaa_provincia.htm"
ccaa_ine <- url |>
  read_html() |>
  html_node(xpath = '//*[@id="contieneHtml"]/table') |>
  html_table(fill = TRUE)
```

La ruta o “xpath” se puede obtener usando las herramientas de desarrollo del navegador, y puede que una vez importada la tabla se requiera algún post-procesamiento antes de poder analizar los datos.

### 1.5.3. Exportación de datos y archivos de datos específicos de R

En algunos proyectos es necesario guardar algunos datos que se han ido creando o transformando, bien para compartir con otras partes interesadas, bien para ser utilizados en el mismo u otros proyectos. Para exportar los datos a Excel, se utiliza la función `write.xlsx()` del paquete `openxlsx` (si no está instalado, se instala de la forma habitual). Si lo que se quiere es exportarlo a texto, se pueden utilizar los equivalentes a las funciones de importación `write.csv()`, `write.csv2()` o `write.table()`.

La siguiente expresión exporta la tabla de datos `tempmin_data` a ficheros Excel y csv (formato en inglés).

---

<sup>9</sup>Con el argumento `header` se indica si la primera fila tiene encabezados (TRUE) o no (FALSE, opción por defecto). También hay que especificar el separador de columnas `sep` y el símbolo decimal `dec`. `fileEncoding` es la especificación de la codificación de texto; las más habituales son `utf-8` y `'latin1'`.

## 1.6. Organización de datos con el tidyverse

19

```
openxlsx::write.xlsx(x = tempmin_data,
                      file = "data/temp_min_Filomena.csv")
write.csv(x = tempmin_data, file = "data/temp_min_Filomena.csv")
```

También se pueden guardar los datos en formato “nativo” de **R**. Los archivos **.RData** almacenan un espacio de trabajo entero, y por tanto pueden guardar varios objetos en el mismo archivo. Cuando posteriormente se importe, los objetos estarán en el espacio de trabajo con su nombre original. Se guardan con la función **save()** y se restauran con la función **load()**, como en el siguiente ejemplo.

```
save(tempmin_data, tempmax_data,
      file = "data/datos_temperaturas.RData")
load("data/datos_temperaturas.RData") #carga de nuevo el objeto
```

Los archivos **.rds** almacenan un único objeto en un archivo. Cuando posteriormente se quieran importar, hay que asignar el resultado al nombre que se quiera. Se guardan con la función **writeRDS()** y se restauran con la función **readRDS()**, como en el siguiente ejemplo.

