

Fundamentos de ciencia de datos con R

Gema Fernández-Avilés y José-María Montero

2023-06-01

Índice general

Prefacio	13
¡Hola mundo!	13
¿Por qué este libro?	14
¿A quién va dirigido?	15
El paquete CDR	16
¿Por qué R?	16
Agradecimientos	17
1. Modelos <i>sparse</i> y métodos penalizados de regresión	19
1.1. Introducción	19
1.2. Selección del mejor subconjunto	20
1.3. Métodos <i>shrinkage</i>	26
2. Modelización de series temporales	39
2.1. Conceptos básicos	39
2.2. Modelos ARIMA	41
2.3. Análisis de series temporales con R	42
3. Análisis discriminante	59
3.1. Introducción	59
3.2. Análisis discriminante lineal	61
3.3. Análisis discriminante cuadrático	71

4. Análisis conjunto	75
4.1. Introducción y conceptos clave	75
4.2. Tipos de análisis conjunto	76
4.3. Etapas de la realización del análisis conjunto	77
4.4. Procedimiento con R: la función Conjoint()	82
5. Análisis de tablas de contingencia	87
5.1. Introducción	87
5.2. Contraste de independencia en tablas 2×2	90
5.3. Contraste de independencia en tablas $R \times C$	96
5.4. Medidas de asociación en tablas 2×2	98
5.5. Medidas de asociación en tablas $R \times C$	101
5.6. Contrastos de independencia en tablas multidimensionales	104
I Machine learning supervisado	107
6. Árboles de clasificación y regresión	109
6.1. Introducción	109
6.2. Procedimiento con R: la función rpart()	112
6.3. Árboles de clasificación	112
6.4. Árboles de regresión	127
7. Máquinas de vector soporte	137
7.1. Introducción	137
7.2. Algoritmo SVM para clasificación binaria	139
7.3. ¿Y si tengo más de dos clases?	140
7.4. Truco del <i>kernel</i> : tratando con la no linealidad	140
7.5. Procedimiento con R: la función svm()	142
7.6. Aplicación de un modelo SVM Radial con ajuste automático en R	142

Índice general

5

8. Clasificador k-vecinos más próximos	149
8.1. Introducción	149
8.2. Decisiones a tener en cuenta	150
8.3. Procedimiento con R : la función <code>knn()</code>	152
8.4. Aplicación del modelo KNN en R	152
9. Naive Bayes	157
9.1. Introducción	157
9.2. Teorema de Bayes	157
9.3. El algoritmo <i>naive</i> Bayes	158
9.4. Procedimiento con R : la función <code>naive_bayes()</code>	161
9.5. Clasificación de clientes utilizando el modelo <i>Naive Bayes</i>	162
10. Métodos ensamblados: bagging y random forest	165
10.1. Introducción a los métodos ensamblados	165
10.2. Bagging	165
10.3. Random Forest	171
11. Boosting y el algoritmo XGBoost	183
11.1. Métodos ensamblados: bagging vs boosting	183
11.2. ¿Qué es el boosting?	184
11.3. Gradient Boosting (GB)	184
11.4. eXtreme Gradient Boosting (XGB)	193
II Machine learning no supervisado	199
12. Análisis cluster: clusterización jerárquica	201
12.1. Introducción	201
12.2. Selección de las variables	202
12.3. Elección de la distancia entre elementos	203
12.4. Técnicas de agrupación jerárquicas	208
12.5. Calidad de la agrupación y número de clusters	219

13. Análisis cluster: clusterización no jerárquica	225
13.1. Métodos de reasignación	225
13.2. Métodos basados en la densidad de elementos	230
13.3. Otros métodos	232
13.4. Nota final	233
14. Análisis de componentes principales	235
14.1. Introducción	235
14.2. Obtención de las componentes principales	236
14.3. Estimación de las componentes principales	240
14.4. Número de componentes a retener	240
14.5. Interpretación de las componentes principales	242
14.6. Reproducción de los datos tipificados y de la matriz de	244
14.7. Limitaciones del análisis de componentes principales	245
15. Análisis factorial	247
15.1. Introducción	247
15.2. Elementos teóricos del análisis factorial	248
15.3. El análisis factorial en la práctica	252
15.4. Relaciones y diferencias entre el AF y el ACP	265
16. Escalamiento multidimensional	267
16.1. Introducción	267
16.2. Medición de distancias y similitudes	268
16.3. Modelo de escalamiento multidimensional	269
16.4. Tipos de escalamiento multidimensional	271
17. Análisis de correspondencias	281
17.1. Introducción	281
17.2. Metodología del análisis de correspondencias	282
17.3. Procedimiento con R: la función ca()	284

Índice general

7

III Deep learning	291
18. Redes neuronales artificiales	293
18.1. ¿Qué es el <i>deep learning</i> ?	293
18.2. Aplicaciones del <i>deep learning</i>	295
18.3. Redes neuronales	298
18.4. Perceptrón o neurona	298
18.5. Perceptrón multiclasa	300
18.6. Funciones de activación	301
18.7. Perceptrón multicapa	305
18.8. Instalación de librerías de <i>deep learning</i> en R : Tensorflow/Keras	308
18.9. Ejemplo de red para clasificación en R	309
18.10. Ejemplo de red para regresión en R	316
19. Redes neuronales convolucionales	321
19.1. Introducción	321
19.2. Convolución	322
19.3. Neuronas convolucionales	324
19.4. Relleno del borde	325
19.5. Capas de agrupación	326
19.6. Desvanecimiento del gradiente	327
19.7. Sobreajuste	328
19.8. Generación de datos de entrenamiento artificiales	329
19.9. Ejemplo en R para el conjunto de datos CIFAR10	331
IV Ciencia de datos de texto y redes	339
20. Minería de textos	341
20.1. Introducción	341
20.2. Conceptos y tareas fundamentales	342
20.3. Análisis de sentimientos	345
20.4. Minería de textos en R	346
20.5. Ejemplo de aplicación	347

21. Análisis de grafos y redes sociales	359
21.1. Introducción	359
21.2. Teoría de grafos	360
21.3. Elementos de un grafo	361
21.4. Procedimiento con R: el paquete <i>igraph</i>	364
21.5. Análisis de influencia en un grafo aplicado a RRSS	366
21.6. Entorno social en el universo cinematográfico Marvel	372
V Ciencia de datos espaciales	377
22. Trabajando con datos espaciales	379
22.1. Introducción	379
22.2. Conceptos clave	380
22.3. Mi primer mapa	388
22.4. ¿Cómo (no) mentir con la visualización?	389
22.5. Mapas espacio-temporales	391
22.6. Mapas interactivos	392
23. Geoestadística	395
23.1. Introducción	395
23.2. Preliminares	396
23.3. Análisis estructural de la dependencia espacial	397
23.4. Kriging	409
24. Modelos econométricos espaciales	413
24.1. La dependencia espacial	413
24.2. Medidas de autocorrelación espacial	417
24.3. Modelos econométricos espaciales de corte transversal	420
25. Procesos de puntos	431
25.1. Introducción	431
25.2. Patrones puntuales espaciales en \mathbb{R}^2	432
25.3. Patrones puntuales espaciales sobre redes lineales	443

Índice general

9

VI Comunica y colabora	449
26.Informes reproducibles con R Markdown y Quarto	451
26.1. ¿Por qué informes reproducibles?	451
26.2. Documentos Quarto	456
26.3. Otros formatos	463
27.Creación de aplicaciones web interactivas con Shiny	465
27.1. Introducción	465
27.2. Componentes mínimos de una aplicación Shiny y disposición básica	466
27.3. Diseño de una aplicación Shiny	467
27.4. Elementos para la introducción de datos	470
27.5. Elementos para visualización (salida)	474
27.6. Reactividad	475
27.7. Publicación de la aplicación en la web	477
27.8. Extensiones de Shiny	479
28.Git y GitHub R	481
28.1. ¿Qué es Git y GitHub?	481
28.2. ¿Por qué usar Git y GitHub?	481
28.3. Instalación y/o actualización de R y RStudio	482
28.4. Configuración de Git y GitHub	483
28.5. Conectar Git y GitHub con Rstudio	486
28.6. Flujo de trabajo general de Git y GitHub en RStudio	490
29.Geoprocесamiento en nube	495
29.1. Introducción	495
29.2. Sintaxis de Google Earth Engine	496
29.3. Primeros pasos	496
29.4. Cálculo de anomalías	497

VII Casos de estudio en ciencia de datos	501
30. Análisis de una red criminal	503
30.1. Introducción	503
30.2. El conjunto de datos <i>Oversize</i>	503
30.3. Creación de la red mafiosa	504
30.4. Visualización de la red mafiosa	504
30.5. Importancia de los actores (delincuentes)	506
30.6. Identificación de comunidades de la mafia	507
30.7. Visualización de comunidades de la mafia	507
31. Optimización de inversiones publicitarias	511
31.1. Metodologías para optimizar las inversiones publicitarias	511
31.2. Robyn como alternativa <i>open-source</i> en R	513
32. ¿Cómo tuitea Elon Musk?	521
32.1. Introducción	521
32.2. Análisis visual de los <i>tweets</i> de Elon Musk	521
33. Análisis electoral: de Rstudio a su periódico	535
33.1. Motivación	535
33.2. Obtención de los datos	535
33.3. Transformación y primeros gráficos	536
34. Crisis: impacto en el paro de Castilla-La Mancha	543
34.1. Planteamiento	543
34.2. Evolución del paro medio anual en Castilla-La Mancha	544
34.3. Evolución del paro medio anual en función de la edad y el sexo	545
34.4. Evolución del paro medio anual según el tiempo de búsqueda de empleo	548
34.5. Evolución del paro medio anual según sexo, edad y sector de procedencia	549
34.6. Conclusiones	550

Índice general

11

35.Segmentación de clientes en el comercio minorista	551
35.1. Motivación y conceptos clave	551
35.2. El modelo <i>Recency, frequency, monetary</i> tradicional	552
35.3. El modelo <i>Recency, frequency, monetary</i> extendido	552
36.Análisis de datos en medicina	559
36.1. Justificación	559
36.2. Introducción al uso de datos en investigación clínica y ensayos clínicos	559
36.3. Análisis de supervivencia	564
36.4. Regresión de COX	566
36.5. Conclusión	568
37.Messi y Ronaldo: dos ídolos desde la perspectiva de los datos	569
37.1. Motivación	569
37.2. Las estadísticas y el fútbol	569
38.Un dato sobre el cambio climático	579
38.1. Introducción	579
38.2. Consideraciones iniciales	580
38.3. Paquetes	580
38.4. Visualización de mapas “pequeños múltiples”	581
39.Predicción de consumo eléctrico con redes neuronales	589
39.1. Introducción	589
39.2. Datos de entrada	589
39.3. Modelización	590
40.Implementación de un sistema experto en el ámbito pediátrico de atención primaria	597
40.1. Introducción	597
40.2. Marco teórico	598
40.3. Sistema experto para el ámbito pediátrico en atención primaria	602

41. El procesamiento del lenguaje natural para tendencias de moda en textil	611
41.1. Introducción	611
41.2. Análisis de tendencias de moda en textil	611
42. Detección de fraude de tarjetas de crédito	619
42.1. Introducción	619
42.2. Modelización del fraude en la compra con tarjetas de crédito	620
A. Información de la sesión	627

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).

- (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

17

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

Modelos *sparse* y métodos penalizados de regresión

María Durbán

Universidad Carlos III de Madrid

1.1. Introducción

El modelo de regresión lineal múltiple: $y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \varepsilon$, visto en el Capítulo 16, a pesar de su simplicidad, tiene importantes ventajas como la **interpretabilidad** y su buen poder **predictivo** en muchas situaciones.

En este Capítulo se va a ver cómo se puede hacer el modelo aún más interpretable y mejor predictor, y para conseguirlo se reemplazará el método de mínimos cuadrados (utilizado hasta ahora para la estimación de los parámetros) por un método alternativo.

Por lo tanto, el objetivo de este Capítulo es aprender técnicas para mejorar:

- **Precisión de la predicción:** en particular cuando el número de variables es mayor que el número de observaciones: $p > n$ (algo que ocurre con mucha frecuencia hoy en día). En este caso no se pueden utilizar mínimos cuadrados ya que la matriz de diseño no es de rango completo, y por lo tanto, no se puede encontrar una solución al problema de minimización. Por ello, se necesita reducir el número de variables, que además, evitará que se sobreajusten los datos.
- **Interpretabilidad del modelo:** al eliminar las variables irrelevantes (es decir, haciendo cero los correspondientes coeficientes) se obtendrá un modelo que es más fácil de interpretar. Por lo tanto, se presentarán varios métodos para llevar a cabo de forma automática la *selección de variables*.

Los métodos para reducir el número de variables en el modelo son:

- **Selección del mejor subconjunto:** su objetivo es identificar un subconjunto de entre los p predictores que se considera que son los que están relacionados con la variable respuesta.
- **Shrinkage:** en este caso no se quieren seleccionar variables explícitamente, sino que se añade una penalización que penaliza el número de coeficientes o su tamaño.
- **Reducción de la dimensión:** el objetivo es proyectar los p -predictores en un subespacio de dimensión más pequeña (mediante el uso de combinaciones lineales de variables, las cuales se usarán como predictores). Dichas combinaciones lineales se llaman **Componentes principales** y a su análisis se dedica el Cap. 14.

Por lo tanto, en este Capítulo se ven los dos primeros métodos.

1.2. Selección del mejor subconjunto

Supóngase que se tiene acceso a p variables predictoras, pero se quiere un modelo más simple que involucre solo a un subconjunto de esos p predictores. La forma lógica de conseguirlo es considerar todos los posibles subconjuntos de los p predictores y elegir el mejor modelo de entre todos los modelos construidos con cada uno de los subconjuntos de variables. Los pasos a seguir serían:

1. Se crea el modelo **nulo**, M_0 , que es aquel que solo contiene la ordenada en el origen y ningún predictor. Este modelo simplemente predice la media muestral para cada observación.
2. Para cada valor de $k = 1, 2, \dots, p$ se calculan los $\binom{p}{k}$ modelos que contienen k predictores. Es decir, los p modelos que contienen 1 predictor, los $p \times (p - 1)/2$ modelos que contienen 2 predictores, etc.
3. Para cada valor de k , se elige el mejor entre los $\binom{p}{k} = \frac{p!}{(p-k)!k!}$ posibles modelos y se denota por M_k . Es decir, M_1 sería el mejor modelo entre los p modelos con una sola variable, M_2 sería el mejor modelo entre los modelos con dos variables, etc. En este caso, el **mejor** modelo sería aquel cuyo RSS (suma de residuos al cuadrado) sea menor, o equivalentemente, aquel cuyo R^2 sea mayor.
4. Elegir entre los modelos: M_1, \dots, M_p aquel que es mejor utilizando un criterio como AIC, BIC o R^2 ajustado.

Este método se puede usar también en el caso de GLMs, es decir se usará el *deviance* en vez de RSS .

1.2.1. Procedimiento con R: la función `regsubset()`

Se va a aplicar el método descrito al conjunto de datos `Hitters` del paquete `ISLR2`. El objetivo es predecir el sueldo, `Salary`, de jugadores de béisbol a partir de varias variables asociadas con su rendimiento el año anterior.

La variable `Salary` no está disponible para algunos de los jugadores que se pueden identificar utilizando la función `is.na()`. Y la función `sum()` permite ver cuántos hay. Se utilizará `na.omit()` para eliminarlas.

```
library("ISLR2")
Hitters <- na.omit(Hitters)
```

La función `regsubsets()` del paquete `leaps` lleva a cabo la selección del mejor subconjunto de variables identificando el mejor modelo que contiene un número dado de variables (1,2,3, etc.) atendiendo a *RSS*. La sintaxis usada es similar a la de la función `lm()`.

```
library("leaps")
regfit_full <- regsubsets(Salary ~ ., Hitters)
```

Los resultados se pueden ver usando `summary()`, donde se muestra el mejor modelo para cada subconjunto de variables. Con un asterisco indica las variables incluidas en cada modelo. Por ejemplo, el mejor modelo con dos variables incluye `Hits` y `CRBI`. Por defecto, `regsubsets()` solo muestra los resultados de los modelos que contienen hasta ocho variables. La opción `nvmax` se puede usar para incrementar esta cantidad, por ejemplo hasta 19 variables (que es el número de variables predictoras en el conjunto de datos):

```
regfit_full <- regsubsets(Salary ~ .,
  data = Hitters,
  nvmax = 19
)
reg_summary <- summary(regfit_full)
```

La función `summary()` devuelve diferentes medias de bondad de ajuste: R^2 , *RSS*, R^2 ajustado, C_p y *BIC*. Se utiliza esta información para elegir el *mejor* de entre todos los modelos.

```
names(reg_summary)
#> [1] "which"    "rsq"      "rss"      "adjr2"    "cp"       "bic"      "outmat"   "obj"
reg_summary$adjr2
#> [1] 0.3188503 0.4208024 0.4450753 0.4672734 0.4808971 0.4972001 0.5007849
#> [8] 0.5137083 0.5180572 0.5222606 0.5225706 0.5217245 0.5206736 0.5195431
#> [15] 0.5178661 0.5162219 0.5144464 0.5126097 0.5106270
```

Por ejemplo el R^2 ajustado mayor corresponde al modelo con 11 variables. Se pueden también visualizar los resultados y dibujar simultáneamente, por ejemplo, los valores de *RSS* y R^2 ajustado de todos los modelos.

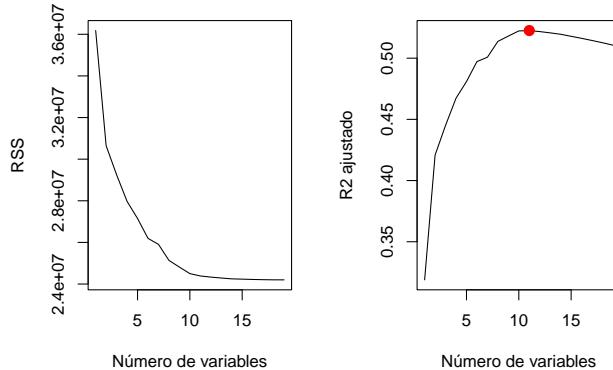


Figura 1.1: valores de R^2 y R^2 ajustados correspondientes a modelos con distinto número de variables

Otra manera de visualizar los resultados es:

```
plot(regfit_full, scale = "adjr2")
```

La primera fila tiene un cuadrado negro en cada una de las variables elegidas de acuerdo al modelo con mayor R^2 ajustado (en este caso, sería similar para los otros criterios).

Varios modelos tienen un valor de R^2 ajustado próximo a 0,52, pero es el modelo con 11 variables el que alcanza el mayor valor. La función `coef()` permite ver los coeficientes estimados de este modelo.

```
coef(regfit_full, 11)
#> (Intercept)      AtBat       Hits       Walks      CATBat      CRUNS
#> 135.7512195 -2.1277482  6.9236994  5.6202755 -0.1389914  1.4553310
#> CRBI          CWalks     LeagueN DivisionW PutOuts      Assists
#>  0.7852528 -0.8228559 43.1116152 -111.1460252   0.2894087  0.2688277
```

1.2.2. Selección *stepwise*

Cuando el número de variables predictoras, p , es grande, el método anterior es computacionalmente muy costoso ya que el número de posibles combinaciones de variables crece de una manera alarmante. En general, la función `regsubset()` puede lidiar con hasta 30-40 variables predictoras. Además, otro problema es el sobreajuste. Si se tienen 40 variables, se estarían ajustando millones de modelos, y puede que el modelo elegido funcione muy bien en los datos utilizados para su construcción, pero no tan bien en un nuevo conjunto de datos. Una alternativa es el método **stepwise**. La idea detrás de este método es similar a la anterior, pero solo se mira un conjunto mucho más pequeño de modelos.

1.2. Selección del mejor subconjunto

23

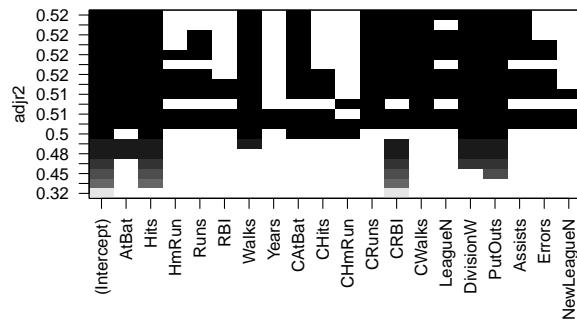


Figura 1.2: Variables seleccionadas en cada uno de los modelos y su correspondiente valor de R^2 ajustado.

Hay dos posibilidades de hacer stepwise: **forward** y **backward**. Ambas son bastante similares; la principal diferencia es el modelo del que se parte: del modelo sin ninguna variable predictora o del modelo con todas ellas.

1.2.2.1. Forward stepwise

En este caso se comienza con el modelo *nulo*, M_0 , y se van añadiendo variables secuencialmente. En particular, en cada paso (*step*) la variable que proporciona la mayor mejora al ajuste es la que se añade al modelo. Los pasos a seguir serían:

1. Se crea el modelo **nulo**, M_0 .
2. Para cada valor de $k = 0, 1, 2, \dots, p$:
 - i. Se consideran todos los $p - k$ modelos que surgen de aumentar el modelo M_k con un predictor.
 - ii. Se elige el **mejor** de esos $p - k$ modelos, que se denotará M_{k+1} . El término **mejor** significa tener el *RSS* más bajo o el R^2 más alto.
3. Se elige entre los modelos M_0, \dots, M_p aquel que es mejor utilizando un criterio como AIC, BIC o R^2 ajustado.

Este enfoque tiene ventajas computacionales claras, ya que el número de modelos ajustados es mucho menor, pero no garantiza que el modelo elegido sea el mejor modelo posible, especialmente si existe correlación entre las variables predictoras.

1.2.2.2. Backward stepwise

En este caso se comienza con el modelo que incluye todas (p) las variables predictoras y se van eliminando de forma iterativa hasta llegar al modelo nulo (M_0). Los pasos serían:

1. Se ajusta el modelo M_p , que contiene todas (p) las variable predictoras.
2. Para cada valor de $k = p, p-1, \dots, 1$:
 - I. Se consideran todos los k modelos que surgen de reducir en el modelo M_k un predictor, es decir, modelos con $k-1$ variables predictoras.
 - II. Se elige el **mejor** de esos k modelos, que se denotará M_{k-1} . El término **mejor** significa tener el RSS más bajo o el R^2 más alto
3. Se elige entre los modelos M_0, \dots, M_p aquel que es mejor utilizando una criterio como AIC, BIC o R^2 ajustado.

Tanto en el caso de forward como backward stepwise, se busca el mejor modelo sólo entre $1+p(p+1)/2$ modelos, lo que permite su uso cuando p es demasiado grande para seleccionarlos mediante la búsqueda del mejor subconjunto.

El método backward necesita que el número de observaciones n sea mayor que el de variables predictoras p (ya que se necesita ajustar el modelo con todas las variables). Por el contrario, el método forward se puede usar incluso cuando $n < p$.

