

Fundamentos de ciencia de datos con R

Gema Fernández-Avilés y José-María Montero

2023-05-22

Índice general

Prefacio	5
¡Hola mundo!	5
¿Por qué este libro?	6
¿A quién va dirigido?	7
El paquete CDR	8
¿Por qué R?	8
Agradecimientos	9
1. Integración y limpieza de datos	11
1.1. Introducción	11
1.2. Integración de datos	12
1.3. Limpieza de datos	15
2. Selección y transformación de variables	31
2.1. Introducción	31
2.2. Selección de variables	32
2.3. Transformación de variables	41
2.4. Reducción de dimensionalidad	46
3. Herramientas para el análisis en ciencia de datos	47
3.1. Introducción	47
3.2. Partición del conjunto de datos	47
3.3. Técnicas para manejar datos no equilibrados	50
3.4. El enfoque de validación	51

Índice general

3.5. Compensación (<i>trade off</i>) entre sesgo y varianza	54
3.6. Ajuste de hiperparámetros	55
3.7. Evaluación de modelos	57
4. Análisis exploratorio de datos	65
4.1. Introducción	65
4.2. Análisis exploratorio de una variable	69
4.3. Análisis exploratorio de varias variables	77

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

Integración y limpieza de datos

Jorge Velasco López^a y José-María Montero^b

^aInstituto Nacional de Estadística de España ^bUniversidad de Castilla-La Mancha

1.1. Introducción

En los proyectos de ciencia de datos, generalmente es necesario realizar un **preprocesamiento** (o **preparación**) de los datos antes de iniciar las fases de modelado. Las labores de preprocesamiento son específicas para cada conjunto de datos, para los objetivos del proyecto y para las técnicas de modelización que se van a utilizar. Sin embargo, hay una serie de tareas comunes, como las de **integración** (combinación de datos de distintas fuentes) a partir de los datos en bruto (o sin procesar) y **limpieza** (identificación y corrección de posibles errores en los datos). Otras tareas que se suelen incluir en el proceso de preparación de datos son: la transformación de la variable objetivo, para cambiar su distribución de probabilidad (normalmente para hacerla gaussiana), la transformación de variables predictoras (o clasificadoras, en su caso) (*feature engineering*), la normalización y la reducción de la dimensionalidad. Estas tareas se abordarán en el Cap. 2.

En **R**, existen varios paquetes para llevar a cabo estos trabajos: **tidyverse**, para la manipulación de ficheros y variables que se han ilustrado en el Cap. ??; **dlookr** (Staniak and Biecek, 2019); **validate**, **errorlocate** y **dcmmodify** (van der Loo and de Jonge, 2019), para realizar validaciones y transformaciones a los datos; **caret** (Kuhn, 2008), para imputar los datos faltantes o perdidos (*missing data*); **sf** (Pebesma et al., 2018), para el manejo de conjuntos de datos espaciales; y **GGally** (Schloerke et al., 2021) y **naniar** (Tierney and Cook, 2018) para labores de visualización.

1.2. Integración de datos

La **integración** es un conjunto de procesos técnicos y de negocio que se utilizan para combinar información proveniente de diferentes fuentes. En términos generales, se puede decir que consiste en acceder a los datos desde todas las fuentes y localizaciones, tanto en entorno local, como en la nube o en una combinación de ambos, de modo que los registros de una fuente de datos enlacen con los registros de otra.

Para ilustrar el proceso de integración, a continuación se integra, por separado,¹ el conjunto de datos `Madrid_Sale` (incluido en el paquete `idealista18`), que contiene el identificador de las viviendas en venta en el municipio de Madrid y 41 variables relativas a dichos inmuebles (como su antigüedad y precio, por ejemplo) con otros dos conjuntos de datos del mismo paquete: `Madrid_POIS`, donde se listan, entre otras, las coordenadas de las estaciones de metro de la ciudad de Madrid; y `Madrid_Polygons`, que contiene los polígonos (en este caso, distritos) del municipio. Ello redundará en un enriquecimiento de los análisis que se lleven a cabo, al disponer en un mismo conjunto de datos un número mayor de variables relativas al problema a solucionar. A modo de ejemplo, la integración de `Madrid_Sale` con `Madrid_POIS` permitirá determinar el número de estaciones de metro a menos de 500 metros de la vivienda y la distancia de cada vivienda a la estación de metro más cercana; la integración de `Madrid_Sale` y `Madrid_Polygons` permitirá la construcción un mapa de precios medios del metro cuadrado de vivienda por distritos. Ambos ejemplos se ilustrarán con detalle en las dos subsecciones siguientes.

La función `glimpse()` permite mostrar la estructura de los tres conjuntos de datos incluidos en el paquete `idealista18`.

```
library("tidyverse")
library("idealista18")
library("sf")
library("GGally")
library("dlookr")

glimpse(Madrid_Sale)
glimpse(Madrid_POIS)
glimpse(Madrid_Polygons)
```

La combinación de conjuntos de datos se realiza, fundamentalmente, con las funciones de unión. En el Cap. ?? se mostraban las cuatro funciones de unión principales del paquete `tidyverse`: `left_join()`, `inner_join()`, `right_join()` y `full_join()`. Sin embargo, también merece la pena mencionar las uniones de filtrado entre dos objetos x e y , que se llevan a cabo mediante las siguientes funciones:

- `semi_join()`: devuelve todas las filas de x con una coincidencia en y .
- `anti_join()`: devuelve todas las filas de x que no tengan una coincidencia en y .
- `nest_join()`: devuelve todas las filas y columnas de x con una nueva columna anidada, que contiene todas las coincidencias de y .

¹Podría parecer que lo lógico es integrar todos los ficheros en un sólo conjunto de datos. Sin embargo, en muchas ocasiones es conveniente realizar integraciones parciales de los ficheros, para llevar a cabo distintas tareas en cada una de ellas, o, simplemente, por cuestiones de rendimiento.

1.2.1. Integración de los ficheros Madrid_Sale y Madrid_POIS

Como se avanzó anteriormente, dos interesantes resultados que se podrían obtener mediante la integración de estos conjuntos de datos son: (i) la determinación del número de estaciones de metro a menos de 500 metros de la localización de la vivienda de interés, y (ii) la distancia a la estación de metro más cercana. Para la integración entre los dos ficheros, se utiliza la función `st_join()`, función de unión para datos espaciales, del paquete `sf`.²

```
vivs_madrid <- Madrid_Sale |>
  st_join(Madrid_Polygons, left = TRUE)
```

Para proceder a la integración de ambos ficheros, primeramente se crean las variables que indican cuál es el sistema de referencia de coordenadas (SRC) que se va a utilizar y que permite determinar la posición de un punto en relación a otro en base a líneas imaginarias (en el ejemplo que nos ocupa, permite representar la ubicación de las viviendas en la superficie de la Tierra). En este caso, la asignación de coordenadas se realiza a través de las variables `projcrc_src` y `projcrs_dest`, en las que se establecen los parámetros de:

- Nombre de la proyección (`proj`).
- Zona UTM (`zone`) donde se ubica el conjunto de viviendas.
- Nombre del elipsoide (`ellips`). La Tierra no es una esfera y tiene accidentes geográficos, por lo cual hay que trabajar con elipsoides y explicitar los parámetros que definen su forma.
- Nombre del datum (`datum`). Define el origen y la orientación de los ejes de coordenadas, es decir, proporciona la información necesaria para dibujar el sistema de coordenadas en el elipsoide. El World Geodetic System (WGS84) es un standard en la industria a nivel mundial; no obstante, existen algunas variantes locales (la más famosa es el North American Datum (NAD83)).
- Tipo de unidades (`units`); en este caso, metros.

Seguidamente, se indica la distancia (en este caso en metros) que se va a usar como radio en la variable `radius_meters`. Finalmente, se lleva a cabo un procesamiento específico para datos espaciales: se crea un objeto espacial, se proyecta a plano para pasar de tres a dos dimensiones (hasta ahora se ha trabajado en la representación de la Tierra en tres dimensiones; sin embargo, estamos acostumbrados a ver mapas, es decir, a ver dos dimensiones), se cambia la geometría y, finalmente, se vuelve al sistema de coordenadas no proyectadas.

```
projcrs_src <- "+proj=longlat +datum=WGS84 +no_defs"
projcrs_dest <- "+proj=utm +zone=30 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
radius_meters <- 500 # Se marca la distancia que interesa.
pois_metro <- Madrid_POIS$Metro |>
  st_as_sf(coords = c("Lon", "Lat"), crs = projcrs_src) |> # Crear objeto espacial sf
  st_transform(crs = st_crs(projcrs_dest)) |> # Proyectar al plano (st_crs recupera la
  ↵ referencia de la coordenada y st_transform realiza la transformación)
```

²Téngase en cuenta que la vivienda es un bien anclado a una localización geográfica.

```
st_buffer(dist = radius_meters) |> # Cambiar la geometría de punto a polígono
  ↵ (círculo)
st_transform(crs = st_crs(projcrs_src)) # Volver al sistema de coordenadas no
  ↵ proyectadas (Ángulos)
```

A continuación, para cada una de las viviendas, se calcula el número de estaciones de metro a menos de 500 metros (variable `N_METRO_STOPS_500_M`). Para ello, primero se realiza el cálculo del objeto `sf metro_count`, seleccionando la variable `pois_metro` y cruzando con la geometría.

```
metro_count <- vibs_madrid |>
  select(ASSETID) |>
  st_join(pois_metro,
    join = st_intersects,
    left = FALSE
  )
# Se elimina la geometría para evitar ralentizar el cálculo
st_geometry(metro_count) <- NULL
```

Los valores de `N_METRO_STOPS_500_M` se obtienen con el siguiente código:

```
metro_count <- metro_count |>
  group_by(ASSETID) |>
  summarise(N_METRO_STOPS_500_M = n()) |>
  ungroup()
```

Al cruzar `metro_count` con `vibs_madrid`, se observa que hay casi 25.000 registros que no cruzan (25.000 viviendas que no tienen ninguna estación de metro a menos de 500 metros). En consecuencia, se retiran del análisis puesto el objetivo de la integración de estos dos conjuntos de datos es (*i*) la determinación del número de estaciones de metro a menos de 500 metros de la localización de las viviendas incluidas en el fichero `Madrid_Sale`.

```
vibs_madrid <- vibes_madrid |>
  inner_join(metro_count, by = "ASSETID")
```

Posteriormente, se determina la estación más cercana a cada vivienda a la venta con la función `st_nearest_feature()`.

```
pois_metro <- Madrid_POIS$Metro |>
  st_as_sf(coords = c("Lon", "Lat"), crs = projcrs_src) # Crear objeto espacial

mascercano_metro_stops <- pois_metro[st_nearest_feature(vibs_madrid, pois_metro), ] #
  ↵ Cálculo de las paradas cercanas
```

Por último, con la función `st_distance()`, se calcula la distancia de cada vivienda a la estación de metro más cercana, creándose la variable `METRO_STOP_MASCERCANO_DISTANCIA`.

1.3. Limpieza de datos

15

```
vivs_madrid <- vivs_madrid |>
  mutate(METRO_STOP_MASCERCANO_DISTANCIA =
    ↵  as.numeric(st_distance(mascercano_metro_stops, geometry, by_element = T)))
```

1.2.2. Integración de los ficheros Madrid_Sale y Madrid_Polygons

En esta subsección se muestran los detalles para construir un mapa de precio medio del metro cuadrado de la vivienda en la ciudad de Madrid, por distritos, tras la integración de los conjuntos de datos `Madrid_Sale` y `Madrid_Polygons`.

Para proceder a la integración de ambos ficheros, primeramente se realiza la conversión del conjunto de datos `Madrid_Polygons` a objeto espacial y se le asocia la coordenada de referencia (de forma similar a como se hizo en la integración de los ficheros `Madrid_Sale` y `Madrid_POIS`):

```
# Se convierten a objetos sf
Madrid_Polygons_sf <- sf::st_as_sf(Madrid_Polygons, wkt = "WKT") # WKT (Well-known
  ↵ text) es un formato de vectores geométricos
Madrid_Sale_sf <- st_as_sf(Madrid_Sale, coords = c("LONGITUDE", "LATITUDE"))
# se asocia la coordenada de referencia del objeto
st_crs(Madrid_Sale_sf) <- "+proj=longlat +datum=WGS84 +no_defs"
st_crs(Madrid_Polygons_sf) <- "+proj=longlat +datum=WGS84 +no_defs"
```

A continuación, se lleva a cabo la unión entre el objeto espacial de `Madrid_Polygons_sf` y `Madrid_Sale_sf` para calcular su precio por metro cuadrado (`preciopm2`) y el área de la geometría (`tract_area`).

```
Madrid_Sale_Polygons <- Madrid_Polygons_sf |>
  dplyr::mutate(tract_area = st_area(WKT)) |>
  sf::st_join(Madrid_Sale_sf) |>
  dplyr:: group_by(LOCATIONNAME) |>
  dplyr:: summarize(tract_area = unique(tract_area), preciopm2 = mean(PRICE /
    ↵ CONSTRUCTEDAREA))
```

A partir del resultado de esta integración, se construye la Fig. 1.1, que muestra un el mapa del precio medio del metro cuadrado de las viviendas a la venta en Madrid, a escala de distrito, lo que da una visión clara de las zonas más o menos económicas.

```
plot(Madrid_Sale_Polygons["preciopm2"])
```

1.3. Limpieza de datos

Es más habitual de lo deseable que algunas variables presenten problemas en la calidad de sus datos. En el Cap. ??, se mencionaban una serie de causas y la posibilidad de realizar el

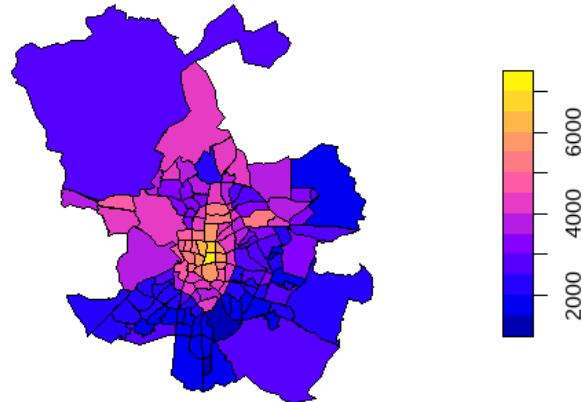


Figura 1.1: Precio por metro cuadrado de viviendas a la venta en Madrid por distrito

perfilado para tener una medición de la calidad de los datos. Si los datos no tienen el nivel de calidad adecuado, deben realizarse tareas de limpieza para transformarlos en datos consistentes, corrigiendo datos incorrectos, corruptos, con formato incorrecto, duplicados o incompletos.

En la Fig. 1.2 se muestra un proceso general de limpieza de datos. Cada rectángulo azul representa datos en un estado determinado, mientras que cada flecha representa las actividades necesarias para pasar de un estado a otro. En el primer estado están los datos tal y como se recogen (**datos en bruto** o **sin procesar**). Pueden carecer de encabezados, contener tipos de datos incorrectos, etiquetas de categoría incorrectas, codificación de caracteres desconocida o inesperada, etc. Una vez realizadas las correcciones necesarias, los datos pueden considerarse **datos técnicamente correctos**. Es decir, en este estado, los datos se pueden leer en un **data.frame** de R, con los nombres, tipos y etiquetas correctos. Sin embargo, esto no significa que los valores estén libres de errores o completos. Los **datos consistentes** son aquellos que están preparados para las fases de modelado.

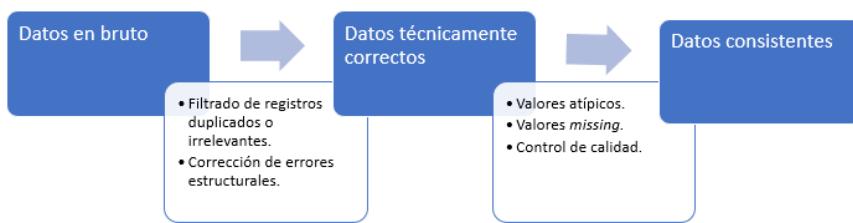


Figura 1.2: Flujo del proceso de limpieza de datos

Si bien las técnicas utilizadas para la limpieza de datos pueden variar según el tipo de datos que se esté procesando, en general, se pueden dividir en cinco grupos:

- Corrección de errores estructurales.

1.3. Limpieza de datos

17

- Filtrado de registros duplicados o irrelevantes.
- Gestión de valores atípicos.
- Gestión de valores faltantes (*missing*).
- Validación y control de la calidad de los datos.

1.3.1. Corrección de errores estructurales

Los **errores estructurales** ocurren cuando se observan formatos erróneos, estructuras incorrectas o errores tipográficos que pueden dar lugar a categorías o clases mal etiquetadas. Por tanto, puede haber errores estructurales a nivel de conjunto de datos (`data.frame`, `sf`, `tibble`,...) y a nivel de variable.

1.3.1.1. A nivel de conjunto de datos

Los **cambios estructurales a nivel de conjunto de datos** consisten en modificar el tipo de objeto o eliminar o agregar variables. Por ejemplo, a continuación se genera el conjunto de datos `Madrid_Sale_int`, con estructura de `data.frame`, a partir del conjunto de datos `vivs_madrid` (de tipo `sf`), fruto del proceso de integración anterior. Además, para este objeto, se elimina la variable `geometry`, que no se va a usar en adelante y ralentiza la computación.

```
Madrid_Sale_int <- as.data.frame(vivs_madrid) |> select(-geometry)
```

También se crea un segundo conjunto de datos reducido, `Madrid_Sale_red`, con una selección de variables que se consideran de interés para ilustrar las tareas de limpieza que se exponen en este capítulo.

```
Madrid_Sale_red <- select(Madrid_Sale_int, ASSETID, PRICE, UNITPRICE, CONSTRUCTEDAREA,
                           ROOMNUMBER, CONSTRUCTIONYEAR, HASNORTHORIENTATION, HASOUTHORIENTATION,
                           HASEASTORIENTATION, HASWESTORIENTATION, CONSTRUCTIONYEAR, DISTANCE_TO_METRO,
                           METRO_STOP_MASCERCANO_DISTANCIA)
```

Por último, se añade la variable `LOCATIONID1`, que indica el código de localización, al conjunto de datos `Madrid_Polygons`.

```
Madrid_Polygons$LOCATIONID1 <- substr(Madrid_Polygons$LOCATIONID, 1, 10)
```

En la siguiente sección, a partir de estos conjuntos de datos, se lleva a cabo un proceso de diagnosis y exploración.

1.3.1.2. A nivel de variable

Los **errores estructurales a nivel de variable** se centran fundamentalmente en el tipo de dato de las variables.

En primer lugar, se visualizan los datos con la función `diagnose()` de `dlookr`.

Nota

El paquete `dlookr` se usa para tareas de diagnosis y exploración, y es de utilidad para la localización de valores duplicados, faltantes, atípicos, tipología de datos, etc. La función `overview()` permite obtener una visión genérica del conjunto de datos, y la función `diagnose()` proporciona información a nivel de variable, como el tipo de dato (`type`), y sobre valores faltantes y únicos. Otras funciones útiles son `diagnose_numeric()` y `diagnose_category()`, que proporcionan información específica para valores numéricos y categóricos, respectivamente.

```
diagnose(Madrid_Sale_red)
```

Al ejecutar la función, se comprueba que todas las variables, excepto el identificador `ASSETID`, son de tipo numérico (o `integer`), lo que es correcto. En caso de tener que modificar el tipo de dato, por considerarse un error estructural, o porque sea conveniente para las fases de modelado, debería hacerse usando las funciones `as.factor()`, `as.numeric()` y `as.character()`, según el caso.

Corregir **errores estructurales tipográficos de variables categóricas** es especialmente relevante en algunas áreas de la ciencia de datos, como la minería de textos o *text mining* (que se verá con más profundidad en el Cap.??), donde la limpieza de textos consiste en eliminar todo aquello que no aporte información sobre su temática, estructura o contenido. A continuación, se muestra una función creada a partir del paquete `stringr` que permite realizar una limpieza básica de un texto, y que se ejecuta sobre la variable `Madrid_Polygons$LOCATIONNAME`, generando la variable `LOCATIONNAME1`.

```
library("stringr")
limpieza_textos <- function(texto) {
  # El orden de la limpieza no es arbitrario
  # Se convierte todo el texto a minúsculas
  nuevo_texto <- tolower(texto)
  # Eliminación de páginas web (palabras que empiezan por "http." seguidas
  # de cualquier cosa que no sea un espacio)
  nuevo_texto <- str_replace_all(nuevo_texto, "http\\S*", "")
  # Eliminación de signos de puntuación
  nuevo_texto <- str_replace_all(nuevo_texto, "[[:punct:]]", " ")
  # Eliminación de números
  nuevo_texto <- str_replace_all(nuevo_texto, "[[:digit:]]", " ")
  # Eliminación de espacios en blanco múltiples
  nuevo_texto <- str_replace_all(nuevo_texto, "[\\\\s]+", " ")
```

1.3. Limpieza de datos

19

```

    return(nuevo_texto)
}
Madrid_Polygons$LOCATIONNAME1 <- limpia_textos(texto = Madrid_Polygons$LOCATIONNAME)
glimpse(Madrid_Polygons)
#> $ LOCATIONNAME <fct> Conde Orgaz-Piovera, Pinar del Rey, Timón, Palacio,
#> ...
#> $ LOCATIONNAME1 <chr> "conde orgaz piovera", "pinar del rey", "timón",

```

1.3.2. Eliminación de observaciones duplicadas o irrelevantes.

Las **observaciones duplicadas** aparecen frecuentemente durante la recogida de datos e integración de las bases de datos, por lo que dichas duplicidades deben ser eliminadas en esta fase de limpieza.

A continuación, se usa la función `overview()` del paquete `dlookr` sobre el conjunto de datos `Madrid_Sale_int`, obtenido en la Sec. 1.3.1.1.

```

head(overview(Madrid_Sale_int), n = 9)
#>      division           metrics   value
#> 1      size       observations 70059
#> 2      size       variables     46
#> 3      size       values     3222714
#> 4      size       memory size 21931336
#> 5 duplicated duplicate observation      0
#> 6 missing   complete observation  26394
#> 7 missing   missing observation  43665
#> 8 missing   missing variables      7
#> 9 missing   missing values     48653

```

Entre otra información, como la existencia de valores faltantes (*missing*) en siete variables, se puede observar que no hay valores duplicados después del proceso de integración. En caso contrario, se podrían usar las funciones `base` de R para (i) localizarlos, con `duplicated()`, y (ii) extraer los registros únicos, con `unique()`. También se puede usar `distinct()`, del paquete `dplyr`, para eliminar los registros duplicados de un *data.frame*.