```
saveRDS(object = tempmin_data,
          file = "data/datos_temperaturas.rds")
nuevo_objeto <- readRDS(file = "data/datos_temperaturas.rds")
```

El paquete **foreign** de **R** base y otros paquetes especializados pueden exportar datos a otros formatos de archivo, que no se tratan en detalle en este capítulo.

## 1.6. Organización de datos con el *tidyverse*

### 1.6.1. El *tidyverse* y su flujo de trabajo

El *tidyverse* es, según se define en su propia página web<sup>10</sup>, un conjunto de paquetes de **R** “opinables” diseñados para ciencia de datos. Las principales ventajas (opinables) de utilizar el *tidyverse* son tres:

1. Utiliza una gramática, estructuras de datos y filosofía de diseño común.
2. El flujo de trabajo es más fluido y, una vez se comprenden las ideas principales, más intuitivo.
3. Para la mayoría de las operaciones, es computacionalmente más eficiente.

---

<sup>10</sup>“... an opinionated collection of R packages designed for data science”. Incluye actualmente 30 paquetes, véase la lista con **tidyverse::tidyverse\_packages(include\_self = TRUE)** y la ayuda de cada paquete para saber más. Los que se vayan usando en el libro se irán explicando oportunamente.

Uno de los paquetes más populares del *tidyverse* es **ggplot2**, que proporciona una “gramática de gráficos” (Wickham, 2016) y es una pieza clave del *tidyverse* actual, junto con los paquetes **dplyr** (gramática para la manipulación de datos) y **tidyverse** (herramienta para crear datos *tidy*). El flujo de trabajo propuesto por el *tidyverse* se describe en el libro “R for Data Science” (Wickham and Grolemund, 2016) y se sintetiza en la Fig. 1.1.

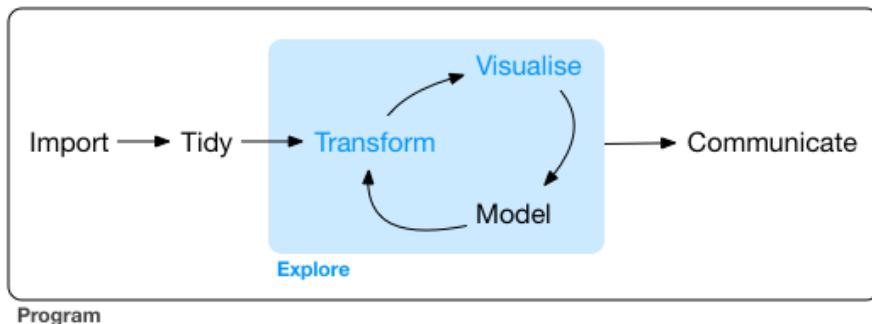


Figura 1.1: Flujo de trabajo en Ciencia de Datos propuesto por el *tidyverse* (fuente: Wickham, H. and Grolemund, G. (2016))

Además del mencionado libro, la web del *tidyverse* (<http://tidyverse.org>) contiene toda la documentación de los paquetes, incluidos artículos para tareas concretas, que merece la pena leer alguna vez. En la web están también las conocidas como *cheatsheets*, algunas de ellas disponibles también en la ayuda de RStudio (menú *Help/Cheatsheets*).

Dentro del flujo de trabajo de la Fig. 1.1, ya se ha tratado la primera etapa (*Import*) en la Sec. 1.5.2. Es importante señalar que, al utilizar las funciones del *tidyverse*, los datos se organizan en objetos de clase **tibble**, que es una extensión del **data.frame** de R base. Las principales diferencias son:

- Permite una representación compacta en la consola al mostrar la tabla de datos.
- La selección con corchetes simples de una única variable siempre devuelve otro **tibble** (a diferencia de un **data.frame**, que devuelve un vector).

Se puede forzar a que una tabla de datos sea de un tipo u otro con las funciones **as.data.frame** (de **tibble** a **data.frame**) y **as\_tibble** (de **data.frame** a **tibble**).

Siguiendo con el esquema de la Fig. 1.1, en este apartado se verán algunas tareas de las etapas *Tidy* (organizar) y *Transform* (transformar), que serán ampliadas en los Cap. ?? y ???. La visualización (*Visualise*) se tratará específicamente en el Cap. ?? y transversalmente en muchos otros. La modelización (*Model*) se trata extensamente en los capítulos de las partes IV a IX, y la comunicación (*Communicate*) se verá en los capítulos de la Parte X. Una de las características de la forma en que están programados los paquetes del *tidyverse* es que se puede trabajar<sup>11</sup> con *pipes*.

<sup>11</sup>Existe una guía de estilo del *tidyverse*, que se puede consultar en <https://style.tidyverse.org>. Hay incluso una serie de *Addins* en RStudio para comprobar y aplicar esta guía de estilo a través del paquete **styler**. Los *Addins* son menús adicionales en RStudio para usar la funcionalidad de algunos paquetes de forma interactiva.

## 1.6. Organización de datos con el tidyverse

21

El *pipe* es, básicamente, un operador compuesto de dos caracteres, `|>`, que se puede obtener con el atajo de teclado **CTRL+MAYUS+M**. El operador se pone en medio de dos expresiones de **R**. Sean `lado_izquierdo` y `lado_derecho` las expresiones que se ponen a izquierda y derecha del *pipe*. Entonces se utiliza de la siguiente manera:

```
lado_izquierdo |> lado_derecho
```

El operador *nativo* de **R**, `|>`, apareció en la versión R-4.1.0. Hay un operador alternativo que proviene del paquete `magrittr`, `%>%`, que había que usar antes de esta versión, y mucha literatura y documentación está escrita usándolo. Hay diferencias, pero a los efectos de este capítulo ambos operadores se pueden utilizar indistintamente.

La expresión `lado_izquierdo` debe producir un valor, que puede ser cualquier objeto de **R**. La expresión `lado_derecho` debe ser una función, que tomará como primer argumento el valor producido en la parte izquierda. Si se desea guardar el resultado final, se debe asignar el resultado a algún nombre de objeto para que se almacene en el espacio de trabajo. La siguiente expresión sería un ejemplo de uso.

```
nombre_objeto <- lado_izquierdo |>
  lado_derecho
```

La ventaja de usar los *pipes* es que se pueden encadenar, de forma que el resultado de cada operación pasa a la siguiente expresión del *pipeline* (secuencia de operaciones con *pipe*), como en el siguiente ejemplo:

```
library("dplyr")
contam_mad |> colnames() |> length()
#> [1] 12
```

## 1.6.2. Transformación de datos con dplyr

En la gramática del *tidyverse*, dentro del paquete `dplyr` se dispone de una serie de “verbos” (funciones) para una sola tabla, que se pueden agrupar en tres categorías: para trabajar con filas, para trabajar con columnas y para resumir datos.

### 1.6.2.1. Operaciones con filas

Los verbos definidos para estas operaciones son:

- `filter()`: elige filas en función de los valores de la columna.

```
pm10 <- contam_mad |>
  filter(nom_abv == "PM10")  # se filtra por PM10
```

- **arrange()**: cambia el orden de las filas con algún criterio.

```
zonas<- contam_mad |>
  arrange(desc(zona), daily_mean)
```

- **slice()**: extrae filas por su índice. También hay una serie de funciones “asistentes” (*helpers*) para obtener los índices que se utilizan con frecuencia. Por ejemplo:

- **slice\_head()** y **slice\_tail()** obtienen las primeras y últimas filas respectivamente (por defecto, una). Se puede especificar **n** (número) o **prop** (proporción) de filas.
- **slice\_sample()** obtiene una muestra aleatoria de **n** filas (o proporción **prop**).
- **slice\_min()**, **slice\_max()** obtienen las filas que contienen los menores o mayores valores respectivamente de la variable indicada en el argumento **order\_by**. Si no se especifica **n** o **prop**, se obtienen sólo las filas que contienen el mínimo o el máximo. Nótese que puede haber más de una fila que cumpla la condición.

Véase el resultado de los siguientes ejemplos:

```
pm10 |> slice(10:15) # extrae filas desde la 10 a la 15
pm10 |> slice_tail(n = 3) # extrae las tres últimas filas
pm10 |> slice_max(order_by = daily_mean) # día con mayor valor medio de PM10
set.seed(1) # Para que la muestra aleatoria sea reproducible
pm10 |> slice_sample(n = 4) # muestra 4 registros
```

### 1.6.2.2. Operaciones con columnas

Los verbos definidos para estas operaciones son:

- **select()**: indica cuando una columna se incluye o no. Se pueden utilizar *helpers* para seleccionar columnas que cumplan cierta condición (por ejemplo, ser numéricas) y también para “quitar” columnas de la selección (con el signo menos (-)).

```
pm10 |> select(longitud, latitud, daily_mean, tipo)
pm10 |> select(where(is.numeric))
pm10 |> select(-c(id:latitud))
```

En cuanto a la **modificación** de datos, existen múltiples posibilidades. Algunas de ellas son:

- **rename()**: cambia el nombre de la columna.

## 1.6. Organización de datos con el tidyverse

23

- **mutate()**: cambia los valores de las columnas y crea nuevas columnas. La función **transmute()** funciona igual que **mutate()**, pero la tabla de datos resultante sólo contiene las nuevas columnas creadas.
- **relocate()**: cambia el orden de las columnas.