1.2.2.3. Procedimiento con R: la función `regsubset()`

La función `regsubset()` permite utilizar el método backward y forward, usando el argumento `method = "forward"` o `method = "backward"`:

```
regfit_fwd <- regsubsets(Salary ~ .,
  data = Hitters,
  nvmax = 19, method = "forward"
)
regfit_bwd <- regsubsets(Salary ~ .,
  data = Hitters,
  nvmax = 19, method = "backward"
)
```

A continuación se ve como, por ejemplo, para el caso del mejor modelo con 2 variables los tres métodos: mejor subconjunto, forward y backward dan lugar conjuntos de variables diferentes:

```
coef(regfit_full, 2)
#> (Intercept)      Hits       CRBI
#> -47.9559022   3.3008446  0.6898994
coef(regfit_fwd, 2)
#> (Intercept)      Hits       CRBI
```

1.2. Selección del mejor subconjunto

25

```
#> -47.9559022  3.3008446  0.6898994
coef(regfit_bwd, 2)
#> (Intercept)      Hits       CRuns
#> -50.8174029   3.2257212  0.6614168
```

Hay que decidir un criterio para elegir el mejor modelo: R^2 ajustado, BIC , etc. Si se usa R^2 ajustado, en ambos casos el mejor modelo es el que tiene 11 variables:

```
which.max(summary(regfit_fwd)$adjr2)
#> [1] 11
which.max(summary(regfit_bwd)$adjr2)
#> [1] 11
```

Con BIC , el modelo elegido no tiene el mismo número de variables:

```
which.min(summary(regfit_fwd)$bic)
#> [1] 6
which.min(summary(regfit_bwd)$bic)
#> [1] 8
```

Otra posibilidad es utilizar como criterio el error de predicción, y para ello se puede utilizar algún esquema de validación cruzada. A continuación se ilustra el caso en el que se divide la muestra en dos subconjuntos: *training* y *testing*, pero se puede utilizar cualquier otro método (*k-fold*, etc.).

```
set.seed(1)
entreno <- sample(c(TRUE, FALSE), nrow(Hitters), replace = TRUE)
test <- (!entreno)
```

Se utiliza `regsubsets()` en la muestra de entrenamiento para obtener los modelos con distinto número de variables predictoras:

```
regfit_best <- regsubsets(Salary ~ ., data = Hitters[entreno, ], nvmax = 19)
```

Para calcular el error de predicción, dado que la función `regsubset()` no tiene asociada una función `predict()`, se han de calcular “manualmente” los valores predichos para la muestra de test. Para eso se necesita la matriz de diseño del modelo.

```
test.mat <- model.matrix(Salary ~ ., data = Hitters[test, ])
```

Ahora, para cada modelo de tamaño k , se extraen los coeficientes de `regfit_best` para el mejor modelo de ese tamaño, se multiplica el vector de coeficientes por la matriz de diseño y se obtienen las predicciones; a continuación se calcula el error cuadrático medio (MSE).

```
val_errors <- rep(NA, 19)
for (i in 1:19) {
  coefi <- coef(regfit_best, id = i)
  pred <- test.mat[, names(coefi)] %*% coefi
  val_errors[i] <- mean((Hitters$Salary[test] - pred)^2)
}
```

El mejor modelo es el que contiene 7 variables:

```
val_errors
#> [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4
#> [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4
#> [17] 141767.4 142339.6 142238.2
```

1.3. Métodos *shrinkage*

Los métodos anteriores se basan en el ajuste de modelos mediante mínimos cuadrados. Ahora se trabajará con un método diferente: ***shrinkage***. Este método se basa en una modificación de mínimos cuadrados añadiendo una penalización que *encoje* los coeficientes del modelo típicamente hacia 0. Una de las ventajas de este método es que reduce la varianza de los coeficientes estimados.

1.3.1. Regresión *ridge*

Se recuerda que el ajuste por mínimos cuadrados estima $\beta_0, \beta_1, \dots, \beta_p$ mediante los valores que minimizan:

$$RSS = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2.$$

La **regresión ridge** añade un término de penalización controlado por un parámetro (que habrá que elegir) que penalizará los coeficientes que se hacen demasiado grandes. Cuanto más grande es el coeficiente mayor es la penalización:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2. \quad (1.1)$$

En realidad lo que se está haciendo es hacer pagar al modelo un precio por el hecho de que los coeficientes no sean cero, y el precio será tanto mayor cuanto mayor sea la magnitud del coeficiente. A esta penalización se le llama **penalización shrinkage** porque “anima” a los coeficientes a que se *contraigan* hacia 0. La magnitud de dicha contracción está gobernada por lambda, el parametro de afinado o regulación (también conocido en la jerga como de tuneado). Si $\lambda = 0$ se está en el caso de mínimos cuadrados, y cuanto mayor sea λ , mayor será el precio a pagar

1.3. Métodos shrinkage

27

para que esos coeficientes sean distintos de 0. Si λ es extremadamente grande los coeficientes estarán muy próximos a 0 para poder hacer el segundo término pequeño (recuérdese que se quiere minimizar RSS más la penalización). Aunque valores más grandes de los coeficientes den un mejor ajuste (y por lo tanto un menor RSS), el término de penalización se hará grande y no se alcanzará el mínimo. Por lo tanto λ sirve como equilibrio entre un buen ajuste del modelo y el tamaño de los coeficientes (y por lo tanto el número de coeficientes distintos de cero).

Elegir un buen valor de λ es crítico. Se utilizará la validación cruzada para elegirlo.

1.3.1.1. Escalado de variables predictoras

Un punto importante en regresión ridge es si las variables predictoras están escaladas o no.

En el caso de mínimos cuadrados, el método es *invariante a la escala* (*scale-invariant*), es decir, que si se multiplica una variable predictora X_j por una constante c , esto solo implica que el coeficiente estimado se ve multiplicado por $1/c$, pero $X_j\hat{\beta}_j$ no cambia. Sin embargo, en el caso de la regresión *ridge* los coeficientes estimados pueden cambiar sustancialmente si se multiplica una variable predictora por una constante, ya que aparecen todos juntos en el término de penalización. Por lo tanto, antes de utilizar la regresión *ridge* (o cualquier método de regularización) es importante **estandarizar las variables predictoras**, dividiendo cada variable por su desviación estándar, de forma que todas tengan desviación estándar igual a 1.

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_{ij})^2}}$$

Con esto se consigue que los coeficientes sean comparables.

La regresión *ridge* en muchas ocasiones da lugar a un menor MSE que el obtenido con mínimos cuadrados ordinarios. Sin embargo, por muy grande que sea λ los coeficientes no serán 0, sino que estarán próximos a cero, por lo que este método no es realmente un método de selección de variables.

La regresión ridge puede ser muy útil cuando hay variables predictoras altamente correlacionadas, pero se desea mantener todas en el modelo. En estos casos, la regresión ridge soluciona los problemas de multicolinealidad.

1.3.1.2. Procedimiento con R: la función `glmnet()`

El paquete que se va a usar para llevar a cabo la regresión *ridge* (y para otros métodos de regresión *shrinkage*) es `glmnet`. La función principal en este paquete se llama también `glmnet()`. Esta función tiene una sintaxis un poco diferente a las funciones usuales para el ajuste de distintos modelos en *R*. Es necesario pasarle la matriz X de variables predictoras (sin la columna correspondiente a la ordenada en el origen), y el vector Y con la variable respuesta. Para ilustrar su uso se utilizarán los datos anteriores sobre béisbol.

```
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

La función `glmnet()` tiene un argumento, `alpha`, que determina el tipo de penalización que se añade en el modelo. En el caso de regresión ridge `alpha=0`.

Por defecto, la función `glmnet()` elige de forma automática el rango de valores de λ . Sin embargo, a modo ilustrativo, se va a elegir la rejilla de valores, desde $\lambda = 10^{10}$ hasta $\lambda = 10^{-2}$, cubriendo de esta forma una gran gama de escenarios, desde el modelo nulo (solo la ordenada en el origen), hasta el caso de mínimos cuadrados. Se verá más adelante que se puede llevar a cabo el ajuste del modelo para un valor determinado de λ que no esté entre los de la rejilla inicial.

```
library("glmnet")
grid <- 10^seq(10, -2, length = 100)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

Por defecto, la función `glmnet()` estandariza las variables predictoras para que estén en la misma escala. Si por alguna razón no se quisiera hacer, se usaría `standardize = FALSE`.

Asociado con cada valor de λ hay un vector de coeficientes estimados mediante regresión ridge almacenados en un matriz accesible utilizando `coef()`. En este caso, el tamaño de la matriz es 20×100 , las 20 filas corresponden a cada uno de los predictores más la ordenada en el origen, y las 100 columnas a cada valor de λ . Lo esperable es que los coeficientes estimados sean más pequeños cuanto mayor sea el valor de λ . A continuación se muestra el valor de los coeficientes cuando $\lambda = 11,498$, así como la suma de sus cuadrados, $\sum_{j=1}^p \beta_j^2$:

```
ridge_mod$lambda[50]
#> [1] 11497.57
sum(coef(ridge_mod)[-1, 50]^2)
#> [1] 40.45739
```

Por el contrario, si λ es más pequeño, 705, el valor es mucho mayor.

```
ridge_mod$lambda[60]
#> [1] 705.4802
sum(coef(ridge_mod)[-1, 60]^2)
#> [1] 3261.554
```

A continuación se dibuja el efecto de λ en los coeficientes (véase Fig. 1.3):

```
plot(ridge_mod, xvar = "lambda", label = TRUE)
```

El lado izquierdo de la Fig 1.3 corresponde a un valor de λ muy pequeño, y por lo tanto no existen restricciones sobre los coeficientes. Conforme aumenta el valor de λ los coeficientes se

1.3. Métodos shrinkage

29

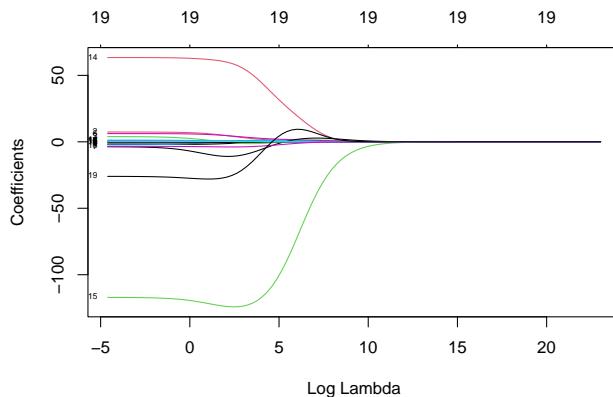


Figura 1.3: Coeficientes estimados para distintos valores del parámetro de regularización (en la escala logarítmica).

aproximan a cero, ya que el precio a pagar por ser distinto de cero es cada vez mayor. Pero no todos se aproximan a cero de la misma manera: hay un conjunto de variables cuyo coeficiente es prácticamente cero para cualquier valor de λ , mientras que para un valor de $\log(\lambda) = 3$ parece que hay solo 4 coeficientes distintos de 0.

La función `predict()` se puede utilizar con diferentes propósitos. Por ejemplo, se pueden obtener los coeficientes de la regresión ridge para un valor específico de λ , por ejemplo $\lambda = 50$:

```
predict(ridge_mod, s = 50, type = "coefficients")[1:20, ]
```

Ahora se van a dividir los datos en una muestra de entrenamiento y otra de test para estimar el error de predicción de la regresión *ridge*.

```
set.seed(1)
entreno <- sample(1:nrow(x), nrow(x) / 2)
test <- (-entreno)
y_test <- y[test]
```

Se ajusta la regresión ridge a la muestra de entrenamiento usando un valor de λ (por ejemplo $\lambda = 4$), y se evalúa su *MSE* en la muestra de test. Para ello se usará la función `predict()`. En este caso, para obtener las predicciones para la muestra de test, se reemplaza `type = "coefficients"` por el argumento `newx`.

```
ridge_mod <- glmnet(x[entreno, ], y[entreno], alpha = 0, lambda = grid)
ridge_pred <- predict(ridge_mod, s = 4, newx = x[test, ])
mean((ridge_pred - y_test)^2)
#> [1] 142226.5
```

El *MSE* es 142,199. Si se usa un valor muy alto de λ , por ejemplo 10^{10} (esto sería equivalente a ajustar un modelo solo con la ordenada en el origen), el resultado es muy distinto:

```
ridge_pred <- predict(ridge_mod, s = 1e10, newx = x[test, ])
mean((ridge_pred - y_test)^2)
#> [1] 224669.8
```

Por lo tanto, en este caso, ajustar un modelo de regresión ridge con $\lambda = 4$ da un *MSE* mucho menor que el obtenido cuando el modelo sólo contiene la ordenada en el origen.

A continuación se compara el resultado para $\lambda = 4$ con el obtenido utilizando mínimos cuadrados ($\lambda = 0$).¹

```
ridge_pred <- predict(ridge_mod, s = 0, newx = x[test, ], exact = T,
                      x = x[entreno, ], y = y[entreno])
mean((ridge_pred - y_test)^2)
#> [1] 167018.2
```

Se observa que el error es menor cuando se usa regresión ridge (con $\lambda = 4$) que cuando se usan mínimos cuadrados.

Hasta ahora se ha elegido el valor $\lambda = 4$ de forma arbitraria, en la siguiente Sección se ve cómo seleccionar el valor de dicho parámetro de una forma automática.

1.3.2. Selección del parámetro de regularización

Se ha visto que el valor de λ tiene un gran impacto en los resultados obtenidos cuando se utiliza un modelo con penalización.

Una buena manera de elegir λ es usar validación cruzada (*cross-validation*), por ejemplo, se puede usar *k-fold cross-validation*:

- Se dividen los datos en k grupos, se ajusta el modelo ridge a $k - 1$ de esos grupos (para una rejilla de valores de λ) y se calcula el error de predicción para el otro grupo.
- La acción anterior se repite tomando como muestra de test cada uno de los k grupos y se suman los errores de predicción.
- Al final se dispondrá de una curva con los errores para cada valor de λ y se elegirá el que dé el mínimo error.

En la práctica, el procedimiento anterior se puede hacer con la función `cv.glmnet()`. Por defecto, esta función usa un *10-fold cross-validation*, pero se puede cambiar usando el argumento `nfolds`.

En el ejemplo del béisbol:

¹Además se ha de añadir ‘exact = T’ en la función ‘predict()’

1.3. Métodos shrinkage

31

```
set.seed(1)
cv_out <- cv.glmnet(x[entreno, ], y[entreno], alpha = 0)
plot(cv_out)
```

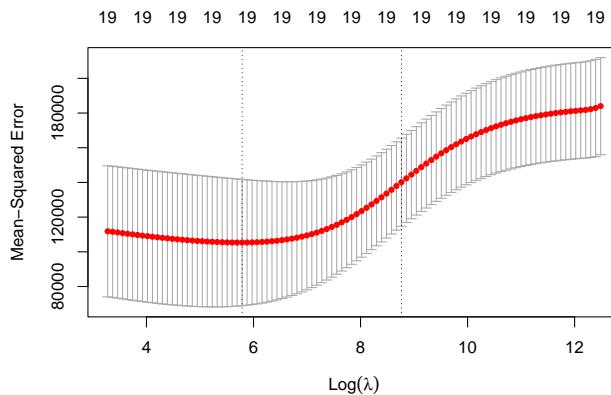


Figura 1.4: Valor del error cuadrático medio y su intervalo de cofianza (calculado sobre los 10-fold) para distintos valores del parámetro de regularización.

```
mejorlam <- cv_out$lambda.min
mejorlam
#> [1] 326.0828
```

En la Fig. 1.4, los puntos rojos corresponden a la media del MSE para los k -folds y las barras superior e inferior corresponden a esa cantidad más/menos una desviación estándar (el ancho será tanto menor cuanto mayor sea k en el k -fold). La primera línea vertical corresponde al valor de λ que hace mínimo el MSE y la segunda es el valor que está a una distancia de una desviación típica del λ mínimo (usar esta valor podría ser una buena opción para evitar el sobre-ajuste, es decir dejar demasiadas variables en el modelo).

Se calcula el valor mínimo del MSE :

```
ridge_pred <- predict(ridge_mod, s = mejorlam, newx = x[test, ])
mean((ridge_pred - y_test)^2)
#> [1] 139833.6
```

Como se puede apreciar, hay una mejora sobre el error de predicción que se había obtenido cuando $\lambda = 4$.

1.3.3. Regresión *lasso*

Uno de los puntos débiles de la regresión ridge es que no hace selección de variables (los coeficientes pueden ser próximos a cero pero no exactamente cero). En el modelo final se incluyen todos los coeficientes y, por lo tanto, la regresión *ridge* sólo es útil cuando la mayoría de las variables predictoras son tienen un impacto significativo en la respuesta.

La regresión *lasso*, introducida por Tibshirani (1996), es una alternativa a la regresión *ridge* cuyo objetivo es precisamente eliminar esa desventaja de esta técnica, y es útil cuando la mayoría de las variables predictoras no son relevantes en el modelo. Los coeficientes *lasso*, $\hat{\beta}^L$, minimizan la siguiente cantidad:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \sum_{j=1}^p |\beta_j|$$

Ahora los coeficientes se *contraen* hacia cero utilizando el valor absoluto en vez de la suma de cuadrados. A esta norma se le llama l_1 , $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$. El cambio que supone es sutil pero importante. En ambos casos los coeficientes se contraen hacia 0, pero en el caso de la regresión *lasso* cuando λ es suficientemente grande los coeficientes serán 0, de modo que se estará haciendo una selección de variables. Es decir, hará los coeficientes exactamente igual a 0 si esas variables no son importantes y λ es suficientemente grande. En este sentido *lasso* es lo que se llama un **modelo sparse** (un modelo con un número *sparse, o escaso, de parámetros).

¿Por qué *lasso* hace que los coeficiente se contraigan exactamente hacia cero? Para entenderlo se va a ver una formulación equivalente a la de los mínimos cuadrados penalizados en el caso de la regresión *lasso*:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \quad \text{sujeto a} \quad \sum_{j=1}^p |\beta_j| < s$$

Se está utilizando mínimos cuadrados con una restricción, o lo que es lo mismo, con un *presupuesto* en la norma l_1 sobre los coeficientes. Las dos formulaciones son equivalentes en el sentido de que si se tiene un *presupuesto* s , habrá un λ en la formulación previa que corresponda al mismo problema y viceversa. Supóngase que se hacen mínimos cuadrados y se obtienen unas ciertas estimaciones de los parámetros (coeficientes) tal que la suma de sus valores absolutos es 10, pero alguien dice que nuestro *presupuesto* es 5 (la suma de los valores absolutos de los coeficientes no puede ser mayor que esa cantidad). Ahora, hay que resolver el problema de mínimos cuadrados pero coeficientes no pueden tomar cualquier valor, ya que se tiene una restricción sobre los mismos. Cuanto más pequeño sea el *presupuesto*, más próximos a cero serán los coeficientes. Si el *presupuesto* es 0, todos los coeficientes serán también 0. Si el presupuesto es muy alto, hay libertad para que los coeficientes tomen el valor que quieran, y se estaría en el caso de mínimos cuadrados. El *presupuesto* impone que haya un equilibrio entre el ajuste a los datos y el tamaño de los coeficientes.

La Fig. 1.5 (tomada de James et al. (2013)) muestra por qué lasso es *sparse*:

El gráfico corresponde a un modelo de regresión con dos variables predictoras. El punto donde está el vector de coeficientes, $\hat{\beta}$, es donde se alcanzaría el valor mínimo de los mínimos cuadrados

1.3. Métodos shrinkage

33

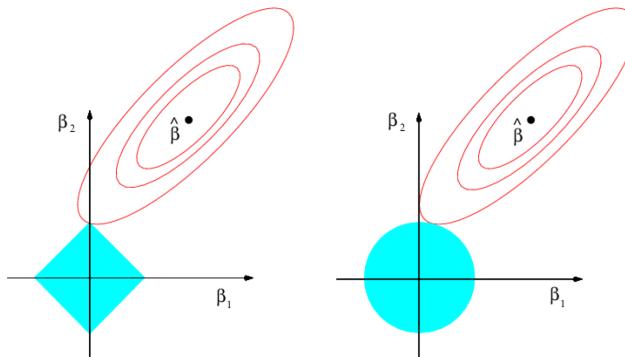


Figura 1.5: Contornos (rojo) de RSS y regiones de restricción (en azul) para lasso (izquierda) y ridge (derecha).

(RSS), los contornos serían combinaciones de valores de β_1 y β_2 que dan lugar al mismo valor de RSS pero que ya no sería el mínimo. Las regiones de restricción son $|\beta_1| + |\beta_2| < s$ (*lasso*) y $\beta_1^2 + \beta_2^2 < s$ (*ridge*). En el caso de *ridge*, el *presupuesto* sería el radio del círculo, y la regresión *ridge* busca el primer lugar en el que el contorno toca a la región de restricción, pero al ser un círculo, difícilmente uno u otro coeficiente va a ser 0. En el caso de *lasso* la región es un diamante, y por lo tanto tiene vértices, en la Fig. 1.5 el contorno toca a la región de restricción en el caso en que $\beta_1 = 0$.

Se vuelve al ejemplo del béisbol para mostrar la regresión lasso; en este caso el argumento α toma valor 1:

```
lasso_mod <- glmnet(x[entreno, ], y[entreno], alpha = 1, lambda = grid)
plot(lasso_mod)
```

En la Fig. 1.6 Se puede ver, que dependiendo del valor del parámetro de regularización, algunos de los coeficientes se hacen exactamente 0. Para elegir el valor de dicho parámetro y calcular el error de predicción resultante se procede como sigue:

```
set.seed(1)
cv_out <- cv.glmnet(x[entreno, ], y[entreno], alpha = 1)
plot(cv_out)

mejorlab <- cv_out$lambda.min
lasso.pred <- predict(lasso_mod, s = mejorlab, newx = x[test, ])
mean((lasso.pred - y_test)^2)
#> [1] 143673.6
```

Este valor es bastante más bajo que MSE en la muestra de test en el caso de mínimos cuadrados (224669,8), y bastante parecido al obtenido con regresión *ridge* (cuando el parámetro de regularización se elige mediante validación cruzada, 139856,6).

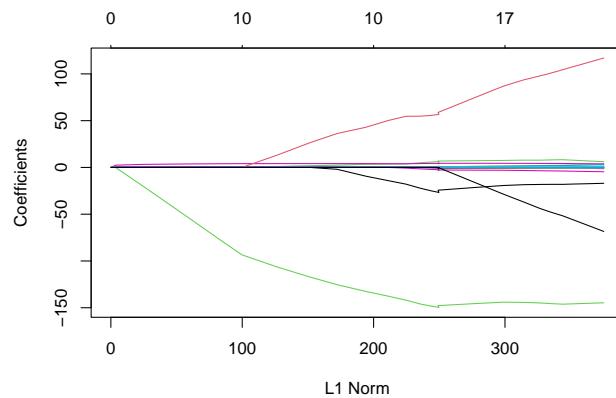


Figura 1.6: Valor de los parámetros estimados para distintos valores de la penalización (que depende del parámetros de regularización.)

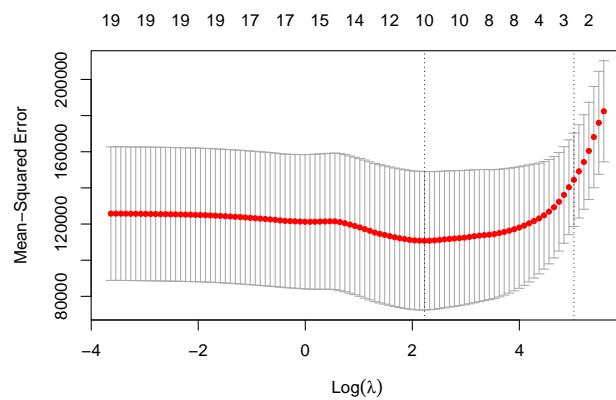


Figura 1.7: Valor del error cuadrático medio y su intervalo de confianza para distintos valores del parámetro de regularización.