Las **observaciones irrelevantes** son aquellas que no encajan en el problema específico que se está analizando. Por ejemplo, si el objeto de estudio son datos de Madrid, se pueden eliminar las observaciones que no correspondan a dicho municipio. A continuación, se puede advertir que todas las observaciones de `Madrid_Polygons$LOCATIONID1` empiezan por el código correspondiente a Madrid (0-EU-ES-28) y, por tanto, no es necesario filtrar registros.

```

head(table(Madrid_Polygons$LOCATIONID1))
#>
#> 0-EU-ES-28
#>      135

```

En caso necesario, se pueden filtrar todos los registros de Madrid en el objeto `Madrid_Polygons1` haciendo:

```
Madrid_Polygons1 <-  
  Madrid_Polygons |> filter(substr(Madrid_Polygons$LOCATIONID, 1, 10) != "0-EU-ES-28")
```

1.3.3. Gestión de valores atípicos no deseados

A menudo, hay observaciones distintas que, aparentemente, no encajan en los datos que se están analizando. Si existe una razón coherente para eliminar un valor atípico (un *outlier*), como una entrada de datos incorrecta, hacerlo mejorará el rendimiento que proporcionan los datos con los que se está trabajando. Sin embargo, el hecho de que exista un valor atípico no significa que sea incorrecto. Si un valor atípico resulta ser irrelevante para el análisis, o es un error, debe considerarse su eliminación. El número de posibles valores atípicos en el conjunto de datos `Madrid_Sale_red` se determina con el siguiente código, que avisa de la posibilidad de que existan para cada una de las variables.

```
diagnose_numeric(Madrid_Sale_red)
```

Otra manera de localizar datos atípicos es a través de la **visualización**. Por ejemplo, en la Fig. 1.3 se relaciona el precio de la vivienda por metro cuadrado con su localización, y se observa que la zona más cara es Recoletos y la más barata es San Cristobal. La simple observación aconsejaría un análisis de los casos extremos (muy baratos o caros en cada uno de los distritos).

```
ggplot(Madrid_Sale_int, aes(x = reorder(LOCATIONNAME, PRICE / CONSTRUCTEDAREA, na.rm =  
  TRUE), y = PRICE / CONSTRUCTEDAREA)) +  
  geom_boxplot() +  
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +  
  labs(x = "Distrito", y = "Precio metro cuadrado")
```

Los box-plots y gráficos de dispersión de variables, para las categorías dadas de otra, así como las correlaciones entre dichas variables, también pueden utilizarse para detectar valores atípicos. Por ejemplo, se puede considerar la relación del precio del metro cuadrado de la vivienda con otras variables, como la superficie construida, la distancia al metro y el número de habitaciones. Para ello, primeramente se crea el conjunto de datos `Madrid_Sale_red2` con la variable derivada `price_bin` (de tipo factor), cuyas categorías o clases (o *bins*) son los cuartiles de la variable `PRICE`.

```
Madrid_Sale_red2 <- mutate(Madrid_Sale_int, price_bin = cut2(PRICE, g=4)) |>  
  select(price_bin, CONSTRUCTEDAREA, DISTANCE_TO_METRO, ROOMNUMBER, LOCATIONNAME)
```

A partir del conjunto de datos `Madrid_Sale_red2` se puede crear construir la Fig. 1.4.

1.3. Limpieza de datos

21

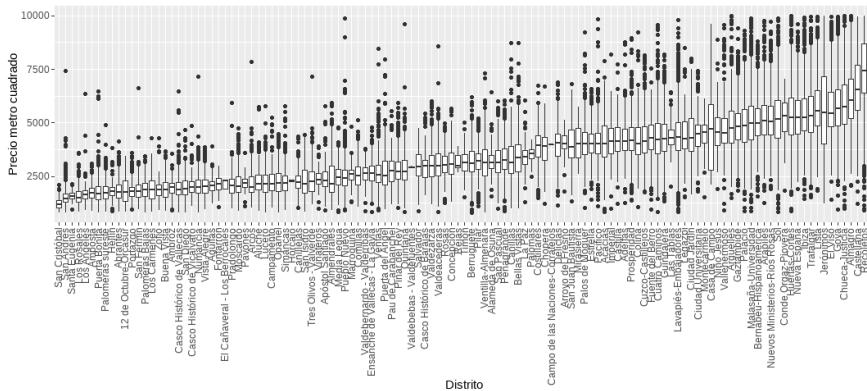


Figura 1.3: Precio medio del metro cuadrado por distritos

```
ggpairs(Madrid_Sale_red2,
        column = 1:4, aes(color = price_bin, alpha = 0.5),
        upper = list(continuous = wrap("cor", size = 2))
      )
```

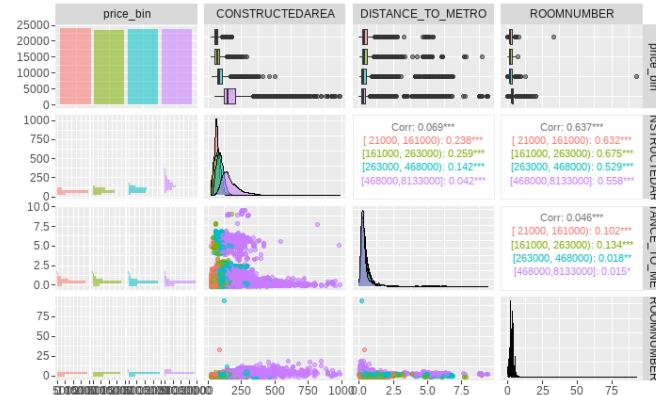


Figura 1.4: Distribuciones y correlaciones cruzadas algunas variables de Madrid-Sale-red

En dicha figura, la diagonal descendente muestra la función de cuantía (para precio medio del metro cuadrado) y las funciones de densidad de CONSTRUCTEDAREA, DISTANCE_TO_METRO y ROOMNUMBER. Los tres últimos paneles de la primera columna muestran los histogramas de las estas tres últimas variables. Los tres últimos paneles de la primera fila, proporcionan los box-plots de estas variables para los cuatro *bins* de la variable price_bin (primer cuartil en rosa, segundo en verde, tercero en azul y cuarto en morado). Los paneles del triángulo lateral derecho muestran sus correlaciones, mientras que los del triángulo inferior izquierdo presentan sus gráficos de dispersión. Dicho lo anterior, por ejemplo, en la primera fila se observa que

las viviendas más económicas suelen tener menos superficie construida (segunda columna), que suelen estar ligeramente más alejadas del metro (tercera) y suelen tener menos habitaciones. Sin embargo, se aprecian algunas cuestiones que llaman la atención. Por ejemplo, que hay una viviendas muy alejadas (a casi 400 kilómetros) de la estación de metro más cercana, lo cual distorsiona algunas de las figuras e impide ver la información que contienen; o que hay viviendas cuyo precio por metro cuadrado pertenece a la primera categoría de la variable `price_bin` (las más económicas) con muchas habitaciones o con mucha superficie construida. A continuación, por ejemplo, se filtran las viviendas con 30 o más habitaciones (aunque la lógica sería válida para muchas menos). Se observa que la superficie construida es de menos de 120 metros lo que, sin mayor conocimiento del conjunto de datos, no parece ser coherente y podrían excluirse (filtrarse) del conjunto de datos, o tratar de recabar la información correcta.

```
Madrid_Sale_red2 |> filter(price_bin == "[ 21000, 168000)", ROOMNUMBER > 30)
#>   price_bin CONSTRUCTEDAREA DISTANCE_TO_METRO ROOMNUMBER LOCATIONNAME
#> 1 [ 21000, 168000)           90       0.3826137      33 Almendrales
```

Finalmente, detectados los valores atípicos, por cualquiera de los procedimientos anteriormente expuestos, el paquete `dlookr`, a través de la función `impute_outlier()`, permite llevar a cabo sofisticadas imputaciones de los mismos, si bien sólo en el caso variables numéricas. Los métodos de imputación que se contemplan son: media, mediana, moda y *capping* (imputar los valores atípicos superiores con el percentil 95, y los inferiores con el percentil 5). Por ejemplo, se podría imputar la variable `CONSTRUCTEDAREA_imp` a partir de `CONSTRUCTEDAREA` con el método media (*mean*): `CONSTRUCTEDAREA_imp <- impute_outlier(Madrid_Sale_red2, CONSTRUCTEDAREA, method = "mean")`.

Otra opción es poner los valores atípicos como valores no disponibles (*not available*, `NA`) y proceder a imputar dichos `NA` tal y como se muestra en el epígrafe siguiente.

```
Madrid_Sale_red2$ROOMNUMBER[Madrid_Sale_red2$ROOMNUMBER >= 30 &
  ~ Madrid_Sale_red2$price_bin == "[ 21000, 168000)" ] <- NA
```

1.3.4. Gestión de datos faltantes (*missing*)

Los datos pueden faltar por multitud de razones, aunque generalmente se suelen agrupar en dos categorías: **valores faltantes informativos** (Kuhn et al., 2013) y **valores faltantes aleatorios** (Little and Rubin, 2019). Los informativos implican una causa estructural, ya sea por deficiencias en la forma en que se recopilaron los datos o por anomalías en el entorno de observación. Los aleatorios son aquellos que tienen lugar independientemente del proceso de recopilación de datos.

Dependiendo de si los valores faltantes son de uno u otro tipo, se procederá de una u otra manera. A los informativos, en general, se les puede asignar un valor concreto (por ejemplo, “Ninguno”), ya que este valor puede afectar a los resultados de las predicciones. Los aleatorios pueden manejarse mediante la eliminación o la imputación. Además, los diferentes algoritmos de aprendizaje automático manejan la falta de información de manera diferente. De hecho, la

1.3. Limpieza de datos

23

mayoría de los algoritmos no incorporan mecanismos para manejarlos (por ejemplo, modelos lineales generalizados y derivados, redes neuronales y *support vector machine*) y, por lo tanto, requieren que se traten previamente. Sólo unos pocos modelos (principalmente basados en árboles) tienen procedimientos incorporados para tratar los valores faltantes.

Como se avanzó anteriormente, en **R**, los valores nulos se representan con el símbolo `NA`. Es importante distinguirlos de los valores indefinidos (p. ej., dividir entre cero), que se representan con el símbolo `NaN` (*Not a Number*). Para visualizar los patrones de datos *faltantes* de la variable `price_bin` del conjunto de datos `Madrid_Sale_red2`, se ejecuta el siguiente código.

```
library("naniar")
gg_miss_fct(x = `Madrid_Sale_red2`, fct = price_bin)

# También puede visualizarse de otra manera con la siguiente opción
gg_miss_var(Madrid_Sale_red2, show_pct = TRUE, facet = price_bin)
```

En la Fig. 1.5 se puede observar claramente que hay datos faltantes en la variable `LOCATIONNAME`, sobre todo en los dos primeros cuartiles (*bins*). Concretamente, hay 42 valores faltantes. No obstante, aunque el degradado del color morado apenas permite apreciarlo, también hay un valor faltante en `ROOMNUMBER`.

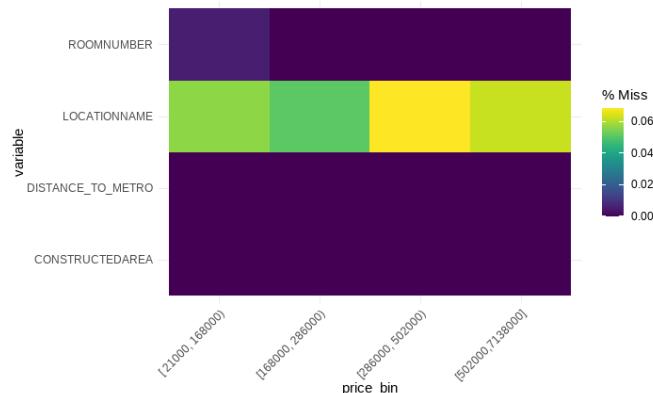


Figura 1.5: Visualización de valores faltantes

La gestión de los valores faltantes debe hacerse considerando la problemática que se quiera resolver. Una primera opción a considerar sería excluirlos, si bien se estaría eliminando información. Para filtrar los registros faltantes, se podría utilizar la función `is.na()`. En el caso de `ROOMNUMBER`:

```
Madrid_Sale_red3 <- Madrid_Sale_red2
Madrid_Sale_red3 |>
  filter(!is.na(ROOMNUMBER))
```

También se puede optar por reemplazarlos, por ejemplo por un 0, de la siguiente manera:

```
Madrid_Sale_red3[is.na(Madrid_Sale_red3)] <- 0
# También puede usarse la función `replace_na()`, que sustituye los valores perdidos en
→ cada variable por el valor especificado.
```

No obstante, estas dos opciones no son acciones recomendables en primera instancia, porque eliminar los registros con valores faltantes, o introducir valores que podrían no respetar la semántica de los datos, puede ocasionar un alto impacto negativo en los niveles globales de calidad de datos del conjunto de datos.

Se puede ir más allá de la eliminación de valores faltantes. A través de diversos métodos se pueden imputar valores que, con mayor o menor probabilidad, podrían ser los que realmente correspondieran a estos valores faltantes. Estos métodos se conocen como **métodos de imputación de valores**. Para imputar valores faltantes se pueden usar diversas alternativas, como la función `preProcess()` del paquete `caret` o la función `impute_na()` del paquete `dlookr`. A continuación, se imputan los valores faltantes del conjunto de datos `Madrid_Sale_red2` con dos métodos. En primer lugar, con el algoritmo de KNN (*k* vecinos más cercanos), que sustituye el valor faltante por la media de los valores de los *k* vecinos más próximos. Después de realizar el preprocessamiento, se comprueba que las imputaciones han sido realizadas.

```
library("caret")
# Se realiza el preprocesamiento:
pre_knn <- preProcess(Madrid_Sale_red2, method = "knnImpute", k = 2)
# Se obtienen los datos
imputed_knn <- predict(pre_knn, Madrid_Sale_red2)
# Con la siguiente función se comprueba que se ha imputado el valor faltante de la
→ variable ROOMNUMBER
diagnose(imputed_knn)
```

A continuación, la imputación se realiza con la mediana (que suele ser preferible a imputar la media, puesto que el promedio puede verse afectado por outliers).

```
# Se realiza el preprocesamiento:
pre_median <- preProcess(Madrid_Sale_red2, method = "medianImpute")
# Se obtienen los datos
imputed_median <- predict(pre_median, Madrid_Sale_red2)
# Con la siguiente función se comprueba que se ha imputado el valor faltante de la
→ variable ROOMNUMBER
diagnose(imputed_median)
```

Nota

El paquete `dlookr`, a través de la función `imputate_na()`, permite imputar valores faltantes. El predictor admite variables numéricas y categóricas. Los métodos que utiliza son: para numéricas `media`, `moda`, `KNN`, `rpart` y `mice`); y para categóricas: `mode`, `rpart` y `mica`. El paquete `recipes` también es recomendable. Por ejemplo, para la imputación de los valores faltantes con la media de la variable se usaría `step_meanimpute(all_numeric())`.

1.3.5. Validación y control de calidad

Al final del proceso de limpieza de datos, éstos deberían ser consistentes y seguir las reglas apropiadas para su campo de negocio. De no ser así, los modelos que estimen en base a ellos no representarán convenientemente la realidad objeto de estudio y las conclusiones que se obtengan de dichos modelos no serán de utilidad para dicha realidad.

La verificación de si los datos son o no consistentes y si siguen o no las reglas del campo de negocio del cual proceden, se puede llevar a cabo con el paquete `tidyverse`, que permite hacer selecciones, filtrados o tablas de frecuencias, entre otras acciones. A modo de ejemplo, en el caso del precio medio del metro cuadrado de los distritos de la ciudad de Madrid, se puede usar la función `count()` para obtener la distribución de frecuencias de la variable `METRO_STOP_MASCERCANO_DISTANCIA` y comprobar si es consistente con el conocimiento que se tiene de esa variable y del conjunto de datos. Se muestran las distancias a la estación más cercana para las viviendas correspondientes a los seis primeros registros.

```
head(count(as.data.frame(Madrid_Sale_red), METRO_STOP_MASCERCANO_DISTANCIA))
#>   METRO_STOP_MASCERCANO_DISTANCIA n
#> 1 1.413845 1
#> 2 1.414032 1
#> 3 2.586694 1
#> 4 3.156593 1
#> 5 4.013776 1
#> 6 4.128947 1
```

Una opción más sofisticada es el paquete `validate`, donde se pueden introducir las reglas de negocio dentro del propio código o bien desde un fichero externo. A continuación, se realiza un ejemplo con las reglas incrustadas en el propio código. Estas reglas pueden ser avisos o normas que indican error en esos datos. En este ejemplo, se han definido siete reglas: por ejemplo, `PRICE ≥ 0`, o que la suma de las variables `HASNORTHORIENTATION`, `HASSOUTHORIENTATION` `HASEASTORIENTATION` y `HASWESTORIENTATION` sea la unidad. La salida que se obtiene se presenta a continuación. A modo de ejemplo, la regla `HASNORTHORIENTATION + HASSOUTHORIENTATION + HASEASTORIENTATION + HASWESTORIENTATION = 1` es la número 3, que, como se puede ver, no se cumple en 48.446 ocasiones.

```

library("validate")

Madrid_Sale_int |>
  check_that(
    HASLIFT >= 0,
    PRICE >= 0,
    HASNORTHORIENTATION + HASSOUTHORIENTATION + HASEASTORIENTATION + HASWESTORIENTATION
  ) == 1,
  is.numeric(PRICE),
  UNITPRICE * CONSTRUCTEDAREA == PRICE,
  if (ROOMNUMBER > 3) PRICE > 100000,
  nrow(.) >= 20000
) |>
  summary()
#> name items passes fails NA error warning
#> 1 V1 70059 70059 0 0 FALSE FALSE
#> 2 V2 70059 70059 0 0 FALSE FALSE
#> 3 V3 70059 21613 48446 0 FALSE FALSE
#> 4 V4 1 1 0 0 FALSE FALSE
#> 5 V5 70059 15280 54779 0 FALSE FALSE
#> 6 V6 70059 70041 18 0 FALSE FALSE
#> 7 V7 1 1 0 0 FALSE FALSE

```

Nota

El proceso de validación puede ser más o menos complejo, según afecte a una única variable en un mismo registro, a más de una variable de un mismo registro o a más de una variable en más de un registro. En el último caso, además, se puede validar en un solo conjunto de datos o en más de uno.

En un esquema tradicional de validación, además de las reglas de validación aportadas por los expertos en el tópico del que se trate, debe incluirse también un listado de reglas de corrección (igualmente aportado por los expertos en la materia) que indiquen cómo hay que corregir un registro cuando no cumple con una determinada regla de validación. Este modo de proceder, además de suponer un doble esfuerzo, puede conducir a inconsistencias o validaciones cíclicas.

El Método de Fellegi y Holt³ (MFH) dà una solución a este problema, evitando dichas inconsistencias, proporcionando un procedimiento que genera un conjunto completo de reglas de validación, incorporando reglas implícitas a las formuladas por los expertos de manera explícita.

En breves palabras, dicho método asegura el cumplimiento de las siguientes tres premisas:

- Minimizar el número de campos a corregir en un registro para hacerlo pasar todas las validaciones.
- Mantener, en la medida de lo posible, la distribución conjunta original del conjunto de datos.

³El MFH es un estándar internacional en la revisión de la integridad de la información de encuestas y censos.

1.3. Limpieza de datos

27

- Derivar las reglas de corrección, directamente y de forma implícita, de las reglas de validación. Por tanto, dichas reglas de corrección no son propuestas el experto o, en su caso, por el validador.

Los detalles sobre el MFH pueden verse en [Boskovitz et al. \(2003\)](#).

El MFH no está exento de limitaciones. La primera es el incremento del coste computacional, que puede llegar a constituir un problema en caso de que el número de reglas implícitas sea muy elevado, lo cual es muy frecuente. De hecho, hay casos en los que hay más reglas implícitas que registros. Para solucionar este problema, denominado “problema de localización del error”, que consiste, básicamente, en determinar el conjunto mínimo de variables a corregir para cada validación, se han propuesto varias alternativas, que incluyen métodos de investigación de operaciones, árboles binarios y metaheurísticas como algoritmos genéticos y similares.

A efectos prácticos, el MFH se puede aplicar con la función `locate_errors()` del paquete `errorlocate`, determinándose así cuáles son las variables a corregir para solventar los errores en las reglas de negocio establecidas (objeto `rules`). Por ejemplo, en el conjunto de datos `Madrid_Sale_red2` (donde se definía la variable `price_bin`), se establecen ahora unas reglas básicas algo más laxas (específicamente una: más de 10 habitaciones los tres primeros cuartiles), obteniéndose que habría que depurar la variable `ROOMNUMBER` en dos ocasiones para que el conjunto de datos quedase totalmente limpio (o depurado).

```
library("errorlocate")

rules <- validator(if (ROOMNUMBER >= 10) price_bin == "[502000,7138000]")
el <- locate_errors(Madrid_Sale_red2, rules) |>
  summary(el)
el
# el$variable
#   names           errors missing
#
#     price_bin      0      0
#   ROOMNUMBER      2      1
#   CONSTRUCTEDAREA 0      0
#   DISTANCE_TO_METRO 0      0
#   LOCATIONNAME    0     42
```

¿Y qué se debe hacer con los registros que no cumplen las normas de validación? La respuesta es, como norma, “siempre que se disponga de información de negocio, ésta debe preponderar sobre cualquier tipo de imputación”. A partir de este punto se puede proceder a realizar imputaciones determinísticas para solucionar los problemas detectados.

En el ejemplo anterior, se propone imputar el valor `ROOMNUMBER=5` a los casos de los tres primeros cuartiles (todos menos el más caro) que tengan más de 10 habitaciones. Para ello, se utiliza la función `modify_so()` del paquete `dcmmodify`. Para comprobar que la imputación se ha llevado a cabo con éxito, se pueden comparar los conjuntos de datos antes y después de la imputación con la función `compare()`, comprobándose que tal imputación se ha realizado exitosamente en los 2 registros que presentaban problemas con la regla `ROOMNUMBER >= 10`.

```
library("dcmodify")

out <- Madrid_Sale_red2 |>
  modify_so(if (ROOMNUMBER >= 10 & price_bin != "[502000,7138000]") ROOMNUMBER <- 5)

rules <- validator(if (ROOMNUMBER >= 10) price_bin == "[502000,7138000]")
compare(rules, raw = Madrid_Sale_red2, modified = out)
#> Object of class validatorComparison:
#>
#> compare(x = rules, raw = Madrid_Sale_red2, modified = out)
#>
# Status           raw modified
# validations      70059   70059
# verifiable       70058   70058
# unverifiable     1       1
# still_unverifiable 1       1
# new_unverifiable 0       0
# satisfied        70056   70058
# still_satisfied  70056   70056
# new_satisfied    0       2
# violated         2       0
# still_violated   2       0
# new_violated     0       0
```

Resumen

- En un proyecto de ciencia de datos deben realizarse procesos de integración y limpieza previos a la fase de modelización, para asegurar niveles adecuados de calidad. Por ello, tras las labores iniciales de depuración, debe comprobarse si los datos son o no consistentes, y si siguen o no las reglas del campo de negocio del cual proceden. En este capítulo se abordan las cuestiones relativas a la integración de conjuntos de datos, su limpieza y depuración, y se proponen procedimientos para la validación de los mismos.
- El conjunto de datos utilizado en este capítulo está disponible en el paquete ‘idealista18'; en concreto, se utilizan los datos de Madrid.
- A partir de estos datos, se muestra un ejemplo de integración de datos espaciales y se diseña un marco de limpieza genérico basado en una serie de pasos básicos.
- Para la realización de estas funciones de integración y limpieza de datos, se proponen distintas funciones de los paquetes `tidyverse` (para la manipulación de ficheros y variables) y `caret` (para la imputación de valores perdidos).
- Para el tratamiento de datos espaciales se utiliza el paquete `sf`. La visualización de los mismos se lleva a cabo con `GGALLY` y `naniar`.
- Finalmente, se utiliza la función `locate_errors()`, del paquete `errorlocate`, para determinar cuáles son las variables con errores, de acuerdo a las reglas establecidas y `dcmmodify` para realizar imputaciones determinísticas.

Capítulo 2

Selección y transformación de variables

Jorge Velasco López^a y José-María Montero^b

^aInstituto Nacional de Estadística de España ^bUniversidad de Castilla-La Mancha

2.1. Introducción

Como se indicó en el Cap. 1, la preparación de datos, en un contexto de ciencia de datos, consiste en transformarlos de tal forma que se puedan utilizar adecuadamente en las fases posteriores de modelado. Esta preparación o pre-preprocesamiento puede ser un proceso laborioso e incluye tareas como la integración y limpieza de datos, que se detallaron en dicho capítulo.

El presente capítulo aborda las tareas relativas a la **selección de variables** (*feature selection*) y **transformación de variables**. La selección de variables tiene como objetivo elegir el elenco de variables más relevantes para el análisis. La transformación de variables hace referencia, básicamente, al uso de determinados procedimientos para modificar la distribución de la variable objetivo, a la ingeniería de variables (*feature engineering*), a la normalización y a la reducción de la dimensionalidad del problema de interés.

Se usará el conjunto de datos `Madrid_Sale` (disponibles en el paquete de R `Idealista18`), con datos inmobiliarios del año 2018 para el municipio de Madrid, y los paquetes `caret` (Kuhn, 2008), para diversas tareas de preparación de datos y `corrplot` (Wei et al., 2017), para visualizar correlaciones, entre otros.