```
pm10 |> rename(zona_calidad_aire = zona)
pm10 |> relocate(fecha, .before = estaciones)
pm10_na <- pm10 |> mutate(isna = is.na(daily_mean))
```

En este punto, es importante señalar que dentro de la función **mutate()** se puede usar cualquier función vectorizada para transformar las variables. Por ejemplo, se podría transformar una columna con las funciones **as.xxx** que se vieron en la Sec. 1.5.1, aplicar formatos a fechas o usar funciones del paquete **lubridate** para trabajar con este tipo de datos. A medida que se avance en el libro irán apareciendo aplicaciones que ahora, quizás, no sean tan evidentes.

### 1.6.2.3. Operaciones de resumen y agrupación

La primera operación de resumen que puede surgir es “contar” filas. La función **tally()** devuelve el número de filas totales de un **data.frame**. La función **count()** proporciona también este número; si, además, se pasa como argumento alguna variable, lo que devuelve es el número de filas para cada valor diferente de dicha/s variable/s. Estos recuentos se pueden añadir a la tabla de datos con las funciones **add\_count()** y **add\_tally()**, lo que permite calcular frecuencias absolutas y relativas fácilmente.

```
pm10 |> tally()
#>      n
#> 1 53794
pm10 |> count(zona)
#>      zona      n
#> 1: Interior M30 20690
#> 2: Noreste 12414
#> 3: Noroeste 4138
#> 4: Sureste 8276
#> 5: Suroeste 8276
```

La función **summarise()** (o, equivalentemente, **summarize()**) aplica alguna función de resumen a la/s variable/s que se especifiquen (**mean()**, **max()**, etc.). El paquete **dplyr** tiene algunas funciones de resumen adicionales, como **n()** (número de filas), **n\_distinct()** (número de filas con valores distintos) y **first()**, **last()**, **nth()** (primero, último y *n*-ésimo valor, en el orden en el que se encuentran, respectivamente).

En muchas ocasiones, las operaciones de análisis se realizan en grupos definidos por alguna variable de agrupación. La función **group\_by()** “prepara” la tabla de datos para realizar operaciones de este tipo. Una vez agrupados los datos, se pueden añadir operaciones de resumen

como las vistas anteriormente. A veces hay que “desagrupar” los datos, para lo que se utiliza la función `ungroup()`.

A continuación, se muestra una expresión un poco más compleja que las anteriores. En el conjunto de datos `contam_mad` del paquete CDR, se filtra por el nombre de contaminante “NOx”. Después se agrupan los datos por zona y se calculan algunos estadísticos resumen para cada zona.

```
contam_mad |>
  filter(nom_abv == "NOx") |> # se filtra por NOx
  group_by(zona) |>
  summarize(
    min = min(daily_mean, na.rm = TRUE),
    q1 = quantile(daily_mean, 0.25, na.rm = TRUE),
    median = median(daily_mean, na.rm = TRUE),
    mean = mean(daily_mean, na.rm = TRUE),
    q3 = quantile(daily_mean, 0.75, na.rm = TRUE),
    max = max(daily_mean, na.rm = TRUE)
  )
#> A tibble: 5 × 7
  zona      min     q1 median   mean     q3   max
  <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 Interior  M30  0.0833  32.4  54.1  72.9  90.0  759.
#> 2 Noreste     1    23.8   39.6  56.2  68.9  516.
#> 3 Noroeste     0    12.0   20.3  29.7  34.5  352.
#> 4 Sureste      0    29.1   45.4  64.6  77.2  453
#> 5 Suroeste    0.667  33.5  59.6  90.5 114.   666.
```

### 1.6.3. Combinación de datos

En el apartado anterior se han tratado los “verbos” de una tabla. Es muy común que haya que combinar datos de distintas tablas, para lo cual se utilizan lo que el *tidyverse* considera *two tables verbs*. En esencia, para combinar tablas que contienen información relacionada, hay que saber cuáles son las columnas que se refieren a lo mismo, para hacer las uniones (*joins*) utilizando esas columnas. Hay cuatro tipos de uniones que se pueden realizar, usando las siguientes funciones:

- `inner_join()`: se incluyen las filas de ambas tablas para las que coinciden las variables de unión.
- `left_join()`: se incluyen todas las filas de la primera tabla y sólo las de la segunda donde hay coincidencias.
- `right_join()`: se incluyen todas las filas de la segunda tabla y sólo las de la primera donde hay coincidencias.
- `full_join()`: se incluyen todas las filas de las dos tablas.

Las funciones requieren como argumentos dos tablas de datos y la especificación de las columnas coincidentes. Si no se especifica, hace las uniones por todas las columnas coincidentes en ambas

## 1.6. Organización de datos con el tidyverse

25

tablas. Para las filas que sólo están en una de las tablas, se añaden valores `NA` donde no haya coincidencias.

A modo de ejemplo, las siguientes expresiones unen dos datasets para combinar datos de municipios con su renta. En el Cap. ?? se verán estas uniones en la práctica.