1.3. Métodos shrinkage

35

Sin embargo, *lasso* tiene una ventaja importante con respecto a la regresión *ridge* ya que los coeficientes estimados son *sparse*. En los resultados que se muestran a continuación, se puede observar que 10 de los 20 coeficientes estimados son 0. Por lo tanto, el modelo *lasso* con λ elegido mediante validación cruzada contiene solo nueve variables predictoras.

```
out <- glmnet(x, y, alpha = 1)
lasso_coef <- predict(out, type = "coefficients", s = mejorlab)[1:20, ]
lasso_coef[lasso_coef != 0]
#>   (Intercept)      Hits       Walks      CHmRun      CRuns
#> -3.04787656  2.02551572  2.26853781  0.01647106  0.21177390
#>     CRBI      LeagueN    DivisionW     PutOuts      Errors
#>  0.41944632 20.48456551 -116.59062083  0.23718459 -0.94739923
```

1.3.4. *Elastic net*

Uno de los problemas de la regresión *lasso* es cuando hay variables predictoras correladas entre sí, pues elegirá una de ellas (y los coeficientes de las demás los hará cero) sin un criterio objetivo. Además, supóngase que se está en una situación en la que el número de variables p es mayor que el número de observaciones N , en este caso *lasso* elegiría como mucho N variables; mientras que la regresión *ridge* las utilizaría todas, aunque disminuye la complejidad del modelo (esto en algunos casos puede ser lo deseable o no). *Elastic net* (Zou and Hastie, 2005) es una generalización de los métodos anteriores que combina la penalización *ridge* y la *lasso*:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j|.$$

También aparece en muchas ocasiones de esta otra forma:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left[\frac{1}{2}(1-\alpha) \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right]$$

donde $\alpha \in [0, 1]$. Se puede ver α como el parámetro que controla la mezcla entre las dos penalizaciones y λ como el que controla la cantidad de penalización. Si $\alpha = 0$ se está en el caso de regresión *ridge*, y si $\alpha = 1$ en el caso de regresión *lasso*.

La función `glmnet()` también sirve para ajustar *elastic net*, pero el parámetro α hay que elegirlo a priori. Otra opción es utilizar el paquete `caret` para hacer validación cruzada sobre α y λ simultáneamente:

```
set.seed(1)
library("caret")
cv_glmnet <- train(
  x = x[entreno, ],
  y = y[entreno],
  method = "glmnet",
```

```

trControl = trainControl(method = "cv", number = 10),
tuneLength = 10
)
# modelo con el MSE más pequeño
cv_glmnet$bestTune
#>   alpha    lambda
#> 9    0.1 99.12337
ggplot(cv_glmnet)

```

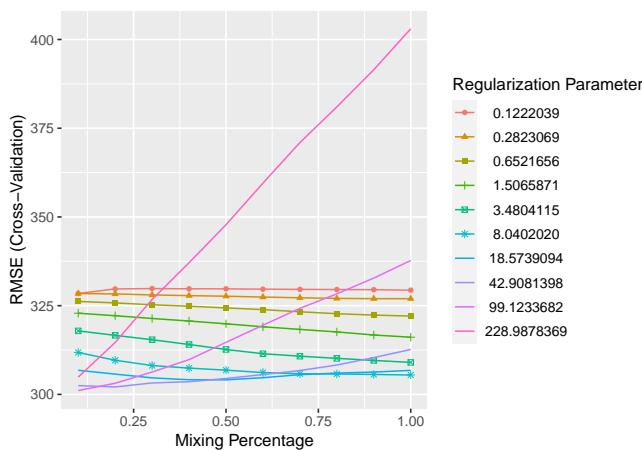


Figura 1.8: Valor de la raíz cuadrada del error cuadrático medio para distintas combinaciones de α y λ .

La Fig. 1.8 muestra como la combinación de α y λ da lugar a diferentes MSE (aquí aparece el $RMSE$, o sea, su raíz cuadrada). Cada línea corresponde a un valor de λ distinto, y en el eje x se representan los valores de α .

Se calcula el error de predicción para estos dos valores de los parámetros de regularización:

```

elastic_mod <- glmnet(x[entreno, ], y[entreno], alpha = cv_glmnet$bestTune$alpha)
elastic_pred <- predict(elastic_mod, newx = x[test, ], s = cv_glmnet$bestTune$lambda)
mean((elastic_pred - y_test)^2)
#> [1] 141626.1

```

Se puede comprobar que es peor que el de la regresión *ridge*, pero mejor que el de *lasso*. Si no se quiere hacer ningún tipo de selección, en este caso se elegiría *ridge*, pero si se quiere reducir al máximo el número de variables predictoras se usaría *lasso* (a costa de que el error de predicción aumente) y el equilibrio vendría con el uso de elastic-net que hace selección de variables pero no aumenta el error de predicción.

Existen otros métodos de regularización que se derivan de estos, como el *group lasso*, *sparse group-lasso*, etc. Se puede encontrar información de estos métodos en (Hastie and Tibshirani, 2015).

Resumen

En este capítulo se introducen una serie de técnicas para mejorar la predicción y la interpretabilidad de los modelos de regresión, en particular:

- Se muestra el uso de la técnica de selección del mejor subconjunto de variables en el modelo, así como los métodos *stepwise*.
- Se presentan 3 métodos tipo *shrinkage*: regresión *ridge*, *lasso* y *elastic net*, bien para la selección de variables, o para solventar problemas de multicolinealidad en el modelo.
- Se muestra cómo seleccionar los parámetros de regularización que controlan la regresión penalizada.
- Se ilustra el uso de todas las metodologías propuestas en el capítulo mediante el análisis de un caso práctico.

Capítulo 2

Modelización de series temporales

M^a Carmen García Centeno

Universidad San Pablo-CEU, CEU Universities

2.1. Conceptos básicos

El análisis de series temporales es muy útil para realizar predicciones, analizar tendencias y el comportamiento de los datos. Con dicho análisis, se tratará de obtener modelos que expliquen su dinámica y puedan utilizarse para predecir valores futuros y tomar decisiones.

Una serie temporal se puede definir como el conjunto de valores observados, en períodos consecutivos de tiempo de la misma amplitud, correspondientes a distintas variables aleatorias. Por ejemplo: las precipitaciones diarias, la inflación mensual o el PIB trimestral. Ya que una serie temporal es la realización de un proceso estocástico, algunos conceptos clave sobre los que se fundamenta el análisis de series temporales son los siguientes:

1. **Proceso estocástico.** Es una sucesión de variables aleatorias $\{Y_t\}$, donde $t = -\infty, \dots, -2, -1, 0, 1, 2, \dots, \infty$, que dependen de un parámetro. En el caso de las series temporales ese parámetro es el tiempo. Así, en cada momento del tiempo, es una única variable aleatoria con una determinada distribución de probabilidad. Desde el punto de vista práctico es muy complicado, y a veces imposible, caracterizar un proceso estocástico según su distribución de probabilidad conjunta. Por esta razón, se recurrirá a una forma menos completa, pero más sencilla y práctica, basada en los dos primeros momentos de una distribución, en concreto: la media ($E(Y_t) = \mu_t$), la varianza ($Var(Y_t) = E(Y_t - \mu_t)^2 = \sigma_t^2$) y las covarianzas ($\gamma(k) = Cov(Y_t, Y_{t-k}) = E(Y_t - \mu_t)(Y_{t-k} - \mu_{t-k})$).

2. Estacionariedad. Un proceso estocástico es estacionario, en sentido simple o amplio, si su media y la varianza se mantienen invariantes a lo largo del tiempo y las covarianzas entre dos variables solo dependen del lapso de tiempo que transcurre entre ellas. Es decir:

- $E(Y_t) = \mu, \quad \forall t,$
- $Var(Y_t) = E(Y_t - \mu)^2 = \sigma^2, \quad \forall t$
- $Cov(Y_t, Y_{t-k}) = E(Y_t - \mu)(Y_{t-k} - \mu) = \gamma(k), \quad \forall t \neq s.$

3. Función de autocovarianzas. Está formada por el conjunto de autocovarianzas calculadas para distintos órdenes (k). Tiene como finalidad determinar las relaciones de dependencia lineales que existen entre las variables del proceso con el fin de identificar el modelo que mejor explica su dinámica a lo largo del tiempo.

4. Función de autocorrelación simple (ACF). Está formada por el conjunto de coeficientes de correlación lineales calculados para distintos órdenes (k). De forma genérica, para medir la correlación lineal existente entre Y_t e Y_{t-k} , el coeficiente de correlación de orden k , $\rho(k)$, se calcula como el cociente entre la covarianza de orden k y la varianza (ya que, al ser estacionario, las desviaciones típicas de Y_t e Y_{t-k} son iguales).

$$\rho(k) = \frac{Cov(Y_t, Y_{t-k})}{(\sqrt{Var(Y_t)})(\sqrt{Var(Y_{t-k})})} = \frac{\gamma(k)}{\gamma(0)} \quad (2.1)$$

La representación gráfica de los coeficientes de correlación $\rho(k)$ para $k=0,1,2,\dots$, se conoce como **correlograma**.

5. Función de autocorrelación parcial (PACF). Está formada por el conjunto de las correlaciones parciales obtenidas para los distintos valores de k . Así, para dos instantes de una serie temporal t e $(t-k)$, mide la correlación lineal existente entre las variables Y_t e Y_{t-k} asociadas a ellos, ajustada de los valores que toma el proceso temporal en los períodos intermedios ($Y_{t-1}, Y_{t-2}, \dots, Y_{t-(k-1)}$).

6. Ergodicidad. Un proceso estocástico es ergódico cuando, a partir de un determinado desfase temporal entre las variables, la correlación lineal existente entre ellas tiende a desaparecer. Esto implica que las covarianzas y el coeficiente de correlación tienden a cero, es decir,

$$\lim_{k \rightarrow \infty} \gamma(k) = 0 \quad y \quad \lim_{k \rightarrow \infty} \rho(k) = 0$$

7. Ruido blanco. Es un proceso puramente aleatorio que se puede expresar de la siguiente forma:

$$Y_t = a_t$$

Se caracteriza por tener esperanza nula, varianza constante y covarianzas nulas. Es decir,

$$E(a_t) = 0 \quad \forall t; \quad E(a_t^2) = \sigma^2 \quad \forall t; \quad E(a_t a_s) = 0 \quad \forall t \neq s$$

Esto implica que un ruido blanco siempre va a ser estacionario.

2.2. Modelos ARIMA

Para determinar las características de un proceso estocástico, subyacente a la serie temporal, se van a utilizar los modelos ARIMA (acrónimo del inglés AutoRegressive Integrated Moving Average). Estos modelos están formados por tres componentes: **AR** (Autorregresivo), **I** (Integrado, es decir, número de diferencias necesarias para convertirlo en estacionario cuando no lo es) y **MA** (Medias móviles). Fueron propuestos por Box y Jenkins y son un caso particular de procesos estocásticos lineales, estacionarios, ergódicos y discretos. Entre estos tipos de procesos lineales, los más frecuentes son:

- Los procesos puramente aleatorios (por ejemplo, un proceso ruido blanco).
- Los procesos puros de medias móviles regulares (MA) o medias estacionales (sMA).
- Los procesos puros autorregresivos (AR) o autorregresivos estacionales (sAR).
- Una combinación de los procesos anteriores que dará lugar a los procesos mixtos regulares (ARMA) o mixtos estacionales (sARMA).

Los modelos ARIMA tratan de captar la dinámica y la dependencia que existe entre los datos de una serie temporal, [Hamilton \(1994\)](#), [Uriel Jiménez and Peiro Giménez \(2000\)](#), [Cryer and Chan \(2010\)](#), [Pemberton \(2011\)](#), [Mínguez Salido and García Centeno \(2011\)](#), [Brockwell and Davis \(2016\)](#), [Shumway and Stoffer \(2017\)](#). Por lo tanto, son muy útiles para describir un valor como una combinación o función lineal de valores pasados y errores debidos al azar.

En la modelización ARIMA se pueden destacar varias fases. La primera se centra en la **identificación** del modelo ARIMA más adecuado que haya podido generar los datos de la serie temporal. En esta fase, es necesario decidir los órdenes del proceso (es decir, el desfase temporal entre el mayor y menor periodo de tiempo de las variables incluidas en el modelo) tanto de la parte regular como de la estacional.

La ACF y la PACF desempeñan un papel clave en la identificación de estos órdenes. Antes de calcular la ACF y la PACF, es necesario comprobar si la serie es estacionaria, es decir, no puede deambular ni tener tendencia creciente o decreciente. En el caso de que no lo sea, se realizarán las transformaciones necesarias para convertirla en estacionaria, ya que la modelización ARIMA exige que los datos sean estacionarios. Las dos razones fundamentales por las cuales pueden no ser estacionarios son: la no constancia en el tiempo de la media o la existencia de fluctuaciones de diferente amplitud que hacen que la varianza no se mantenga constante.

Si la serie no es estacionaria en varianza, de las transformaciones posibles de Box-Cox se utilizará la transformación logarítmica. Esta transformación suele hacer más simétricas las distribuciones y, como consecuencia, más próximas a la normal. Además, esta transformación es útil, ya que la diferencia del logaritmo de la serie representa su tasa de variación porcentual.

Si no es estacionaria en media, puede ser debido a la existencia de tendencia en el tiempo o de estacionalidad. En el caso de tendencia se calcularán diferencias regulares (una o dos como máximo según el tipo de tendencia lineal o no lineal, es decir, $Y_t - Y_{t-1} = \Delta Y_t$ o $\Delta^2 Y_t = \Delta(\Delta Y_t)$, respectivamente). En el caso de estacionalidad, se calculará una diferencia estacional (es decir, $Y_t - Y_{t-s} = \Delta_s Y_t$) como máximo.

Después de proponer el modelo en la segunda fase se procede a su **estimación**. En esta fase se obtienen los valores de los parámetros correspondientes al modelo propuesto, así como sus desviaciones típicas y los residuos del modelo. Posteriormente, en la fase de **validación** o **diagnóstico** se realizarán los contrastes necesarios para determinar si el modelo propuesto es adecuado o no, es decir si los parámetros estimados son estadísticamente significativos o no; si el modelo estimado es estacionario e invertible; y, si sus residuos siguen un proceso ruido blanco o no. En el caso de que no lo sean, será necesario corregir el modelo con la información proporcionada por los residuos hasta obtener un modelo cuyos residuos sean ruido blanco. Finalmente, se procederá a la utilización del modelo para **predecir**.

2.3. Análisis de series temporales con R

Algunas de las librerías que normalmente se suelen utilizar en la modelización de series temporales con **R** son:

```
library("tseries")
library("astsa")
library("forecast")
library("lubridate")
library("foreign")
library("quantmod")
library("ggplot2")
```

Brevemente comentar que la librería “tseries” permite manipular datos de series temporales; “astsa” es adecuada para analizar series de tiempo en los dominios de frecuencia y tiempo; “forecast” es necesaria para mostrar y analizar predicciones con series temporales; “tidyverse” es muy útil para la modelización y la visualización de datos; “lubridate” facilita el trabajo con fechas y horas; “foreign” es esencial para crear objetos en **R** importando datos de casi cualquier formato conocido, como por ejemplo SAS, SPSS, Stata, etc.; “quantmod” ayuda de forma quantitativa en el desarrollo de estrategias y modelización mediante la utilización de estadísticas; “readxl” facilita la obtención de datos de Excel a **R**; “ggplot2” es muy útil para la realización de gráficos.

En el caso real que se propone, se utilizarán datos mensuales del INE (www.ine.es) correspondientes al índice general de precios al consumo (IPC) en el periodo muestral comprendido entre enero de 2002 y marzo de 2022. La fecha en la que se observan los datos está incluida en la primera columna de este fichero. Los datos se leerán desde el paquete **CDR** del libro:

```
ipc <- CDR:::ipc
```

Si los datos se cargan sin fecha, para trabajar con series temporales es necesario ponerles la fecha. Por ejemplo, para series mensuales, el código sería:

2.3. Análisis de series temporales con **R**

43

```
ipc_ts <- ts(ipc$ipc, start = c(2002, 1), end = c(2022, 3), frequency = 12)
```

La representación gráfica de la serie original puede ayudar a saber si la serie ha sido generada por un proceso estacionario o no. Para obtener este gráfico, se podrá ejecutar el siguiente código:

```
ipc$Time <- as.Date(ipc$Time)
ggplot(
  data = ipc,
  aes(x = Time, y = ipc)
) +
  theme(
    axis.title.x = element_text(size = 15),
    axis.title.y = element_text(size = 15)
) +
  geom_line(colour = "blue")
```

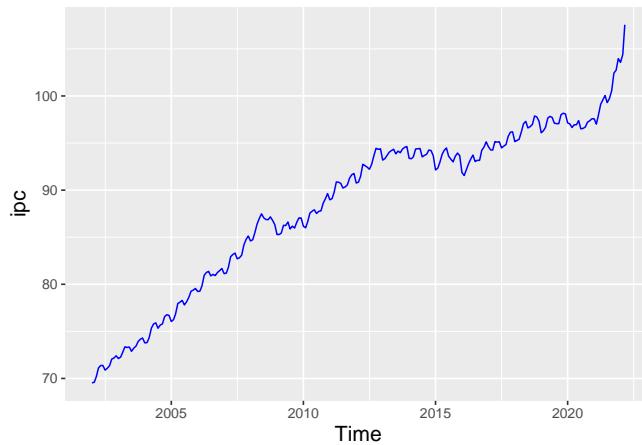


Figura 2.1: Evolución del IPC entre enero 2002 y marzo 2022.

La descomposición aditiva o multiplicativa en tendencia, componente estacional, componente cíclico e irregular de la serie del IPC y su representación gráfica, también puede ayudar a determinar si la serie es estacionaria o no. El código para la descomposición aditiva de la serie es el siguiente:

Tanto en el gráfico de la serie original del IPC (Fig. 2.1) como en el de su descomposición (Fig. 2.2) se aprecia que la serie tiene tendencia y componente estacional, por lo tanto, no es estacionaria en media, es decir, la media es distinta para diferentes meses y años. Tampoco lo es en varianza, ya que las dispersiones respecto de la media cambian a lo largo del tiempo.

Si una serie no es estacionaria ni en media ni en varianza es necesario empezar corrigiendo la no estacionariedad en varianza y, posteriormente, la no estacionariedad en media.

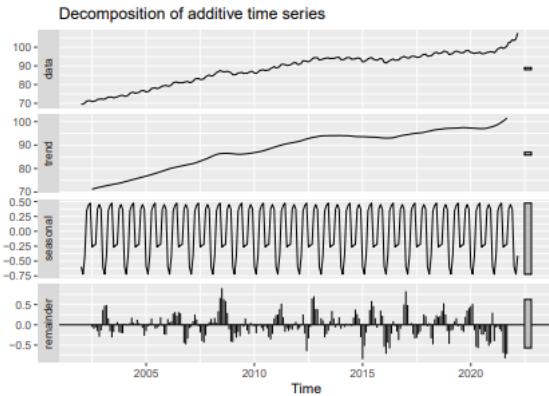


Figura 2.2: Descomposición aditiva del IPC

Las transformaciones de Box-Cox son las más utilizadas para corregir el problema de no estacionariedad en varianza. De todas estas transformaciones, la más habitual es el logaritmo. Así, para obtener la representación gráfica del logaritmo de Y_t , se puede ejecutar el siguiente código:

```
logipc <- log10(ipc_ts)
ts_logipc <- data.frame(value = logipc, Time = time(logipc))
ggplot(
  data = ts_logipc,
  aes(x = Time, y = logipc)
) +
  theme(
    axis.title.x = element_text(size = 15),
    axis.title.y = element_text(size = 15)
  ) +
  geom_line(colour = "red")
```

Corregido el problema de la no estacionariedad en varianza, en la Fig. 2.3 se aprecia que sigue sin ser estacionaria, ya que al tener estacionalidad y una tendencia creciente no es estacionaria en media. Para conseguir que la serie sea estacionaria es necesario corregir tanto su tendencia como su estacionalidad.

Es indiferente cual de los dos problemas se resuleva primero, ya que el resultado final de la transformación es el mismo. En este caso, se empezará corrigiendo la tendencia. Para ello, es necesario calcular una diferencia regular del logaritmo del IPC ($d\log ipc_t = \log(ipc_t) - \log(ipc_{t-1})$). Al ser datos mensuales, esta transformación representa la tasa de variación relativa mensual del IPC. El código para calcular esta diferencia regular y su represenación gráfica (Fig. 2.4) es:

2.3. Análisis de series temporales con **R**

45

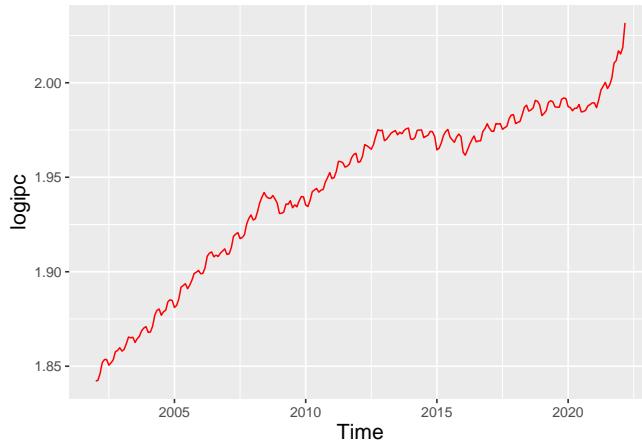


Figura 2.3: Logaritmo del IPC.

```
dlogipc <- diff(logipc, differences = 1)
ts_dlogipc <- data.frame(value = dlogipc, Time = time(dlogipc))
ggplot(
  data = ts_dlogipc,
  aes(x = Time, y = dlogipc)
) +
  theme(
    axis.title.x = element_text(size = 15),
    axis.title.y = element_text(size = 15)
  ) +
  geom_line(colour = "blue")
```

Corregida la serie de tendencia, quedaría por corregir su estacionalidad. Para ello, sobre la diferencia regular del logaritmo del IPC, se calculará una diferencia estacional ($d12dlogipc_t$). El código para calcular esta diferencia estacional y su representación gráfica (Fig. 2.5) es:

```
d12dlogipc <- diff(dlogipc, 12)
ts_d12dlogipc <- data.frame(value = d12dlogipc, Time = time(d12dlogipc))

ggplot(
  data = ts_d12dlogipc,
  aes(x = Time, y = d12dlogipc)
) +
  theme(
    axis.title.x = element_text(size = 15),
    axis.title.y = element_text(size = 15)
  ) +
  geom_line(colour = "purple")
```

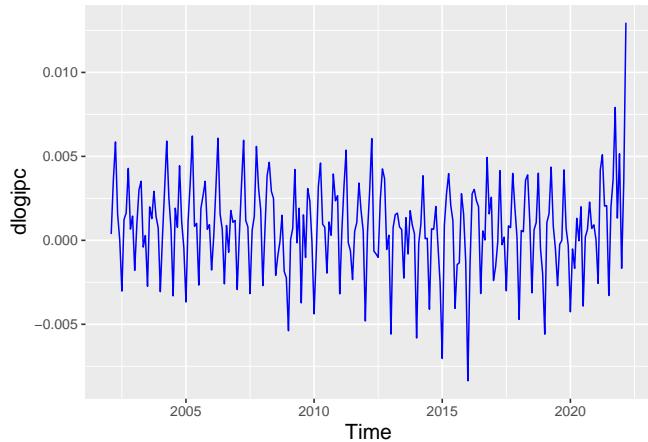


Figura 2.4: Diferencia regular del logaritmo del IPC.