2.2. Selección de variables

Quizás, el primer gran reto al que se enfrenta el científico de datos cuando maneja grandes conjuntos de datos es la identificación de las variables que proporcionen información valiosa sobre la variable objetivo, bien se trate de un problema de regresión o de clasificación. En caso de que el científico de datos salga exitoso de este primer gran reto, un determinado subconjunto de variables del conjunto de datos de interés proporcionará la misma información sobre la variable objetivo que la totalidad de variables incluidas en el conjunto de datos.

En consecuencia, la selección de variables involucra un conjunto de técnicas cuyo objetivo es seleccionar el subconjunto de variables predictoras más relevante para las fases de modelización. Esto es importante porque:

- Variables predictoras redundantes pueden distraer o engañar a los algoritmos de aprendizaje, lo que posiblemente se traduzca en un menor rendimiento, no solo predictivo (exactitud y precisión), sino también en términos de tiempo de computación.
- Igualmente, la inclusión de variables irrelevantes aumenta el coste computacional y dificulta la interpretabilidad.

Una adecuada selección de variables tiene ventajas importantes: (*i*) elimina las variables con información redundante; (*ii*) reduce el grado de complejidad de los modelos; (*iii*) evita o reduce el sobreajuste; (*iv*) incrementa de la precisión de las predicciones; y (*iv*) reduce la carga computacional.

No obstante, es importante señalar que, antes de llevarse a cabo la selección de variables propiamente dicha, debe comprobarse la magnitud de la varianza de las variables candidatas a ser seleccionadas y de sus correlaciones dos a dos, así como si existen combinaciones lineales entre ellas (multicolinealidad). Y ello, porque estas tres comprobaciones sirven para realizar una primera pre-selección de variables, si bien por razones técnicas y no de capacidad de explicación del comportamiento de la variable respuesta.

Los métodos de selección de variables (tras la pre-selección anteriormente mencionada) se suelen clasificar en: (*i*) los que utilizan la variable objetivo (supervisados); y (*ii*) los que no (no supervisados). Debido a la complejidad de la cuestión, se pasará revista únicamente a los métodos supervisados más relevantes, que se pueden dividir en:

- **Métodos tipo filtro**, que puntuán de mayor a menor cada variable predictora en base a su capacidad predictiva y seleccionan un subconjunto de ellas en base a dichas puntuaciones (Brownlee, 2020).
- **Métodos tipo envoltura (wrapper)**, que eligen el subconjunto de variables que dan como resultado el modelo con mayores prestaciones en cuanto a calidad de resultados y eficiencia: error de predicción o clasificación, precisión, tiempo de computación...
- **Métodos intrínsecos** (o *embedded*), que seleccionan las variables automáticamente como parte del ajuste del modelo durante el entrenamiento (tal es el caso de algunos modelos de regresión penalizados, como Lasso, árboles de decisión y bosques aleatorios (*random forests*)).

2.2.1. Pre-selección de variables

2.2.1.1. Varianza nula

Uno de los aspectos fundamentales en la selección de variables es comprobar si su varianza es cero o cercana a cero porque, si es así, sus valores son iguales o similares, respectivamente, y, por tanto, esas variables estarán perfectamente o quasi-perfectamente correladas con el término independiente del modelo, con lo cual, en el mejor de los casos, solo añadirán ruido al modelo. Además, este tipo de variables causan problemas a la hora de dividir el conjunto de datos en subconjuntos de entrenamiento, validación y test. Las causas de una nula o muy pequeña variabilidad pueden estar en haber medido la variable en una escala inapropiada para la variable o en haber expandido una variable politómica en varias dicotómicas (una por categoría), entre otras. En el primer caso, un cambio de escala puede evitar el problema de la colinealidad. Otra opción más drástica la eliminación de la variable.

A continuación, se comprueba si las variables del conjunto de datos `Madrid_Sale` tienen **varianza cero**. Para ello se utiliza la función `nearZeroVar()` del paquete `caret`.

Se seleccionan en primer lugar las variables numéricas en el conjunto de datos `Madrid_Sale_num`.

```
library("idealista18")
library("tidyverse")
library("caret")

Madrid_Sale <- as.data.frame(Madrid_Sale)
numeric_cols <- sapply(Madrid_Sale, is.numeric)
Madrid_Sale_num <- Madrid_Sale[, numeric_cols]
```

Se observa que se devuelve el valor `nzv=FALSE` (`nzv: near zero variance`) para casi todas las variables, con la excepción de `PARKINGSPACEPRICE`, `ISDUPLEX`, `ISSTUDIO`, `ISINTOPFLOOR` y `BUILTYPEID_1`, que podrían descartarse como variables predictoras.

```
varianza <- nearZeroVar(Madrid_Sale_num, saveMetrics = T)
# Con el argumento saveMetrics, se guardan los valores que se han utilizado para los
# cálculos.
# Se muestran los primeros resultados
head(varianza, 2)
#>      freqRatio percentUnique    zeroVar    nzv
#> PERIOD  2.019617   0.004218742 FALSE  FALSE
#> PRICE   1.076923   2.911986500 FALSE  FALSE
```

Para filtrar (excluir) las variables que se descartan como predictoras, se procede como sigue:

```
Madrid_Sale_num <- Madrid_Sale_num|>
  select(-c(PARKINGSPACEPRICE, ISDUPLEX, ISSTUDIO, BUILTYPEID_1))
```

2.2.1.2. Correlación entre variables

Como se avanzó anteriormente, otra de las cuestiones a tener en cuenta en el proceso de selección de variables es la magnitud de las **correlaciones entre las variables** candidatas, pues la existencia de correlaciones elevadas tiene consecuencias perversas sobre la fiabilidad de las predicciones (o de la clasificación realizada). En el caso extremo el modelo tendrá problemas de colinealidad o multicolinealidad (véase Sec. 2.2.1.3).

Para detectar las variables con muy elevada correlación entre ellas, se le pasa la función `findCorrelation()` de `caret`, con valor 0,9, a la matriz de correlaciones lineales entre las variables susceptibles de ser seleccionadas.

```
madrid_cor <- cor(Madrid_Sale_num[, 1:20])
alta_corr <- findCorrelation(madrid_cor, cutoff = .9)
```

Con ello, se comprueba que la variable `HASPARKINGSPACE` tiene correlaciones superiores a 0,9 con varias de las variables predictoras, procediéndose a su eliminación.

```
Madrid_Sale_num <- Madrid_Sale_num[, -alta_corr]
```

la Fig. 2.1, generada con el paquete `corrplot`, muestra las correlaciones existentes entre las primeras variables predictoras.

```
library("corrplot")

matriz_corr <- cor(Madrid_Sale_num[, 1:8])
corrplot(matriz_corr, method = "circle")
```

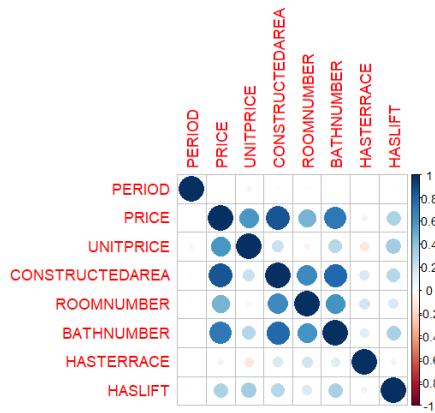


Figura 2.1: Matriz de correlaciones topada en 0,9

Se aprecia que ya no hay variables altamente correlacionadas.

2.2.1.3. Combinaciones lineales

En la práctica, en la mayoría de los casos, por ejemplo en las regresiones lineales, las variables que se utilizan como predictores no son ortogonales sino que tienen cierto grado de dependencia lineal entre ellas. Si dicho grado es moderado, las consecuencias de la no ortogonalidad en la predicción no son graves, pero en los casos de dependencia lineal quasi-perfecta las inferencias resultantes del modelo estimado distan mucho de la realidad. Dichas consecuencias son aún más graves en el caso de que las combinaciones lineales sean perfectas. Por ello, la existencia de colinealidad o combinaciones lineales entre las variables seleccionables también es una circunstancia a evitar. En el caso de que los predictores (o varios de ellos) conformen una o varias combinaciones (o quasi-combinaciones) lineales, no se puede conocer el impacto específico de cada uno de ellos en la variable objetivo, pues dichos impactos se solapan unos con otros. Además, como se ha avanzado, las predicciones no son fiables, entre otras cosas (véase (?)). Y es que se le está pidiendo al conjunto de datos en estudio más información sobre la variable objetivo de la que realmente tiene. Entre otros modelos, la regresión lineal y la regresión logística parten del supuesto de no colinealidad o multicolinealidad entre las variables, por lo que no debería haber variables correlacionadas, ni dos a dos, ni en forma de combinación lineal entre varias de ellas.

Las principales fuentes de multicolinealidad son:

- El método utilizado en la recogida de datos (subespacios).
- Restricciones en el modelo o en la población (existencia de variables correlacionadas).
- Especificación del modelo (polinomios).
- Más variables que observaciones.

En cuanto al detalle de las consecuencias más importantes de la multicolinealidad, hay que señalar las siguientes:

- Los estimadores tendrán grandes varianzas y covarianzas.
- Las estimaciones de los coeficientes del modelos serán demasiado grandes.
- Los signos de los coeficientes estimados suelen ser distintos a los esperados.
- Pequeñas variaciones en los datos, o en la especificación del modelo, provocarán grandes cambios en los coeficientes.

En el ejemplo con los datos del conjunto `Madrid_Sale` se utiliza la función `findLinearCombos()` de `caret` para encontrar, en caso de que las haya, combinaciones lineales de las variables predictoras.

```
Madrid_Sale_num_na <- tidyrr::drop_na(Madrid_Sale_num) # Es necesario eliminar los NA.
combos <- findLinearCombos(Madrid_Sale_num_na)
combos
#$remove
#NULL
```

Como puede comprobarse, no se encuentra ninguna combinación lineal en las variables numéricas de `Madrid_Sale`. En caso de existir, una solución al problema de la multicolinealidad pasa por:

- Eliminar variables predictoras que se encuentren altamente relacionadas con otras que permanecen en el modelo.
- Sustituir las variables predictoras por componentes principales (véase Cap. ??).
- Incluir información externa a los datos originales. Esta alternativa implica utilizar estimadores contraídos (de Stein o ridge) o bayesianos.

A continuación se muestra, caso de existir, cómo se eliminarían las combinaciones lineales:

```
Madrid_Sale_num_na[, -combos$remove]
```

2.2.2. Métodos de selección de variables

Tras la pre-selección de variables llevada a cabo en el epígrafe anterior, procede la selección de variables, propiamente dicha, de entre las que han superado la fase previa, en base, principalmente, a criterios de capacidad predictiva. No obstante, también se utilizan para:

- Simplificar de modelos para hacerlos más interpretables.
- Mejorar la precisión del modelo (si se ha escogido bien el subconjunto de variables).
- Reducir el tiempo de computación; sobre todo, entrenar algoritmos a mayor velocidad.
- Evitar la maldición de la dimensionalidad (o efecto Huges), que se refiere a las consecuencias no deseadas que tienen lugar cuando la dimensionalidad de un problema es muy elevada.
- Reducir la probabilidad de sobreajuste.

2.2.2.1. Métodos tipo filtro

Los **métodos de selección de variables tipo filtro** usan técnicas estadísticas para evaluar la relación entre cada variable predictora (o de entrada, o independiente) y la variable objetivo (o de salida, o dependiente). Generalmente, consideran la influencia de cada variable predictora sobre la variable objetivo por separado. Las puntuaciones obtenidas se utilizan como base para clasificar y elegir las variables predictoras que se utilizarán en el modelo.

La elección de las técnicas estadísticas depende del tipo de variables (objetivo y predictoras). Por ejemplo, si las variables de entrada (predictoras) y salida (objetivo) fueran numéricas, se utilizaría el coeficiente de correlación de Pearson o el de Spearman (dependiendo de si la relación entre la variable predictora y la variable objetivo es lineal o no) o el método de información mutua (véase [Vergara and Estévez \(2014\)](#)). Si ambas fuesen categóricas, podrían usarse medidas de asociación para tablas de contingencia 2×2 o $R \times C$ (véanse Sec. ?? y ??). Si la de entrada

fuese categórica y la de salida numérica, la técnica adecuada sería el Análisis de la Varianza (ANOVA, véase Sec. ??). Si la categórica fuese la de salida y la numérica la de entrada, entonces habría que acudir a la regresión logística (véase Sec. ??), por ejemplo. Sin embargo, el conjunto de datos no tiene porqué tener sólo un tipo de variable de entrada. Para manejar diferentes tipos de variables de entrada, se pueden seleccionar, por separado, variables de entrada numéricas y variables de entrada categóricas, usando en cada caso las técnicas apropiadas.

Estos métodos suelen eliminar sólo las variables de menor interés a la hora de predecir/clasificar. Permiten ahorrar tiempo y son especialmente robustos para el sobreaprendizaje. Sin embargo, no tienen en cuenta las relaciones entre las variables, lo que puede dar lugar a seleccionar variables redundantes si es que no se ha llevado a cabo una fase de pre-selección.

Nota

Existen diversos paquetes, como **FSelector** (Romanski et al., 2013) y el mismo **caret**, para implementar técnicas de selección de variables. Aquí se utiliza **FSinR**, que contiene una serie de métodos de filtro y envoltura, que se combinan con algoritmos de búsqueda para obtener el subconjunto óptimo de variables, usando funciones para entrenar modelos de clasificación y regresión disponibles en el paquete **caret**. La selección de variables o características se lleva a cabo con la función **FeatureSelection()**, y la del algoritmo de búsqueda que se utilizará en el proceso de selección de funciones se realiza con la función **searchAlgorithm()**. Por su parte, los métodos de filtrado se implementan a través de la función **filterEvaluator()**. No debe olvidarse que, antes de realizar el proceso de selección de variables, el usuario tiene que dividir el conjunto de datos convenientemente para llevar a cabo cada operación sobre el subconjunto correcto (véase Cap. 3). Igualmente, también de manera previa, se tiene que resolver el problema de los datos faltantes.

A continuación se muestra un ejemplo para variables predictoras numéricas. Para ello, se toma una muestra del conjunto de datos **Madrid_Sale_num**, obtenido en la Sec. 2.2.1.1. Una vez en disposición de la muestra, primeramente se transforma la variable objetivo en categórica, siendo las categorías (intervalos) cuatro cortes de la distribución de sus valores; dicha categorización se lleva a cabo mediante **binning**.¹ También se eliminan los registros con datos faltantes.

¹ *Binning* (anglicismo que deriva de la palabra *bin*: cubo, cesta, contenedor) es una técnica de discretización que agrupa datos numéricos en intervalos. Se suele utilizar para simplificar el análisis de datos continuos y aumentar la interpretabilidad del modelo, si bien a costa de reducir las combinaciones de las categorías de las variables predictoras que pueden realizarse, con lo cual el modelo sólo podrá hacer predicciones para unas pocas combinaciones de categorías de las variables predictoras. El *binning* puede ser supervisado o no (agrupamiento automático o manual). En este último caso, hay que tomar muchas precauciones porque, como señala Kuhn et al. (2013), (i) el *binning* en las variables predictoras puede llevar a una pérdida significativa en la capacidad del modelo a la hora de determinar la relación (sobre todo si es compleja) entre los predictores y la variable objetivo; y (ii) en el entorno clasificadorio, puede dar lugar a una alta tasa de falsos positivos. Estas limitaciones pueden superarse en el caso de que el *binning* se lleve a cabo de forma supervisada (tal es el caso de los árboles de regresión y clasificación y de la regresión adaptativa multivariante con splines), si bien debe tenerse en cuenta que, aunque se utilizan todos los predictores para llevar a cabo el proceso de *bining*, la categorización está guiada por un único objetivo (por ejemplo, maximizar la exactitud).

```

library("rsample")

# Se toma una muestra con el paquete rsample
set.seed(7)
Madrid_Sale_num_sample <- sample(1:nrow(Madrid_Sale_num), size = 5000, replace = FALSE)
Madrid_Sale_num_sample <- Madrid_Sale_num[Madrid_Sale_num_sample, ]
# Se realiza binning con cuatro bins
Madrid_Sale_num_sample_bin <- Madrid_Sale_num_sample |>
  mutate(price_bin = cut(PRICE, breaks = c(0, 250000, 500000, 750000, 10000000), labels
    ↪ = c("primerQ", "segundoQ", "tercerQ", "c"), include.lowest = TRUE)) |>
  select(price_bin, CONSTRUCTEDAREA, ROOMNUMBER, BATHNUMBER, HASTERRACE, HASLIFT)
# Se eliminan los registros con valores missing
Madrid_Sale_sample_na <- drop_na(Madrid_Sale_num_sample_bin)

```

Una vez discretizada la variable objetivo, se selecciona el conjunto de variables predictoras de la variable objetivo `price_bin`, que es la variable `PRICE` transformada mediante *binning*. Como método tipo filtro se utiliza `minimum description length` (MDLM), que es un método de selección de variables que se basa en una medida de la complejidad del modelo denominada “longitud mínima de la descripción” (de ahí el nombre del modelo), por lo que su objetivo es encontrar el modelo más sencillo que proporcione una explicación aceptable de los datos. Como algoritmo de búsqueda se utiliza `sequential forward selection`.²

```

library("FSinR")

# Método tipo filtro MDLC (Minimum-Description-Length-Criterion)
evaluador <- filterEvaluator("MDLC")
# Se genera el algoritmo de búsqueda
buscador <- searchAlgorithm("sequentialForwardSelection")
# Se implementa el proceso, pasando a la función los dos parámetros anteriores
resultados <- featureSelection(Madrid_Sale_sample_na, "price_bin", buscador, evaluador)
# Se muestran los resultados
resultados$bestFeatures
#>   CONSTRUCTEDAREA ROOMNUMBER BATHNUMBER HASTERRACE HASLIFT
#> [1,]          0          0          0          1          0
resultados$bestValue
#> [1] 355.3439

```

En este caso, con los argumentos propuestos, el modelo seleccionado para explicar el comportamiento de la variable objetivo `price_bin` contiene únicamente el término independiente y una variable predictora: `HASTERRACE`.

²Este método (selección hacia adelante) consiste en ajustar primero un modelo que contenga únicamente un término independiente (o intercepto); es decir, sin variables predictoras. Posteriormente, se ajusta otro con término independiente y una sola variable predictora. Después, se ajusta otro modelo con término independiente y dos variables predictoras. Y así sucesivamente. El criterio de parada suele ser que el valor del criterio de información de Akaike no experimente una reducción significativa al añadir una variable más al modelo. Los otros dos métodos o criterios para moverse en el espacio de búsqueda de subconjuntos de variables predictoras son el `backward` (selección hacia atrás), que funciona justo al revés que el `forward`, por eliminación de variables, y la selección paso a paso, `stepwise` que es una combinación de los dos anteriores.

2.2.2.2. Métodos de selección de variables tipo envoltura (*wrapper*)

Este enfoque realiza una búsqueda a través de diferentes combinaciones o subconjuntos de variables predictoras/clasificadoras para comprobar el efecto que tienen en la precisión del modelo (Saeys et al., 2007).

Hay varias alternativas:

- Evaluar las variables individualmente y seleccionar las n variables principales que obtienen unas buenas prestaciones, aunque se pierde la información de las dependencias entre variables.
- Observar el rendimiento del modelo para todas las combinaciones de variables posibles. En este sentido, se puede utilizar un algoritmo de búsqueda global estocástica, como los algoritmos genéticos que, si bien pueden ser efectivos, también pueden ser computacionalmente muy costosos.

Los métodos **wrapper** son de gran eficacia a la hora de eliminar variables irrelevantes y/o redundantes (cosa que no ocurre en los de tipo filtro porque se centran en el poder predictor de cada variable de forma aislada). Además, tienen en cuenta la circunstancia de que dos o más variables, aparentemente irrelevantes en cuanto a su capacidad predictiva o clasificatoria cuando se consideran una por una, pueden ser relevantes cuando se consideran conjuntamente. Sin embargo, son muy lentos, ya que tienen que aplicar muchísimas veces el algoritmo de búsqueda, cambiando, cada vez, el número de variables, siguiendo, cada vez, algún criterio tanto de búsqueda como de paro. En lo que respecta a los criterios de búsqueda, estos son similares a los de los métodos tipo filtro. Por lo que se refiere a los criterios de paro, los usados en los métodos *wrapper* son menos eficientes que los criterios basados en algún tipo de medida de ganancia de información, distancia o consistencia, entre el predictor y la variable objetivo (o clase) que utilizan los de tipo filtro.

Nota

Las principales diferencias entre los métodos tipo filtro y tipo envoltura son las siguientes:

- Los métodos de filtro cuantifican la relevancia de las variables por su correlación con la variable salida, mientras que los métodos de tipo envoltura cuantifican las prestaciones del modelo para diferentes subconjuntos de variables.
- Los métodos de filtro tienen una carga computacional enormemente inferior a la de los envolventes, ya que no necesitan entrenar ningún modelo.
- Los métodos de filtro utilizan métodos estadísticos para evaluar la selección de variables; los de tipo envoltura utilizan métodos de validación cruzada.
- En la mayoría de ocasiones, la selección de variables realizada por los métodos tipo envoltura suele ser más exitosa que la proporcionada por los métodos de filtro.
- Los métodos de envoltura tienen una probabilidad de sobreajuste mucho mayor que los de filtro.

A continuación, sobre el conjunto de datos `Madrid_Sale_sample_na` y sobre la variable objetivo `price_bin` se establecen tanto los parámetros de los algoritmos de búsqueda como los métodos

de filtrado y se calculan los resultados, usando para ello de **FSinR**. Como método de selección de variables se utiliza un método **wrapper** (con la función `wrapperEvaluator()`de **FSinR**) y como algoritmo de búsqueda **sequential forward selection**.

```
# Se fijan los parámetros
evaluador <- wrapperEvaluator("rpartISE")
buscador <- searchAlgorithm("sequentialForwardSelection")
# Se evalúan sobre Madrid_Sale_sample_na
results <- featureSelection(Madrid_Sale_sample_na, "price_bin", buscador, evaluador)
resultados$bestFeatures
resultados$bestValue
```

El resultado es el mismo que con el con el método tipo filtro anteriormente utilizado: el modelo seleccionado para explicar el comportamiento de la variable objetivo `price_bin` contiene únicamente el término independiente y una variable predictora: `HASTERRACE`.

Nota

Se puede sofisticar más el modelo ajustando los parámetros del modelo con parámetros de remuestreo, que son los mismos argumentos que se pasan a la función `trainControl()` del paquete `caret`. En segundo lugar, se pueden establecer los parámetros de ajuste, que son los mismos que para la función de `train` de `caret`.

2.2.2.3. Métodos de selección tipo intrínseco (*embedded*)

Finalmente, hay algunos algoritmos de aprendizaje automático que realizan la selección automática de variables como parte del aprendizaje del modelo. Estos son los métodos de selección de tipo intrínseco, que aglutinan las ventajas de los métodos de filtro y envoltura.

Un ejemplo son los relativos a los modelos de regresión penalizados, como Lasso, o *ridge* (que tienen funciones de penalización incluidas para reducir el sobreajuste), árboles de decisión y bosques aleatorios.

En el siguiente ejemplo se modeliza un bosque aleatorio (usando el paquete `randomForest`) y, tras dicha modelización, se identifica el conjunto óptimo de variables con la función `varImp()` de `caret`.

```
library("randomForest")

# Usar random forest para la selección de variables
rf_modelo <- randomForest(price_bin ~ ., data = Madrid_Sale_num_sample_bin)

# Listar las variables más importantes
varImp(rf_modelo)
```

Con este método de selección de variables, el modelo con mayor poder predictivo de la variable salida `price_bin` es el que contiene un término independiente y los predictores `CONSTRUCTEDAREA`, `ROOMNUMBER` y `BATHNUMBER` (ejecútese el código para comprobarlo).

2.3. Transformación de variables

La transformación y creación de variables predictoras a partir de los datos en bruto tiene una componente técnica y otra más creativa; en esta última, son de gran relevancia la intuición y la experiencia en trabajos de modelado, así como el dominio de los datos en cuestión. Para labores de transformación también se utilizará el paquete `caret`.