```
library("sf")
munis_renta <- municipios |>
  left_join(renta_municipio_data) |>
  select(name, cpro, cmun, `2019`)
#> Joining, by = "codigo_ine"
```

Otra forma de unir tablas es, simplemente, añadiendo columnas (que tengan el mismo número de filas) o filas (que tengan el mismo número de columnas). Para ello se usan las funciones `bind_cols()` y `bind_rows()`, respectivamente. Otra forma conveniente de añadir nuevas filas o columnas son las funciones `add_row()` y `add_column()`. Se pueden añadir antes o después de una fila/columna especificada con el argumento `.before`, y pasando los valores como pares “variable = valor” para cada variable en el conjunto de datos.

Como comentario final del paquete `dplyr`, una característica importante es que se pueden usar las funciones vistas sobre tablas de una base de datos, sin necesidad de utilizar sentencias SQL y con la ventaja de que las operaciones se realizan en el motor de la base de datos. En el Cap. ?? se tratarán las cuestiones relacionadas con los gestores de bases de datos y SQL.

#### 1.6.4. Reorganización de datos

A lo largo del capítulo se ha visto la importancia de disponer los datos de forma rectangular, de forma que se tenga una columna para cada variable y una fila para cada observación. Algunas veces es conveniente reorganizar los datos más “a lo ancho” o más “a lo largo” de lo que se encuentran.

Para estas operaciones se utilizan las funciones `pivot_longer()` y `pivot_wider()` del paquete `tidyverse` de la siguiente forma:

- `pivot_longer()`: el argumento `names_to` asigna el nombre de la nueva variable que va a indicar de qué columna vienen los datos; y el argumento `values_to` asigna el nombre de la nueva variable que va a contener el valor de la tabla original.
- `pivot_wider()`: el argumento `names_from` indica el nombre de la variable que contiene los nombres de las nuevas columnas a crear a lo ancho; y el argumento `values_from` indica el nombre de la variable que contiene los valores en la tabla original. Las observaciones deben estar identificadas de forma única por varias variables. Si no es el caso, se puede aplicar una función al estilo de las tablas dinámicas de las hojas de cálculo con el argumento `values_fn`.

Las funciones `pivot_longer()` y `pivot_wider()` admiten otros argumentos `names_xx` y `values_xx` para personalizar la forma de reestructurar los datos. En la mayoría de las ocasiones será suficiente con las comentadas (`xx_from` y `xx_to`). Si fuera necesario, se recomienda consultar la ayuda de las funciones, o la lectura del artículo sobre *pivoting*.

A modo de ejemplo, el conjunto de datos `contam_mad` tiene los datos “mezclados” de varias variables medioambientales en la columna `daily_mean`. La columna `nom_abv` contiene el parámetro al que se refiere la columna de datos. Entonces, interesa “extender” la tabla para tener cada parámetro en una columna, de forma que se pueda hacer un análisis de datos adecuado, como en el siguiente código:

```
library("tidyverse")
extendida <- contam_mad |>
  pivot_wider(names_from = "nom_abv",
              values_from = "daily_mean",
              values_fn = mean)
colnames(extendida)
#> [1] "estaciones"    "id"          "id_name"      "longitud"
#> [5] "latitud"       "nom_mag"     "ud_med"       "fecha"
#> [9] "zona"          "tipo"        "BEN"          "SO2"
#> [13] "NO2"          "EBE"         "CO"           "NO"
#> [17] "PM10"         "PM2.5"      "TOL"          "NOx"
```

Se deja como ejercicio volver a obtener la tabla original usando la función `pivot_longer()` a partir del objeto `extendida`.

El paquete `tidyverse` también contiene funciones para reorganizar las columnas de la tabla uniendo columnas con la función `unite()`, o separando una columna en dos o más con la función `separate()` (véanse los detalles en la ayuda de las funciones).

Para terminar este apartado de reorganización de datos, se da una primera aproximación al tratamiento de valores perdidos, que se tratará en el Cap. ???. En R, un valor perdido se representa por el valor especial `NA` (*not available*). Brevemente, las funciones más utilizadas en este campo son:

- `drop_na()` del paquete `tidyverse`: permite eliminar las filas que tienen valores perdidos en ciertas variables (o en cualquiera, si no se especifica ninguna).
- `replace_na()`: sustituye los valores perdidos en cada variable por el valor especificado.
- `fill()`: permite “rellenar” valores perdidos con los últimos encontrados.

Los datos de contaminación a menudo tienen muchos valores perdidos. La siguiente expresión elimina las filas del conjunto de datos `contam_mad` con valores perdidos y, después, cuenta las filas.

```
contam_mad |>
  drop_na() |> # se omiten los NAs para el análisis
  count()
#>      n
#> 1: 505773
```

## Resumen

- R es software libre y gratuito, mantenido por una enorme comunidad.
- La forma de interactuar con R es mediante expresiones, que se escriben en *scripts*, y al ejecutarlas se obtienen los resultados.
- Los objetos de datos que se vayan a usar deben estar en el espacio de trabajo.
- RStudio es un “envoltorio” de R, y por tanto R tiene que estar instalado en el sistema para poder usar RStudio.
- Los paquetes se instalan una sola vez, y deben cargarse con `library()` para usar sus funciones.
- La tabla de datos o `data.frame` es la estructura de datos más adecuada para análisis de datos y cada columna es un `vector`.
- El *tidyverse* es un conjunto de paquetes que facilita las tareas de análisis de datos.
- El operador *pipe*, `|>`, permite “pasar” valores a funciones de forma encadenada.
- Las operaciones básicas con una tabla son filtrado, selección y resumen.
- Para crear nuevas columnas en las tablas de datos se usa la función `mutate`.
- Para combinar tablas con columnas comunes se usan las funciones `xx_join`.



## Bibliografía

- Ihaka, R. and Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314.
- R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Venables, W. and Ripley, B. (2002). *Modern applied statistics with S*. Statistics and computing. Springer.
- Wickham, H. (2015). *Advanced R*. Chapman & Hall/CRC The R Series. CRC Press.
- Wickham, H. (2016). *ggplot2*. Use R! Springer International Publishing, Cham, second edition.
- Wickham, H. and Grolemund, G. (2016). *R for Data Science*. O'Reilly Media.



# Índice alfabético

apilar datos, 25  
big data, 11  
cadenas de caracteres, 16  
data.frame, 15  
extender datos, 25  
extraer datos, 22  
factor, 16  
fechas, 16  
filtrar datos, 21  
hoja de cálculo, 11  
importar datos, 17  
internet of things, 11  
lista, 16  
matriz, 15  
missing, 26  
ordenar datos, 22  
paquete, 12  
proyectos, 14  
R, 11  
script, 11  
seleccionar columnas, 22  
strings, 16  
tidyverse, 19  
valores perdidos, 26  
vector, 15