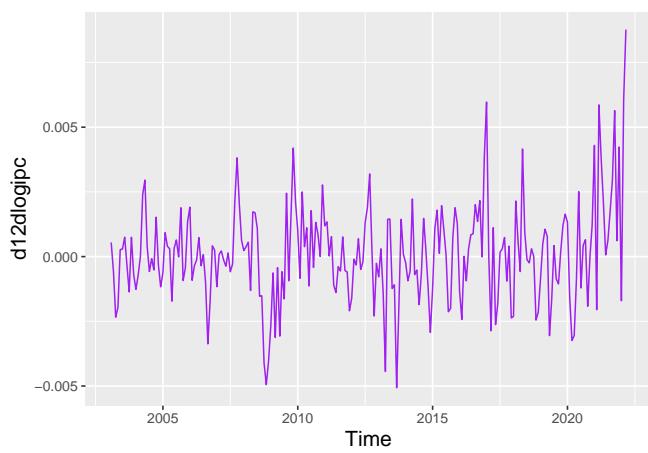


Figura 2.5: Diferencia estacional de la diferencia regular del logaritmo del IPC.

2.3. Análisis de series temporales con **R**

47

También, para comprobar si la serie es estacionaria en media o no, se puede utilizar un test de raíces unitarias¹. En este caso se utilizará el test de raíces unitarias de Dickey-Fuller. Éste es un contraste unilateral en el que la hipótesis nula implica la existencia de raíces unitarias y, por lo tanto, que la serie no es estacionaria (es decir, es I(1)), mientras que la hipótesis alternativa implica que sí es estacionaria (es decir, I(0)). Si se realiza el test sobre la serie original (ipc_ts) y sobre la serie transformada del IPC (d12dLogIPC), se puede comprobar que esta última es la transformación estacionaria. El código utilizado para realizar este contraste es:

```
adf.test(ipc_ts)
#>
#> Augmented Dickey-Fuller Test
#>
#> data: ipc_ts
#> Dickey-Fuller = -1.8396, Lag order = 6, p-value = 0.6433
#> alternative hypothesis: stationary
adf.test(d12dlogipc)
#>
#> Augmented Dickey-Fuller Test
#>
#> data: d12dlogipc
#> Dickey-Fuller = -3.3727, Lag order = 6, p-value = 0.06002
#> alternative hypothesis: stationary
```

Teniendo en cuenta el p-valor para los valores originales del IPC, si se preestablece un nivel de significación del 10 %, se acepta la hipótesis nula (y, por lo tanto, la serie no es estacionaria). Sin embargo, para la transformación estacionaria (d12dLogipc) se rechaza la H_0 . Por lo tanto, se puede concluir que dicha transformación la ha convertido en estacionaria.

2.3.1. Identificación o especificación del modelo

A partir de la transformación estacionaria del modelo es necesario calcular la ACF y la PACF muestrales, con el fin de identificar el modelo ARIMA más adecuado, es decir, especificar tanto el modelo como su correspondiente orden. Antes de hacerlo para el caso concreto del IPC, se analizarán brevemente los principales modelos teóricos que pueden explicar la dinámica de una serie temporal. Estos modelos para la parte regular (es decir, la dependencia asociada a observaciones consecutivas de tiempo) pueden ser: autorregresivos puros (AR(p)), medias móviles (MA(q)) o mixtos ARMA(p,q), donde p y q representan sus correspondientes órdenes respectivamente. O bien, para la parte estacional (s), es decir, la dependencia asociada a observaciones que distan entre sí s-periodos de tiempo o múltiplos de s: sAR(P), sMA(Q) o sARMA(P,Q), donde P y Q representan, respectivamente, los órdenes de los modelos correspondientes a la parte estacional. Destacar que para identificar el modelo de la parte estacional es necesario analizar los coeficientes correspondientes a los retardos estacionales. Así, por ejemplo, en el

¹El término “customer churn” se suele traducir como perdida de clientes o rotación de clientes. Se compone de las palabras inglesas “change” (en castellano cambio) y “turn” (en castellano abandonar)

caso de una serie mensual, habría que fijarse en los coeficientes correspondientes a los retardos 12, 24, 36, etc., para una serie trimestral en los retardos, 4, 8, 12, 16, etc.

Antes de identificar el modelo correspondiente al IPC en el periodo muestral analizado, se mostrarán algunos ejemplos de modelos ARIMA, su ecuación general y el comportamiento de la ACF y la PACF.

- En el caso de MA(q) o sMA(Q):

- La ACF se anula para órdenes superiores a q (o a Q en el caso estacional). Es decir, solo los q primeros coeficientes son significativos (o Q en el caso estacional). El resto de los coeficientes son nulos (o estadísticamente nulos en el caso muestral).
- La PACF decrece de forma exponencial o sinusoidal hacia cero.

Ejemplo de la ACF y PACF teóricas, para un proceso estacionario (Y_t), generado por un medias móviles de primer orden o MA(1).

La ecuación que describe la dinámica de un modelo MA(1) o ARMA(0,1) es:

$$Y_t = a_t - \theta a_{t-1}$$

O bien, en forma polinómica:

$$Y_t = (1 - \theta L)a_t$$

donde, L es el operador de retardos y a_t es un ruido blanco, es decir: $E(a_t) = 0 \quad \forall t$; $E(a_t^2) = \sigma^2 \quad \forall t$; $E(a_t a_s) = 0 \quad \forall t \neq s$.

Este modelo siempre es estacionario (ya que, se obtiene como una combinación de procesos ruido blanco) y, para que sea invertible (es decir, que se pueda expresar en función del pasado de la variable), es necesario que las raíces del polinomio estén fuera del círculo unidad o lo que es lo mismo que $|\theta| < 1$.

La ACF teórica es:

$$\rho(k) = \begin{cases} \frac{-\theta}{(1+\theta^2)} & \text{si } k = 1 \\ 0 & \forall k > 1 \end{cases}$$

La PACF teórica viene dada por:

$$\phi_{kk} = \frac{-\theta^k(1-\theta^2)}{1-\theta^{2(k+1)}} \quad \text{para } k \geq 1$$

- En un AR(p) o sAR(P):

- La ACF decrece rápidamente de forma exponencial o sinusoidal hacia cero.

2.3. Análisis de series temporales con **R**

49

- La PACF se anula para órdenes superiores a p (o a P , en el caso estacional). Es decir, solo los p (o P en el caso estacional) primeros coeficientes son significativos. El resto de los coeficientes son nulos (o estadísticamente nulos en el caso muestral).

Ejemplo de la ACF y PACF teóricas, para un proceso estacionario (Y_t), generado por un autorregresivo de primer orden o AR(1).

La ecuación que describe la dinámica de un modelo AR(1) o ARMA(1,0) es:

$$Y_t = \phi Y_{t-1} + a_t$$

O bien, en forma polinómica: $Y_t(1 - \phi L) = a_t$, donde L es el operador de retardos y a_t es un ruido blanco.

Este modelo siempre es invertible (ya que, está expresado en función del pasado de la variable) y, para que sea estacionario, es necesario que las raíces del polinomio de retardos estén fuera del círculo unidad o lo que es lo mismo que $|\phi| < 1$.

La ACF teórica es:

$$\rho(k) = \phi \rho(k-1) = \phi^k$$

La PACF teórica viene dada por:

$$\phi_{kk} = \begin{cases} \rho_1 = \phi & \text{si } k = 1 \\ 0 & \forall k > 1 \end{cases}$$

- En un ARMA(p,q) o sARMA(P,Q):

- La ACF tiene un comportamiento irregular en los q primeros (o en los Q en el caso estacional) coeficientes. A partir del orden q (o Q en el caso estacional), se comporta como la de un AR(p) (o un sAR(P) en el caso estacional).
- La PACF tiene un comportamiento irregular en los p primeros coeficientes (o P en el caso estacional). A partir del orden p (o P en el caso estacional), se comporta como la de un MA(q) (o de un sMA(Q) en el caso estacional).

Ejemplo de la ACF y PACF teóricas, para un proceso estacionario (Y_t), que sigue un ARMA(1,1).

La ecuación que describe la dinámica de un modelo ARMA(1,1) es:

$$Y_t = \phi Y_{t-1} + a_t - \theta a_{t-1}, \text{ o bien, en forma polinómica, } Y_t(1 - \phi L) = (1 - \theta L)a_t,$$

donde L es el operador de retardos y a_t es un ruido blanco.

Para que el modelo sea estacionario, es necesario que las raíces del polinomio de la parte autorregresiva estén fuera del círculo unidad (o que $|\phi| < 1$).

Para que sea invertible es necesario que las raíces del polinomio de las medias móviles estén fuera del círculo unidad (o que $|\theta| < 1$).

Además, es necesario que no existan raíces comunes, es decir, $\phi \neq \theta$.

La ACF teórica es igual a:

$$\rho(k) = \begin{cases} \frac{(1-\phi\theta)(\phi-\theta)}{1+\theta^2-2\phi\theta} & \text{si } k = 1 \\ \phi\rho(k-1) & \forall k > 1 \end{cases}$$

La FACP teórica se obtiene como:

$$\begin{aligned} \phi_{11} &= \rho_1 \\ \phi_{22} &= \frac{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & \rho_2 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{vmatrix}} = \frac{\rho_2 - \rho_1^2}{1 - \rho_1^2} \\ \phi_{33} &= \frac{\begin{vmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_2 \\ \rho_2 & \rho_1 & \rho_3 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_2 \\ \rho_2 & \rho_1 & 1 \end{vmatrix}} = \frac{\rho_1^3 - \rho_1\rho_2(2-\rho_2) + \rho_3(1-\rho_1^2)}{1 - \rho_2^2 - 2\rho_1^2(1-\rho_2)} \\ &\vdots \end{aligned}$$

Nota:

La forma de obtener los diferentes coeficientes de la ACF y PACF de los modelos ARIMA estacionales es la misma que para los modelos ARIMA utilizados para la parte regular.

La diferencia fundamental reside en que para la parte regular se calculan los coeficientes de correlación entre las variables que se encuentran en períodos consecutivos de tiempo, mientras que para la parte estacional se hace entre las variables que están separadas entre si s períodos o múltiplos de s (donde, por ejemplo, $s=12$, si la serie es mensual; $s=4$ si la serie fuese trimestral, etc.). Por esta razón, solo se muestra la forma de calcular la ACF y la PACF correspondiente a la parte regular.

Teniendo en cuenta lo anterior, en el caso concreto del IPC, después de obtener su transformación estacional, la ACF y la PACF muestrales son muy útiles para identificar el modelo ARIMA más adecuado, así como, su correspondiente orden en el periodo muestral estudiado.

2.3. Análisis de series temporales con **R**

51

Los códigos **R** para ejecutar los comandos que permiten calcular y representar la ACF y la PACF muestrales de la transformación estacionaria del IPC (Fig. 2.6 y 2.7 respectivamente), con 40 retardos (lags) para incluir al menos 3 retardos estacionales (12, 24 y 36), son:

```
acf(ts(d12dlogipc, frequency = 1), lag.max = 40, main = "")
```

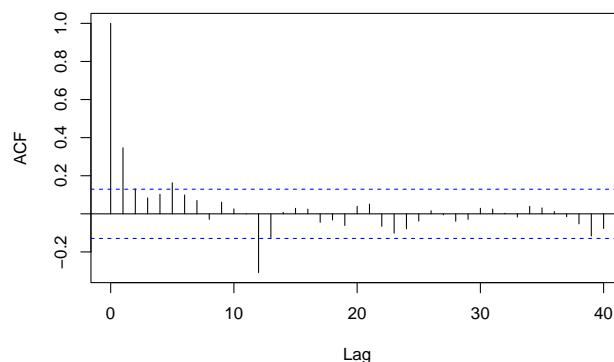


Figura 2.6: ACF. Función de autocorrelación muestral de d12dLogIPC.

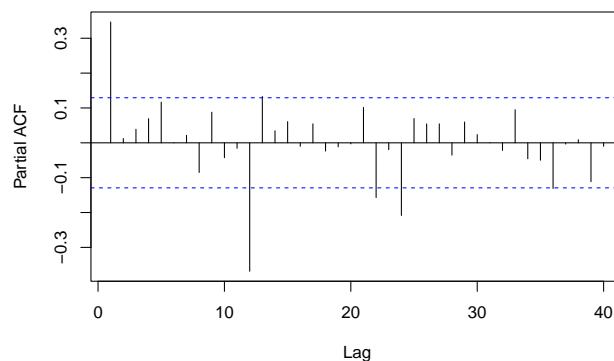


Figura 2.7: PACF. Función de autocorrelación parcial muestral de d12dLogIPC.

Observando el comportamiento de estas funciones, para la parte regular se podría proponer un AR(1) y para la parte estacional un MA(1)12. Por lo tanto, ya que se ha calculado una diferencia regular y otra estacional para convertir a la serie original en estacionaria, el modelo ARIMA para el IPC, en el periodo muestral analizado, sería un ARIMA(1,1,0)(0,1,1)12 o

sARIMA(1,1,0)(0,1,1). Señalar que **R** permite la identificación automática de los órdenes del modelo.

2.3.2. Estimación del modelo

Para estimar los parámetros del modelo (ya que, la función de verosimilitud que se utiliza para estimar los parámetros del modelo es no lineal) se utiliza un procedimiento iterativo de estimación no lineal. El código en **R** que permite obtener las estimaciones de los parámetros del modelo y sus correspondientes desviaciones típicas es:

```
modelo <- arima(ipc_ts, c(1, 1, 0), c(0, 1, 1))
modelo
#>
#> Call:
#> arima(x = ipc_ts, order = c(1, 1, 0), seasonal = c(0, 1, 1))
#>
#> Coefficients:
#>         ar1      sma1
#>     0.4665 -0.8302
#> s.e.  0.0701  0.0860
#>
#> sigma^2 estimated as 0.1082:  log likelihood = -77.76,  aic = 161.51
```

2.3.3. Validación

En la metodología ARIMA desarrollada por Box y Jenkins es necesario determinar si el modelo propuesto es correcto o no, es decir, si se ajusta o no correctamente a los datos de la muestra. Para ello, en la fase de validación, fundamentalmente, es necesario comprobar que:

1. Los parámetros del modelo estimado cumplen con las condiciones de estacionariedad e invertibilidad.

El modelo estimado para el IPC, es invertible y estacionario, ya que la parte regular al ser un autorregresivo de primer orden (AR(1)) siempre va a ser invertible y como su parámetro estimado es en valor absoluto menor que uno, es decir, $|0,4665| < 1$, también es estacionario. La parte estacional sigue un medias móviles de primer orden (sMA(1)), el cual siempre es estacionario y como su parámetro estimado en valor absoluto es menor que uno, es decir, $|-0,8302| < 1$, también es invertible.

Si alguna de las raíces del polinomio de retardos de la parte autorregresiva estuviese próxima a uno, entonces es posible que la serie original este subdiferenciada y, por lo tanto, no sería estacionaria, lo que implicaría la necesidad de calcular alguna diferencia adicional. Si alguna de las raíces del polinomio de retardos de las medias móviles fuese la que estuviese próxima a uno, entonces el modelo podría estar sobrediferenciado y habría que eliminar alguna diferencia.

2.3. Análisis de series temporales con **R**

53

Además, si existiesen raíces comunes en ambos polinomios de retardos de la parte regular sería necesario simplificarlas y el modelo correcto sería un ARIMA(p-1,d,q-1). Se haría lo mismo si las raíces comunes fuesen en la parte estacional.

2. Los parámetros estimados son estadísticamente significativos.

Para comprobar si los parámetros son estadísticamente significativos o no, (o en otros términos, para comprobar si se ha sobreparametrizado o no), se realiza un contraste de significatividad individual para cada uno de ellos. Por ejemplo, para el parámetro del AR(1) la hipótesis nula y alternativa serían:

$$\begin{aligned} H_0 &: \phi = 0 \\ H_1 &: \phi \neq 0 \end{aligned}$$

Para realizar el contraste se utiliza el estadístico t que sigue una distribución t-Student con $(n-k)$ grados de libertad. Este estadístico t se calcula dividiendo el valor estimado del parámetro entre su correspondiente error estándar. En concreto, para los dos parámetros estimados de este modelo (el correspondiente al AR(1) de la parte regular y al sMA de la parte estacional) se tiene:

$$t = \frac{0,4665}{0,0701} = 6,654 \quad y \quad |t| = \left| \frac{-0,8302}{0,0860} \right| = 9,653.$$

Como, para un nivel de significación del 5 % y los 248 grados de libertad, el valor crítico de la t-Student es aproximadamente 1,96, se rechaza la H_0 , lo que implica que los dos parámetros estimados del modelo son estadísticamente distintos de cero.

3. Los residuos del modelo son ruido blanco.

Uno de los supuestos de la modelización ARIMA es que las perturbaciones aleatorias tienen que ser ruido blanco. Como éstas no son observables es necesario calcular los residuos y comprobar si son ruido blanco o no. Existen varias formas de comprobar si los residuos son ruido blanco o no. Entre ellas las más utilizadas son: el gráfico de la serie original de residuos, la ACF y la PACF estimadas y el contraste de Portmanteau (planteado inicialmente por Box-Pierce y actualizado posteriormente por Ljung-Box).

En primer lugar, de forma intuitiva, el gráfico de los residuos puede mostrar si la media es constante e igual a cero y si su varianza también es constante. Además, puede reflejar si existen valores atípicos u outliers (se consideran como tal aquellos que superen tres veces su desviación típica). El siguiente código R, permite obtener los residuos y su representación gráfica (Fig. 2.8).

```
residuos <- residuals(modelo)
ts_residuos <- data.frame(value = residuos, Time = time(residuos))
ggplot(
  data = ts_residuos,
```

```

aes(
  x = Time,
  y = residuos
)
) +
  geom_line(colour = "red")

```

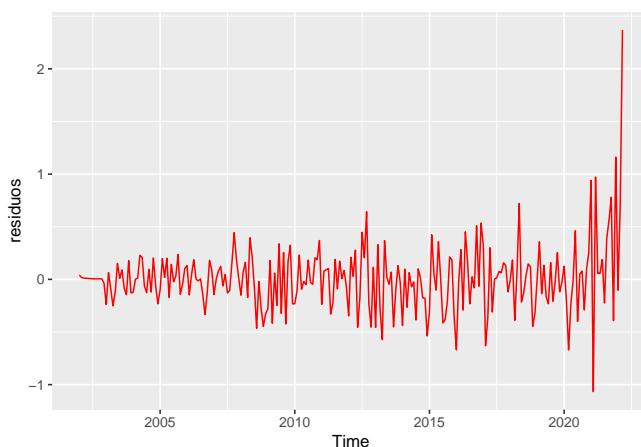


Figura 2.8: Gráfico de los residuos.

En esta Fig. 2.8 se observa que la media es constante e igual a cero y que a partir de febrero de 2022, como consecuencia de las tensiones inflacionistas acaecidas, los residuos son mayores que en el resto del periodo muestral.

En segundo lugar, se puede calcular la ACF y la PACF muestrales de los residuos para comprobar que están incorrelacionados, es decir, que los coeficientes de correlación calculados son estadísticamente nulos. Si éstos fuesen estadísticamente significativos, los residuos no serían ruido blanco. Entonces, para la correcta modelización habría que identificar el proceso e incorporarlo al modelo inicial propuesto para volver a estimarlo. Los correlogramas obtenidos (Fig. 2.9), con el código que se muestra a continuación, indican que no hay coeficientes de correlación estadísticamente significativos (ya que, se encuentran dentro de los intervalos de confianza) y, por lo tanto, los residuos son ruido blanco.

```

par(mfrow = c(2, 1), mar = c(1, 1, 1, 1) + 0.1)
acf(ts(residuos, frequency = 1), lag.max = 40)
pacf(ts(residuos, frequency = 1), lag.max = 40)

```

Finalmente, se puede utilizar el contraste de Portmanteau (Ljung-Box) para comprobar si los residuos están incorrelacionados y se comportan como un ruido blanco o no. En este contraste global de significación la hipótesis nula que se plantea, en concreto, es que los primeros 40 coeficientes de correlación son cero frente a la hipótesis alternativa de que no lo son. Para un

2.3. Análisis de series temporales con **R**

55

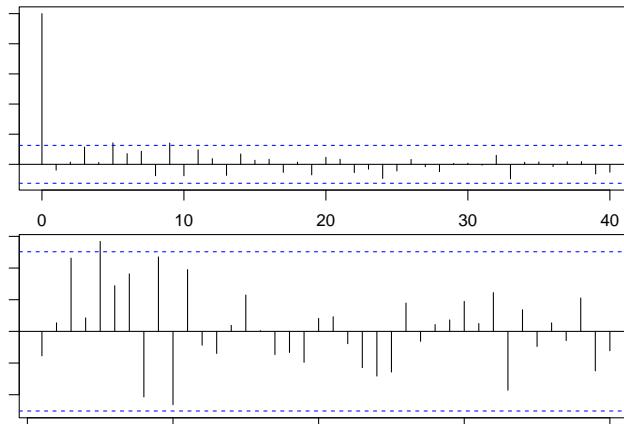


Figura 2.9: ACF y PACF estimadas de los residuos.

nivel de significación del 5 %, la evidencia empírica no es suficiente para rechazar H_0 , ya que el p-valor correspondiente al estadístico del contraste (X-squared o Chi-cuadrado) es 0.5448 mayor que el nivel de significación del 5 % y, por lo tanto, los residuos están incorrelacionados. El código **R** para llevar a cabo el test de Box-Ljung es:

```
Box.test(residuos, type = "Ljung-Box")
#>
#> Box-Ljung test
#>
#> data: residuos
#> X-squared = 0.36682, df = 1, p-value = 0.5447
```

4. Análisis de la bondad del ajuste.

También se debe comprobar la capacidad de ajuste del modelo comparando los valores observados y los estimados. Una forma sencilla e intuitiva de hacerlo es a través de su representación gráfica. El código **R** utilizado para ello es el siguiente:

```
plot(ipc_ts)
lines(ipc_ts - modelo$residuals, col = "red")
```

La Fig. 2.10 muestra como el modelo estimado se ajusta bastante bien a los valores observados y, por lo tanto, evidencia que el modelo sARIMA (1,1,0)(0,1,1) propuesto capta la dinámica del IPC en el periodo muestral analizado.

Por último, para comprobar la adecuación entre el modelo estimado y los valores observados no suele ser adecuado utilizar el coeficiente de determinación o el coeficiente de determinación

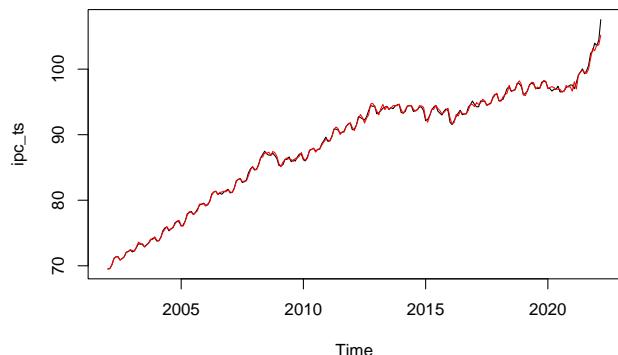


Figura 2.10: Ajuste con el modelo estimado.

corregido, ya que si es necesario calcular diferencias para convertir a la serie en estacionaria, la variable dependiente no es la misma (es decir, no es lo mismo Y_t , que $dlog(Y_t)$, que $d12dlog(Y_t)$). Si se quieren comparar diferentes modelos para elegir cuál es el que mejor se ajusta a los datos, es necesario utilizar otros métodos de información, como el criterio de información de Akaike (AIC), el criterio bayesiano o de Schwarz (BIC) o el de Hannan-Quinn (HQC). Si, por ejemplo, se eligiese el AIC será mejor el modelo con menor AIC.

2.3.4. Predicción

Después de comprobar que el modelo ARIMA estimado es adecuado se puede utilizar para obtener valores futuros de la variable objeto de análisis. Las predicciones que se obtienen pueden ser de dos tipos: puntuales o por intervalos. Las predicciones puntuales se obtienen calculando el valor futuro en $T+h$ esperado de la variable (\hat{Y}_{T+h}) condicionado al conjunto de información disponible hasta el momento actual (T). Para obtener las predicciones por intervalos es necesario sumar y restar a la predicción puntual la desviación típica del error de predicción multiplicada por el valor crítico tabulado para el nivel de confianza de fijado.