Nota

`Caret` se ha elegido como herramienta principal para la parte de preprocesamiento por su amplia difusión y porque también se utiliza en la parte de *machine learning* supervisado de este libro. No obstante, se podrían usar otros paquetes, como `recipes`, incluido en `tidymodels`. Este tipo de paquetes, comúnmente llamados metapaquete (*meta-packages*), permiten agrupar varios programas junto a sus dependencias para su instalación de una vez. Por tanto, un metapaquete permite ahorrar tiempo y esfuerzo a la vez que facilita la implementación de múltiples modelos en paralelo para, posteriormente, vincular sus resultados.

La fase de modelización puede condicionar la fase previa de preparación de datos. Por ejemplo, determinadas técnicas imponen requisitos y expectativas sobre el tipo y forma de las variables predictoras (Boehmke and Greenwell, 2019). Así, podría ser necesario que la variable objetivo tenga una distribución de probabilidad específica, o la eliminación de variables predictoras altamente correlacionadas con otras y/o que no estén fuertemente relacionadas con la variable objetivo.

Generalmente, estas transformaciones son más útiles para algoritmos como los de regresión, métodos basados en instancias (también llamados *memory-based learning methods*, como *k*-vecinos más cercanos -KNN- y *Learning Vector Quantization* -LVQ-), máquinas de vectores de soporte -SVM- y redes neuronales -NN-, que para métodos basados en árboles y reglas.³

2.3.1. Transformación de la distribución de la variable objetivo

Aunque no siempre es necesario, la transformación de la distribución de la variable objetivo puede llevar a una mejora predictiva significativa, especialmente en el caso de modelos paramétricos. Por ejemplo, los modelos de regresión lineal ordinarios asumen que el término de error, y, por consiguiente, la variable objetivo, se distribuyen normalmente. Pero puede ocurrir, por

³Una de las varias clasificaciones existentes de los métodos de aprendizaje los divide en basados en instancias (muestras u observaciones del conjunto de entrenamiento) o en modelos. Los algoritmos basados en instancias "memorizan" dichas instancias, y utilizan esta información a la hora de realizar una predicción. El aprendizaje basado en modelos tiene como objetivo la creación de un modelo a partir de los datos de entrenamiento, con el cual se harán las predicciones.

ejemplo, que la variable objetivo tenga valores atípicos y la suposición de normalidad no se cumpla por asimetría.

Para simetrizar la distribución de probabilidad de la variable objetivo (mejorando así la dispersión de valores y, a veces, desenmascarando las relaciones lineales y aditivas entre los predictores y el objetivo) se puede usar una transformación log (entre otras). Para corregir la asimetría positiva de la distribución probabilística de la variable objetivo se suele utilizar una de las dos opciones siguientes:

- **Normalizar con una transformación logarítmica**, que proporciona buenos resultados en la mayoría de los casos. En la Fig. 2.2, se puede comprobar que, en el ejemplo que se viene arrastrando, una transformación logarítmica normaliza, en gran medida, la distribución de la variable PRICE. Nótese que, si la variable objetivo tiene valores negativos o cero, una transformación logarítmica producirá *Nan* y *-Inf*, respectivamente. Si los valores de respuesta no positivos son pequeños (por ejemplo, entre -0,99 y 0), se puede aplicar una pequeña compensación (por ejemplo, la función `log1p()` agrega un 1 al valor antes de aplicar la transformación).

```
respuesta_log <- log(Madrid_Sale$PRICE)
```

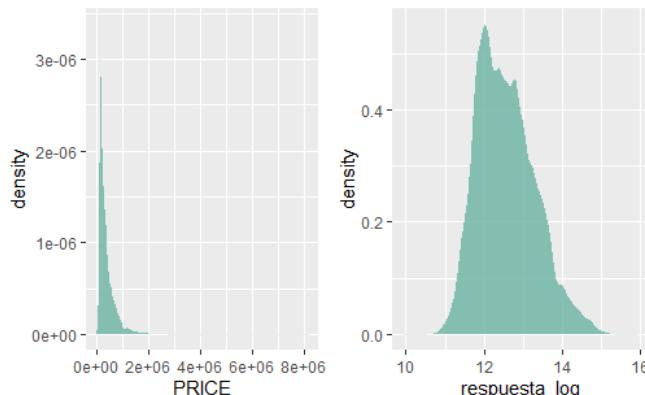


Figura 2.2: Normalización logarítmica

- Como segunda opción, se puede usar una transformación de la familia de transformaciones Box-Cox (o simplemente una **transformación de Box-Cox**), de carácter potencial y con mayor flexibilidad que la transformación logarítmica. Generalmente, se puede encontrar la función adecuada a partir de una familia de transformadas de potencia, que llevarán la distribución de la variable transformada tan cerca como sea posible de la distribución normal [Sakia (1992)]⁴. No obstante, igual que la transformación logarítmica, las transformaciones del tipo Box-Cox también tienen la limitación de ser sólo aplicables a variables

⁴La piedra angular de la transformación de Box-Cox es el exponente de dicha transformación (λ), que varía entre -5 y 5.

2.3. Transformación de variables

43

cuyos valores sean positivos. Por consiguiente, tanto si se usa una transformación log como una Box-Cox, no se deben centrar los datos primero, ni realizar ninguna operación que pueda hacer que los valores de la variable transformada no sean positivos.

- En caso de valores nulos o negativos, una muy buena opción, la tercera, es la transformación Yeo-Johnson, que es una extensión de la transformación Box-Cox que no está limitada a los valores positivos.

```
respuesta_boxcox <- preProcess(Madrid_Sale_num_sample, method = "BoxCox")
trainBC <- predict(respuesta_boxcox, Madrid_Sale_num_sample)
respuesta_boxcox
#> Created from 5000 samples and 2 variables
#>
#> Pre-processing:
#> - Box-Cox transformation (2)
#> - ignored (0)
#>
#> Lambda estimates for Box-Cox transformation:
#> -0.3, -0.3
```

Hay que tener en cuenta que, cuando se modela con una variable objetivo transformada, las predicciones también estarán en la escala transformada. Es posible que haya que deshacer (o volver a transformar) los valores pronosticados a su escala original, para que los responsables de la toma de decisiones puedan interpretar los resultados más fácilmente.

Nota

El paquete **recipes** (recetas de cocina), incluido en **tidymodels**, permite la transformación de variables de forma secuencial. La transformación de la distribución probabilística de la variable objetivo con **recipes** se lleva a cabo en 4 etapas: (i) **recipe()**, donde se especifica la fórmula (variables predictoras y variable objetivo); (ii) **step()**, donde se definen los pasos a seguir: imputación de valores perdidos, creación de variables ficticias (dummies), normalización, etc.; (iii) **prep** (preparar, o en otros términos, entrenar), donde que se utiliza un conjunto de datos para analizar cada paso en él; y (iv) **bake** (hornear/cocinar), donde, una vez aplicada la receta, se aplica al conjunto de datos. La idea detrás de **recipes** es similar a **caret::preProcess()**. Sin embargo, a diferencia de **caret**, no maneja automáticamente las variables categóricas y requiere crear variables ficticias manualmente.

2.3.2. Cambios de origen y escala en las variables (normalizaciones)

La escala en que se miden las variables individuales no es una cuestión baladí a la hora de la modelización. Los modelos que incorporan funciones lineales en las variables predictoras, son sensibles a la escala de esas variables. Lo mismo puede decirse de los algoritmos que utilizan medidas de distancia, como los de agrupación y clasificación, o los de escalamiento multidimensional, entre otros; o los de reducción de la dimensionalidad. Cuando se estiman modelos,

a menudo es aconsejable modificar la escala de las variables predictoras; el objetivo es evitar que unas variables tengan mayor influencia que otras en el resultado obtenido. Por ejemplo, en el conjunto de datos `Madrid_Sale` la superficie de las viviendas, medida en metros cuadrados, tiene una media y una desviación típica mayores que la antigüedad de la misma, medida en años. En consecuencia, los algoritmos basados en la magnitud de los errores pueden dar más importancia a las variables con mayor desviación típica, pero no porque tengan mayor variabilidad real que las otras, sino porque la medida de dicha variabilidad (la desviación típica) es más grande debido a la distinta escala en la que están medidas dichas variables. La consecuencia: efectos perniciosos indeseados sobre las predicción o la clasificación.

La normalización de variables tiene como objetivo que las comparaciones entre estas variables, en cuanto a su contribución al análisis de interés, sean objetivas; es decir, ponerlas en igualdad de condiciones en lo que respecta a su influencia (más allá de la que realmente tienen) en la variable objetivo.

La estandarización (o normalización *z-score*) es el método de normalización de variables más popular. Consiste en restar la media de la variable a sus los valores y, posteriormente, dividir esta diferencia entre la desviación típica de la variable. De esta manera, las variables (numéricas) transformadas tendrán media nula y varianza unitaria, lo que proporciona una unidad de medida comparable común a todas las variables: la distancia a la media medida en términos de desviaciones típicas.

A modo de ejemplo, a continuación se estandarizan las variables del conjunto de datos `Madrid_Sale` con la función `preProcess()` de `caret` y el `method=c('center', 'scale')`, de tal manera su media sea nula y su desviación típica unitaria.

```
prep_centrado <- preProcess(Madrid_Sale_num, method = c("center", "scale"))
pred_centrado<-predict(prep_centrado, Madrid_Sale_num)[1:3]
head(pred_centrado, n = 3)
#>      PRICE    CONSTRUCTEDAREA      ROOMNUMBER
#1 -1.523435   -0.64763050   -0.5764192
#2 -1.523435   -0.38628625   0.4062338
#3 -1.523435   -0.05541004   0.7717038
```

Otra normalización también popular es la **min-max**, que re-escala los valores de la variable entre 0 y 1, o entre -1 y 1, y cuya expresión general es:

$$X_{norm} = \frac{X - \min(X)}{\max(X) - \min(X)}.$$

Si se desea re-escalar entre dos valores arbitrarios, a y b, la expresión anterior se transforma como sigue:

$$X_{norm} = a + \frac{(X - \min(X))(b - a)}{\max(X) - \min(X)},$$

Otras opciones de normalización pueden verse en la amplia literatura sobre la cuestión.

Finalmente, recordar que, cuando se lleva a cabo un proceso de normalización de variables, hay que hacerlo tanto en el subconjunto de entrenamiento como en el de test, para que ambos se basen en la misma media y varianza.

2.3.3. Ingeniería de variables (*feature engineering*)

La ingeniería de variables consiste en el proceso de conseguir, a partir de la información disponible, las variables idóneas (y el en número apropiado) para que los modelos o clasificadores proporcionen los mejores resultados posibles, dados los datos disponibles y el modelo a ejecutar. En otros términos, es el proceso de transformación de las variables seleccionadas, de forma que se obtenga el mejor rendimiento posible de los modelos de *machine learning*. Por ejemplo, transformar las variables relacionadas con la fecha de tal manera que se diferencie según el tipo de horario (“de oficina” y “de descanso”), o que se considere la cercanía al momento actual (los datos más cercanos contienen más información); los filtros de imagen (desenfocar una imagen) y la conversión de texto en números (utilizando el procesamiento avanzado del lenguaje natural, que asigna palabras a un espacio vectorial) son también ejemplos interesantes.

La mayoría de los modelos requieren que los predictores tengan forma numérica, por lo que, en caso de tener predictores de carácter categórico, hay que transformarlos en numéricos. Para implementar otro tipo de modelos, conviene transformar alguna(s) variable numérica en categórica. En el primer caso, conviene aplicar técnicas de **agrupamiento** (o *binning*), que crean agrupaciones o intervalos a partir de variables continuas; en el segundo, las técnicas de **codificación**, permiten tratar variables categóricas como si fueran continuas. Hay casos, como el de los modelos basados en árboles, que manejan, de manera natural, variables numéricas y categóricas; pero incluso en estos modelos se puede mejorar su rendimiento si se preprocesan las variables categóricas.

La identificación entre las labores de selección y de transformación de variables es bastante frecuente; sin embargo, es errónea, pues, si bien tienen algunos solapamientos, sus objetivos son claramente distintos. La ingeniería de variables tiene como objetivo la construcción de modelos más sofisticados y más interpretables que los que se pueden implementar con los datos tal y como están en el fichero raíz. La selección de variables permite que el modelo sea manejable, mejorando su interpretabilidad sin que por ello se reduzca significativamente el rendimiento del modelo.

El proceso de agrupamiento ya ha sido referido e ilustrado en la Sec. 2.2.2.1. En cuanto al proceso de codificación, se pueden distinguir dos tipos:

- **Codificación de etiquetas:** consiste en asignar a cada etiqueta un entero o valor único según el orden alfabético. Es la codificación más popular y ampliamente utilizada.
- **Codificación one-hot:** consiste en crear una nueva variable ficticia (*dummy*) binaria por cada categoría existente en la variable a codificar. Estas nuevas variables contendrán un 1 en aquellas observaciones que pertenezcan a esa categoría, y un 0 en el resto.⁵

⁵En muchas tareas, como, por ejemplo, en la regresión lineal, es común usar $k - 1$ variables binarias en lugar de k , siendo k el número total de categorías. Esto se debe a que la k -ésima variable binaria es redundante, ya que no es más que una combinación lineal de las otras, y, además, provocará problemas numéricos. Por otra

Para ejemplificar este tipo de codificación, a continuación, en el conjunto de datos `Madrid_Sale_num_sample_bin`, se crean *dummies*, una para cada cada categoría de las variables objeto de codificación. Para ello, se utiliza la función `dummyVars()` de `caret`. El resultado puede verse con la función `predict()`.

```
dummies <- dummyVars(~ ., data = Madrid_Sale_num_sample_bin)
head(predict(dummies, newdata = Madrid_Sale_num_sample_bin))
```

No debe olvidarse, igual que para todas las transformaciones descritas, hacer las mismas transformaciones en el conjunto de test.

2.4. Reducción de dimensionalidad

La **reducción de dimensionalidad** es un enfoque alternativo para filtrar las variables no informativas sin eliminarlas (como se hacía en la Sec. 2.2, que generalmente se usa para variables numéricas). La diferencia es que las técnicas de reducción de la dimensionalidad crean una proyección de los datos que da como resultado variables predictoras completamente nuevas, que son combinaciones lineales independientes formadas a partir de las variables originales, solucionando así, también, los problemas de colinealidad y multicolinealidad (perfecta o quasi-perfecta). Como se explica en el Cap. ??, el espacio de un conjunto de variables puede reducirse proyectándolo a un subespacio de variables de menor dimensión utilizando componentes principales (la técnica de reducción de la dimensionalidad por anonomasia).

Resumen

- Se presentan las principales técnicas y métodos de *feature selection* para llevar a cabo la selección (pre-selección y selección propiamente dicha) de las variables predictoras o clasificadoras más relevantes para obtener predicciones o clasificaciones exitosas.
- Se describen las principales transformaciones que se realizan en la fase de pre-procesamiento de un proyecto de modelado predictivo: las transformaciones de la escala o de la distribución de la variable objetivo, la transformación de variables (*feature engineering*) y la reducción de la dimensionalidad.
- La creación de variables predictoras a partir de los datos en bruto tiene una componente creativa, que requiere de herramientas adecuadas y de experiencia para encontrar las mejores representaciones, apoyándose, en la medida de lo posible en el conocimiento que se tenga de los datos.
- Las labores de selección y transformación de variables se ilustran con el conjunto de datos de `Madrid_Sale`, utilizándose los paquetes `caret` y `rsample`.

parte, la no inclusión de dicha variable no implica pérdida de información alguna, ya que se entiende que, si el resto de las categorías contienen un 0, la categoría correspondiente es la de la categoría eliminada.

Capítulo 3

Herramientas para el análisis en ciencia de datos

José-María Montero^a y Jorge Velasco López^b

^aUniversidad de Castilla-La Mancha ^bInstituto Nacional de Estadística de España

3.1. Introducción

En este capítulo se describen una serie de herramientas necesarias para desarrollar proyectos de ciencia de datos. Son herramientas que se utilizan pre- o post- modelado de los datos y que aumentan significativamente el rendimiento de los modelos. Tal caja de herramientas incluye el particionado del conjunto de datos, el manejo de datos no equilibrados¹, los métodos de remuestreo, el equilibrio entre sesgo y varianza, el ajuste de hiperparámetros y la evaluación de modelos, entre otras.

Para ilustrar el manejo de las herramientas anteriormente mencionadas, se utiliza el conjunto de datos `Madrid_Sale` (disponible en el paquete de **R** `Idealista18`), con datos inmobiliarios del año 2018 para el municipio de Madrid. En cuanto al software **R**, se utiliza `caret` (Kuhn, 2008), para diversas tareas de preparación de datos, y `rsample`, para muestreo.

3.2. Partición del conjunto de datos

El objetivo principal del proceso de ciencia de datos es encontrar el modelo o algoritmo que mejor resuelva la pregunta de investigación o, lo que es lo mismo, que proporcione mejores resultados.

¹El término inglés *unbalanced data* se suele traducir, en español, en lenguaje formal, por “datos no equilibrados”, si bien en la jerga de ciencia de datos, a nivel de divulgación, también se utiliza la expresión “datos desbalanceados”.

Por ejemplo, en el caso de los modelos de predicción (y en general de aquellos en los que el aprendizaje es supervisado), muy populares en la ciencia de datos, el que prediga con mayor exactitud los valores futuros de la variable objetivo a partir de los predictores seleccionados en el conjunto de datos disponible. En otras palabras, un algoritmo que, no sólo ajuste bien los datos pasados sino, lo que es más importante, que proporcione predicciones (futuras) acertadas (y precisas). Para ello, inicialmente, se dividen los datos en dos subconjuntos:

- **de entrenamiento (train):** se utiliza para desarrollar conjuntos de funciones, entrenar algoritmos, ajustar hiperparámetros, comparar modelos y realizar todas las demás actividades necesarias para seleccionar un modelo final.
- **de prueba (test):** se utiliza para validar la precisión del modelo seleccionado en la fase de entrenamiento.

A la hora de dividir el conjunto de datos en los dos subconjuntos anteriores, hay que tomar dos decisiones:

- ¿Qué porcentaje de los datos (casos, observaciones) se incluye en cada subconjunto?
- ¿Cómo se seleccionan los casos u observaciones que van a cada subconjunto?

Por lo que se refiere a la primera decisión, cuanto más grande sea el subconjunto de entrenamiento mejor será el predictor (o clasificador), aunque las mejoras serán cada vez más pequeñas. Por el contrario, cuanto más grande sea el subconjunto de prueba o test, más precisa será la estimación del error de predicción. En otros términos, lo ideal sería tener un conjunto de datos muy grande y que ambos subconjuntos fueran grandes. De esta manera, los errores de predicción serían pequeños y tendrían poca variabilidad. Sin embargo, con frecuencia, este no es el caso en la práctica, y el dilema es elegir un buen predictor (o clasificador) o una buena estimación del error de predicción. En la práctica, lo más frecuente es incluir el 70 % de los datos en el subconjunto de entrenamiento y el 30 % restante en el de test, aunque los repartos 80 %-20 % y 60 %-40 % también son muy populares.

En cuanto a la segunda decisión, la respuesta es: mediante métodos de muestreo, siendo los más utilizados el muestreo aleatorio simple y el muestreo aleatorio estratificado (véase Cap. ??).

3.2.1. Muestreo aleatorio simple

La forma más sencilla de asignar los datos a los subconjuntos de entrenamiento y prueba, es tomar una **muestra aleatoria simple** (m.a.s.) (véase Sec. ??) del conjunto de casos u observaciones del tamaño deseado, y asignarlos al subconjunto de entrenamiento, asignándose los restantes al conjunto de test.

Un problema que puede surgir con las m.a.s. es que, cuando el conjunto de datos es pequeño y los valores de uno (o más) de los predictores estén muy desequilibrados (por ejemplo, el predictor es binario y el 95 % de sus valores pertenecen a una clase o categoría y el 5 % restante a la otra), hay una probabilidad nada desdeñable de que en alguno de los dos subconjuntos (sobre todo en el de test) dicho predictor no esté representado. Si esta circunstancia ocurriese

3.2. Partición del conjunto de datos

49

en el conjunto de entrenamiento, algunos algoritmos darían error al aplicarlos al conjunto de test (donde habría datos de un predictor más). Si, por el contrario, ocurriese en el conjunto de test, los problemas surgirían por haber un predictor menos que en el conjunto de entrenamiento. Los problemas se agravarían si la desproporción anterior tuviese lugar en la variable objetivo.

A continuación, se realiza una división² 70%-30% en el conjunto de datos `Madrid_Sale_num`, generado en el Cap. 2. Para que se pueda reproducir, se establece al principio una semilla determinada.

```
library ("caret")

set.seed(123) # para permitir reproducirlo
index <- createDataPartition(Madrid_Sale_num$PRICE, p = 0.7, list = FALSE)
train <- Madrid_Sale_num[index, ]
test <- Madrid_Sale_num[-index, ]
dim(Madrid_Sale_num) # 94815
dim(train) # 66373
dim(test) # 28442
```

Como puede comprobarse, de los 94.815 datos que contiene `Madrid_Sale_num`, 66.373 (el 70%), pasan a formar parte del conjunto de entrenamiento y, el resto, 28.442, constituyen el conjunto de test.

3.2.2. Muestreo estratificado

Si se desea controlar el muestreo para que los subconjuntos de entrenamiento y prueba tengan distribuciones similares en las clases de la variable objetivo³, se puede usar **muestreo estratificado** (véase Sec. ??).⁴ Sin embargo, este tipo de muestreo, estratificando por la variable objetivo, no garantiza que ocurra lo mismo con los predictores. Es decir, presenta la misma limitación que el muestreo aleatorio simple en caso de que algún (o algunos) predictores estén muy desequilibrados y se quiera garantizar que en ambos subconjuntos las clases de la variable respuesta estén representadas de forma similar. Una posible solución es la eliminación de dichos predictores (que tendrán varianza próxima a cero), si bien ello implica pérdida de información.

A continuación, se estratifica el conjunto de datos `Madrid_Sale_num_sample_bin` del Cap. 2 por la variable objetivo (`price_bin`, el precio de venta con *binning*), que tiene cuatro categorías.

```
library("rsample")

set.seed(123) # para permitir reproducirlo
```

²Por defecto, salvo que la variable sea categórica, la división se realiza mediante muestreo aleatorio simple; por tanto, a diferencia del caso de muestreo aleatorio estratificado, que se verá a continuación, no se indica el argumento de las funciones `training` y `testing`.

³Es decir, que cada clase esté representada con, aproximadamente, las mismas proporciones en los dos subconjuntos.

⁴El muestreo estratificado también puede ser de utilidad en problemas de predicción cuando el conjunto de datos es pequeño y la distribución probabilística de la variable objetivo se desvía mucho de la normalidad.

```
table(Madrid_Sale_num_sample_bin$price_bin) |> prop.table()
#      0.4776    0.3062    0.1024    0.1116
split_estrat <- initial_split(Madrid_Sale_num_sample_bin, prop = 0.7, strata =
  ~ "price_bin")
train_estrat <- training(split_estrat)
test_estrat <- testing(split_estrat)
```

Como puede comprobarse debajo, al generar muestras aleatorias estratificadas por la variable objetivo, la distribución de éstas en los subconjuntos de entrenamiento y de prueba es aproximadamente igual:

```
table(train_estrat$price_bin) |> prop.table()
# 0.4777015 0.2913093 0.1132075 0.1177816
table(test_estrat$price_bin) |> prop.table()
# 0.4799886 0.3061750 0.1023442 0.1114923
```

3.3. Técnicas para manejar datos no equilibrados

A menudo, los datos utilizados en determinadas áreas tienen menos del 1% de eventos raros, pero precisamente su rareza es lo que los hace “interesantes”: por ejemplo, estafas en operaciones bancarias o usuarios que hacen *clic* en anuncios. En otros términos, una de las clases de la variable objetivo es dominante, pero la clase minoritaria es la que presenta interés. Sin embargo, la mayoría de los algoritmos no funcionan bien con variables cuyas clases están desequilibradas (Kuhn et al., 2013). Hay varias técnicas para manejar este problema:

- ***Downsampling***⁵: equilibra el conjunto de datos reduciendo el tamaño de las clases abundantes para que coincida con el de la clase menos prevalente. Este método es de utilidad cuando el tamaño del conjunto de datos es suficientemente grande para ser aplicado.
- ***Upsampling***⁶: equilibra el conjunto de datos aumentando el tamaño de las clases más raras. En lugar de deshacerse de datos de las clases abundantes, se generan nuevos datos para las clases raras mediante repetición o *bootstrapping*. Este procedimiento es de utilidad cuando no hay suficientes datos en la clase (o clases) rara.
- **Creación de datos sintéticos**: esta técnica consiste en equilibrar el conjunto de entrenamiento generando nuevos registros sintéticos, esto es, inventados, de la clase minoritaria. Existen diversos algoritmos que realizan esta tarea, siendo uno de los más conocidos la técnica de SMOTE (*Synthetic Minority Oversampling Technique*) (Chawla et al., 2002).
- **Otras técnicas**: como que el algoritmo implemente mecanismos para dar mayor peso a los casos de la clase minoritaria, etc.