Algunas características generales de las predicciones obtenidas con modelos ARIMA son:

- En modelos MA(q) o sMA(Q). Si el horizonte temporal de predicción es mayor que el orden del proceso, entonces la predicción es la media del proceso.
- En los modelos AR(p) o sAR(P) a medida que aumenta el horizonte temporal, la predicción tiende a la media del proceso.
- En los modelos ARMA(p,q) o SARMA(P,Q), para ordenes superiores al medias móviles la función de predicción se comporta como la de un autorregresivo, lo que implica que tiende a la media del proceso.

2.3. Análisis de series temporales con **R**

57

- Es importante que la serie sea estacionaria para que las predicciones sean estables, ya que si no es estacionaria el modelo tendrá un comportamiento explosivo.
- Cuanto más alejado esté el horizonte temporal de predicción mayor será la incertidumbre respecto de las predicciones obtenidas.

En el caso concreto del IPC, para obtener 12 predicciones puntuales, así como sus correspondientes intervalos de predicción al 80 % y 95 % de confianza y su representación gráfica con dicho modelo (Fig. 2.11), el código **R** que se puede utilizar es:

```
forecast::forecast(modelo, h = 12)
#>      Point Forecast   Lo 80    Hi 80    Lo 95    Hi 95
#> Apr 2022     109.7116 109.2900 110.1332 109.0668 110.3564
#> May 2022     110.5926 109.8443 111.3410 109.4481 111.7371
#> Jun 2022     111.1030 110.0714 112.1346 109.5253 112.6806
#> Jul 2022     110.5518 109.2748 111.8289 108.5987 112.5050
#> Aug 2022     110.7714 109.2787 112.2641 108.4885 113.0543
#> Sep 2022     111.0288 109.3435 112.7140 108.4515 113.6060
#> Oct 2022     111.9631 110.1034 113.8228 109.1189 114.8073
#> Nov 2022     112.1740 110.1540 114.1939 109.0847 115.2632
#> Dec 2022     112.3896 110.2209 114.5583 109.0728 115.7064
#> Jan 2023     111.6049 109.2968 113.9130 108.0750 115.1349
#> Feb 2023     111.6666 109.2270 114.1061 107.9355 115.3976
#> Mar 2023     112.5012 109.9369 115.0656 108.5794 116.4231

prediccciones <- forecast::forecast(modelo, h = 12)
autoplot(prediccciones)
```

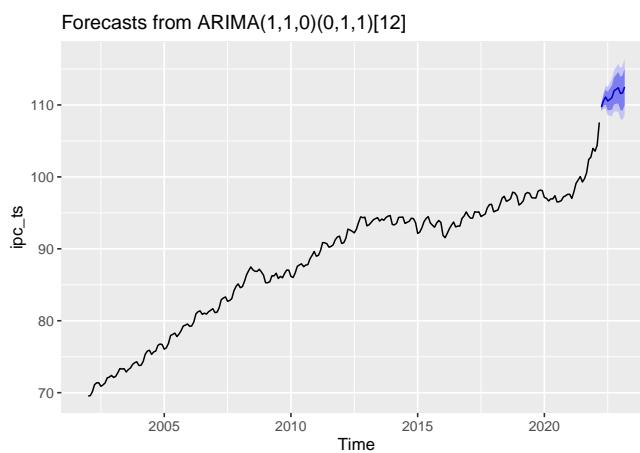


Figura 2.11: Predicciones con el modelo ARIMA(1,1,0)(0,1,1)12 estimado.

Resumen

Los modelos univariantes ARIMA analizan desde un punto de vista moderno y estocástico las series temporales y son muy útiles para captar su dinámica. Es importante destacar que:

- Es necesario tener en cuenta conceptos básicos que son fundamentales para la correcta modelización ARIMA, entre ellos se puede destacar: proceso estocástico, estacionalidad, estacionariedad, invertibilidad o ruido blanco.
- Las principales fases para la obtención y utilización de un modelo ARIMA son: especificación, estimación, validación y predicción.
- Es necesario que los datos sean estacionarios en media (no pueden presentar tendencia creciente o decreciente) y en varianza (no pueden tener fluctuaciones de diferente amplitud) para identificar el modelo ARIMA que mejor capte la dinámica de la serie temporal objeto de análisis.
- La ACF y la PACF muestrales son muy útiles para identificar el modelo más adecuado para explicar la dinámica de los datos.
- Para comprobar que el modelo estimado es adecuado es necesario realizar: un análisis de los parámetros estimados, un análisis de los residuos y de la bondad del ajuste.
- Validado el modelo, fundamentalmente, se suele utilizar para predecir y tomar decisiones.

Capítulo 3

Análisis discriminante

M^a Leticia Meseguer Santamaría^a y Manuel Vargas Vargas^a

^aUniversidad de Castilla-La Mancha

3.1. Introducción

El **análisis discriminante (AD)** es una técnica de dependencia orientada a la clasificación de individuos en grupos (o poblaciones) preexistentes y con ciertas características conocidas, utilizando para ello la información proporcionada por un conjunto de variables clasificadoras.¹ La clasificación se realiza mediante **funciones discriminantes**, combinaciones de las clasificadoras originales y que se emplean como criterio para asignar cada individuo a un grupo o población.

De esta forma, en el análisis discriminante se identifican dos **finalidades**: la descriptiva, caracterizando la separación entre grupos al proporcionar la contribución de cada variable clasificadora a dicha separación; y la predictiva, estableciendo el criterio de clasificación de un individuo nuevo a alguno de los grupos conociendo los valores de las variables clasificadoras.

El problema de la *discriminación* puede plantearse de diversas formas y aparece en numerosas áreas de investigación. El **AD** se agrupa dentro de los modelos denominados **supervisados**, puesto que se conoce a priori a qué grupo o población está asignado cada individuo de la muestra, y es utilizado en campos tan diferentes como los sistemas automáticos de concesión de créditos bancarios (*credit scoring*), clasificación de pacientes en función de pruebas diagnósticas, atribución de obras literarias o pictóricas a autores, o en control de calidad, cuando la información es muy costosa o requiera la destrucción de las unidades. En el campo de la ingeniería, la discriminación se conoce como *reconocimiento de patrones (pattern recognition)* y es utilizada para el diseño de máquinas de clasificación automática (reconocimiento de billetes o monedas, sonidos, etc.)

¹La referencia a individuos es en sentido amplio, entendiéndose por individuos no solo personas, sino también objetos, entes, elementos, casos, etc.

Aunque existen varios enfoque diferentes, en este capítulo se adoptará el enfoque clásico de Fisher, que asume la normalidad multivariante de las variables clasificadoras. Como punto de partida, para un **AD** se considera:

- Un conjunto de **N** individuos, de los que se conoce su grupo de pertenencia. Esta información se resume en una variable categórica **Y** cuyas categorías son los distintos grupos.
- Un conjunto de **k** grupos ($k \geq 2$) con, al menos, dos individuos en cada uno de ellos.
- Un conjunto de **p** variables clasificadoras, medidas en intervalo o razón. Estas variables no deben presentar multicolinealidad, es decir, ninguna clasificadora puede ser combinación lineal de otras clasificadoras. Además, dado el enfoque adoptado, se asume que estas variables siguen una distribución normal multivariante.²

El número de variables discriminantes debe ser inferior en más de dos al número de individuos ($p < N - 2$) para poder identificar los parámetros (véase Cap. ??). Además, en la práctica, es útil disponer de algún criterio o método que permita seleccionar qué variables se considerarán clasificadoras. Una alternativa pueden ser los métodos de jerarquización de variables desarrollados en análisis de regresión o la selección de variables (*feature selection*, véase Cap. ??). Como punto inicial, es frecuente que se considere que una variable puede ser clasificadora si presenta diferencias en su distribución entre los grupos, utilizando para ello un **ANOVA**.

Así, el **AD** busca determinar un criterio o *regla discriminante* que clasifique a cada individuo, j , en uno de los k grupos conociendo las observaciones de cada una de las p variables X_i , es decir, el vector $X_j = (X_{1,j}, X_{2,j}, \dots, X_{p,j})'$. Estas reglas discriminantes están basadas en la información muestral y en los supuestos que sobre ésta se hacen; en el planteamiento clásico de Fisher, al asumir la normalidad de las variables, se basan en el comportamiento en los k grupos de los vectores de medias y de las matrices de varianzas-covarianzas (véase Sec. ??). Por ello, se suelen distinguir varios casos, que conducen a distintos métodos de obtención de reglas discriminantes, por lo que reciben nombres diferentes:

- El caso más sencillo (e históricamente el más antiguo), además de la normalidad, supone que las matrices de varianzas-covarianzas son iguales en todos los grupos (supuesto de *homocedasticidad*). El método se conoce como **análisis discriminante lineal** (*linear discriminant analysis* o *LDA*). En este caso, detallado en la Sec. 3.2, la diferencia en la distribución de las variables entre los grupos se produce en los vectores de medias, y la función discriminante obtenida es una combinación lineal de las variables clasificadores que minimiza los errores de clasificación.
- Otra posibilidad es que se asuma la normalidad pero no la igualdad de las matrices de varianzas-covarianzas entre los grupos. En este caso, la función discriminante es una función cuadrática, por lo que el método se conoce como **análisis discriminante cuadrático** (*quadratic discriminant analysis* o *QDA*), detallado en la Sec. 3.3.

²Este supuesto garantiza que el método clásico propuesto por Fisher es óptimo. En la práctica, el AD es robusto frente a incumplimientos de la normalidad p-dimensional, por lo que también se aplica en muchos casos prácticos donde no se puede garantizar este requisito.

3.2. Análisis discriminante lineal

61

Sea cual sea el método elegido, las *reglas discriminantes* que se obtengan para clasificar a un individuo en uno de los grupos deben determinarse minimizando los errores de clasificación, que pueden ser evaluados probabilísticamente al disponer de la distribución de las variables en cada grupo. Así, para cada individuo j y sus valores de las variables clasificadoras $X_j = (x_{1j}, x_{2j}, \dots, x_{pj})'$, se dispone de las verosimilitudes para cada uno de los k grupos, $L_i(X_j; \theta_i)$, $1 \leq i \leq k$.

Conociendo la probabilidad *a priori* de pertenencia de un individuo a cada grupo³ π_i , $1 \leq i \leq k$, aplicando el teorema de Bayes (véase (??)), se puede calcular la probabilidad de que el individuo pertenezca a cada grupo, G_i

$$P(G_i/x_j) = \frac{L_i(X_j; \theta_i)\pi_i}{\sum_m L_m(X_j; \theta_m)\pi_m} \quad (3.1)$$

A partir de esta ecuación, la *regla discriminante* consiste en asignar al individuo al grupo más probable. Dado que el denominador de (3.1) es constante para todos los grupos, la regla equivale a asignar al individuo al grupo donde sea *ponderadamente* más verosímil:

$$j \text{ se clasifica en } G_i \text{ si } L_i(X_j; \theta_i)\pi_i = \max_m L_m(X_j; \theta_m)\pi_m \quad (3.2)$$

ecuación que se simplifica en el caso de igual probabilidad *a priori*, resultando la *regla discriminante* en asignar a cada individuo al grupo más verosímil.

En general, se pueden cometer dos tipos de error: no clasificar al individuo en un grupo cuando realmente pertenece a él; o clasificarlo en un grupo al que realmente no pertenece. Si no se conocen dichos costes (o son iguales), no afectan a la *regla discriminante*; sin embargo, si son conocidos y han de ser tenidos en cuenta, la regla se modificaría, ponderando cada verosimilitud por los costes asociados.

En las secciones siguientes se abordarán ambos modelos de **AD** que, aunque no son los únicos, sí representan la gran mayoría de las aplicaciones prácticas.

3.2. Análisis discriminante lineal

Es un modelo de **AD** basado en los supuestos generales expuestos en el epígrafe anterior (N individuos, k grupos y p variables clasificadoras con distribución normal y sin multicolinealidad) y caracterizado por la **igualdad de las matrices de varianza-covarianza** de las variables en todos los grupos. Para la exposición de la metodología, se presentará el caso más sencillo, con sólo dos grupos y probabilidades *a priori* iguales, para generalizarlo posteriormente al caso general de k grupos.

Dos grupos y una variable clasificadora.

³Suele ser frecuente asignar idéntica probabilidad *a priori* a todos los grupos $\pi_i = \frac{1}{k}$ o proporcional al tamaño de cada grupo.

Es el supuesto más simple posible, donde se han de clasificar N individuos en con dos grupos (I y II) a partir de la información de una única variable clasificadora, X . En este caso, las distribuciones de probabilidad de X en los grupos I y II solo difieren en la media, como se muestra en la Fig. 3.1.

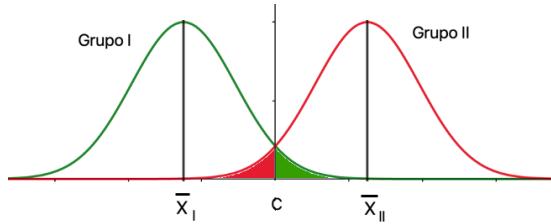


Figura 3.1: Dos grupos y una variable clasificadora.

La *regla discriminante* consistirá en asignar cada individuo al grupo donde su verosimilitud es mayor. Como se aprecia, esta regla divide la recta real en dos partes, cuyo **punto de corte (C)** es:

$$C = \frac{\bar{x}_I + \bar{x}_{II}}{2} \quad (3.3)$$

quedando la asignación de cada individuo:⁴

$$\text{si } x_j < C \in \text{ Grupo I y si } x_j > C \in \text{ Grupo II} \quad (3.4)$$

Las probabilidades de los errores que se pueden cometer en la asignación corresponderían a las áreas resaltadas en rojo (individuo asignado al grupo I cuando realmente pertenece al grupo II) y en verde (individuo asignado al grupo II cuando realmente pertenece al grupo I), constituyendo la zona de error de clasificación.

Dos grupos y dos variables clasificadoras.

Si, bajo los mismos supuestos, se dispone de dos variables clasificadoras, X_1 y X_2 , se proyectan los elipsoides de ambos grupos sobre las dos variables, se obtendría la Fig. 3.2:

Se obtienen, sobre cada variable, zonas de error de clasificación amplias (marcadas en amarillo) que conllevarán errores de clasificación grandes. Sin embargo, si se proyectan ambos elipsoides sobre un nuevo eje, obtenido como una combinación lineal de ambas variables clasificadoras ($w_1X_1 + w_2X_2 - D = 0$), es posible reducir la zona de error de clasificación y, como consecuencia, la probabilidad de error de clasificación.

La obtención de la combinación lineal que minimiza la probabilidad de error de clasificación fue resuelto por Fisher buscando una **función discriminante** que maximiza la separación entre

⁴De forma intuitiva, se asigna cada individuo al grupo cuya media está más cercana al valor de la variable. Esta interpretación se generaliza a más variables clasificadoras, asignando cada individuo al grupo cuyo centroide esté más cercano a él. Si la probabilidad *a priori* fuese proporcional al tamaño de los grupos, el punto de corte se calcularía como $C = \frac{n_I\bar{x}_I + n_{II}\bar{x}_{II}}{N}$.

3.2. Análisis discriminante lineal

63

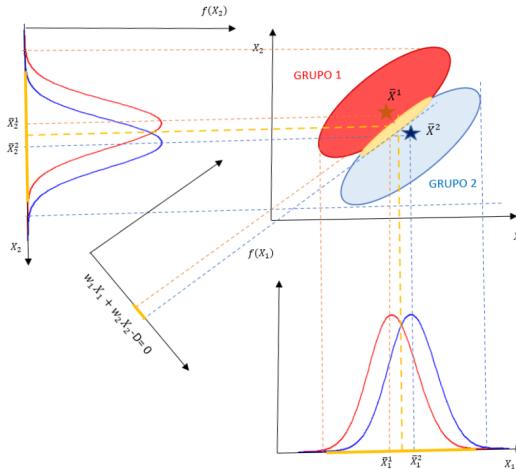


Figura 3.2: Dos grupos y dos variables clasificadoras.

ambos grupos, maximizando la distancia entre sus centroides y minimizando la variabilidad dentro de cada grupo. El procedimiento se detalla para el caso general de p variables.

Dos grupos y p variables clasificadoras.

El objetivo es encontrar una *regla discriminante* que permita *separar* ambos grupos. Se busca la **función discriminante de Fisher**, que se plantea como una combinación lineal de las p variables clasificadoras:

$$D = w_1X_1 + w_2X_2 + \dots + w_pX_p \quad (3.5)$$

que asignaría al individuo j -ésimo una **puntuación discriminante** $D_j = w_1X_{1j} + w_2X_{2j} + \dots + w_pX_{pj}$; expresando matricialmente estas puntuaciones en diferencias respecto a las medias:

$$\begin{pmatrix} D_1 - \bar{D} \\ D_2 - \bar{D} \\ \vdots \\ D_N - \bar{D} \end{pmatrix} = \begin{pmatrix} X_{11} - \bar{X}_1 & X_{21} - \bar{X}_2 & \dots & X_{p1} - \bar{X}_p \\ X_{12} - \bar{X}_1 & X_{22} - \bar{X}_2 & \dots & X_{p2} - \bar{X}_p \\ \vdots & \vdots & \ddots & \vdots \\ X_{1N} - \bar{X}_1 & X_{2N} - \bar{X}_2 & \dots & X_{pN} - \bar{X}_p \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix} \quad (3.6)$$

donde $\bar{D} = w_1\bar{X}_1 + w_2\bar{X}_2 + \dots + w_p\bar{X}_p$ por ser D una combinación lineal de variables normales. En notación abreviada, la ecuación (3.6) se puede expresar como $d = Xw$.

La suma de cuadrados de las desviaciones de la función discriminante respecto a su media quedaría entonces como:

$$\mathbf{d}' \mathbf{d} = \mathbf{w}' \mathbf{X}' \mathbf{X} \mathbf{w} \quad (3.7)$$

donde $\mathbf{X}'\mathbf{X}$ es la matriz simétrica de las desviaciones cuadráticas respecto a sus medias de las variables clasificadoras (o matriz **suma de cuadrados y productos cruzados**, SCPC). Esta matriz se puede descomponer en la suma de dos matrices, la SCPC entregrupos, \mathbf{F} y la SCPC residual o intragrupos, \mathbf{U} , por lo que la ecuación (3.7) se puede reexpresar como:

$$\mathbf{d}' \mathbf{d} = \mathbf{w}' \mathbf{Fw} + \mathbf{w}' \mathbf{Uw} \quad (3.8)$$

Fisher propuso determinar los pesos $\{w_i\}$ buscando que discriminen entre los grupos, maximizando la variabilidad entre grupos respecto a la intragrupos, es decir:

$$\max_{\mathbf{w}} \frac{\mathbf{w}' \mathbf{Fw}}{\mathbf{w}' \mathbf{Uw}} \quad (3.9)$$

Como esta función es invariante frente a cambios de escala, maximizar (3.9) es equivalente a maximizar $\mathbf{w}' \mathbf{Fw}$ con la condición $\mathbf{w}' \mathbf{Uw} = 1$ que, aplicando los multiplicadores de Lagrange, implica:

$$\begin{aligned} L &= \mathbf{w}' \mathbf{Fw} - \lambda(\mathbf{w}' \mathbf{Uw} - 1) \Rightarrow \frac{\partial L}{\partial w} = 2\mathbf{Fw} - 2\lambda\mathbf{Uw} = 0 \Rightarrow \\ &\Rightarrow \mathbf{Fw} = \lambda\mathbf{Uw} \Rightarrow (\mathbf{U}^{-1}\mathbf{F})\mathbf{w} = \lambda\mathbf{w} \end{aligned} \quad (3.10)$$

Así, el vector propio asociado al mayor autovalor de la matriz $\mathbf{U}^{-1}\mathbf{F}$ proporcionará la **función discriminante lineal de Fisher** que mejor separa ambos grupos.

El **punto de corte (C)** se obtiene evaluando la función discriminante en la media de cada grupo y promediando por el tamaño de los grupos:

$$\begin{aligned} \bar{D}_I &= w_1 \bar{X}_{1I} + w_2 \bar{X}_{2I} + \dots + w_p \bar{X}_{pI} \\ \bar{D}_{II} &= w_1 \bar{X}_{1II} + w_2 \bar{X}_{2II} + \dots + w_p \bar{X}_{pII} \end{aligned} \quad (3.11)$$

$$C = \frac{n_I \bar{D}_I + n_{II} \bar{D}_{II}}{N} \quad (3.12)$$

Y el **criterio de asignación para cada individuo, j** , será:

$$\text{si } D_j < C \in \text{ Grupo I y si } D_j > C \in \text{ Grupo II} \quad (3.13)$$

G grupos y p variables:

En caso de existir más de dos grupos, la generalización del caso anterior es relativamente sencilla. Siguiendo la misma idea utilizada para dos grupos, se debería obtener un número de **funciones discriminantes de Fisher** suficiente para separar los k grupos; este número es $T = \min(k - 1, p)$.⁵

⁵Para separar linealmente k grupos hacen falta $k - 1$ hiperplanos, pero su obtención está también limitada por el número p de variables clasificadoras.

3.2. Análisis discriminante lineal

65

Así, cada una de las T funciones discriminantes será una combinación lineal de las p variables clasificadoras:

$$D_t = w_{t1}X_1 + w_{t2}X_2 + \dots + w_{tp}X_p \text{ para } t = 1, \dots, T \quad (3.14)$$

donde se exige que $\text{Corr}(D_i, D_j) = 0, \forall i \neq j$.

La suma de cuadrados de las desviaciones de la matriz \mathbf{D} de funciones discriminantes respecto a sus medias tendría una expresión equivalente a la ecuación (3.7):

$$\mathbf{D}'\mathbf{D} = \mathbf{W}'\mathbf{X}'\mathbf{X}\mathbf{W} \quad (3.15)$$

Para que las funciones discriminen lo máximo posible a los k grupos, las combinaciones lineales han de maximizar la variabilidad entre los grupos respecto a la intragrupos, en un razonamiento análogo la expuesto en la ecuación (3.9):

$$\max \frac{\mathbf{W}'\mathbf{F}\mathbf{W}}{\mathbf{W}'\mathbf{U}\mathbf{W}} \quad (3.16)$$

Al ser una función homogénea, su maximización equivale a maximizar $\mathbf{W}'\mathbf{F}\mathbf{W}$ con la condición $\mathbf{W}'\mathbf{U}\mathbf{W} = 1$ que, aplicando los multiplicadores de Lagrange, implica:

$$\begin{aligned} L &= \mathbf{W}'\mathbf{F}\mathbf{W} - \lambda(\mathbf{W}'\mathbf{U}\mathbf{W} - 1) \Rightarrow \frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{F}\mathbf{W} - 2\lambda\mathbf{U}\mathbf{W} = 0 \Rightarrow \\ &\Rightarrow \mathbf{F}\mathbf{W} = \lambda\mathbf{U}\mathbf{W} \Rightarrow (\mathbf{U}^{-1}\mathbf{F})\mathbf{W} = \lambda\mathbf{W} \end{aligned} \quad (3.17)$$

Así, el vector propio asociado al mayor autovalor de la matriz $\mathbf{U}^{-1}\mathbf{F}$ (generalmente no simétrica) proporcionará la **primera función discriminante lineal de Fisher**, siendo el autovalor la proporción de varianza total explicada por las T funciones discriminantes que recoge la primera función.