A modo de ejemplo, a continuación se utiliza *downsampling* en el conjunto de datos *Madrid_Sale_num_sample_bin*, para mejorar la precisión del modelo, mediante el algoritmo *gradient-boosting*:

⁵También denominado *undersampling*.

⁶También denominado *oversampling*.

3.4. El enfoque de validación

51

```
# Se especifica que el modelo se entrene con downsampling
ctrl <- trainControl(
  method = "repeatedcv", repeats = 5,
  classProbs = TRUE,
  sampling = "down"
)
Madrid_Sale_num_sample_bin_downsample <- train(price_bin ~.,
  data = Madrid_Sale_num_sample_bin,
  method = "gbm",
  preProcess = c("range"),
  verbose = FALSE,
  trControl = ctrl
)
```

No existe una ventaja absoluta de un método sobre otro. La aplicación de estos métodos depende del caso de uso al que se aplique y del conjunto de datos. La función de `caret` para implementar estas técnicas está en `?caret::trainControl()`.

Una alternativa a los métodos anteriores es la implementación de algoritmos que proporcionen un buen rendimiento con variables cuyas clases están desequilibradas, de tal manera que se refuerce el aprendizaje en la clase minoritaria. La idea detrás de esta segunda opción es incluir una penalización o un sesgo que pondere las clases de tal manera que se le dé más importancia a la predicción, clasificación, etc. correcta en la clase minoritaria (para más detalles, véase ([García Abad et al., 2021](#))).

3.4. El enfoque de validación

Anteriormente, se indicaba que los datos deben dividirse en dos subconjuntos, uno de entrenamiento y otro de prueba, y que no debía usarse el subconjunto de prueba para evaluar la exactitud del modelo durante la fase de entrenamiento. Si la exactitud de las predicciones (por ejemplo, porcentaje de casos bien clasificados en un problema de clasificación) en el conjunto de test es (además de elevada) similar a la que se obtiene en el conjunto de entrenamiento, entonces el modelo entrenado generalizará bien para otros conjuntos de datos y puede darse por bueno. En otro caso, el modelo no ha entrenado bien. Por ejemplo, si la exactitud de las predicciones en el conjunto de entrenamiento es del 80 % y en el de test es del 25 % o del 97 %, entonces el modelo no puede darse por válido. En el caso del 97 %, la diferencia de porcentajes está indicando un aprendizaje “excesivo” del conjunto de datos de entrenamiento, de tal manera que el modelo en cuestión puede proporcionar muy buenas predicciones para el conjunto de datos utilizado, pero no para nuevos datos, o conjuntos de datos (esta circunstancia se conoce como sobreajuste u *overfitting*.⁷

En el caso en el que la exactitud de las predicciones sea muy distinta en los subconjuntos de entrenamiento y test, hay varias opciones (no excluyentes) para salvar dicha circunstancia:

⁷Subajuste o *underfitting* indica la imposibilidad de identificar o de obtener resultados correctos debido a un insuficiente tamaño de muestra en el conjunto de entrenamiento, o un entrenamiento muy pobre.

mejorar el modelo, ajustar sus hiperparámetros, incluir más casos en el conjunto de datos, modificar el preprocessado de los datos, ver si hay desequilibrio entre las clases, analizar si las variables predictoras son o no las adecuadas, revisar el proceso de limpieza de datos... y, posteriormente, entrenar de nuevo el modelo y determinar si es o no válido. Otra opción más drástica es, simplemente, cambiar de modelo.

Una mejor opción sería utilizar desde el principio un enfoque de validación, que implica dividir el subconjunto de entrenamiento en dos partes: un subconjunto de entrenamiento propiamente dicho y un conjunto de **validación**. Así, se puede entrenar el modelo en el nuevo subconjunto de entrenamiento y estimar su exactitud en el conjunto de validación. Es importante tener claro que el subconjunto de validación no es un subconjunto que se deje aparte, como el de test, durante la fase de entrenamiento, sino que se utiliza en dicha fase.

En resumen, con el enfoque de validación, para dar por válido un modelo, se procede como sigue:

- Dividir el conjunto de datos en subconjunto de entrenamiento y subconjunto de test.
- Dividir el subconjunto de entrenamiento en un subconjunto de entrenamiento propiamente dicho y un subconjunto de validación.
- Entrenar el modelo con los datos del subconjunto de entrenamiento propiamente dicho.
- Comprobar que la exactitud de las predicciones en dicho subconjunto de entrenamiento y en el de validación es similar (y aceptable para la exigencia que se requiere).
- Realizar predicciones con el conjunto de test y comprobar que se obtiene un porcentaje de buenas predicciones aceptable para los requisitos exigidos.
- Agregar el conjunto de test al de entrenamiento (global) y entrenar de nuevo el modelo (que será el definitivo); de esta manera se aprovecha el 100 % de los datos. Este último entrenamiento debería mejorar el modelo final, aunque la única manera de comprobarlo es mediante su comportamiento en el entorno real.

La limitación del enfoque de validación con un solo subconjunto de reserva (de validación) es que dicha validación puede ser muy variable y poco confiable, a menos que se esté trabajando con conjuntos de datos muy grandes (Molinaro et al., 2005). Y aquí es donde entran en juego los procedimientos de validación que utilizan remuestreo. El procedimiento de validación con remuestreo más utilizado es la validación cruzada (VC) k -grupos (*k-fold cross validation*). También es muy popular el que utiliza remuestreo por *bootstrapping*, que se abordará tras el VC k -grupos.

Para llevar a cabo una VC k -grupos, se divide aleatoriamente el subconjunto de datos de entrenamiento en k grupos (*folds*) de aproximadamente el mismo tamaño. El modelo se ajusta en los $k-1$ primeros grupos y el último se usa como conjunto de validación, para “validar” la bondad del modelo. A continuación, se separa el penúltimo grupo y se ajusta el modelo con los restantes, usándose el penúltimo grupo como subconjunto de validación para validar la bondad del modelo. Después se separa el antepenúltimo grupo, y así sucesivamente, hasta separar el primero. Como resultado, se obtienen k conjuntos de errores, cuyo promedio (véase Sec. @ref(evaluación)) podría servir como estimación de la exactitud y precisión (o error⁸) esperada en un conjunto de datos nuevo.

⁸Boehmke and Greenwell (2019) lo denominan error de generalización puesto que tiene lugar al “generalizar” el modelo entrenado a nuevos conjuntos de datos.

3.4. El enfoque de validación

53

El procedimiento descrito puede repetirse varias veces (VC con repetición), mediante nuevas particiones aleatorias del conjunto de entrenamiento y procediendo igual que en la iteración anterior.

En la práctica, normalmente se usa $k = 5$ o $k = 10$ (las Fig. 3.1 y 3.2 ilustran el caso de CV 5-grupos). No existe una regla formal en cuanto al tamaño de k , pero a medida que k aumenta, la diferencia entre el rendimiento estimado y real precisión estimada y el real que se obtendrá en el conjunto de test, disminuirá. En el lado negativo de la balanza, un k demasiado grande puede aumentar notablemente la carga computacional y, además, no generar mejoras significativas. A este respecto, en Molinaro et al. (2005) se concluye que CV con $k = 10$ funciona de manera similar a CV con $k = n$, la CV más extrema, también conocida como VC “dejando uno fuera” (*leave one out cross validation, LOOCV*).



Figura 3.1: CV 5-grupos (i)

y si la exactitud de las predicciones en el subconjunto de entrenamiento propiamente dicho y en el de validación es similar (y aceptable para la exigencia que se requiere):



Figura 3.2: CV 5-grupos (ii)

Aunque $k \geq 10$ contribuye a minimizar la variabilidad del error de predicción (es decir, tiende a aumentar la precisión de las predicciones), en general la CV k -grupos suele proporcionar mayores variabilidades que el *bootstrapping* (que se analiza a continuación); no ocurre lo mismo con el sesgo (Boehmke and Greenwell, 2019). Kim (2009) demostró que repetir el CV k -grupos puede ayudar a reducir la estimación del error de generalización.

Una implementación del CV k -grupos, con tres repeticiones, utilizando el conjunto de datos `Madrid_Sale_num_sample_bin`, previa mejora de la precisión del modelo mediante *downsampling*, se llevó a cabo en la Sec. 3.3 con la siguiente orden:

```
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

Bootstrapping es un procedimiento de muestreo aleatorio con reemplazamiento (Efron and

Tibshirani, 1986). Esto significa que, después de seleccionar un dato para incluirlo en el subconjunto que sea, sigue disponible para una selección posterior. Una muestra *bootstrap* tiene el mismo tamaño que el conjunto de datos original a partir del cual se obtiene. Las observaciones originales seleccionadas (una o varias veces) en la muestra conforman el subconjunto de entrenamiento, mientras que aquellas que no aparecen en ella (se les denomina *out-of-bag*) conforman el subconjunto de test.

La Fig. 3.3, tomada de Boehmke and Greenwell (2019), muestra un esquema de muestreo *bootstrap*, donde cada muestra contiene 12 observaciones, al igual que en el conjunto de datos original. Como puede observarse, el muestreo *bootstrap* lleva aproximadamente a la misma distribución de valores (representados por colores) que el conjunto de datos original.

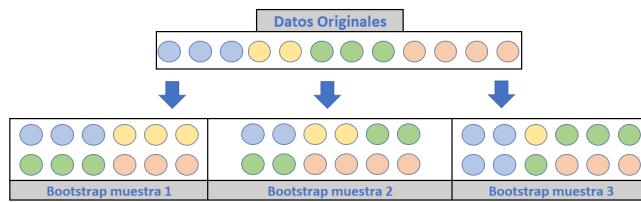


Figura 3.3: Remuestreo bootstrap

El hecho de que *bootstrapping* replique el conjunto de observaciones implica, como se dijo anteriormente, que la variabilidad del error es menor que en CV k -grupos. Sin embargo, dicha replicación puede aumentar el sesgo de la estimación dicho error. Esto puede ser un problema con conjuntos de datos muy pequeños, pero no para la mayoría de los conjuntos de datos, que suelen ser de tamaño medio o grande (por ejemplo, $n \geq 1000$).

Las muestras *bootstrap* pueden crearse fácilmente con `rsample::bootstraps()`, como se ilustra a continuación.

```
bootstraps(Madrid_Sale_num_sample_bin, times = 10)
```

Si se usa la función `trainControl()`, se debe especificar: `method = "boot"`.

3.5. Compensación (*trade off*) entre sesgo y varianza

En el entorno predictivo, el objetivo es que el error de predicción, en términos generales, o por término medio, sea lo más pequeño posible. Sin embargo, no se pueden promediar los errores porque se compensarían los positivos con los negativos. Por ello, se promedian elevados al cuadrado (considerando sólo su magnitud), denominándose dicho promedio “error cuadrático medio”, ECM (en este caso, de predicción): $E(y_j - \hat{y}_j)^2$ en términos probabilísticos, o $\sum_{j=1}^N (y_j - \hat{y}_j)^2 \frac{n_j}{N}$ en términos descriptivos. Pues bien, el ECM se puede descomponer como suma de dos componentes:

3.6. Ajuste de hiperparámetros

55

- Uno debido a la diferencia entre el valor correcto de la variable objetivo o respuesta y el que se espera que proporcione el modelo. Dicha diferencia se denomina sesgo (en inglés, *bias*), y aparece elevado al cuadrado en dicha descomposición.
- Otro debido a que, dado un conjunto de valores de las variables predictoras, la respuesta del modelo no es siempre la misma. Esta variabilidad aparece en la descomposición en forma de varianza, y por ello se denomina varianza del error de predicción o, simplemente, varianza de predicción.

Lógicamente, el incremento/reducción de uno de los componentes implica la reducción/incremento del otro.

Un sesgo muy elevado es un indicador de que el modelo es muy simple y no ha ajustado bien los datos de entrenamiento (*underfitting*), lo cual se traduce en errores de predicción elevados. Una varianza de predicción elevada (es decir, pequeños cambios en los datos de entrada producen salidas muy distintas), es un signo de complejidad en el modelo y sobreajuste (*overfitting*) de los datos de entrenamiento.

Los algoritmos con tendencia a un elevado porcentaje del ECM debido al sesgo, tienen, lógicamente, un porcentaje del ECM debido a la varianza de predicción pequeño; es decir, tienen los problemas que se derivan del infra-ajuste de los datos: malas predicciones (sesgadas) y, encima, muy precisas. Y al contrario, aquellos que tienen un porcentaje del ECM debido al sesgo pequeño, tienen un porcentaje del ECM por varianza de predicción elevado.

Los modelos lineales (regresión lineal, análisis discriminante lineal, regresión logística...) suelen tener errores por sesgo elevados.⁹ Modelos como los árboles de decisión, el *k*-vecinos más cercanos y los *support vector machine* tienen errores por sesgo pequeños (y, por tanto, varianza de predicción grande, siempre en relación al ECM total), son muy adaptables y ofrecen una flexibilidad extrema en cuanto a los patrones a los que pueden ajustarse. Sin embargo, plantean sus propios problemas, especialmente el de sobreajuste de los datos de entrenamiento, cuya consecuencia es que el modelo no se generalizará bien con datos nuevos.¹⁰

Lógicamente el modelo predictivo o clasificador deseado es el que tenga el menor ECM posible.¹¹ Si este no fuera el caso, habría que encontrar la combinación sesgo cuadrático-varianza de predicción que minimizase el ECM. La Fig. 3.4, se ha generado a partir de datos sintéticos de sesgo (al cuadrado) y varianza de predicción. En ella se puede apreciar que el valor mínimo del ECM, que es la suma de los valores del sesgo cuadrático y varianza de predicción determinados por la linea vertical amarilla.

3.6. Ajuste de hiperparámetros

Los denominados “hiperparámetros” de un modelo son los valores de las configuraciones utilizadas durante el proceso de entrenamiento. A diferencia de los parámetros, son valores que no

⁹Sin embargo, rara vez se ven afectados por el error introducido en el remuestreo.

¹⁰Por ello, el remuestreo es clave para reducir este riesgo.

¹¹Sin embargo, hay una parte del ECM que no se puede reducir: aquél que se debe a la aleatoriedad, o a la no inclusión en el modelo de variables relevantes, entre otras.

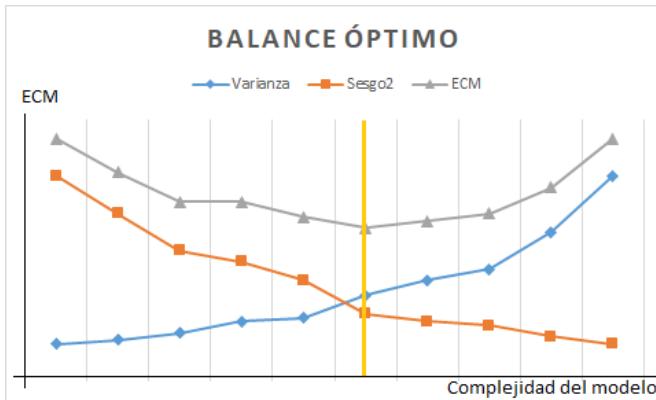


Figura 3.4: Trade-off entre sesgo y varianza

se obtienen a partir de los datos, sino que los propone el científico de datos. Podría decirse que son conjeturas (buenas conjeturas) realizadas sin utilizar las observaciones disponibles.

Los hiperparámetros, a diferencia de los parámetros, se fijan antes del entrenamiento. Siendo más específicos, al entrenar un modelo de aprendizaje automático se fijan los valores de los hiperparámetros para que con estos se estimen los parámetros. Podría decirse que son los ajustes del modelo para que éste pueda resolver de manera óptima el problema de aprendizaje automático.

Algunos ejemplos de hiperparámetros utilizados para entrenar los modelos son la ratio de aprendizaje en el algoritmo del descenso del gradiente, el número de vecinos en el algoritmo de k -vecinos más cercanos, la profundidad máxima en un árbol de decisión, el número de árboles en un bosque aleatorio (*random forest*)... Como puede apreciarse, sirven para controlar la complejidad de los algoritmos de aprendizaje automático y, por tanto, la compensación entre sesgo y varianza.

En conclusión, hiperparámetros y parámetros son conceptos bien diferentes.

A la luz de la definición de hiperparámetro, lo natural sería que el científico de datos los fijase de acuerdo con su experiencia en el pasado en problemas similares, asignándoles los mismos, o parecidos, valores. Sin embargo, existen métodos más sofisticados para resolver el problema de la “optimización de hiperparámetros”, es decir, de la obtención del conjunto óptimo de valores de los mismos que proporciona la configuración que, tras el entrenamiento, dará lugar a los mejores resultados. Entre estos procedimientos cabe destacar los siguientes por ser los que incorporan los algoritmos más populares:

- Optimización bayesiana: utiliza la moda para elegir qué hiperparámetros considerar, en función del rendimiento de las elecciones anteriores.
- Búsqueda en cuadrícula (*grid search*): prueba con todas las combinaciones posibles de la cuadrícula.

3.7. Evaluación de modelos

57

- Búsqueda aleatoria: muestrea y evalúa aleatoriamente conjuntos de una distribución de probabilidad específica.
- Optimización secuencial basada en modelos: son una formalización de la optimización bayesiana.

A modo de ejemplo, los dos hiperparámetros que más influencia tienen en un modelo de *random forest* (véase Cap. ?), son `mtry` (número de variables muestreadas aleatoriamente como candidatas en cada *split*) y `ntree` (número de árboles). Pues bien, a continuación, como viene siendo habitual, se utiliza el conjunto de datos `Madrid_Sale_num_sample_bin` para buscar el valor óptimo de `mtry` mediante la técnica de búsqueda en cuadrícula (se usa la métrica `Accuracy` (exactitud), que hace referencia a la proporción de predicciones correctas, véase Sec. @ref(evaluación)).

```
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
seed <- 7
metrica <- "Accuracy"
set.seed(seed)
mtry <- sqrt(ncol(Madrid_Sale_num_sample_bin))
tunegrid <- expand.grid(.mtry = mtry)
```

Nota

La implementación del algoritmo de *random forest* del paquete `randomForest` proporciona la función `tuneRF()`, que busca valores `mtry` óptimos dados los datos disponibles.

Se entrena el modelo y se observa que la mayor exactitud, 68,76 %, se obtiene con un `mtree` de 2.449489.

```
rf_default <- train(price_bin ~ ., data = Madrid_Sale_num_sample_bin, method = "rf",
                     metric = metrica, tuneGrid = tunegrid, trControl = control)
print(rf_default)
# Accuracy
# 0.6876006
```

3.7. Evaluación de modelos

La última fase del proceso de modelización contesta a la pregunta: ¿Cómo de bueno es el modelo entrenado? ¿Cómo de bien generaliza los (buenos) resultados obtenidos en la fase de entrenamiento a nuevos conjuntos de datos (datos *out-of-sample*)? Por ello, en este epígrafe se presentan las métricas más populares de rendimiento de modelos en los entornos de regresión (predicción) y clasificación.

En el entorno de regresión, es prácticamente imposible predecir valores exactos, y hay que conformarse con muy buenas aproximaciones a dichos valores exactos, por lo que las métricas

que se utilizan para medir la bondad del modelo entrenado suelen estar basadas en la diferencia entre los valores reales y los que predice el modelo, es decir, en los errores de predicción. La medida natural sería la media de los errores de predicción, pero, para evitar la compensación de los errores positivos con los negativos, y puesto que el interés se centra en la magnitud de los errores y no en su signo, las métricas de evaluación más populares en el entorno de regresión son:

- **Error cuadrático medio (ECM)**: es la más utilizada en tareas de regresión. Como se avanzó en 3.5, se define como la media de las diferencias cuadráticas de entre los valores objetivo (y_j) y los predichos por el modelo (\hat{y}_i), evitando así la compensación de errores positivos y negativos. Su expresión es, por tanto, $ECM = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$. Al considerar los errores de predicción al cuadrado, exacerba los errores grandes, por lo que hay que utilizar esta métrica con cuidado cuando se tengan valores anómalos en el conjunto de datos y no hayan sido tratados en las fases previas a la modelización y validación. En ese caso, incluso un modelo cuasiperfecto podría tener un ECM elevado. Si todos los errores son inferiores a la unidad hay un riesgo elevado de subestimar lo malo que es el modelo (en caso de que lo fuese).
- **Error absoluto medio (EAM)**: es una alternativa al ECM que se define como la media del valor absoluto de los errores de predicción (para evitar compensaciones): $EAM = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$. Al no elevar al cuadrado, no magnifica los errores grandes, por lo que es menos sensible que el ECM a valores anómalos, si bien tampoco es recomendable en caso de que éstos no hayan sido tratados previamente. Una ventaja que tiene es que su unidad de medida es la misma que la de la variable objetivo.
- **Raíz cuadrada del error cuadrático medio (RECM)**: “deshace”, no en un sentido matemático, sino aproximadamente, la elevación al cuadrado de los errores en el ECM y, por consiguiente, viene dada en las mismas unidades que la salida del modelo, lo que la hace más interpretable. Su expresión es: $RECM = +\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$.
- **Coeficiente de determinación (R^2)**: se define como $R^2 = 1 - \frac{\sum_{i=1}^n (y_j - \hat{y}_j)^2}{\sum_{i=1}^n (y_j - \bar{y})^2}$ (véanse detalles adicionales en el Cap. 5 de [Montero \(2007\)](#)) y su campo de variación es $[-\infty, 1]$. En la práctica totalidad de las situaciones reales toma valores entre 0 y 1, puesto que sólo toma valores negativos cuando el modelo entrenado sea muy deficiente y prediga peor que cuando se establece como predicción la media de las salidas observadas, sean cuales sean los valores de los predictores. Obviando estas situaciones, y aquéllas en las que la varianza de la variable salida no se pueda descomponer en varianza debida al modelo y varianza debida al error (por ejemplo, en el caso de una regresión potencial)¹², R^2 se puede interpretar como la reducción proporcional en el ECM que tiene lugar al predecir las salidas del modelo mediante los predictores (cualquiera que sea la función que los ligue con la salida) en vez de mediante la media de la variable output (que es la predicción óptima en ausencia de predictores). Mide, por tanto, lo bueno que es disponer de predictores para predecir los valores de la variable output o de salida; o, en otros términos, el porcentaje de varianza de la variable salida que explican los predictores a través del modelo que liga a ambos. Cuanto más cercano esté a la unidad mejor es el modelo a efectos predictivos.

¹²En estos casos es mejor utilizar otra métrica, como el ECM.

3.7. Evaluación de modelos

59

- **Coeficiente de determinación ajustado (R_{adj}^2)**: una limitación importante del R^2 es que su valor puede aumentarse artificialmente mediante la inclusión de más y más variables predictoras, pues la inclusión de las mismas o mantiene o mejora dicha métrica. Esta circunstancia puede dar lugar a confusión, pues el hecho de que un modelo utilice más variables predictoras que otro, no quiere decir que sea mejor. El R_{adj}^2 corrige dicha circunstancia penalizando la complejidad del modelo, entendiéndose que un modelo es más complejo que otro si utiliza un mayor número de variables predictoras que ese otro. Su expresión viene dada por $R_{adj}^2 = 1 - \left(\frac{n-1}{n-p-1} \right) (1 - R^2)$, y su valor nunca supera el del R^2 .
- **Deviance**: es una métrica relacionada con la estimación de modelos (especialmente modelos lineales generalizados) por el método de la máxima verosimilitud. Compara, por cociente, la verosimilitud del modelo estimado con la del modelo saturado (aquel que tiene tantos parámetros como observaciones¹³ y que, por tanto, tiene la máxima verosimilitud alcanzable). Mide el grado en el que un modelo explica la variabilidad en un conjunto de datos cuando se utiliza la estimación de máxima verosimilitud. En términos de log-verosimilitud (l) se define como $D = 2(l_{\text{Modelo saturado}} - l_{\text{Modelo propuesto}})$, y, lógicamente, cuanto menor es la deviance mejor es el modelo.
- **Raíz del error logarítmico cuadrático medio (RELCM)**: similar a RMSE, pero tomando logaritmos en los valores reales y predichos. De especial interés cuando lo que importa es la magnitud relativa (porcentual) de los errores. No se puede utilizar cuando la variable objetivo toma valores negativos. Para salvar la problemática de que la variable objetivo tome el valor cero, generalmente se agrega una constante a los valores reales y predichos de la variable salida antes de aplicar la operación logarítmica. Dependiendo del problema, se puede elegir otro tipo de constante. Su expresión viene dada por: $RMSLE = \sqrt{\frac{1}{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$.

En este manual se usan estas medidas en repetidas ocasiones. Por ejemplo, en el Cap. ?? se ajusta la regresión *ridge* en el subconjunto de entrenamiento y se evalúa su ECM en el subconjunto de test.

```
ridge_pred <- predict(ridge.mod, s = 1e10, newx = x[test, ])
mean((ridge_pred - y.test)^2)
```

En el entorno clasificadorio, las salidas del modelo pueden ser de clase (tal es el caso de los algoritmos de máquinas de vectores soporte y k -vecinos más cercanos, por ejemplo) o de probabilidad (caso de la regresión logística, los bosques aleatorios, el adaboost...). Dado que pasar de salidas probabilísticas a salidas de clase consiste únicamente en fijar umbrales de probabilidad, y que algunos algoritmos ya proporcionan el paso de salidas de clase a salidas probabilísticas, en lo que sigue no se hará distinción entre ellas.

En dicho entorno clasificadorio, es muy frecuente el uso de la **matriz de confusión**, que compara las clases (niveles categóricos) reales con las predichas en el subconjunto de test (cuyos

¹³Por tanto, pasa por todas ellas.

resultados se conocen). La Fig. 3.5 muestra un ejemplo de clasificación multiclas (en concreto, 3 clases) basado en el famoso conjunto de datos “Flor iris” de Fisher, que considera tres especies (iris setosa, iris virginica e iris versicolor). La predicción de la clase a la que pertenece una flor se hace en función del largo y el ancho del sépalo y del pétalo.

En la diagonal ascendente figura el número de flores, de cada color, cuya clase ha sido correctamente predicha. Los elementos fuera de dicha diagonal indican las flores, de cada clase, que el clasificador utilizado ha clasificado erróneamente. Como puede apreciarse, 47 de las 50 flores iris que se consideran fueron bien clasificadas. Sin embargo, dicho clasificador clasificó una flor versicolor como virgínica, y dos virgínicas como versicolores.

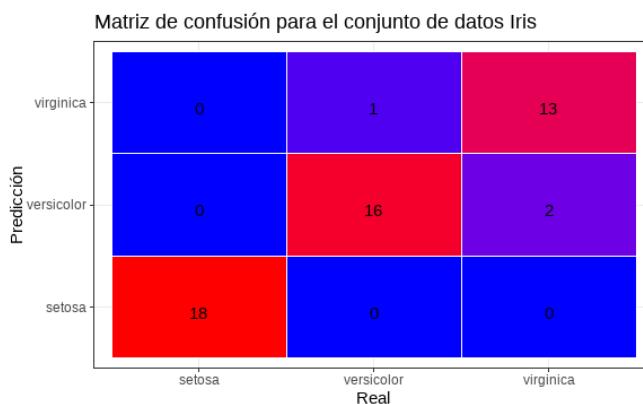


Figura 3.5: Matriz de confusión con tres clases

Aunque el concepto de matriz de confusión es muy sencillo, la terminología que lo rodea no lo es tanto; incluso podría decirse que es confusa. La predicción proporcionada por el modelo puede ser (véase Fig. 3.6):

- Un verdadero positivo (VP): predicción de verdadero y verdadero en realidad.
- Un verdadero negativo (VN): predicción de falso y falso en realidad.
- Un falso positivo (FP): predicción de verdadero y falso en la realidad.
- Un falso negativo (FN): predicción de falso y verdadero en la realidad.

		Predicción	
		Positivo	Negativo
Realidad	Positivo	VP (Verdaderos Positivos)	FN (Falsos Negativos)
	Negativo	FP (Falsos Positivos)	VN (Verdaderos Negativos)

Figura 3.6: Terminología de una matriz de confusión

Como se avanzó anteriormente, esta terminología es confusa y, por ello, se ilustra a continuación con un ejemplo. Supóngase que se está interesado en conocer si un determinado tratamiento

3.7. Evaluación de modelos

61

médico tiene efectos positivos sobre una enfermedad. Echando mano de la teoría de la contrasteación de hipótesis (véase Sec. ??), supóngase que se toma como hipótesis nula (H_0): SÍ y como hipótesis alternativa (H_1): NO. Pues bien:

- Si es cierto que el tratamiento en cuestión tiene un efecto positivo en la enfermedad y el modelo no rechaza la hipótesis nula, se tiene un VP.
- Si la hipótesis nula es falsa, es decir, si el tratamiento no tiene un efecto positivo sobre la enfermedad, y el modelo rechaza la hipótesis nula, entonces se tiene un VN.
- Si la hipótesis nula es falsa y el modelo concluye que no se rechaza la hipótesis nula de que el tratamiento cura la enfermedad, entonces se tiene un FP.
- Si es cierto que el tratamiento tiene un efecto positivo en la enfermedad y el modelo rechaza la hipótesis nula, se tiene un FN.

Las siguientes medidas se pueden calcular a partir de una matriz de confusión (es decir, a partir del número de VPs, VNs, FPs y FNs) para un clasificador binario:

- **Exactitud.** Es la proporción de predicciones correctas: $Exactitud = \frac{VP+VN}{Total}$. Responde a la pregunta: ¿Con qué frecuencia funciona correctamente el clasificador? Lógicamente, la tasa de clasificación errónea se obtiene como $\frac{FP+FN}{Total}$. En caso de desequilibrio notable de clase, por ejemplo, la clase A contiene 999 casos y la B tan sólo 1, siendo B la clase rara, la clase positiva, la precisión no sería una métrica fiable para evaluar el rendimiento clasificatorio del modelo. Por ejemplo, el clasificador podría consistir en una regla que dijese que todos los casos pertenecen a la clase A, ¡y acertaría en el 99,9% de los casos! En estos casos, sería mejor utilizar como métricas la precisión y la sensibilidad.
- **Precisión.** Da respuesta a la pregunta: cuando el clasificador predice “SÍ”, ¿con qué frecuencia predice correctamente? Su expresión viene dada por: $Precisión = \frac{VP}{VP+FP}$.
- **Sensibilidad**¹⁴. Restringe el denominador de la precisión a los SÍ reales. Responde a la pregunta: ¿cuándo en realidad es un SÍ, cuál es el porcentaje de aciertos del clasificador? Su expresión es $Sensibilidad = \frac{VP}{VP+FN}$. Por consiguiente, la sensibilidad es una medida de la probabilidad de que un caso real positivo se clasifique como positivo.
- **Especificidad.** Se define como $Especificidad = \frac{VN}{FP+VN}$. Es, por tanto, el porcentaje de verdaderos negativos respecto de todo lo que debería haber sido clasificado como negativo. Su complementario, la 1-especificidad ($\frac{FP}{FP+VN}$) es, básicamente, una medida de la frecuencia (relativa) con la que se producirá una falsa alarma, o la frecuencia con la que un caso real negativo se clasifique como positivo.
- **Puntuación F1.** Es la media armónica de la precisión y la sensibilidad. Su campo de variación es [0, 1], donde 0 indica falta total de precisión y sensibilidad y 1 significa precisión y sensibilidad perfectas. La puntuación F1 penaliza los valores extremos y se suele utilizar en caso de conjuntos de datos desequilibrados.

¹⁴A veces denominada “exhaustividad”.

- **Área bajo la curva de características operativas del receptor (área bajo la curva ROC).** Al graficar la sensibilidad (tasa de verdaderos positivos) frente a la tasa de falsos positivos (también denominada 1-especificidad), se obtiene la curva ROC. La diagonal ascendente representa la aleatoriedad. Cuanto más grande sea el área bajo la curva ROC, mejor será la precisión obtenida. Es una medida recomendable en el caso de clases desequilibradas. Un ejemplo de área bajo la curva ROC puede verse en el Cap. ??, y se reproduce en la parte derecha de la Fig. 3.7.

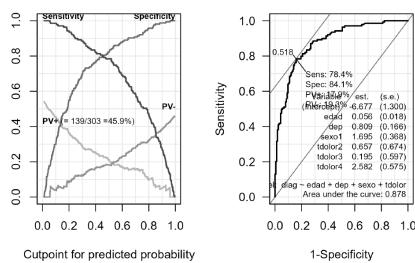


Figura 3.7: Ejemplo de curva ROC a partir de la estimación de un modelo lineal generalizado (parte derecha)

- **Índice de Gini.** Bien conocido en la literatura estadística sobre concentración, se trata de un indicador útil en el caso de clases desequilibradas. Su campo de variación es $[0,1]$, donde 0 representa la igualdad perfecta y 1 la concentración en una única clase. Puede calcularse a partir del área bajo la curva ROC de la siguiente manera: $IG = 2 \text{ Área bajo la curva ROC} - 1$. En caso de $IG=0$, el área bajo la curva ROC es $1/2$ y la curva ROC coincide con la diagonal ascendente. En caso de $IG=1$, el área bajo la curva ROC será 0 y dicha curva vale cero para todos los valores del eje de abscisas, excepto para el último, para el cual vale 1. En caso de que las observaciones no se vayan acumulando de una en una para la configuración de la curva bajo ROC, o para el cálculo directo del índice de Gini, es mejor utilizar una versión del mismo denominada “índice E” (IE, véase Cap. 3 de [Montero \(2007\)](#)), pues el índice de Gini tan sólo proporcionará una aproximación de la concentración existente; buena aproximación, pero aproximación.

- **Índice de Jaccard.** Mide las similitudes entre el conjunto clases reales y predichas por el clasificador. Se define como el cociente entre el número de coincidencias en los conjuntos real y predicho (clases predichas correctamente) y el tamaño de la unión de los dos conjuntos (el doble del número de clases a etiquetar menos el número de clases predichas correctamente): $I_{Jaccard} = \frac{VP+VN}{2Total-(Fp+FN)}$.

Otras medidas de interés son la pérdida logarítmica, el coeficiente de correlación de Matthews, el gráfico de Kolmogorov-Smirnov y el gráfico de ganancia y elevación. Éstas, entre otras, pueden verse en [Dembla \(2020\)](#), [Hernández-Orallo et al. \(2011\)](#) y [Vujović et al. \(2021\)](#), entre otros.

Con tantas métricas, ¿cómo decidir entre ellas? Será el conocimiento de la problemática que se esté estudiando (el negocio) el que normalmente guie la elección de la métrica: si se trata de

3.7. Evaluación de modelos

63

un problema de predicción de la producción de un determinado bien cuyo coste de almacenaje es elevado, lo más prudente sería utilizar el ECM para penalizar la sobreproducción; si lo que interesa son valores altos de la precisión y la sensibilidad (tal es el caso en numerosas situaciones del ámbito sanitario), la mejor medida sería la puntuación F1. No obstante, se habrá de tener siempre en cuenta que la exactitud, la precisión, la sensibilidad, la especificidad y el índice de Jaccard son buenas formas de evaluar clasificadores cuando las clases están equilibradas. En caso contrario, el área bajo la curva ROC y el IG (o el IE) son dos buenas alternativas.

Resumen

En este capítulo se describen una serie de herramientas necesarias para desarrollar proyectos de ciencia de datos. Son herramientas que se utilizan se utilizan pre- o post-modelado de los datos y que aumentan significativamente el rendimiento de los modelos. Tal caja de herramientas incluye el particionado del conjunto de datos, el manejo de datos no equilibrados, los métodos de remuestreo, el equilibrio entre sesgo y varianza, el ajuste de hiperparámetros y la evaluación de modelos, entre otras.

Se hace especial hincapié en las métricas para evaluar modelos, tanto en el entorno de regresión como en el de clasificación, y, en ambos casos, se ofrece un amplio abanico de ellas.

A efectos ilustrativos, se utiliza, fundamentalmente, el paquete `caret` y conjuntos de datos derivados del dataset `Madrid_Sale`.

Capítulo 4

Análisis exploratorio de datos

Emilio L. Cano

Universidad Rey Juan Carlos

4.1. Introducción

El análisis exploratorio de datos (AED), y en particular su visualización, es el primer análisis que se debe hacer sobre cualquier conjunto de datos. El AED se realiza mediante dos herramientas: los resúmenes numéricos y las visualizaciones gráficas. La “historia” que nos esté contando el gráfico de los datos, nos guiará hacia las técnicas de aprendizaje estadístico más adecuadas. Incluso, en muchas ocasiones será suficiente el AED para tomar una decisión sobre el problema en estudio.

4.1.1. El cuarteto de Anscombe

Un ejemplo clásico de la importancia del **AED** y, concretamente, de las representaciones gráficas es el “cuarteto de Anscombe” ([Anscombe, 1973](#)), el cual está compuesto por 11 filas de 8 variables numéricas que conforman 4 conjuntos de datos (disponibles en el objeto `anscombe`), con los mismos resúmenes estadísticos pero con propiedades muy distintas, lo que se ve fácilmente cuando se representan en forma gráfica. Si se calcula, por ejemplo, la media y la desviación típica de cada variable, se observa que son prácticamente iguales. Incluso los coeficientes de correlación de cada X con su Y son también prácticamente idénticos.

```
library(dplyr)
anscombe |> summarise(across(.fns = mean))
#> #> x1 x2 x3 x4      y1      y2      y3      y4
#> #> 9  9  9  9    7.5009  7.5009  7.5   7.5009
anscombe |> summarise(across(.fns = sd))
```

```
#>   x1    x2    x3    x4    y1    y2    y3    y4
#> 3.316 3.316 3.316 3.316 2.031 2.031 2.030 2.030
```

Sin embargo, la Fig. 4.1 muestra que, a pesar de tener medias y desviaciones típicas prácticamente iguales, los datos son muy diferentes.

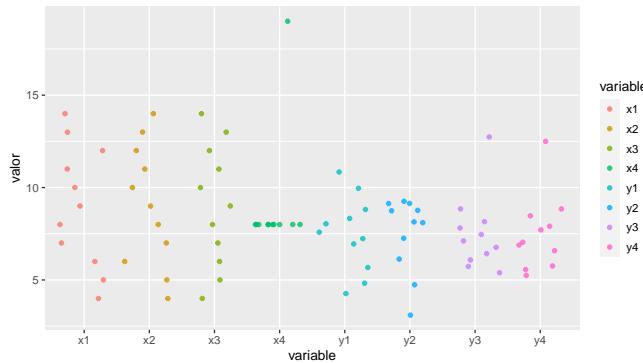


Figura 4.1: Representación de las variables del cuarteto de Anscombe

Si en el análisis por separado ya se ve la necesidad de hacer un gráfico, ésta es más evidente cuando se analizan las variables conjuntamente. La Fig. 4.2 muestra los cuatro gráficos que constituyen “el cuarteto de Anscombe”, y que se puede obtener de la propia ayuda del conjunto de datos (`example(anscombe)`). La línea de regresión que se ajusta es prácticamente la misma, y los coeficientes de correlación entre las variables X e Y de los cuatro gráficos, idénticos: 0.8163. Es evidente que la relación entre las variables es muy distinta en cada uno de los casos, y si no se visualizan los datos para elegir el mejor modelo de regresión y después interpretarlo, se pueden tomar decisiones erróneas. El cuarteto de Anscombe es muy ilustrativo, al igual que *The Datasaurus Dozen* (Matejka and Fitzmaurice, 2017) en <https://www.autodeskresearch.com/publications/samestats>.

4.1.2. Conceptos generales

Muy brevemente, se presentan una serie de conceptos esenciales para la mejor comprensión de este manual¹. Los datos que se analizan, provienen de una determinada **población**, y no son más que una **muestra**, es decir, un subconjunto de toda la población. La **Estadística Descriptiva** se ocupa del AED en sentido amplio, que se aplica sobre los datos concretos de la muestra. La **Inferencia Estadística** (véase Cap. ??) hace referencia a los métodos mediante los cuales, a través de los datos muestrales, se toman decisiones, se analizan relaciones, o se hacen predicciones sobre la población. Para ello, se hace uso de la **Probabilidad** aplicando el modelo adecuado (véase Cap. ??). Además, será muy importante considerar el método de obtención de la muestra (véase Cap. ??) que, en términos generales, debe ser representativa

¹Para un análisis extenso de los conceptos aquí expuestos puede consultarse, por ejemplo, Montero Lorenzo (2007).

4.1. Introducción

67

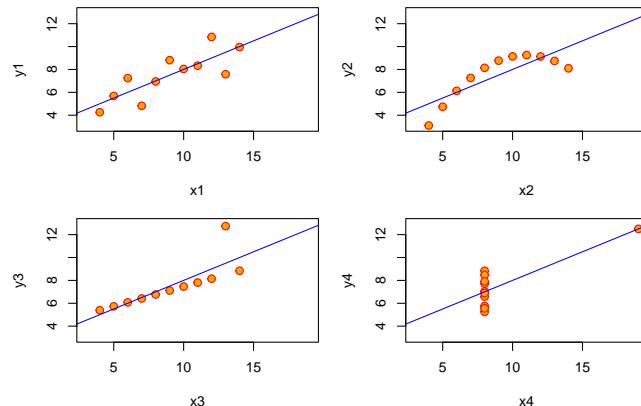


Figura 4.2: Los cuatro gráficos que constituyen el cuarteto de Anscombe junto con un ajuste lineal

de la población para que las conclusiones sean válidas. La Fig. 4.3 representa la esencia de la Estadística y sus métodos.

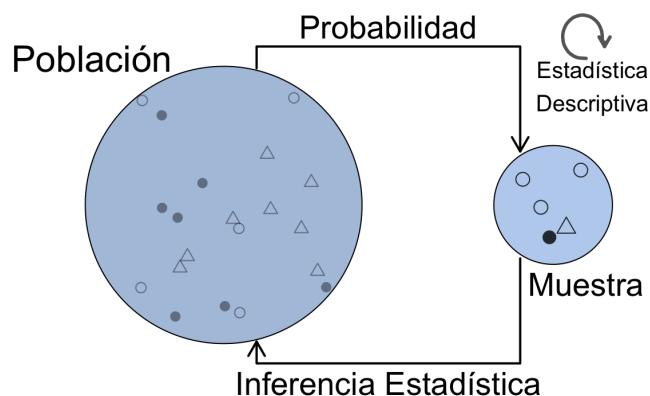


Figura 4.3: La esencia de los métodos estadísticos

Las características a observar en los elementos de una población pueden dar lugar a diferentes tipos de datos o variables. El análisis a realizar será diferente dependiendo del tipo de variable, que puede ser:

1. **Cuantitativa** (se pueden medir o contar). Se denomina **variable cuantitativa** a cualquier característica observable que pueda expresarse en valores numéricos. Se clasifican como **variables discretas** (se puede *contar* el número de valores que toma) y **continuas** (pueden tomar cualquier valor en un intervalo dado).

2. **Cualitativa** (no se pueden expresar como un número). Se denomina **variable cualitativa, atributo o factor** a cualquier característica observable que indica una *cualidad* o *atributo*. Éstas pueden tener varios niveles o solo dos (dicotómicas). Si en una variable categórica se pueden *ordenar* las categorías, entonces se tienen **variables ordinales**.

4.1.3. Componentes de un gráfico y su representación en R

De los diferentes sistemas que dispone **R** para representar gráficos (los “base”, paquete **graphics**, y los “grid”, paquete **lattice** ([Sarkar, 2008](#))), este capítulo se centra en el paquete **ggplot2** ([Wickham, 2016](#)), que forma parte del *tidyverse*, por su amplio uso y popularidad.

El flujo de trabajo con **ggplot2** se puede resumir en los siguientes pasos:

1. Proporcionar una **tabla de datos** a la función **ggplot**. Es el primer argumento (**data**) y se puede utilizar el operador *pipe*.
2. Proporcionar las **columnas** de la tabla de datos que serán representadas en el gráfico. Este será el segundo argumento (**mapping**) de la función **ggplot**, y se especifica con la función **aes** (*aesthetics*) como una lista de pares *aesthetic = variable*, de forma que el elemento especificado como *aesthetic* será “mapeado” a los valores de la variable. Esta especificación se puede hacer también en las funciones que añaden capas, que se explican a continuación. Los *aesthetics* más comunes (para muchos tipos de gráficos obligatorios) son **x** e **y**, es decir, las columnas que se usarán para el eje horizontal y el eje vertical respectivamente. Además, se pueden especificar columnas para el color, el tamaño, el símbolo de los puntos, el tipo de línea, el texto, y otros específicos del tipo de gráfico. Los *aesthetics* se pueden especificar también de forma “fija” (sin depender de ninguna variable) fuera de la función **aes**.
3. Añadir las **capas** del gráfico con los *geoms*, es decir, los objetos geométricos que representan a cada variable. Esto se indica con el operador **+**, como si se “sumasen” componentes al gráfico mediante funciones **geom_xxx**.
4. Añadir otras capas al gráfico: por ejemplo, una capa de etiquetas del gráfico (función **labs**); de ejes, para modificar los ejes y leyendas creados por defecto (funciones **scale_*_xxx**); de estadísticos, para crear nuevas variables a representar basadas en los datos (funciones **stat_xxx**).
5. Añadir un tema al gráfico: por ejemplo, en blanco y negro, o con especificaciones concretas, como el posicionamiento de la leyenda.
6. Añadir “facetas” (*facets*). De esta forma se divide el gráfico en varios subgráficos basándose en los valores de una o más variables discretas (normalmente categóricas).

En las secciones que siguen se verán ejemplos de todo el proceso. El siguiente es un ejemplo de una expresión que contiene los elementos anteriores. Se anima al lector a que los identifique con dicha lista:

```
ggplot(data, aes(x = variable)) + geom_histogram() +
  labs(title = "Título") + theme_bw() + facet_wrap(~factor)
```

4.2. Análisis exploratorio de una variable

4.2.1. Variables cualitativas

El resumen numérico de variables cualitativas se muestra en la tabla de frecuencias, la cual se puede representar con un gráfico de barras o con un gráfico de sectores². Las frecuencias absolutas son el número de observaciones en cada categoría, y las frecuencias relativas son la proporción de observaciones en cada categoría con respecto al total. Por ejemplo, el conjunto de datos `accidentes2020_data` disponible en el paquete CDR, describe los datos de accidentes de tráfico con víctimas y/o daños al patrimonio en la ciudad de Madrid registrados por Policía Municipal. Entre sus variables, contiene la variable cualitativa tipología del accidente `tipo_accidente`. Un resumen puede obtenerse tanto con la función `table()` como con el paquete dplyr, como se vio en la Sec. ???. En variables cualitativas, la categoría más frecuente se denomina **moda** de la variable³.