Para obtener el resto de funciones discriminantes, basta con ir eligiendo los siguientes vectores propios asociados a los autovalores, ordenados decrecientemente. Como los vectores propios son linealmente independientes, las funciones de discriminación son incorreladas.⁶

De esta forma, la primera función discriminante, D_1 , será la que proporcione mayor discriminación entre los centroides de los grupos; D_2 será la que proporcione mayor discriminación, después de D_1 , y que esté incorrelada con ella; y así sucesivamente, D_t será la que produzca mayor discriminación entre los centroides de los grupos, después de las $t - 1$ anteriores, e incorrelada con todas las anteriores.

⁶Como la capacidad discriminante de la funciones va decreciendo, puede haber casos donde no se consideren relevantes todas, sino el conjunto de las m primeras. En ese caso, la variabilidad explicada sería $\sum_{i=1}^m \lambda_i$, por lo que la proporción de variabilidad atribuible a cada función discriminante D_t sería $\frac{\lambda_t}{\sum_{i=1}^m \lambda_i}$.

3.2.1. Discriminante lineal con R: la función `lda()`

A continuación, se va a ejemplificar la aplicación de un **discriminante lineal** con **R**. Para ello, se utilizará y cargará la base de datos `iris`, que consta de 150 observaciones y 5 variables, 4 numéricas, que serán las clasificadoras, y una categórica, sobre la que se realiza el análisis, con tres categorías: setosa, versicolor y virginica.

```
library("caret")
library("MASS")
library("klaR")
data("iris")
```

Se clasificarán las flores iris, identificadas con la variable `Species` (especies de iris), utilizando como variables clasificadoras: `Sepal.Length` (Longitud del sépalo), `Sepal.Width` (anchura del sépalo), `Petal.Length` (longitud del pétalo) y `Petal.Width` (anchura del pétalo).

Para evaluar la capacidad predictiva del análisis discriminante, se dividen los datos de la muestra en un 80 % para la estimación (o entrenamiento) y un 20 % para el test.⁷

Las distribuciones univariadas deben ser normales; si no fuera así, se podrían transformar utilizando log y root (distribuciones exponenciales) y Box-Cox (distribuciones sesgadas), como se muestra en la Sec. ???. Igualmente, es conveniente estandarizar las variables para evitar que la diferencia de escalas influya en la importancia relativa de cada variable clasificadora en las funciones discriminantes.

```
# División de los datos: 80% para entrenamiento y 20% para test
set.seed(123)
muestra <- iris$Species |>
  createDataPartition(p = 0.8, list = FALSE)
entrenamiento_d <- iris[muestra, ]
test_d <- iris[-muestra, ]
# Estimación de los parámetros de preprocessamiento (estandarización)
preproc_param <- entrenamiento_d |>
  preProcess(method = c("center", "scale"))
# Transformación de los datos usando los parámetros estimados
entrenamiento_t <- preproc_param |> predict(entrenamiento_d)
test_t <- preproc_param |> predict(test_d)
```

Una inspección previa de los datos puede ayudar a detercar si las variables clasificadoras pueden contribuir a la discriminación entre los grupos. En este ejemplo, la Fig. 3.3 muestra la densidad de cada variable sobre cada grupo para el conjunto de entrenamiento:

⁷ Esta estrategia es muy común en modelos predictivos, y tiene como objetivo evitar el **sobreajuste** a los datos muestrales; así, los datos del conjunto de test son realmente “*nuevos*” para el modelo, porque no han sido utilizados en la estimación.

3.2. Análisis discriminante lineal

67

```

library("ggplot2")
library("ggsignif")

p1 <- ggplot(data = entrenamiento_t, aes(x = Sepal.Length, fill = Species, colour =
  Species)) +
  geom_density(alpha = 0.3) +
  theme_bw()
p2 <- ggplot(data = entrenamiento_t, aes(x = Sepal.Width, fill = Species, colour =
  Species)) +
  geom_density(alpha = 0.3) +
  theme_bw()
p3 <- ggplot(data = entrenamiento_t, aes(x = Petal.Length, fill = Species, colour =
  Species)) +
  geom_density(alpha = 0.3) +
  theme_bw()
p4 <- ggplot(data = entrenamiento_t, aes(x = Petal.Width, fill = Species, colour =
  Species)) +
  geom_density(alpha = 0.3) +
  theme_bw()
ggarrange(p1, p2, p3, p4, ncol = 2, nrow = 2, common.legend = TRUE, legend = "bottom")

```

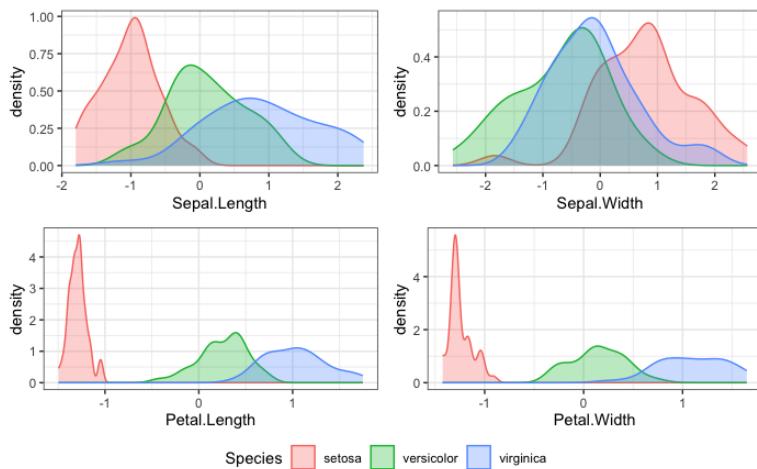


Figura 3.3: Densidad de cada variable clasificadora sobre los grupos.

Igualmente, los gráficos bivariantes pueden ser ayudar a ver si hay “distancias” entre los centroides de los grupos para las variables clasificadoras, como muestra la Fig. 3.4:

```

pairs(x = entrenamiento_t[, -5], col = c("firebrick", "green3",
  "darkblue")[entrenamiento_t$Species], pch = 20)

```

Como se observa en estos gráficos, las variables clasificadoras pueden contribuir a la discriminación entre las tres especies de flores *iris*.

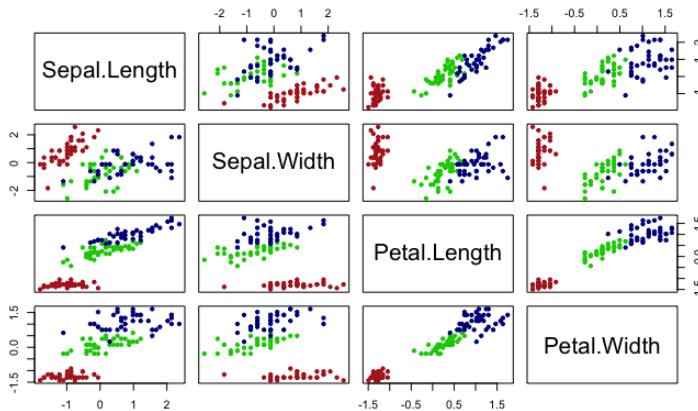


Figura 3.4: Diagramas bivariantes de dispersión de las variables clasificadoras.

Para aplicar la función `lda()` debemos especificar la variable de agrupación (`Species`) y el conjunto de datos (`entrenamiento_t`); de forma opcional, se pueden especificar las probabilidades *a priori* (`prior`, por defecto se usa `proportions`), el método de estimación de las medias y varianzas (`method`, por defecto `moment`) o el argumento `CV` para obtener los grupos pronosticados y las probabilidades a posteriori (por defecto, `CV=FALSE`)

```
options(digits = 4)
modelo_lda <- lda(Species ~ ., data = entrenamiento_t)
modelo_lda
#> Call:
#> lda(Species ~ ., data = entrenamiento_t)
#>
#> Prior probabilities of groups:
#>      setosa  versicolor  virginica
#> 0.3333 0.3333 0.3333
#>
#> Group means:
#>           Sepal.Length Sepal.Width Petal.Length Petal.Width
#> setosa       -1.0113     0.78049    -1.2900    -1.2453
#> versicolor      0.1014    -0.68675     0.2566     0.1473
#> virginica      0.9099    -0.09374     1.0334     1.0981
#>
#> Coefficients of linear discriminants:
#>           LD1         LD2
#> Sepal.Length  0.6795  0.04464
#> Sepal.Width   0.6565 -1.00330
#> Petal.Length -3.8365  1.44176
#> Petal.Width   -2.2722 -1.96516
#>
```

3.2. Análisis discriminante lineal

69

```
#> Proportion of trace:
#>   LD1     LD2
#> 0.9902 0.0098
```

La salida muestra las **probabilidades previas** (*Prior probabilities of groups*) y los **centroides de cada grupo** (*Group means*). A continuación muestra las **funciones discriminantes de Fisher** mediante los respectivos coeficientes w_{it} . En este caso, las dos funciones discriminantes son:

$$D_1 = 0,6795 * SL + 0,6565 * SW - 3,8365 * PL - 2,2722 * PW$$

$$D_2 = 0,0446 * SL - 1,0033 * SW + 1,4418 * PL - 1,9651 * PW$$

con una proporción de discriminación de 0.9902 y 0.0098, respectivamente.

La proyección de los individuos en el plano formado por las dos funciones discriminantes se recoge en la Fig. 3.5:

```
datos_lda <- cbind(entrenamiento_t, predict(modelo_lda)$x)
ggplot(datos_lda, aes(LD1, LD2)) +
  geom_point(aes(color = Species)) +
  ggtitle("Gráfico LDA")
```

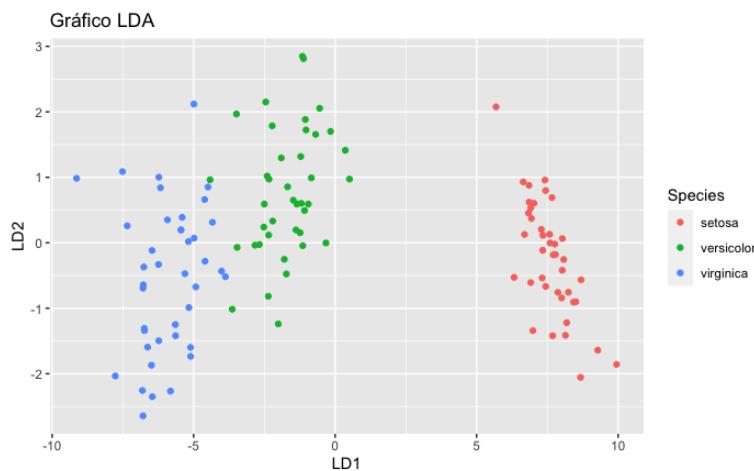


Figura 3.5: Proyección sobre las dos funciones discriminantes de los individuos.

Como se aprecia, la primera función discriminante es la que mayor contribución tiene a la separación entre los grupos, separando muy claramente a la especie *setosa* y, en menor medida, a las especies *virginica* y *versicolor*, grupos entre los que hay un pequeño grado de solapamiento. Por otro lado, la segunda función discriminante, con una proporción de 0.0098 apenas contribuye a la separación entre grupos.

Por último, es posible visualizar cómo quedarían las regiones bivariantes que clasifican a los individuos en cada clase mediante la función `partimat()` del paquete `klaR`, como muestra la Fig. 3.6:

```
partimat(Species ~ ., data = entrenamiento_t, method = "lda", image.colors =
  c("skyblue", "lightgrey", "yellow"), col.mean = "red")
```

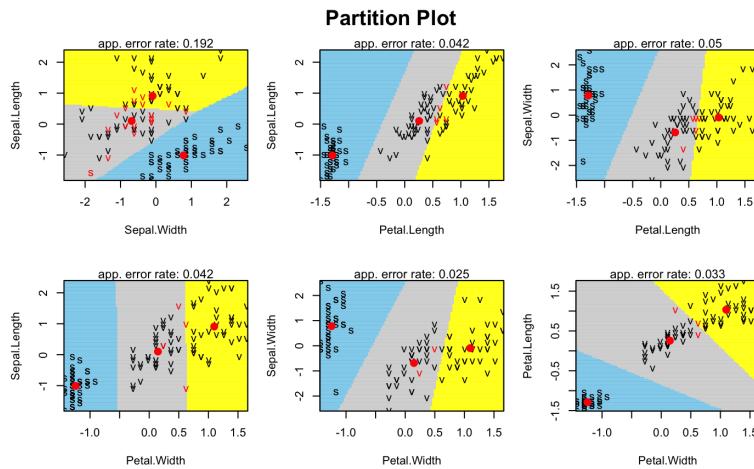


Figura 3.6: Regiones bivariantes de clasificación en cada grupo: setosa (celeste), versicolor (gris) y virginica (amarillo). Centroides en rojo.

Por último, aplicando las funciones discriminantes a los datos reservados para estudiar la capacidad predictiva del modelo:

```
predicciones_lda <- modelo_lda |> predict(test_t)
table(test_t$Species, predicciones_lda$class, dnn = c("Grupo real", "Grupo
  → pronosticado"))
#>           Grupo pronosticado
#> Grupo real   setosa versicolor virginica
#>   setosa       10      0      0
#>   versicolor    0     10      0
#>   virginica     0      1      9
mean(predicciones_lda$class == test_t$Species)
#> [1] 0.9667
```

se obtiene la tabla conocida como **matriz de confusión**, donde se compara el grupo real con el pronosticado por el modelo. En este caso, se clasifican correctamente 29 de las 30 “nuevas” flores, indicando un grado de ajuste del 96.9667 %.

3.3. Análisis discriminante cuadrático

En el discriminante lineal visto anteriormente, se asume que las variables clasificadoras tienen idénticas matrices de varianza-covarianza en los distintos grupos, supuesto que garantiza que las funciones discriminantes son combinaciones lineales de las variables

Es posible eliminar esta restricción, permitiendo diferencias en las matrices de varianzas-covarianzas en los grupos, lo que introduce términos cuadráticos en las funciones discriminantes.

Así, denominando π_t a la probabilidad *a priori* de pertenecer al grupo G_t , μ_t y Σ_t al vector de medias y matriz de varianzas-covarianzas respectivamente de dicho grupo, a partir del vector de observaciones X se puede obtener la **función discriminante cuadrática** como:

$$Q(\mathbf{X}) = \frac{1}{2}\mathbf{X}'(\Sigma_i^{-1} - \Sigma_j^{-1})\mathbf{X} + \mathbf{X}'(\Sigma_i^{-1}\mu_i - \Sigma_j^{-1}\mu_j)\mathbf{X} + \frac{1}{2}\mu_j'\Sigma_j^{-1}\mu_j - \frac{1}{2}\mu_i'\Sigma_i^{-1}\mu_i + \frac{1}{2}\log(|\Sigma_j|) - \frac{1}{2}\log(|\Sigma_i|) \quad (3.18)$$

$\forall i \neq j, i, j = 1, 2, \dots, k$

A partir de aquí, la **regla de clasificación** para un individuo consiste en evaluar la función discriminante (3.18) para cada grupo y asignarlo a aquél que verifique:

$$G = \underset{t}{\operatorname{argmax}} \ln\pi_t + \frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(X - \mu_t)^t\Sigma_t^{-1}(X - \mu_t) \quad (3.19)$$

En este caso, los límites de la región de clasificación son ecuaciones cuadráticas de x.

3.3.1. Discriminante cuadrático con R: la función qda()

Para ilustrar la realización de un análisis discriminante cuadrático en R, se va a ejemplificar la aplicación de la función qda() a los datos `iris` utilizados en el modelo lineal. La elección de la misma base de datos responde a un planteamiento didáctico, para poder comparar los resultados de ambos métodos y las diferencias que produce asumir la igualdad de matrices de varianza-covarianza (método lineal) o no asumirlo (método cuadrático).⁸

```
options(digits = 4)
modelo_qda <- qda(Species ~ ., data = entrenamiento_t)
modelo_qda
#> Call:
#> qda(Species ~ ., data = entrenamiento_t)
#>
#> Prior probabilities of groups:
#>      setosa versicolor virginica
#>      0.3333    0.3333    0.3333
#>
```

⁸En una situación real, la estrategia razonable sería decidir previamente sobre la hipótesis de igualdad de las matrices de varianza-covarianza (utilizando, por ejemplo el contraste *M de Box*, aunque es muy sensible al supuesto de normalidad multivariante) y, en función del resultado, optar por uno de las dos alternativas.

```
#> Group means:
#>           Sepal.Length Sepal.Width Petal.Length Petal.Width
#> setosa      -1.0113     0.78049    -1.2900     -1.2453
#> versicolor   0.1014    -0.68675     0.2566     0.1473
#> virginica    0.9099    -0.09374     1.0334     1.0981
```

La representación gráfica de las áreas por las que se clasifican los individuos se representa en la Fig. 3.7.

```
partimat(Species ~ ., data = entrenamiento_t, method = "qda", image.colors =
  c("skyblue", "lightgrey", "yellow"), col.mean = "red")
```

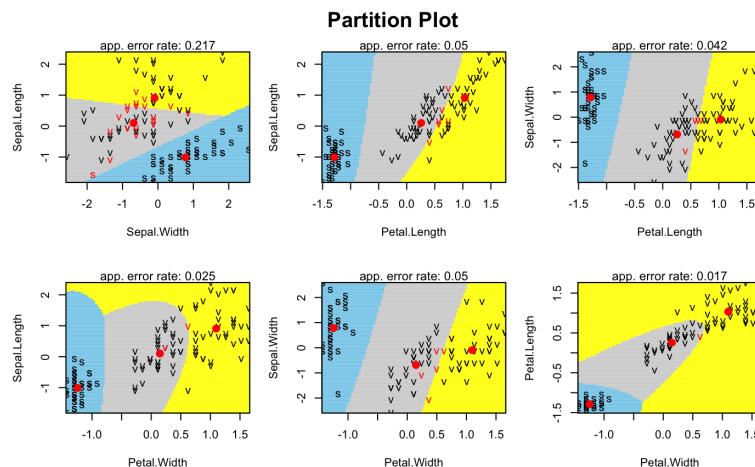


Figura 3.7: Regiones bivariantes de clasificación en cada grupo: setosa (celeste), versicolor (gris) y virginica (amarillo). Centroides en rojo.

Como se aprecia, ahora los contornos de las áreas no son siempre lineales, sino que incluyen fronteras cuadráticas.

```
predicciones_qda <- modelo_qda |> predict(test_t)
table(test_t$Species, predicciones_qda$class, dnn = c("Grupo real", "Grupo
  ↪ pronosticado"))
#>           Grupo pronosticado
#> Grupo real  setosa versicolor virginica
#> setosa        10         0         0
#> versicolor     0        10         0
#> virginica      0         1        9
mean(predicciones_qda$class == test_t$Species)
#> [1] 0.9667
```

Resumen

El *análisis discriminante* permite clasificar individuos en distintos grupos preexistentes en relación a una variable cualitativa, a partir de las variables clasificadoras. La información se sintetiza en las funciones discriminantes. Su uso puede tener una finalidad descriptiva, identificar la separación entre grupos y la contribución de cada variable clasificadora; y una finalidad predictiva, para clasificar un individuo nuevo. Los principales tipos son el lineal y el cuadrático, que se desarrollan en **R** con las funciones `lda()` y `qda()`, respectivamente.

Capítulo 4

Análisis conjunto

M^a Leticia Meseguer Santamaría

Universidad de Castilla-La Mancha

4.1. Introducción y conceptos clave

El **análisis conjunto** (o *Conjoint analysis*) estudia situaciones de elección múltiple. Funciona dividiendo un producto o servicio en sus componentes (atributos y niveles) y analizando las utilidades parciales de cada uno; después se realizan diferentes combinaciones de éstos para identificar las preferencias del consumidor. Permite conocer las preferencias del público ante el lanzamiento de nuevos productos o servicios, para adaptarlo a ellas y maximizar el éxito. Evalúa la sensibilidad al precio u otras características del producto y predice su comportamiento en el mercado. Mediante este análisis se puede establecer qué atributo y qué categoría (nivel) son los más valorados y cuantificarlos de forma relativa.

Los principales elementos clave de un análisis conjunto son:

- **Atributos:** características de un producto o servicio sobre las que se basará la elección.
- **Niveles:** valores que puede tomar cada atributo. El número de niveles no tiene por qué ser igual en todos los atributos, pero es conveniente que sea similar para facilitar la elección del entrevistado.
- **Diseño experimental:** proceso estadístico por el que se confeccionan las opciones de las preguntas de la encuesta, y que realiza el investigador.
- **Utilidades parciales:** valoración numérica que representa el grado de preferencia por cada nivel de atributo. Se hace en referencia a las otras opciones. También se conocen como “*Partworth utility*”, “Niveles de preferencia”, “Niveles relativos de preferencias” o “Valoraciones relativas”.
- **Importancia:** valores numéricos que señalan la preferencia por cada atributo.

- **Perfil:** combinación concreta de niveles de los atributos de un producto o servicio. También se denomina alternativa.
- **Escenarios:** conjunto de perfiles entre los que el entrevistado tiene que señalar sus preferencias.
- **Pregunta:** conjunto de opciones, normalmente entre 8 y 12.
- **Probabilidad de elección:** probabilidad de tomar una decisión determinada considerando las utilidades parciales.
- **Modelo de comportamiento:** modelo de decisión que se infiere a partir de las probabilidades de elección, después del análisis de las preferencias (utilidades parciales) de los entrevistados.
- **Análisis:** estimación de las utilidades parciales.
- **Valoración:** impacto del nuevo producto o servicio, basado en el modelo de comportamiento que se haya establecido.

Ejemplo: elección de gimnasio. Sea una muestra de gimnasios en la que se identifican 4 atributos (características) con distintos niveles. Se realiza una encuesta sobre qué gimnasio se prefiere. Los participantes eligen una opción entre las distintas combinaciones ofrecidas (**preguntas**). Mediante el análisis de los resultados de la encuesta se extrae el peso de cada atributo y nivel en las respuestas (**utilidades parciales**), que describen las preferencias medias de los encuestados; así mismo, se identifican los atributos y niveles más valorados y su importancia relativa.

Por ejemplo, los resultados pueden indicar que el horario más demandado es el de 09:00 a 23:00 durante 7 días a la semana.

HORARIO	CLASSESPROGRAMADAS	LIMPIEZA	PRECIO
24 horas (7d)	5 clases/día	3 veces al día	100 €/mes
		2 veces al día	
		1 vez al día	
		50 €/mes	
		25 €/mes	

ATRIBUTOS

NIVELES DE CADA ATRIBUTO

Figura 4.1: elección de gimnasio. Atributos: horario, clases programadas, limpieza y precio

4.2. Tipos de análisis conjunto

En la literatura científica sobre este análisis se diferencian tres formas de abordar la investigación:

El análisis tradicional de perfil completo (*Full Profile*)

Este análisis, también llamado *Conjoint Value Analysis (CVA)*, muestra al entrevistado una selección de productos resultantes de la combinación de determinados niveles de una serie de atributos (perfiles) y se le pide que los valore según su preferencia en una escala numérica (utilidad). Las preferencias globales de cada individuo se descomponen en utilidades independientes

4.3. Etapas de la realización del análisis conjunto

77

y compatibles para cada atributo y nivel mediante métodos basados en regresión lineal múltiple (véase Cap. @ref(#cap-lm)), que proporcionan las funciones de utilidad para los niveles, *partworth*.

En el diseño original se detallan todos los perfiles hipotéticos (perfiles completos). Sin embargo, su número suele ser alto y se hace uso de técnicas de investigación que indiquen los más significativos. Está limitado a un análisis con muy pocos atributos y niveles.