```
library(CDR)
library(dplyr)
accidentes2020_data |>
  count(tipo_accidente) |>
  mutate(porc = 100 * n / sum(n))
#>          tipo_accidente     n      porc
#> 1:           Alcance 7294 22.4936010
#> 2:   Atropello a animal    75  0.2312887
#> 3:   Atropello a persona 2127  6.5593487
#> 4:           Caída 2118  6.5315940
#> 5: Choque contra obstáculo fijo 4667 14.3923274
#> 6:       Colisión frontal   899  2.7723810
#> 7: Colisión fronto-lateral 8081 24.9205909
#> 8:       Colisión lateral 4386 13.5257656
#> 9:       Colisión múltiple 2231  6.8800691
#> 10:        Despeñamiento     2  0.0061677
#> 11:            Otro 251  0.7740463
#> 12: Solo salida de la vía 151  0.4656613
#> 13:           Vuelco 145  0.4471582
```

Para representar el gráfico de barras con la función `ggplot()`, se añade la capa de geometría con la función `geom_bar()` (ver Fig. 4.4).

```
library("ggplot2")
library("CDR")
accidentes2020_data |>
  ggplot() +
  geom_bar(aes(y=tipo_accidente), fill = "pink")
```

²El gráfico de sectores no es recomendable, ya que proporciona la misma información que el gráfico de barras y es mucho más difícil para el ojo humano distinguir ángulos que alturas.

³Nótese que el orden por defecto que utiliza R es el alfabético. Se puede cambiar este comportamiento reordenando los niveles del factor, por ejemplo para poner el último “Otro”.

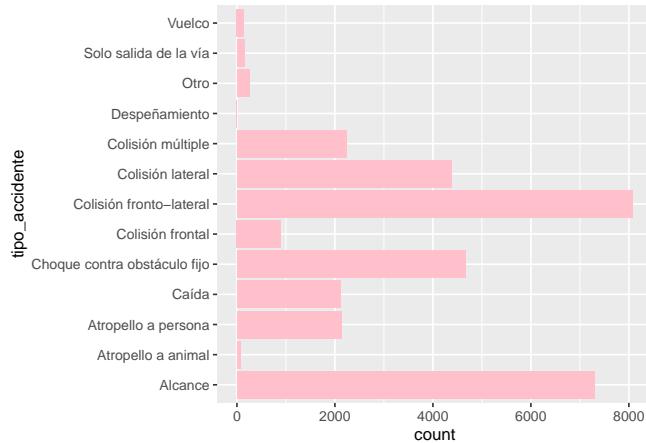


Figura 4.4: Gráfico de barras con ggplot2

El código anterior es la forma más básica de hacer un gráfico con `ggplot2`. Opciones más avanzadas pueden encontrarse en ([Wickham and Gromlund, 2016](#)).

Ya se ha comentado que los gráficos de sectores no se recomiendan a menos que se incluya información numérica. El paquete `ggstatsplot` realiza gráficos que incluyen análisis estadísticos. Por ejemplo, la función `ggpiestats()` proporciona un gráfico de sectores con algunos tests estadísticos (ver la ayuda de la función) y podría utilizarse para determinar en qué medida un conjunto de 80 ayuntamientos de distinto signo político presta o no un determinado servicio `serv` (ver el conjunto de datos en el paquete del libro `?CDR::ayuntam`). El siguiente código produce el gráfico de la Fig. 4.5.

```
library(ggstatsplot)
ayuntam |>
  ggpiestats(x = serv)
```

Una alternativa a los gráficos de sectores son los *waffle charts* (gráficos de gofre, o de tableta de chocolate). La siguiente expresión produce el gráfico de la Fig. 4.6 usando el paquete `waffle`. Con el argumento `use_glyph` se pueden incluir iconos en vez de cuadrados.

```
library(waffle)
freq <- ayuntam |>
  count(serv)
m <- setNames(freq$n, freq$serv)
waffle(m,
  rows = 4, colors = c("red", "green"))
```

4.2. Análisis exploratorio de una variable

71

$$\chi^2_{\text{gof}}(1) = 20.00, p = 7.74e-06, \widehat{C}_{\text{Pearson}} = 0.45, \text{CI}_{95\%} [0.30, 1.00], n_{\text{obs}} = 80$$

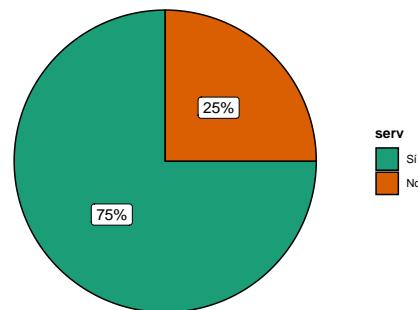


Figura 4.5: Gráfico de sectores con tests. Prestación o no de un determinado servicio X en ayuntamientos de distinto signo político

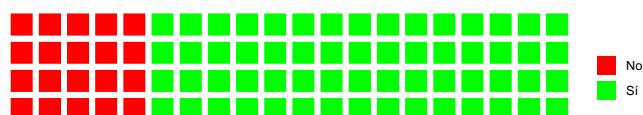


Figura 4.6: Gráfico waffle: Prestación o no de un determinado servicio X en 80 ayuntamientos de distinto signo político

4.2.2. Variables cuantitativas

Los estadísticos descriptivos más importantes que se utilizan en un AED se dividen en tres grandes grupos:

- **Medidas de posición**, que su vez se divide en (i) centrales: media (`mean()`), mediana (`median()`) y moda y (ii) no centrales: cuantiles `quantile()`, mínimo (`min()`) y máximo (`max()`).
- **Medidas de dispersión**. Las más importantes son: varianza (`var()`), desviación típica (`sd()`), rango intercuartílico (`IQR()`), desviación absoluta mediana (`mad()`) y coeficiente de variación (`sd(x)/mean(x)`).
- **Medidas de forma**: asimetría (*skewness*) y apuntamiento (*kurtosis*).

La función `summary()` de R base es una función de las llamadas “genéricas” y solo aborda las medidas de posición.

```
library('CDR')
summary(renta_municipio_data$`2019`)
#>   Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
#> 4053 9914 11595 12247 13690 32183 5697
```

Sin embargo, los estadísticos descriptivos suelen presentarse juntos “describiendo” al conjunto de datos. Existen distintos paquetes, como `summarytools`, que proporcionan un resumen completo de un vector numérico con la función `descr()` y de un conjunto de datos completo (ver opciones del paquete).

```
library(summarytools)
renta_municipio_data |>
  select(`2019`) |>
  descr()

#>Descriptive Statistics
#>2019
#>N: 55273
#>                               2019
#>-----
#>      Mean    12246.84
#>      Std.Dev 3562.94
#>      Min     4053.00
#>      Q1     9914.00
#>      Median 11595.00
#>      Q3     13690.50
#>      Max    32183.00
#>      MAD    2742.81
#>      IQR    3776.25
```

4.2. Análisis exploratorio de una variable

73

#>	<i>CV</i>	0.29
#>	<i>Skewness</i>	1.82
#>	<i>SE.Skewness</i>	0.01
#>	<i>Kurtosis</i>	5.77
#>	<i>N.Valid</i>	49576.00
#>	<i>Pct.Valid</i>	89.69

Se proporciona a continuación una breve definición de cada estadístico y su fórmula matemática como referencia general y por su uso en otras partes del libro. En las fórmulas siguientes, x_i representa cada valor de la característica en la muestra y n es el número de observaciones en la muestra.

- Media (*Mean*) \bar{x} : Es el centro de gravedad de los datos, en torno al cual varían.

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}.$$

- Desviación típica (*Std.Dev*) s : Es la raíz cuadrada de la varianza s^2 . Representan la variación promedio alrededor de la media, en las unidades de la variable y en sus unidades al cuadrado respectivamente. La siguiente fórmula corresponde a la varianza muestral, dividiendo por $n - 1$. Para la varianza poblacional, se dividiría por el tamaño de la población N . Para calcular la desviación típica, bastaría con hacer la raíz cuadrada.

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

- El mínimo (*Min*) y el máximo (*Max*) son los extremos de los datos.
- La mediana (*Median*) es el segundo cuartil Q_2 . Es el punto central de los datos, dejando la mitad de las observaciones por abajo y la otra mitad por arriba. Los cuartiles Q_1 y Q_2 , dividen los datos dejando por debajo de su valor el 25 % y el 75 % respectivamente.
- La desviación absoluta mediana (*MAD*) es la mediana de las desviaciones a la mediana.
- El rango intercuartílico es la diferencia entre el tercer y el primer cuartil. Es decir, el rango del 50 % de las observaciones centrales:

$$IQR = Q_3 - Q_1.$$

- El coeficiente de variación es el cociente entre la desviación típica y la media (en valor absoluto). Es una medida de variabilidad relativa muy útil para comparar la variabilidad entre distintas muestras o poblaciones.

$$CV = \frac{s}{|\bar{x}|}.$$

- Coeficiente de asimetría (*Skewness*). En variables simétricas sería igual a cero. Si es mayor de cero, los datos presentan asimetría positiva (una cola a la derecha, en los valores altos con respecto a la media) y si es menor de cero, los datos presentan asimetría negativa (una cola a la izquierda, en valores bajos con respecto a la media).

$$\gamma_1 = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^3}{s^3}.$$

- Coeficiente de apuntamiento (*Kurtosis*). Si los datos presentan un apuntamiento similar al de la distribución normal, será próximo a cero. Cuanto más grande, más apuntado, y cuanto más pequeño, más aplanado.

$$\gamma_2 = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{s^4} - 3.$$

Los otros tres valores que aparecen en la salida de la función `descr()` son el error típico del coeficiente de asimetría (*SE.Skewness*), el número total de valores válidos (*N.Valid*) y el porcentaje respecto al total de datos (*Pct.Valid*). El complementario de este último por tanto sería el porcentaje de valores perdidos (*missing*).

La representación de la tabla de frecuencias de una variable cuantitativa es el **histograma**⁴. Para representarlo, se cuenta el número de observaciones (frecuencia) por intervalo (*bin*). Una posible regla sería el método de Sturges⁵, que se puede hallar con la función `nclass.Sturges()`.

Para obtener tabla de frecuencias de la renta neta per cápita en 2019 usando el número de intervalos según la regla de Sturges se haría:

```
renta_municipio_data |>
  mutate(clases_sturges_renta = cut(`2019`,
    breaks = nclass.Sturges(`2019`))
)) |>
  count(clases_sturges_renta)
```

⁴En el caso de las variables discretas con un número de posibles valores pequeño, es mejor proceder igual que si fuera una variable cualitativa, obteniendo una tabla de frecuencias y gráfico de barras con la diferencia de que el orden de los posibles valores será el numérico.

⁵Este es el método que utiliza por defecto la función `hist` de **R** base, que además redondea la amplitud del intervalo para facilitar la interpretación. Otra regla muy sencilla es tomar como número de intervalos en torno a la raíz cuadrada del número total de datos.

4.2. Análisis exploratorio de una variable

75

Sin embargo, esta regla no siempre es la más apropiada, como se verá en la Sec. ??, pues debe estudiarse bien la naturaleza de la variable a analizar.

El histograma proporciona mucha información sobre la variable: (i) si es aproximadamente simétrica, (ii) si tiene forma de campana (se parece a la distribución Normal), (iii) si hay valores extremos y cómo son de frecuentes, y (iv) si puede haber mezcla de poblaciones (más de una moda).

La función `geom_histogram()` del paquete `ggplot2` añade una capa con un histograma al gráfico. El color de las barras se controla con el *aesthetics fill*, y la altura puede representar las frecuencias absolutas (recuentos) o relativas (proporciones). El número de intervalos se indica con el argumento `bins`, o alternativamente la anchura de intervalo con `bin_width`, véase la Fig. 4.7.

```
p <- renta_municipio_data |>
  tidyverse::drop_na() |>
  ggplot(aes(`2019`))

h1 <- p + geom_histogram(color = "yellow", fill = "pink")
h2 <- p + geom_histogram(
  color = "yellow", fill = "pink",
  bins = nclass.Sturges(renta_municipio_data$`2019`)
)
h3 <- p + geom_histogram(color = "yellow", fill = "pink", bins = 20)

library(patchwork)
h1 + h2 + h3
```

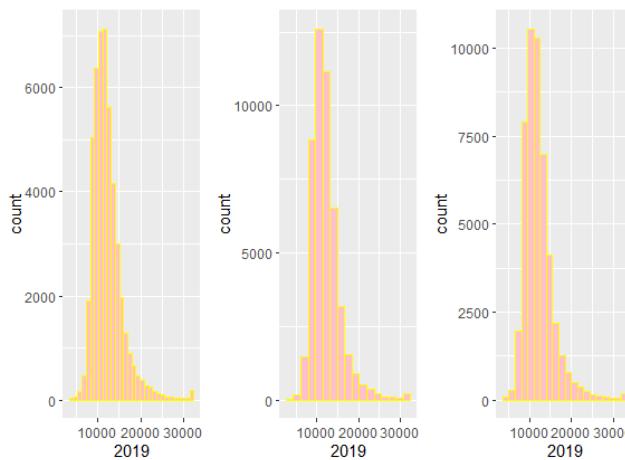


Figura 4.7: Histogramas de la renta neta per cápita en 2019 con distintos bins. Izquierda: bins por defecto ($n=30$); Centro: bins con la regla de Sturges; Derecha: bins = 20

Una representación alternativa al histograma es la línea de densidad, que sustituye las barras por una línea continua, generalmente suavizada. A continuación, se añade la linea de densidad a uno de los histogramas de la Fig. 4.7 y el resultado se puede ver en la Fig. 4.8.

```
p <- renta_municipio_data |>
  tidyrr::drop_na() |>
  ggplot(aes(`2019`))
p + geom_histogram(aes(y = after_stat(density)),
  position = "identity",
  color = "yellow", fill = "pink") +
  geom_density(lwd = 1, colour = 4)
```

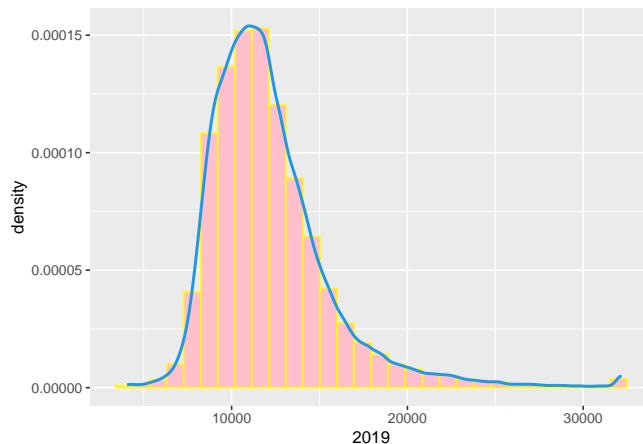


Figura 4.8: Histograma y linea de densidad de la renta neta per capita española en 2019

Otras representaciones gráficas muy útiles de las variables continuas son el gráfico de **cajas y bigotes** y el **diagrama de violín**, que se obtienen fácilmente combinando en `ggplot()` las capas `geom_boxplot()` y `geom_violin()`, respectivamente (véase Fig. 4.9).

```
p <- renta_municipio_data |>
  tidyrr::drop_na() |>
  ggplot(aes(x=0, y= `2019`))
boxplot <- p + geom_boxplot(color = "yellow", fill = "pink")
violin <- p + geom_violin(aes(), color = "yellow", fill = "pink")
boxplot + violin
```

Otra visualización básica para una variable numérica es la visualización secuencial de las observaciones, bien a través de puntos (`geom_point()`) o a través de líneas (`geom_line()`). El orden de las observaciones puede indicar cuándo se ha producido un cambio u otros patrones.

4.3. Análisis exploratorio de varias variables

77

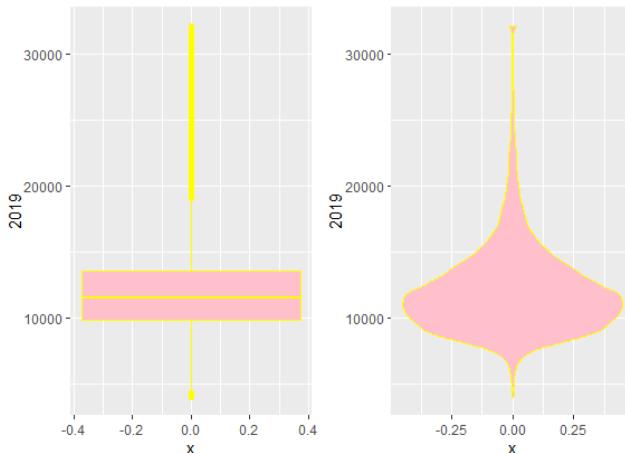


Figura 4.9: Boxplot y violin plot de la renta neta per cápita en 2019

4.3. Análisis exploratorio de varias variables

En la Sec. 4.2 se ha realizado un AED de variables aisladas, pero lo usual es incluir las relaciones entre variables dentro del AED. Las herramientas estadísticas utilizadas son: (i) las tablas de frecuencias conjuntas, que, en el caso de dos atributos, son tablas de doble entrada, con un atributo en filas y el otro en columnas, para determinar si existe asociación entre las variables, como se verá en el Cap. ??; (ii) los resúmenes numéricos, como la covarianza, el coeficiente de correlación, coeficientes de asociacion, etc. y (iii) los gráficos en los que se puede representar más de una variable.

4.3.1. Variables cualitativas

El resumen numérico sigue siendo la tabla de frecuencias, en este caso conjuntas para las distintas combinaciones de los niveles de las variables. Este tipo de tablas se denominan **tablas de contingencia** (ver Cap. ??). Para dos atributos, se puede representar en forma de tabla de doble entrada.

El resultado de la función `table()` se puede utilizar dentro de las funciones `prop.table()` y `addmargins()` para obtener las frecuencias relativas, añadir las marginales, o ambas cosas. Para el ejemplo de la prestación de servicio o no por parte de 80 ayuntamientos, `table()` podría utilizarse para dar respuesta a la siguiente pregunta: ¿La prestación pública del servicio *X* es independiente del signo político del Ayuntamiento o depende de dicho signo?

```
table(ayuntam$signo_gob , ayuntam$serv)
#>
#>           No  Sí
```

```
#> Avanzados 14 28
#> Ilustrados 6 32
```

No obstante, la representación gráfica más habitual siguen siendo los gráficos de barras, como se muestra en la Fig. 4.10 producida con el siguiente código:

```
p <- ayuntam |>
  ggplot(aes(signo_gob, fill = serv))

frecuencias <- p + geom_bar()
proporciones <- p + geom_bar(position = position_fill())

frecuencias + proporciones
```

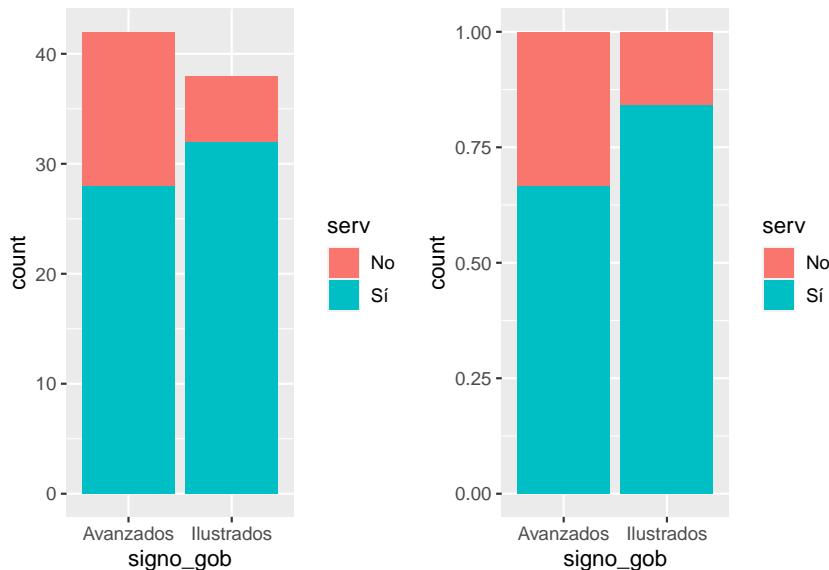


Figura 4.10: Grafico de barras de la prestación pública del servicio X por parte de 80 Ayuntamientos de distinto signo político. Izquierda: frecuencias absolutas. Derecha: frecuencias relativas.

Una visualización interesante de tablas de doble entrada son los gráficos en los que se representan las frecuencias conjuntas por medio de puntos cuyo área es proporcional a la frecuencia. La Fig. 4.11 muestra gráficamente la tabla de frecuencias conjunta de los atributos `signo_gob` y `serv` del conjunto de datos `ayuntam`.

4.3. Análisis exploratorio de varias variables

79

```
library('gplots')
balloonplot(table(ayuntam$signo_gob , ayuntam$serv))
```

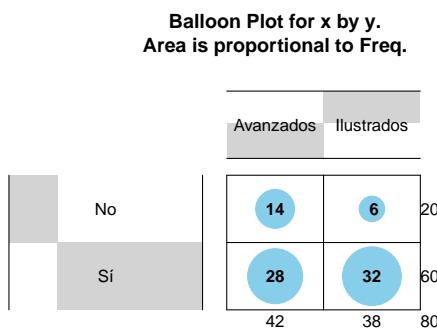


Figura 4.11: Representación gráfica de tabla de frecuencias con la función ‘balloonplot()’

Para representar dos o más factores a la vez en un único gráfico, se dispone de los gráficos de mosaico con la función `mosaicplot()` de R base, o bien el paquete `ggmosaic`, que incluye una función `geom_mosaic()` para usar en gráficos `ggplot2`. El siguiente ejemplo produce el gráfico de la Fig. 4.12:

```
library('ggmosaic')
accidentes2020_data |>
  ggplot() +
    geom_mosaic(aes(x = product(tipo_accidente, sexo),
                    fill=sexo))
```

En cualquier caso, se pueden representar más variables creando “subgráficos” o facetas (*facets*). Basta con añadir una capa al gráfico `ggplot2` con la función `facet_wrap` y argumento `facets` una lista de variables (categóricas o discretas) para cuyos valores se quiere hacer un gráfico distinto. En el siguiente código⁶ se realiza un gráfico para cada nivel del factor `tipo_accidente`. Cada uno de estos gráficos es una “faceta” del gráfico, que se muestra en la Fig. 4.13.