El análisis conjunto adaptativo o *Adaptative Conjoint Analysis (ACA)*

La recolección de datos se hace en dos fases: en la primera, el encuestado señala la importancia para él de cada atributo, y en la segunda, asigna utilidades a un número limitado de perfiles. Es decir, se le conduce a través de una investigación sistemática por diferentes secciones, en las que se le presentan uno o pocos atributos, y que se va adaptando a sus respuestas. Se obtienen las valoraciones de los niveles de interés (utilidades parciales). Está limitado por su complejidad y por el uso de software especializado, aunque permite un gran número de atributos y niveles.

El análisis conjunto basado en elecciones o *Choice-Based Conjoint (CBC)*

Pretende un mayor realismo, mostrando a cada entrevistado un grupo de productos distintos y se le pide que elija cuál de ellos prefiere. Además, contempla la posibilidad de no elegir ninguna alternativa. Mediante un modelo de regresión no lineal se calculan las utilidades de los atributos-niveles, conocido como modelo de elección discreta. Como inconvenientes se pueden señalar que tiene una mayor complejidad en su diseño y análisis, que requiere de muestras muy grandes para tener validez estadística y que el número máximo de atributos que admite es 10. Su mayor ventaja es que presenta un mejor sistema de elección que las alternativas anteriores.

4.3. Etapas de la realización del análisis conjunto

Para la aplicación correcta de un análisis conjunto, es conveniente seguir una serie de etapas, que facilitarán las elecciones metodológicas que han de hacerse y la interpretación de los resultados.

Planteamiento del problema. Se analiza la oportunidad de aplicar esta técnica dadas el objetivo y los datos del proyecto. Para ello, se debe especificar tanto el producto o servicio como los atributos que se quieren estudiar.

Elección de la metodología conjoint a aplicar. Ésta dependerá, básicamente, de las características y cantidad de atributos estudiados; también se debe considerar la forma en que se valorarán por parte de los individuos. A modo de guía de elección, se optaría por cada metodología cuando:

- CVA: se analizan hasta 6 atributos, siendo los productos o servicios habituales, de forma que se permita una elección rápida, sin demasiada reflexión.
- ACA: se analizan más de 10 atributos, procediendo a una elección rápida pero reflexiva, ya que se trata de un proceso adaptativo cuyo resultado final depende, en buena medida, de la idoneidad de las primeras valoraciones.

- CBC: se analizan entre 6 y 10 atributos, con elecciones reflexivas sobre productos, puesto que el individuo no asigna directamente utilidades, sino que se limita a elegir una (o ninguna) de las opciones presentadas.

La selección de elementos. Se escogerán sólo los **atributos** que condicionan la elección, es decir, los que expliquen las preferencias de los individuos y permitan diferenciar bien entre los productos o servicios; deben estar lo más cercanos posible a la independencia entre ellos. Para cada atributo se eligen sus **niveles**; deben ser mutuamente excluyentes y cubrir todo el rango de posibilidades. Por último, y dependiendo de la metodología utilizada, se determinan los **perfíles y escenarios**.

Creación de estímulos. Se utilizarán diseños factoriales fraccionados ortogonales, que reducen el número de perfíles que se le muestran al entrevistado. Como el número de posibles perfíles es la multiplicación del número de niveles de todos los atributos, puede ser imposible en la práctica que el individuo indique sus preferencias entre todos ellos. Por ello, se seleccionan sólo algunos de los perfíles, que sean representativos del resto, es decir, que en los perfíles incluidos en los estímulos aparezca cada nivel de cada factor combinado con el resto de niveles de forma lo más proporcional posible.¹ La selección se efectúa mediante diseño factorial de experimentos, fraccionado (que jerarquiza los efectos, permitiendo la reducción de perfíles) y ortogonal (equilibrado en los niveles).

Forma de presentación. Dependerá de la metodología elegida. La Fig. 4.2 muestra algunos ejemplos.

- CVA: *Matriz de comparaciones o trade-off*, en la que el entrevistado valora la combinación de atributos y niveles (sólo es válida para dos atributos). *Perfiles completos para ordenar*, donde se elaboran perfíles de cada producto o servicio utilizando sólo un nivel de cada atributo y el encuestado los valora (ordena) según sus preferencias. *Perfil determinado para valorar*, combinación que el encuestado valora según sus preferencias.
- ACA: *Comparaciones pareadas* en la que se comparan dos perfíles incompletos.
- CBC: *Elección de un perfil* en el que los encuestados señalan el perfil preferido entre el subconjunto que se les muestra, sin valorarlos ni ordenarlos.

Trabajo de campo y tratamiento de los datos

- CVA: la recogida de datos es en papel u ordenador, y el análisis en ordenador con software especializado.
- ACA y CBC: la recogida y el análisis son con ordenador.

¹Por ejemplo, si se consideran 4 atributos con 3 niveles cada uno, habría un total de 81 perfíles diferentes. Se elige un número menor de perfíles (suele ser habitual considerar como mínimo el número total de niveles menos el de atributos más uno; en este ejemplo $4 \times 3 - 4 + 1 = 9$ perfíles) y se busca que en ellos aparezca, aproximadamente, el mismo número de veces cada nivel de cada atributo.

4.3. Etapas de la realización del análisis conjunto

79

Matriz Trade-off o de comparaciones			
		Forma	
PAN		Pan redondo	Barra
Tipo	Artesano	6	4
	Industrial	2	5
	Molde	3	1

Tarjetas perfiles-valoración							
Alternativa A Artesano Pan redondo 125 gr	Alternativa B Artesano Pan redondo 250 gr	Alternativa C Artesano Barra 125 gr	Alternativa D Artesano Barra 250 gr				
Alternativa E Artesano Molde 125 gr	Alternativa F Artesano Molde 250 gr	Alternativa G Industrial Pan redondo 125 gr	Alternativa H Industrial Pan redondo 250 gr				
Alternativa I Industrial Barra 125 gr	Alternativa J Industrial Barra 250 gr	Alternativa K Industrial Molde 125 gr	Alternativa L Industrial Molde 250 gr				

Matriz Trade-off o de comparaciones				
¿Valore de 1 a 5 su gusto por este pan?				
Artesano Pan redondo 125 gr				
1	2	3	4	5
Nada	Poco	Normal	Bastante	Mucho

Comparaciones pareadas							
¿Valore de 1 a 5 estas dos opciones de pan?							
							
Opción A	Artesano Pan redondo 125 gr	Opción B	Industrial Barra 250 gr				
Prefiero opción A	Indiferencia	Prefiero opción B					
1	2	3	4				
5							

Elección de un perfil entre varios				
Alternativa A Artesano Pan redondo 125 gr	Alternativa B Artesano Barra 250 gr	Alternativa C Industrial Pan redondo 250 gr		
1	2	3		
Alternativa D Industrial Barra 250 gr	Alternativa D No me gusta ninguna opción			
4	5			

Figura 4.2: formas de presentación

Estimación de las utilidades

CVA: se utiliza, por lo general, un modelo regresión por mínimos cuadrados ordinarios (OLS), mediante el cual se determinan las utilidades parciales o *partworth*, que indican las preferencias del encuestado mediante un modelo aditivo lineal en relación a los niveles de los atributos considerados como referencia.

Considerando la variable X_{jk}^i para indicar si el nivel k del atributo j está en el perfil i , ésta tomará sólo dos posibles valores $X_{jk}^i = 1$ (si está) o $X_{jk}^i = 0$ si no lo está. Denominando Y^i la preferencia que tiene un individuo sobre el perfil i , la función de utilidad que se debería estimar es:

$$Y^i = \beta_0 + \sum_{j,k} U_{jk} X_{jk}^i \quad (4.1)$$

donde los coeficientes U_{jk} forman el vector de utilidades parciales, o *partial partworth*, indicando la utilidad que el individuo asigna a cada nivel de cada atributo.

Si se estima directamente la ecuación (4.1), habrá multicolinealidad, ya que todos los atributos deben estar presentes en todos los perfiles, es decir $\sum_k X_{jk}^i = 1$. Para introducir estas restricciones en la función de utilidad inicial, se elimina uno de los niveles de cada factor (sin pérdida de generalidad, el primero), por lo que se estima por mínimos cuadrados ordinarios la función de utilidad restringida:

$$Y^i = \delta + \sum_{j,k,k \neq 1} \gamma_{jk} X_{jk}^i \quad (4.2)$$

donde $\delta = \beta_0 + \sum_j U_{j1}$ y $\gamma_{jk} = U_{jk} - U_{j1}$. Utilizando estas dos ecuaciones tras la estimación de la ecuación (4.2), se pueden calcular los valores de la función de utilidad original (4.1). ²

Si hay Z individuos, cada uno de ellos, z , presentará una valoración diferente de los perfiles i , es decir valores diferentes de Y_z^i , produciendo una estimación diferente de la función de utilidad (4.1) y, por tanto, diferentes utilidades parciales $\widehat{U}_{jk,z}$. Para obtener las utilidades para el conjunto de individuos, se procede al cálculo de sus valores medios, por lo que serán:

$$\widehat{U}_{jk} = \frac{\sum_z \widehat{U}_{jk,z}}{Z} \quad (4.3)$$

Se obtienen, así, las **utilidades** de cada nivel k de cada atributo j , reflejando la importancia que conceden los individuos a cada uno de esos niveles.

Para determinar la importancia de cada atributo j se utiliza la diferencia entre la utilidad más alta y más baja de sus niveles, es decir,

²Los valores U_{j1} se pueden calcular imponiendo que $\sum_k U_{jk} = 1$ para cada atributo j . Menos frecuente, aunque también válido, es expresar la utilidades relativas a un nivel de referencia (en este caso, el primero) por lo que se consideraría cada $U_{j1} = 0$.

4.3. Etapas de la realización del análisis conjunto

81

$$Imp_j = \max_k \{U_{jk}\} - \min_k \{U_{jk}\} \quad (4.4)$$

En términos relativos, la importancia de cada atributo j respecto al conjunto de atributos, se puede expresar como:

$$ImpRel_j = \frac{Imp_j}{\sum_{t=1}^J Imp_t} \quad (4.5)$$

Por último, con las estimaciones de las utilidades, la *utilidad total* de un perfil i es la suma de las utilidades de los niveles de cada atributo que lo definen:³

$$\hat{U}_i = \beta_0 + \sum_{j=1}^J \hat{U}_{jk} \text{ para los niveles k presentes en el perfil} \quad (4.6)$$

donde: \hat{U}_i es la utilidad total del perfil i ; \hat{U}_{jk} es la utilidad parcial asociada con el nivel k del atributo j ; y β_0 es la constante de regresión.

En los casos de las metodologías **ACA** y **CBC**, la estimación de las utilidades parciales es más compleja, debiendo recurrir a modelos de regresión no lineal. Sin embargo, su interpretación es la misma que en el caso de la metodología **CVA**.

Interpretación de resultados. Algunos de los resultados del análisis conjunto que frecuentemente se analizan son:

- La importancia de los atributos y niveles. La serie de importancias relativas de cada atributo ($ImpRel_j$) reflejan la opinión de los individuos sobre la relevancia (en porcentaje) de cada atributo para evaluar los perfiles. Igualmente, para cada atributo j , las utilidades de sus niveles U_{jk} muestran la importancia relativa asignada a dicho nivel dentro de la valoración de cada atributo.
- Las utilidades totales de cada perfil permiten analizar cuáles son los preferidos por los individuos, o la comparación relativa entre un conjunto determinado de perfiles. Esta información puede orientar en la determinación de la configuración final del producto o servicio analizado.
- La influencia de un determinado atributo sobre la elección de los individuos. Así, sería posible estimar los cambios de utilidad que se producirían por la modificación de sus niveles (generalmente conocida como *elasticidad*); por ejemplo, si un atributo fuese el precio, se podría dar una aproximación a la variación de utilidad producida por un incremento del precio.

³Si se desea obtener la utilidad total del perfil i para un individuo concreto, z , bastaría con utilizar la (4.6) para los valores concretos de β_{0z} y $U_{ij,z}$.

- Al disponer de las utilidades parciales de cada individuo, es posible estudiar si éstas son homogéneas para toda la muestra o, por el contrario, si existen *grupos* de individuos con utilidades parecidas entre ellos y diferenciadas del resto. Este planteamiento podría ayudar a segmentar el mercado y a ofrecer productos o servicios diferenciados a cada nicho de mercado.

4.4. Procedimiento con R: la función Conjoint()

Para exemplificar el uso del análisis conjunto, se hará uso del paquete `conjoint` y de su base de datos `tea`, compuesta por 5 listados: niveles (`tlevn`), perfiles (`tprof`), preferencias (`tpref`), preferencias en forma matricial (`tprefm`), y la simulación de perfiles (`tsimp`).

El listado `tprof` recoge los 13 perfiles que se presentan a los individuos, cada uno de ellos con una combinación de niveles de cada factor. El hecho de utilizar un diseño factorial fraccionario ortogonal ha permitido encontrar una muestra de perfiles *representativa* del total (habría $3 \times 3 \times 3 \times 2 = 54$ perfiles distintos) donde cada nivel de cada atributo está representado, aproximadamente, de forma proporcional.

La matriz `tprefm` contiene las preferencias que cada uno de los 100 individuos asigna a cada uno de los perfiles i que se le presentan (estas preferencias se recogen ordenadas en el vector `tpref`). Por último, `tsimp` muestra cuatro simulaciones de perfiles distintos a los mostrados a los individuos.

En este ejemplo, se desea conocer el por qué unos téns son preferidos a otros. Para ello, en el contexto de un análisis tradicional con perfiles incompletos, se seleccionan 4 atributos, con sus correspondiente niveles (`tlevn`):

- Precio, distinguiendo entre bajo, medio y alto.
- Variedad, distinguiendo entre negro, verde y rojo.
- Forma de presentación, distinguiendo entre bolsita, granulado y hojas.
- Aroma, distinguiendo entre aromático y no aromático.

Del total de 54 posibles combinaciones se seleccionan 13 perfiles, valorados de 0 a 10 según las preferencias de cada uno de los 100 encuestados.

Para realizar el análisis conjunto propuesto, primeramente se carga tanto el paquete como los datos:

```
library("conjoint")
data("tea")
```

Con la función `Conjoint()` devolverá las utilidades de los niveles (*partworth*), el vector de porcentajes de la importancia de los atributos, y las gráficas correspondientes.

4.4. Procedimiento con **R**: la función *Conjoint()*

83

```
Conjoint(y=tpref, x=tprof, z=tlevn)
#> Call:
#>
#> lm(formula = frml)#
#> Residuals:#>      Min       1Q   Median      3Q     Max#> -5,1888 -2,3761 -0,7512
#>  2,2128  7,5134#>#> Coefficients:#>                               Estimate Std. Error t value
#> (Intercept)            3,55336    0,09068 39,184 < 2e-16 ***#>
#> factor(x$price)1      0,24023    0,13245  1,814   0,070 .#> factor(x$price)2
#> -0,14311   0,11485 -1,246   0,213#> factor(x$variety)1  0,61489    0,11485
#>  5,354 1,02e-07 ***#> factor(x$variety)2  0,03489    0,11485   0,304   0,761
```

En la Tabla 4.4 se exponen otras funciones disponibles para obtener algunos resultados concretos:

Tabla 4.1: Funciones en **R** para cálculos sobre análisis conjunto

Función en R	Utilidad Resultado
caUtilities(y=tprefm , x=tprof, z=tlevn)	Utilidades medias para cada nivel
caPartUtilities(y=tprefm , x=tprof, z=tlevn)	Matriz de utilidades de los niveles para cada individuo
caTotalUtilities(y=tprefm , x=tprof)	Utilidades totales de los perfiles mostrados para cada individuo
colMeans(caTotalUtilities(y=tprefm , x=tprof))	Utilidades totales medias para cada perfil
ShowAllUtilities(y=tprefm , x=tprof, z=tlevn)	Todas las utilidades
caImportance(y=tprefm , x=tprof)	Importancia relativa media de cada atributo
caModel(y=tprefm[20,], x=tprof)	Utilidades para un solo individuo, por ejemplo el 20.

Interpretación del resultado:

- La primera parte de la salida corresponde a Los coeficientes del modelo de regresión (4.2), donde no se incluye el último nivel de cada atributo (para evitar la multicolinealidad); también se señala con asteriscos los que resultan estadísticamente significativos y el grado de ajuste del modelo. A continuación, están las utilidades parciales para todos los niveles de todos los atributos.
- La última parte de la salida recoge la importancia relativa de cada atributo, medida en porcentaje.

En este ejemplo el atributo con más peso es la **variedad de té**, con una importancia relativa del 32.22 %; el siguiente es la **presentación**, con un 27.15 %; después, el **precio**, con un 24.76 %; y por último, el **aroma**, con un 15.88 %. Es decir, el conjunto de 100 individuos, al valorar los perfiles que se les presentan, consideran principalmente el atributo variedad, seguido de forma

de presentación y precio; por el contrario, que sea aromático o no aporta relativamente poca valoración.

Analizando las utilidades de cada atributo (*parthworth*) se puede señalar que:

- Para el atributo **precio**, es el nivel bajo (*low*) el que recibe mayor preferencia (0.2402), siendo el nivel medio (*medium*) el menos preferido (-0.1431), algo por debajo de la preferencia del nivel alto (*high*), -0.0971.
- Para el atributo **variedad**, el té negro (*black*) es el nivel con mayor utilidad (0.6149), seguido del té verde (0.0349), quedando el nivel de té rojo con una preferencia mucho menor (-0.6498).
- En el caso del atributo **presentación**, en hojas (*leafy*) es el nivel más preferido (0.7529), seguido de la modalidad en bolsitas (*bags*) con una utilidad de 0.1369; por último, el nivel granulado presenta la preferencia más baja (-0.8898).
- Por último, para el atributo **aroma**, que sea aromático (*yes*) es el nivel más preferido, con una utilidad de 0.4108; al ser un atributo con solo dos niveles, la preferencia del otro nivel (*no*) es -0.4108.

También es posible obtener sólo la información relativa a las utilidades, mediante la función `caUtilities()`, o sólo la correspondiente a la importancia de los atributos, usando la función `caImportance()`, como se recoge en la Tabla .

De forma adicional, el análisis conjunto también permite profundizar en el conocimiento de los individuos en función de sus preferencias. Como ejemplo, se puede abordar si existen *grupos* de individuos con preferencias similares entre ellos y diferencias de las del resto de grupos, cuestión conocida generalmente como *segmentación*. En este caso, el objetivo del proyecto sería identificar grupos de encuestados con preferencias similares, segmentando el mercado y permitiendo adaptar los atributos de cada producto o servicio a las características concretas de ese nicho de mercado.

Para ello, se puede utilizar la función `caSegmentation()`, que devuelve un análisis de conglomerados dividiendo a los individuos en *n clusters*, usando el método *k-means* (véase Sec. 13). Como ejemplo, se consideran tres clusters.

El vector de agrupación resume las características de los tres grupos formados.

```
segments<-caSegmentation(y=tperf, x=tprof, c=3)
segments$segm
#> K-means clustering with 3 clusters of sizes 40, 40, 20
#>
#> Cluster means:
#>      [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
#> 1 5.480275 2.9381 1.3681 4.540275 1.9731 3.782900 1.382900 0.965750 2.820750
#> 2 4.754975 4.6918 3.6718 6.964975 6.6918 3.500525 4.385525 2.717225 3.062225
#> 3 2.623500 6.6211 4.7511 2.933500 2.5211 4.189050 4.549050 5.066950 2.086950
#>      [,10]  [,11]  [,12]  [,13]
```

4.4. Procedimiento con **R**: la función *Conjoint()*

85

```
#> 1 0.111225 3.450750 0.442900 0.692900
#> 2 1.840925 6.292225 6.595525 7.105525
#> 3 5.312100 4.266950 4.859050 3.569050
#>
#> Clustering vector:
#> [1] 2 3 2 2 2 1 2 3 2 2 2 2 1 1 1 1 3 1 3 1 1 2 1 2 3 2 3 3 2 2 1 2 3 2 2 2 2
#> [38] 1 1 1 1 3 1 3 1 2 2 1 1 1 2 1 1 2 2 1 2 1 3 1 1 2 3 3 3 2 1 1 1 2 2 1 2 3
#> [75] 2 2 2 1 2 2 3 3 2 2 1 1 1 3 1 3 1 3 1 1 2 1 3 2 2
#>
#> Within cluster sum of squares by cluster:
#> [1] 1605.654 2690.267 1131.293
#> (between_SS / total_SS = 41.4%)
#>
#> Available components:
#>
#> [1] "cluster"      "centers"       "totss"        "withinss"     "tot.withinss"
#> [6] "betweenss"    "size"          "iter"         "ifault"
```

La salida contiene las utilidades medias de cada nivel para cada uno de los tres grupos, mostrando las diferencias entre ellas.

Resumen

El **análisis conjunto** estudia situaciones de elección múltiple. Divide un producto o servicio en atributos y niveles y analiza las utilidades parciales de cada uno; después, se realizan diferentes combinaciones de éstos para identificar las preferencias del consumidor, estableciendo qué atributos y niveles son los más valorados y cuantificarlos de forma relativa. Permite evaluar las preferencias del público ante el lanzamiento de nuevos productos o su sensibilidad de alguna característica, como el precio, el formato, cambios en la imagen del producto, etc. Función principal de uso en **R**: `conjoint()`.

Capítulo 5

Análisis de tablas de contingencia

José-María Montero

Universidad de Castilla-La Mancha

5.1. Introducción

Las **tablas de contingencia** analizan la relación existente entre variables categóricas, o susceptibles de categorizar, con un número de categorías finito. Dada su naturaleza, no permiten el uso de las tradicionales operaciones aritméticas, con lo cual, en el ámbito de la Estadística Descriptiva su análisis suele basarse en diagramas de barras y porcentajes (véase Cap. ??), y en la esfera de la Inferencia Estadística (Cap. ??) se centra en los contrastes de hipótesis no paramétricos y, básicamente, en el contraste de independencia entre dos o más de estas variables. Una pregunta que suele hacerse toda aquella persona que se acerca por primera vez al análisis de tablas de contingencia es el significado del término “contingencia”. Pues bien, este término fue acuñado por Pearson ([Pearson, 1904](#)) al apuntar: “... Este resultado nos permite partir de la teoría matemática de la independencia probabilística, tal como se desarrolla en los libros de texto elementales, y construir a partir de ella una teoría generalizada de la **asociación** o, como yo la llamo, **contingencia**.”

El análisis de tablas de contingencia (o de asociación) permite dar respuesta, entre otras, a preguntas como: los factores involucrados en una tabla de contingencia, ¿son independientes o están asociados? Si están asociados, ¿qué niveles de dichos factores son los que están asociados?, ¿cuál es la intensidad de dicha asociación?

5.1.1. Notación

Sea una población (o una muestra) de N elementos sobre la que se pretende analizar, simultáneamente, dos (por simplicidad) *atributos* o *factores* (A y B) con R y C *niveles*, *modalidades* o

categorías, respectivamente. Sean $\{A_1, A_2, \dots, A_R\}$ y $\{B_1, B_2, \dots, B_C\}$ los niveles anteriormente aludidos. Sea n_{ij} el número de elementos que presentan a la vez las modalidades i y j de los factores A y B , respectivamente. La tabla estadística que describe, conjuntamente, estos N elementos (en otros términos, que muestra las frecuencias conjuntas de los niveles de ambos factores) se denomina **tabla de contingencia**.