```
niveles <- levels(factor(accidentes2020_data$tipo_accidente))
etiquetas <- set_names(str_wrap(niveles, width = 20),
                        niveles)
accidentes2020_data |>
  ggplot(aes(sexo, fill = estado_meteorológico)) +
```

⁶Una sintaxis alternativa para especificar las facetas en la función `facet_wrap()` sería la siguiente: `facet_wrap(~tipo_accidente)`.

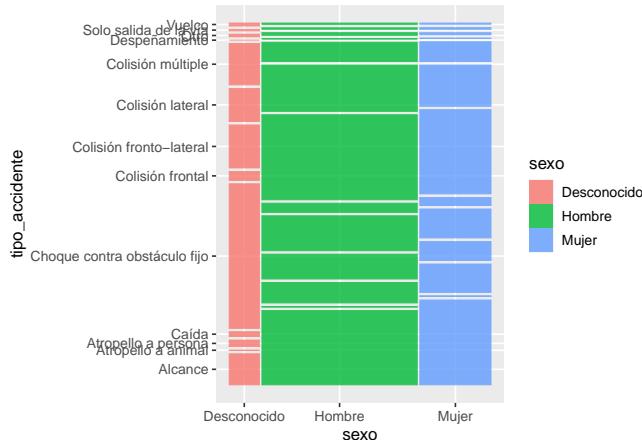


Figura 4.12: Gráfico de mosaico para relacionar el tipo de accidente y el sexo en los datos de accidentes

```
facet_wrap(vars(tipo_accidente),
           labeller = as_labeller(etiquetas)) +
  geom_bar() +
  labs(fill = "Estado Meteorológico") +
  theme(axis.text.x = element_text(angle = 90))
```

4.3.2. Variables cuantitativas

La descripción conjunta de variables numéricas se puede resumir con el vector de medias (medias de cada variable) y la matriz de varianzas-covarianzas. La covarianza s_{xy} (función `var()`) es una medida del grado de dependencia lineal entre dos variables numéricas. Si la covarianza es cero, no hay relación lineal (pero podría haber otro tipo de relación, recuérdese el cuarteto de Anscombe). Pero la covarianza es una medida que depende de la escala de las variables, y es más fácil interpretar el coeficiente de correlación lineal r_{xy} (función `cor()`), que está acotado entre -1 y 1. Cuanto más se acerque a 1, en valor absoluto, más fuerte será la dependencia lineal. Las fórmula para calcular ambos estadísticos son las siguientes:

$$s_{xy} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}),$$

$$r_{xy} = \frac{s_{xy}}{s_x \cdot s_y}.$$

Además, el cálculo de la matriz de correlaciones puede suponer un punto de partida en las técnicas de reducción de la dimensionalidad (Véanse los Cap. ??, ?? y ??). Si, por ejemplo,

4.3. Análisis exploratorio de varias variables

81

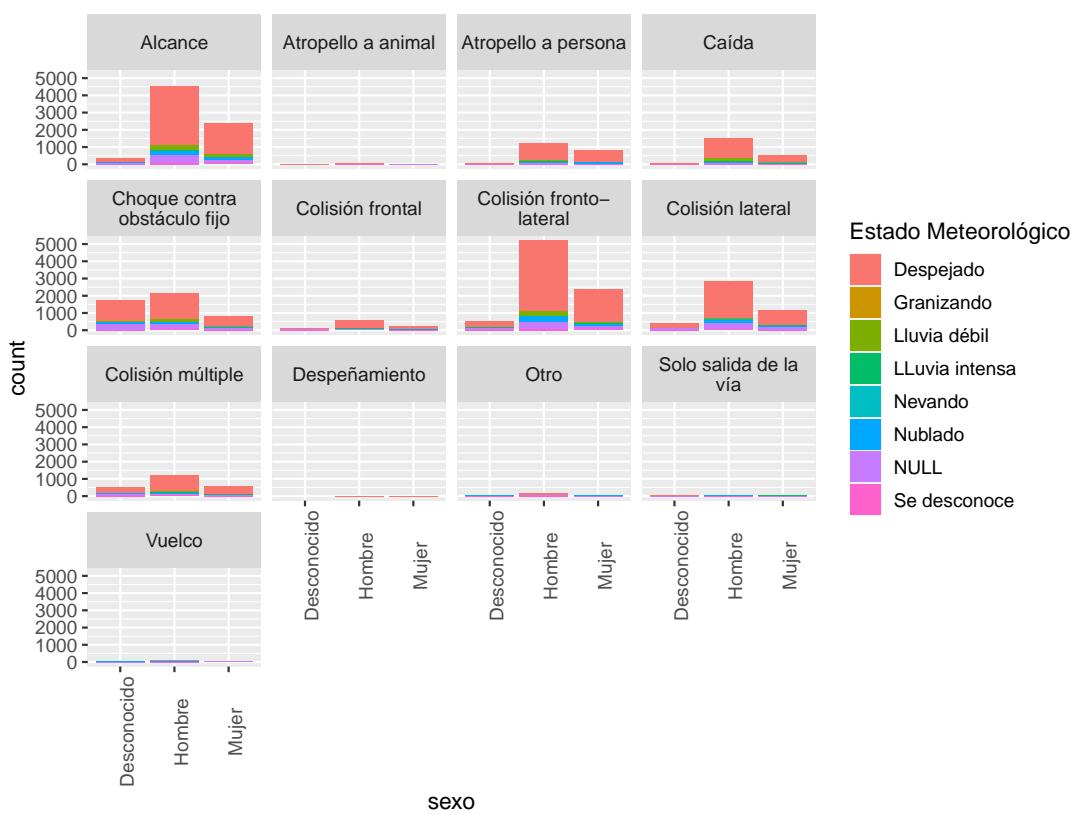


Figura 4.13: Representación de tres atributos mediante gráficos de barras conjuntos y facetas

se desea calcular la matriz de correlaciones del conjunto de datos TIC2021, que presenta las estadísticas de uso de las TIC en la Unión Europea 2021, se puede utilizar el paquete `corrplot`, que proporciona una forma elegante y versátil de representarla⁷, véase la Fig. 4.14.

```
library('corrplot')
mcor_tic <- cor(TIC2021)
corrplot.mixed(mcor_tic, order = 'AOE')
```

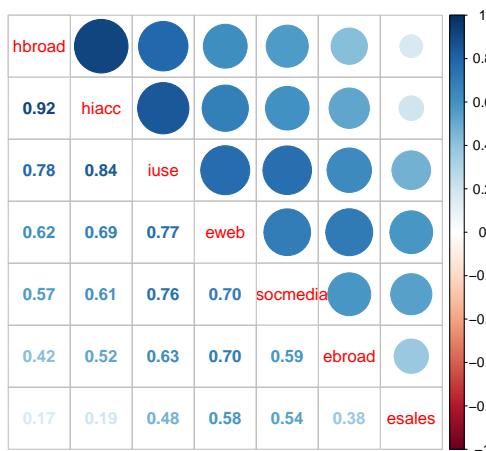


Figura 4.14: Representación gráfica de la matriz de correlaciones entre las variables del conjunto de datos TIC2021

La matriz de correlaciones se puede representar mediante “mapas de calor” (*heatmap*), es decir, un cuadrado que representa las filas y columnas de la matriz de correlaciones (variables) y donde el color de las celdas es una gradación que depende del valor de las mismas. Un mapa de calor de la matriz de correlaciones guardada anteriormente, `mcor_tic`, puede obtenerse con la expresión `heatmap(mcor_tic)`.

En cuanto a los resúmenes gráficos, el **diagrama de dispersión** es el gráfico más popular. La función `geom_point()` de `ggplot2` añade una capa con los puntos (x, y), que ya nos da una idea de la relación entre las variables, y permite interpretarla conjuntamente con el coeficiente de correlación. Se puede añadir una línea de regresión, incluida una banda de confianza, por diversos métodos (función `geom_smooth()` por defecto, una curva *loess* o *gam* dependiendo del número de filas). Alternativamente a los puntos como objeto geométrico, se pueden representar líneas (`geom_line()`).

Por ejemplo, antes de llevar a cabo un ajuste lineal, o de otro tipo, con los datos `airquality`, tal y como se hará en los Cap. ?? a ??, se podría hacer un AED previo entre las variables `Ozone` y `Temp` con el gráfico de la Fig. 4.15.

⁷Una gran cantidad de ejemplos puede verse ejecutando `example(corrplot)`.

4.3. Análisis exploratorio de varias variables

83

```
airquality |>
  dplyr::select(Ozone,Temp) |>
  ggplot(aes( x= Temp, y=Ozone)) +
  geom_point() +
  geom_smooth()
```

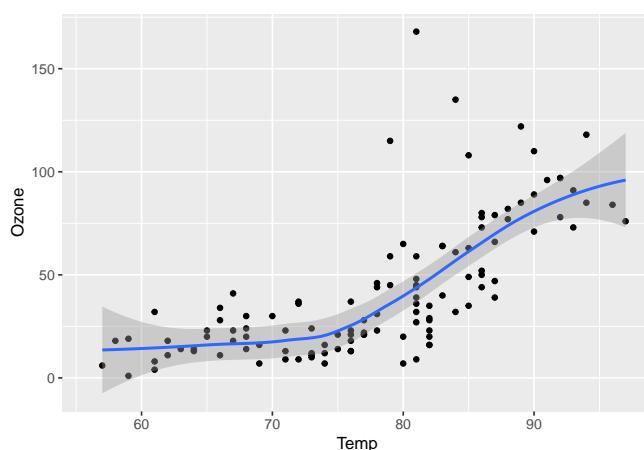


Figura 4.15: Gráfico de dispersión del Ozono frente a la Temperatura

Un caso particular es cuando la variable explicativa es el tiempo. En este caso, se tiene una serie temporal, y la representación con líneas es más adecuada (véase la Fig. 4.16).

```
contam_mad |>
  filter(nom_abv == "NOx") |>
  group_by(fecha, nom_mag) |>
  summarise(media_estaciones = mean(daily_mean, na.rm = TRUE)) |>
  ggplot(aes(x = fecha, y = media_estaciones)) +
  geom_line(aes(color = nom_mag)) +
  geom_smooth(lineWidth = 0.5, color = "black", se = TRUE) +
  theme(legend.position = "none")
```

4.3.3. Variables cualitativas y cuantitativas

Cuando se trabaja en un proyecto de ciencia de datos, lo normal es tener tanto variables cualitativas como cuantitativas. Para representar conjuntamente ambos tipos de variables existen múltiples posibilidades, algunas de las cuales se enumeran a continuación, con el tipo de gráfico adecuado:

- Una variable numérica y una variable categórica: gráficos de cajas o de violín para cada nivel de la categórica (Fig. ??), o bien gráficos de densidad para cada categoría (Fig. 4.17).

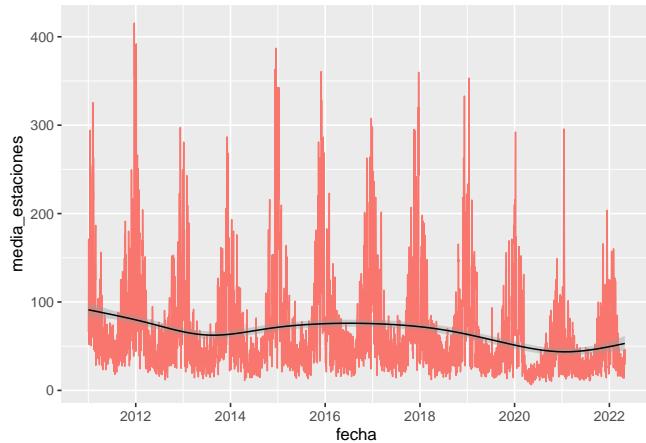


Figura 4.16: Concentración media semanal de NOx en las estaciones de medición de Madrid (enero 2011- marzo 2022)

```
contam_mad |>
  na.omit() |>
  filter(nom_abv == "PM10") |>
  filter(between(fecha, left = as.Date("2022-03-10"), right = as.Date("2022-03-20")))
  |>
  ggplot(aes(zona, daily_mean)) +
  geom_violin() +
  geom_jitter(height = 0, width = 0.01) +
  aes(x = zona, y = daily_mean, fill = zona)
```

```
library('ggridges')
contam_mad |>
  filter(nom_abv == "NOx") |>
  ggplot(aes(x = daily_mean, y = tipo, fill = tipo)) +
  geom_density_ridges()
```

- Dos variables numéricas y varias variables categóricas: gráfico de dispersión de las numéricas, y mapeado del color, tamaño y símbolo por cada nivel de las categóricas, como en la Fig. 4.18.

```
# periodo del estado de alarma
pm10_nox_mad <- contam_mad |>
  na.omit() |>
  filter(nom_abv %in% c("PM10", "NOx")) |>
  filter(between(fecha, left = as.Date("2020-03-14"), right = as.Date("2020-06-30")))
  |>
```

4.3. Análisis exploratorio de varias variables

85

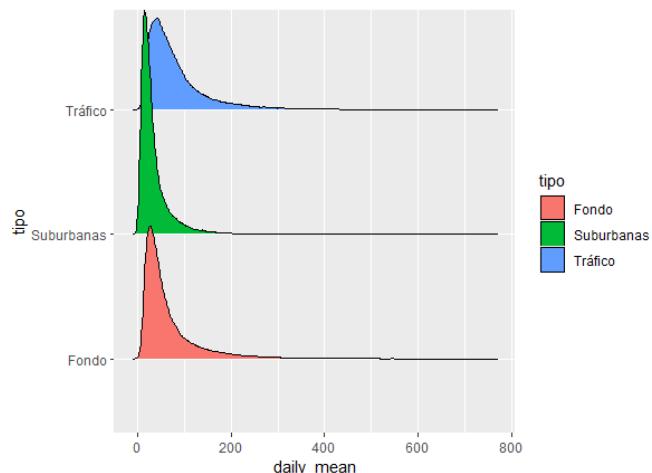


Figura 4.17: Comparación de concentraciones de NOx por tipo de estación de medición

```

select(estaciones, zona, tipo, nom_abv, daily_mean, fecha) |>
tidyrr::pivot_wider(names_from = "nom_abv", values_from = "daily_mean", values_fn =
  mean)

pm10_nox_mad |>
ggplot(
  aes(x = PM10, y = NOx, colour = tipo, size = zona)) +
geom_point()

```

- Más de dos variables numéricas y más de una categórica: gráfico de dispersión y mapeado de las otras variables a otros *aesthetics*. Combinación de geometrías y *aesthetics*. Por ejemplo, añadir puntos con efecto *jitter* a un gráfico de cajas.

En todos los casos anteriores, se pueden crear “facetas” para hacer un gráfico por cada combinación de variables categóricas, de forma que se tenga un buen número de variables representadas en un mismo “lienzo”, como en la Fig. 4.19.

```

pm10_nox_mad |>
tidyrr:: drop_na() |>
ggplot(aes(y=NOx, x= PM10, colour = tipo, shape = zona)) +
geom_point() +
geom_smooth() +
facet_wrap(vars(estaciones))

```

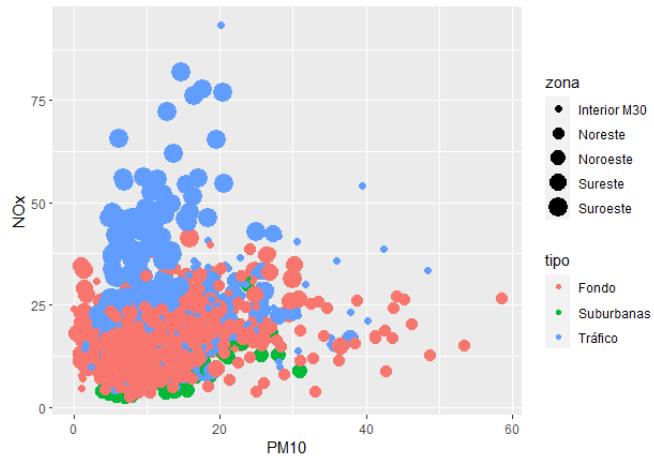


Figura 4.18: Gráfico de dispersión entre las variables ‘NO_x’, ‘PM10’, ‘zona’ y ‘tipo’ (de emplazamiento) durante el estado de alarma en la ciudad de Madrid (todas las estaciones de medición)

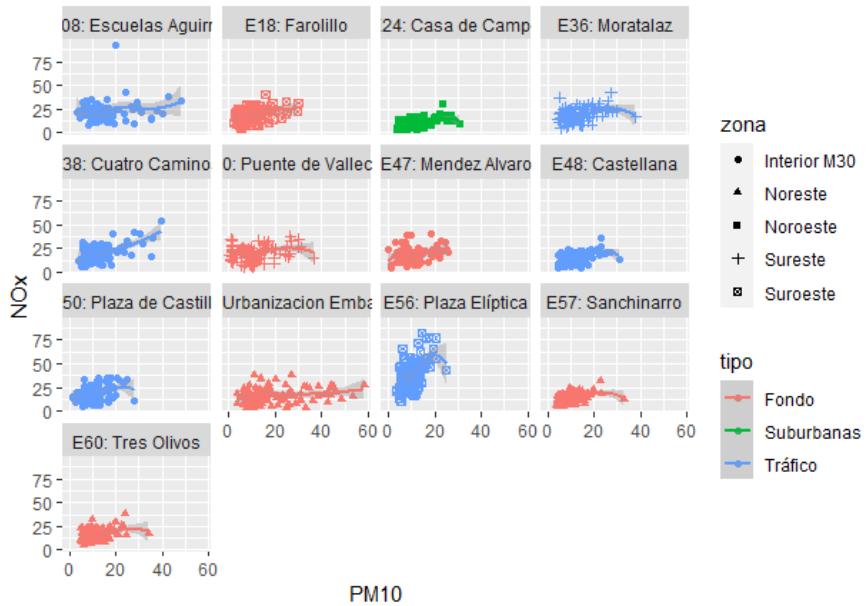


Figura 4.19: Gráfico de dispersión entre las variables ‘NO_x’, PM10, ‘zona’ y ‘tipo’ (de emplazamiento) por estación de medición durante el estado de alarma en la ciudad de Madrid

Resumen

Análisis exploratorio de una característica

- El análisis exploratorio es una tarea fundamental antes de abordar cualquier otra técnica estadística.
- Las variables categóricas se resumen con tablas de frecuencias y gráficos de barras.
- Las variables discretas se pueden resumir también con tablas de frecuencias y gráficos de barras, pero si hay muchos valores distintos también pueden ser apropiados los histogramas.
- Las variables numéricas se pueden resumir con tablas de frecuencias por intervalos, medidas de posición y de dispersión, histogramas y gráficos de cajas.
- Los gráficos de cajas sirven además para identificar valores atípicos.

Análisis exploratorio de varias características

- Las variables cualitativas se pueden resumir con tablas de frecuencias conjuntas y su representación gráfica, y con combinaciones de gráficos de barras.
- La principal medida conjunta de dos variables numéricas es el coeficiente de correlación. Para más de dos variables se suelen representar en forma de matriz.
- El gráfico de dispersión es la representación básica para dos variables numéricas. Se pueden representar estos gráficos por pares en forma de matriz de gráficos.
- Para añadir más variables, se pueden “mapear” variables a *aesthetics* (tamaño, color, etc.), añadiendo más objetos geométricos, o bien añadiendo “facetas” (subgráficos) para cada variable o para cada posible valor de una variable cualitativa.

Bibliografía

- Anscombe, F. J. (1973). Graphs in statistical analysis. *American Statistician*.
- Boehmke, B. and Greenwell, B. M. (2019). *Hands-on machine learning with R*. CRC press.
- Boskovitz, A., Goré, R., and Hegland, M. (2003). A logical formalisation of the fellegi-holt method of data cleaning. In *Advances in Intelligent Data Analysis V: 5th International Symposium on Intelligent Data Analysis, IDA 2003, Berlin, Germany, August 28-30, 2003. Proceedings 5*, pages 554–565. Springer.
- Brownlee, J. (2020). *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.
- Dembla, G. (2020). Intuition behind log-loss score. *Towards Data Science*.
- Efron, B. and Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pages 54–75.
- García Abad, J. et al. (2021). Comparativa de técnicas de balanceo de datos. aplicación a un caso real para la predicción de fuga de clientes.
- Hernández-Orallo, J., Flach, P. A., and Ramirez, C. F. (2011). Brier curves: a new cost-based visualisation of classifier performance. In *Icml*, pages 585–592.
- Kim, J.-H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational statistics & data analysis*, 53(11):3735–3745.
- Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of statistical software*, 28:1–26.
- Kuhn, M., Johnson, K., et al. (2013). *Applied predictive modeling*, volume 26. Springer.
- Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

- Matejka, J. and Fitzmaurice, G. (2017). Same Stats, Different Graphs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1290–1294, New York, NY, USA. ACM.
- Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307.
- Montero, J.-M. (2007). *Estadística descriptiva*. ALFA CENTAURO.
- Montero Lorenzo, J. (2007). Estadística descriptiva, editorial thomson-paraninfo.
- Pebesma, E. J. et al. (2018). Simple features for r: standardized support for spatial vector data. *R J.*, 10(1):439.
- Romanski, P., Kotthoff, L., and Kotthoff, M. L. (2013). Package fselector. *URL* <http://cran.r-project.org/web/packages/FSelector/index.html>.
- Saeys, Y., Inza, I., and Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517.
- Sakia, R. M. (1992). The box-cox transformation technique: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(2):169–178.
- Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer, New York. ISBN 978-0-387-75968-5.
- Schloerke, B., Cook, D., Larmarange, J., Briatte, F., Marbach, M., Thoen, E., Elberg, A., Toomet, O., Crowley, J., Hofmann, H., et al. (2021). Ggally: Extension to ggplot2.
- Staniak, M. and Biecek, P. (2019). The landscape of r packages for automated exploratory data analysis. *arXiv preprint arXiv:1904.02101*.
- Tierney, N. J. and Cook, D. H. (2018). Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations. *arXiv preprint arXiv:1809.02264*.
- van der Loo, M. P. and de Jonge, E. (2019). Data validation infrastructure for r. *arXiv preprint arXiv:1912.09759*.
- Vergara, J. R. and Estévez, P. A. (2014). A review of feature selection methods based on mutual information. *Neural computing and applications*, 24:175–186.
- Vujović, Ž. et al. (2021). Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, 12(6):599–606.
- Wei, T., Simko, V., Levy, M., Xie, Y., Jin, Y., Zemla, J., et al. (2017). Package corrplot. *Statistician*, 56(316):e24.
- 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

- análisis de la varianza, 37
- análisis exploratorio de datos, 67
- asimetría, 74
- atributo, 71
- binning, 37, 45
- bootstrapping, 54–56
- codificación, 45
 - de etiquetas, 45
 - one-hot, 45
- coeficiente
 - de asimetría, 74
 - de correlación, 84
 - de Pearson, 37
 - de Spearman, 37
 - de variación, 74
- coeficiente de
 - determinación, 59
 - determinación ajustado, 59
- coeficiente
- colinealidad, 35
- combinación lineal, 35
- control calidad, 25
- correlación, 34, 84
- covarianza, 84
- cuarteto de Anscombe, 67
- cuasicombinación lineal, 35
- curtosis, 74
- datos
 - sintéticos, 52
 - atípicos, 20
 - brutos, 11
 - duplicados, 19
 - faltantes, 17
- integración, 11, 15
- limpieza, 15
- missing, 22
- no equilibrados, 49, 52, 63
- problemas de calidad, 15
- desviación
 - absoluta mediana, 74
 - típica, 74
- deviance, 59
- downsampling, 52
- EAM, 59
- ECM, 56, 57, 59
- error
 - estructural, 17
 - a nivel de conjunto de datos, 17
 - a nivel de variable, 18
- estadística descriptiva, 67, 68
- evaluación de modelos, 49
- exactitud, 50, 62
- expecificidad, 62
- facetas (gráficos), 82
- factor, 71
- falso negativo, 62
- falso positivo, 62
- feature engineering, 31, 45
- feature selection, 31, 32
- gráfico, 70
 - de barras, 72
 - de cajas, 79
 - de densidad, 77
 - de dispersión, 84
 - de violín, 79
- hiperparámetro, 49, 57

- histograma, 76
- inferencia estadística, 67
- matriz
 - de correlaciones, 84
 - de varianzas-covarianzas, 84
- matriz de confusión, 61
- media, 73
- mediana, 73
- medidas de asociación, 37
- meta-pakage, 41
- metapaqete, 41
- moda, 73
- muestra, 67, 68
 - muestra aleatoria simple, 50
 - muestra bootstrap, 56
- muestreo
 - aleatorio estratificado, 51
 - aleatorio simple, 50
- multicolinealidad, 35
 - consecuencias, 35
 - fuentes, 35
- máximo, 73
- método de información mutua, 37
- mínimo, 73
- normalización, 44
 - z-score, 44
 - min-max, 44
- outlier, 20
- overfitting, 54, 57
- oversampling, 52
- partición del conjunto de datos, 49
- población, 67, 68
- precisión, 62
- preprocesamiento, 11
- rango intercuartílico, 74
- RECM, 59
- regresión logística, 37
- RELCM, 59
- remuestreo, 49
- selección
- tipo envoltura (wrapper), 32, 39
- tipo filtro, 32, 36
- tipo intrínseco (embedded), 32, 40
- selección de variables, 31, 32
- sensibilidad, 62
- serie temporal (gráfico), 85
- sesgo, 56
- subconjunto
 - de entrenamiento, 50, 54
 - de entrenamiento propiamente dicho, 54
 - de test, 50
 - de validación, 54
 - detest, 54
 - de validación, 53
- tabla
 - de contingencia, 79
 - de frecuencias, 71, 76
- tablas de contingencia, 37
- trade off
 - sesgo-varianza, 49, 56
- transformación de variables, 31
- underfitting, 54, 57
- undersampling, 52
- uniones
 - filtrado, 12
- upsampling, 52
- validación, 25, 53
 - cruzada con repetición, 54
 - cruzada k-grupos, 54
 - cruzada, 54
 - dejando uno fuera, 55
- Variable
 - cuantitativa, 73
- variable
 - continua, 73
 - cualitativa, 71
 - discreta, 73
 - irrelevante, 32
 - redundante, 32
- varianza, 74
- varianza cercana a cero, 33
- varianza cero, 33
- varianza de predicción, 56
- verdadero negativo, 62

Índice alfabético

93

verdadero positivo, 62

visualización, 20, 67

área bajo la curva ROC, 63