		Factor B				
		Nivel B_1	Nivel B_2	...	Nivel B_C	Total
Nivel A_1		n_{11}	n_{12}	...	n_{1C}	$n_{1.}$
Nivel A_2		n_{21}	n_{22}	...	n_{2C}	$n_{2.}$
.	
Factor A	
.	
Nivel A_R		n_{R1}	n_{R2}	...	n_{RC}	$n_{R.}$
Total		$n_{.1}$	$n_{.2}$...	$n_{.C}$	N

A modo de ejemplo, considérese una muestra de 80 ayuntamientos de una CC.AA., anotándose en la base `ayuntam`, incluida en el paquete CRD del libro, el signo político del equipo gubernamental (`signo_gob`) y si prestan o no públicamente el servicio X (`serv`). Los resultados obtenidos fueron los siguientes:

```
library("CDR")
data("ayuntam")

summarytools::ctable(ayuntam$signo_gob, ayuntam$serv, headings = TRUE) |>
  print()
#> Cross-Tabulation, Row Proportions
#> signo_gob * serv
#> Data Frame: ayuntam
#>
#> -----
#>           serv      No       Sí     Total
#>   signo_gob
#>   Avanzados      14 (33.3%)  28 (66.7%)  42 (100.0%)
#>   Ilustrados       6 (15.8%)  32 (84.2%)  38 (100.0%)
#>   Total          20 (25.0%)  60 (75.0%)  80 (100.0%)
#> -----
```

Del estudio de las distribuciones marginales de ambos factores se deduce que el 52,5 % de los ayuntamientos de la CC.AA. están regidos por los Avanzados y el 47,5 % por los Ilustrados. Y, más interesante, que en el 75 % de los ayuntamientos prestan el servicio X .

El análisis de la tabla de contingencia daría respuesta a las siguientes preguntas: ¿La prestación pública del servicio X es independiente del signo político del ayuntamiento o depende de dicho signo? En este último caso: ¿Qué signo político está asociado con la prestación pública y cuál

5.1. Introducción

89

no?, ¿la asociación entre los factores “Signo político del equipo gubernamental” y “Prestación pública del servicio X ” es muy intensa? Pero dicho análisis se abordará posteriormente.

En función del número de factores involucrados en la tabla y del número de niveles de cada uno de ellos se tiene la siguiente tipología de tablas de contingencia:

- Tablas $R \times C$: 2 factores, el primero con R niveles y el segundo con C niveles.
- Tablas $R \times C \times M$: 3 factores, con R , C y M niveles, respectivamente.
- Y así sucesivamente.

Dentro de las tablas $R \times C$ se distinguen las tablas 2×2 de las demás, por su especial interés en la realidad y por criterios pedagógicos, al ser las más sencillas.

5.1.2. Diseños experimentales o procedimientos de muestreo que dan lugar a una tabla de contingencia

Una cuestión a la que no se le da la suficiente importancia es la forma en la que se toma la información contenida en la tabla (el diseño del experimento o procedimiento de muestreo). Dada una determinada tabla de contingencia, ésta puede haber sido obtenida mediante uno u otro diseño de experimento o procedimiento de muestreo, y esta circunstancia no es baladí, puesto que condiciona su análisis, sobre todo cuando el tamaño muestral es pequeño.

Sin ánimo de exhaustividad, los diseños experimentales o procedimientos de muestreo más habituales que dan lugar a una tabla de contingencia son los siguientes¹:

- **Tipo 1: se fijan los totales marginales de ambos factores**

Ejemplo: se desea investigar si la preferencia de la larva de gorgojo por el tipo de judía es independiente de la cubierta de la semilla o depende de ésta. Para ello se toman 22 judías de tipo A y 18 de tipo B , que se introducen en un recipiente con 33 larvas. Dadas las condiciones de densidad, no entrará más de una larva por judía. Pasado un tiempo prudencial, para que las larvas entren en las judías se hace el recuento de las que han sido atacadas de cada tipo y las que no.

		Presencia de larva	atacante		Total
			NO	SÍ	
Tipo de judía	A	N_{11}	N_{12}	22	
	B	N_{21}	N_{22}	18	
	Total	7	33	40	

Como puede apreciarse, los totales marginales de ambos factores han sido fijados en el diseño del experimento.

¹Otros procedimientos de muestreo o diseños experimentales no habituales pueden verse en Ruiz-Maya et al. (1995).

■ **Tipo 2:** sólo se fijan los totales marginales de uno de los factores

Ejemplo: en un municipio se desea investigar si el desempleo es o no independiente del sexo del desempleado. Se seleccionan aleatoriamente 100 varones y 100 mujeres y se les pregunta por su situación laboral (trabajando; en paro).

		Situación laboral		
		Trabajando	En paro	Total
Sexo	Varón	N_{11}	N_{12}	100
	Mujer	N_{21}	N_{22}	100
	Total	$N_{.1}$	$N_{.2}$	200

■ **Tipo 3:** únicamente se fija el tamaño muestral

Ejemplo: un estudio transversal sobre la prevalencia de osteoporosis y su relación con dietas pobres en calcio incluyó a 400 mujeres entre 50 y 54 años. Cada una de ellas realizó una densíometría de columna y llenó un cuestionario sobre sus antecedentes dietéticos para determinar si su dieta era o no pobre en calcio.

		Dieta pobre en calcio		
		SI	NO	Total
Osteoporosis	SI	N_{11}	N_{12}	$N_{1.}$
	NO	N_{21}	N_{22}	$N_{2.}$
	Total	$N_{.1}$	$N_{.2}$	400

5.2. Contraste de independencia en tablas 2×2

Como se avanzó en la Sec. 5.1, la primera pregunta a la que debe dar respuesta el análisis de tablas de contingencia es si los factores involucrados en la tabla son independientes o, por el contrario, están asociados. La respuesta a esta pregunta exige llevar a cabo un contraste de independencia y, para ilustrarlo, se aborda, inicialmente, el caso de las tablas 2×2 . Dicho contraste se llevará a cabo de tres formas: (i) exacta, (ii) aproximada, y (iii) aproximada con corrección de continuidad.

5.2.1. Planteamiento general del contraste exacto de independencia

- Hipótesis: H_0 : los factores son independientes. H_1 : están asociados.²

²También puede establecerse como hipótesis alternativa la asociación en un determinado sentido: “el nivel 1 del factor A está asociado con el 1 del B y el 2 del A con el 2 del B” (asociación positiva); o “el nivel 1 del factor A está asociado con el 2 del B y el 2 del A con el 1 del B” (asociación negativa). En estos dos casos el contraste no sería bilateral sino unilateral.

- Filosofía del contraste: se trata de un contraste de significación. Por tanto, la tabla observada será “rara” (bajo H_0) si su probabilidad, más la probabilidad de obtener tablas más alejadas de H_0 que ella, es inferior al nivel de significación, α , prefijado para el contraste. En ese caso, se rechaza la hipótesis de independencia entre los factores involucrados en la tabla.

5.2.2. Algoritmo para la realización del contraste exacto de independencia

De acuerdo con la filosofía de los contrastes de significación (Sec. ??), el algoritmo para la realización del contraste de independencia en tablas de contingencia es como sigue:

1. Selección de las tablas del espacio muestral que se alejen de la hipótesis de independencia, en la dirección marcada por la hipótesis alternativa, tanto o más que la tabla observada, incluida esta última.
2. Cálculo, bajo la hipótesis de independencia, de la probabilidad de ocurrencia de cada una de las tablas seleccionadas en el punto 1.
3. Suma de dichas probabilidades y comparación con el α prefijado.
4. Toma de la decisión relativa al rechazo o no de la hipótesis de independencia.

Nótese que (i) los pasos 1 y 2 dependen del diseño del experimento o procedimiento de muestreo llevado a cabo; (ii) en ausencia del software adecuado, la realización de un test exacto es un procedimiento laborioso (a veces un reto), con lo cual, si ese fuera el caso, los test aproximados de independencia son bienvenidos.

A continuación, se expone el contraste de independencia, en sus versiones exacta, aproximada y aproximada con corrección de continuidad, cuando el procedimiento de muestreo o diseño experimental es el de tipo 1. En la Sec. 5.2.4 se comentan algunas cuestiones de interés cuando el diseño de muestreo es de tipo 2 o tipo 3.

5.2.3. Contraste de independencia: diseño tipo 1

5.2.3.1. Contraste exacto (test exacto de Fisher)

Considérese el ejemplo del diseño tipo 1 expuesto en 5.1.2³. Supóngase que el resultado obtenido fue el siguiente:

```
datos_jud = c(1,6,21,12)
tabla = cbind(expand.grid(list(Tipo_de_judía = c("A","B"),
                                Presencia_larva = c("No","Sí"))),
              count = datos_jud)
tabla_jud <- ftable(xtabs(count~Tipo_de_judía+Presencia_larva, tabla))
```

³Se trata de un ejemplo clásico de Sokal and Rolf (2012).

Tipo de judía		Presencia de larva		Total
		NO	SI	
	A	1	21	22
	B	6	12	18
	Total	7	33	40

Según el algoritmo expuesto en la Sec. 5.2.2, el contraste es como sigue:

1. Selección de las tablas que se alejan de H_0 tanto o más que la observada.⁴ Como se señalaba en Pearson (1904), la teoría de la independencia probabilística indica que, bajo la hipótesis de independencia, el porcentaje de judías de tipo A y de tipo B no atacadas (o atacadas) por una larva de gorgojo tiene que ser el mismo. En otros términos, bajo la hipótesis de independencia, en cada una de las cuatro celdas se tiene que verificar que: $n_{ij} = \frac{n_{i\cdot}n_{\cdot j}}{N}, \forall i, j$, donde $\frac{n_{i\cdot}n_{\cdot j}}{N} = \hat{E}_{ij}$ se denomina estimación de la frecuencia esperada bajo la hipótesis de independencia. Se puede que comprobar que en una tabla 2x2, $\hat{D}_{11} = \hat{D}_{22} = -\hat{D}_{12} = -\hat{D}_{21}$, con lo cual las tablas que se alejan tanto o más que la observada de la hipótesis de independencia son aquellas que verifican, en valor absoluto, y tomando de referencia, por ejemplo, la celda {1,1}: $\hat{D}_{11} = n_{11} - \hat{E}_{11} = n_{11} - \frac{n_{1\cdot}n_{\cdot 1}}{N} \geq \hat{D}_{11obs}$.

En el ejemplo que se considera, las \hat{D}_{11} son las siguientes (en negrita las de la tabla observada y aquellas otras que se alejan tanto o más que ella de H_0):

$$T_0: -3,85; T_1: -2,85; T_2: -1,85; T_3: -0,85; T_4: 0,15; T_5: 1,15; T_6: 2,15; T_7: 3,15,$$

donde el subíndice de T indica el valor de n_{11} en dicha tabla.

Nótese que el criterio anterior no es otro que el criterio general de seleccionar las tablas en las que la diferencia de porcentajes, por ejemplo, por fila, en valor absoluto, sea superior a la de la tabla observada, puesto que $\left| \frac{n_{11}}{n_{1\cdot}} - \frac{n_{21}}{n_{2\cdot}} \right| = |\hat{D}_{11}| \frac{N}{n_{1\cdot}n_{2\cdot}}$.

2. Cálculo, bajo la hipótesis de independencia, de la probabilidad de ocurrencia de cada una de las tablas seleccionadas en 1. La probabilidad de ocurrencia de una tabla de contingencia con los totales marginales fijos se puede obtener como el cociente entre el número de disposiciones de las frecuencias observadas favorables a dicha tabla y el número de disposiciones posibles. El número de disposiciones favorables coincide con el coeficiente multinomial (maneras de que de n frecuencias, n_{11} caigan en la celda {1,1}, n_{12} lo hagan en la celda {1,2}, n_{21} lo hagan en la celda {2,1} y n_{22} lo hagan en la celda {2,2}): $\frac{n!}{n_{11}!n_{12}!n_{21}!n_{22}!}$.

El número de disposiciones posibles, supuesta H_0 , es: $\binom{n}{n_{1\cdot}} \binom{n}{n_{\cdot 1}} = \frac{n!}{(n_{1\cdot}!n_{2\cdot}!)^2} \frac{n!}{(n_{\cdot 1}!n_{\cdot 2})^2}$.

Por tanto, el cociente entre ambas es: $P = \frac{n_{11}!n_{12}!n_{21}!n_{22}!}{n!n_{11}!n_{12}!n_{21}!n_{22}!}$.

En consecuencia, las probabilidades de las tablas seleccionadas en 3 son: $T_0 : 0,0017; T_1 : 0,0219; T_7 : 0,0091$.

⁴El contraste se ha llevado a cabo de forma bilateral. En caso de una alternativa unilateral (asociación positiva o en el sentido de la diagonal principal; o negativa, en el sentido de la diagonal no principal), el procedimiento sería el mismo y las tablas seleccionadas serían, en el primer caso, todas menos la T_7 , y en el segundo, T_0 y T_1 .

3. Suma de dichas probabilidades: 0,0327.

4. Comparación con α y decisión sobre el rechazo o no de la hipótesis de independencia: La decisión depende del valor de α . Si fuera, por ejemplo, 0,05, se rechazaría la independencia entre el tipo de judía y si es o no atacada por la larva de gorgojo.

El código R necesario para tomar llevar a cabo el test exacto de Fisher anterior es: {contraste de independencia!en tablas 2x2!test exacto de Fisher}

```
#Ho: Los factores son independientes.
#H1: Los factores están asociados
fisher <- fisher.test(tabla_jud, alternative = "two.sided")
fisher$p.value
#> [1] 0.0327607
```

```
#Ho: Los factores son independientes.
#H1: Existe asociación negativa.
fisher_less <- fisher.test(tabla_jud, alternative = "less")
fisher_less$p.value
#> [1] 0.02361309
```

```
#Ho: Los factores son independientes.
#H1: Existe asociación positiva.
fisher_greater <- fisher.test(tabla_jud, alternative = "greater")
fisher_greater$p.value
#> [1] 0.998293
```

Como puede apreciarse, se rechaza la hipótesis de independencia frente a la de asociación (test bilateral). Esto no significa que las larvas ataquen a un tipo de judía y no al otro. Atacan a ambos tipos, ¡y bastante! Este es el primer hecho que se constata. Sin embargo, atacan más a las judías de tipo A (un 95 % son atacadas) que a las de tipo B (dos terceras partes son atacadas). Esa diferencia porcentual de judías atacadas se considera significativa bajo el supuesto de independencia y, en ese sentido, se dice que existe asociación entre el tipo de judía y la presencia o no de larva atacante. La asociación sería A-SI y B-NO. Sin embargo, ¡cuidado!, las larvas atacan siempre. La asociación anterior debe entenderse como “el porcentaje de ataque es muy grande en ambos casos, pero en A (mucho) más que en B. Este es el segundo hecho importante que se constata: las larvas muestran una preferencia significativa por las judías tipo A. Aunque ya se ha visto la dirección de la asociación (en el sentido de la diagonal ascendente), en la Sec. 5.4, dedicada a las medidas de asociación en tablas 2x2, se cuantificará su intensidad.

5.2.3.2. Contraste aproximado

En este caso (tipo 1), bajo H_0 : independencia, la frecuencia conjunta de una celda, N_{ij} , cualquiera que sea, se distribuye según una ley hipergeométrica con $E(N_{ij}) = \frac{N_{ij}}{n_i \cdot n_j}$ y $V(N_{ij}) =$

$\frac{n_{i..}n_{.j}(n-n_{i.})(n-n_{.j})}{n^2(n-1)}$. Por consiguiente,

$$P\left(\left(N_{11} - \frac{n_{1..}n_{.1}}{n}\right)^2 \geq \left(n_{11} - \frac{n_{1..}n_{.1}}{n}\right)^2\right) = P\left(\frac{(N_{11} - \frac{n_{1..}n_{.1}}{n})^2}{\frac{n_{1..}n_{.1}n_{2..}n_{.2}}{n^2(n-1)}} \geq \frac{(n_{11} - \frac{n_{1..}n_{.1}}{n})^2}{\frac{n_{1..}n_{.1}n_{2..}n_{.2}}{n^2(n-1)}}\right)$$

Y si ninguna $\hat{E}_{ij} = \frac{n_{ij}}{n_{i..}n_{.j}}$ es inferior a 5, la probabilidad anterior puede aproximarse (Teorema Central del Límite) por:

$$P\left(\chi_1^2 \geq \frac{\left(n_{11} - \frac{n_{1..}n_{.1}}{n}\right)^2}{\frac{n_{1..}n_{.1}n_{2..}n_{.2}}{n^2(n-1)}}\right) = P\left(\chi_1^2 \geq \frac{(n-1)(n_{11}n_{22} - n_{21}n_{12})^2}{n_{1..}n_{.1}n_{2..}n_{.2}}\right),$$

donde el estadístico $\frac{(n-1)(n_{11}n_{22} - n_{21}n_{12})^2}{n_{1..}n_{.1}n_{2..}n_{.2}}$ se denomina chi-cuadrado ajustado (χ_{adj}^2) y es tal que $\chi_{adj}^2 = \frac{n-1}{n} \frac{n(n_{11}n_{22} - n_{21}n_{12})^2}{n_{1..}n_{.1}n_{2..}n_{.2}}$, donde $\frac{n(n_{11}n_{22} - n_{21}n_{12})^2}{n_{1..}n_{.1}n_{2..}n_{.2}}$ es el estadístico chi-cuadrado (χ^2) que proporcionan todos los softwares de contraste de independencia en tablas de contingencia.

En el ejemplo propuesto:

```
chisq.test(tabla_jud)$expected
#>      [,1] [,2]
#> [1,] 3.85 18.15
#> [2,] 3.15 14.85
chisq.test(tabla_jud, correct=FALSE)
#>
#> Pearson's Chi-squared test
#>
#> data: tabla_jud
#> X-squared = 5.6828, df = 1, p-value = 0.01713
```

Como $\chi^2 = 5,6828$, entonces $\chi_{adj}^2 = 5,54073$ y $P\left(\chi_{adj}^2 \geq 5,54073\right) = 0,0186$.⁵

Nótese que la probabilidad exacta de obtener una tabla tan alejada o más de la hipótesis de independencia que la observada (incluida esta) es 0,0327, mientras que la probabilidad aproximada es 0,0186. La aproximación no es muy buena, y ello se debe a la existencia de frecuencias esperadas menores que 5.

5.2.3.3. Contraste aproximado con corrección de continuidad

Como se vio en la subsección anterior, al aproximar la probabilidad de obtención de tablas tanto o más alejadas de H_0 que la observada (que se calcula con una distribución hipergeométrica, que es discreta) mediante una distribución χ_1^2 (que es continua), se comete un “error de continuización”. Dicho error se intenta corregir incluyendo en el contraste una “corrección

⁵Este “cálculo extra” es un pequeño peaje que hay que pagar en aras de la exactitud. Y es importante, porque pudiera hacer cambiar la decisión resultante del contraste.

5.2. Contraste de independencia en tablas 2×2

95

de continuidad”. Hay varias correcciones que han tenido cierto éxito en la literatura. La más popular es la corrección de Yates, si bien solo se recomienda cuando las \hat{E}_{ij} sean múltiplos de 0,5.

En el contraste aproximado con corrección de Yates, se rechaza H_0 si:

$$P\left(\chi_1^2 \geq \frac{(n-1)(|n_{11}n_{22} - n_{21}n_{12}| - 0,5n)^2}{n_{1.}n_{.1}n_{2.}n_{.2}}\right) \leq \alpha,$$

donde el estadístico $\frac{(n-1)(|n_{11}n_{22} - n_{21}n_{12}| - 0,5n)^2}{n_{1.}n_{.1}n_{2.}n_{.2}}$ se denomina estadístico chi-cuadrado ajustado corregido de continuidad de Yates ($\chi_{adj,CCY}^2$).

En el ejemplo propuesto, el test chi-cuadrado con corrección de continuidad de Yates se obtiene directamente con la función `chisq.test()` que, por defecto, incluye el argumento `correct = TRUE`.

```
chisq.test(tabla_jud)
#>
#> Pearson's Chi-squared test with Yates' continuity correction
#>
#> data: tabla_jud
#> X-squared = 3.8637, df = 1, p-value = 0.04934
```

Como el estadístico Chi-cuadrado corregido de continuidad χ_{CCY}^2 vale 3,8337, entonces $\chi_{adj,CCY}^2 = \frac{39}{40} \times 3,8337 = 3,7636$; y como $P(\chi_1^2 \geq 3,7636) = 0,0524$, H_0 se rechazaría cuando $\alpha > 0,0524$. Nótese que si, por ejemplo, $\alpha = 0,05$, la decisión sobre el rechazo o no de H_0 es distinta con χ_{CCY}^2 y $\chi_{adj,CCY}^2$; de ahí la importancia de utilizar el estadístico ajustado.

Por tanto, la corrección de Yates ha transformado la infraestimación de la probabilidad exacta en una sobreestimación de más o menos el mismo tamaño. Ello se debe a la existencia de \hat{E}_{ij} que distan mucho de ser múltiplos de 0,5. La corrección de Yates es la que incluye la librería utilizada (`stats`). Otras correcciones pueden verse en Ruiz-Maya et al. (1995) y Montero (2002).

5.2.4. Contraste de independencia: diseños tipo 2 y tipo 3

En el caso tipo 2, para la realización del test exacto, las tablas que se alejan tanto o más que la observada de la hipótesis de independencia son las que verifican:

$$\left| \frac{N_{11}}{n_{1.}} - \frac{N_{21}}{n_{2.}} \right| \geq \left| \frac{n_{11}}{n_{1.}} - \frac{n_{21}}{n_{2.}} \right|$$

y la probabilidad de ocurrencia de una tabla de contingencia viene dada por:

$$P(N_{11} = n_{11}; N_{12} = n_{12}; N_{21} = n_{21}; N_{22} = n_{22} | N = n) = \binom{n_{1.}}{N_{11}} \binom{n_{2.}}{N_{21}} \left(\frac{N_{.1}}{n} \right)^{N_{.1}} \left(\frac{N_{.2}}{n} \right)^{N_{.2}} \quad (5.1)$$

El estadístico de contraste en el test aproximado viene dado por $\chi^2 = \frac{n(n_{11}n_{22}-n_{21}n_{12})^2}{n_1.n_1n_2.n_2}$, y por $\chi^2_{CC} = \frac{n(|n_{11}n_{22}-n_{21}n_{12}|-\frac{f}{2})^2}{n_1.n_1n_2.n_2}$ en el caso de estar corregido de continuidad, siendo f el mayor factor común de los tamaños muestrales fijados.

En el caso tipo 3, las tablas que se alejan tanto o más que la observada de la hipótesis de independencia son las que verifican la condición expuesta en el tipo 2 (y tipo 1), siendo su probabilidad de ocurrencia:

$$P(N_{11} = n_{11}; N_{12} = n_{12}; N_{21} = n_{21}; N_{22} = n_{22} | N = n) = \frac{n!}{n_1!n_2!n_1!n_2!} \left(\frac{n_1}{n}\frac{n_1}{n}\right)^{n_{11}} \left(\frac{n_1}{n}\frac{n_2}{n}\right)^{n_{12}} \left(\frac{n_2}{n}\frac{n_1}{n}\right)^{n_{21}} \left(\frac{n_1}{n}\frac{n_2}{n}\right)^{n_{22}} \quad (5.2)$$

El test aproximado, en este caso, es un test razón de verosimilitudes donde el estadístico de contraste, $G = -2\ln \frac{n_1^{n_1}n_2^{n_2}n_1^{n_1}n_2^{n_2}}{n_{11}^{n_{11}}n_{21}^{n_{21}}n_{12}^{n_{12}}n_{22}^{n_{22}}n^n}$, también se distribuye como una χ^2_1 en caso de independencia. Apenas hay literatura sobre correcciones de continuidad en este modelo y la poca que hay sugiere la aplicación de la corrección de Yates.

Nota En el diseño de muestreo tipo 3 se recomienda no usar ninguna corrección de continuidad, salvo que el tamaño muestral sea muy pequeño y sea imprescindible la realización del test.

El código R de estos dos contrastes aproximados puede verse en la Sec. 5.3.1.

5.3. Contraste de independencia en tablas $R \times C$

El análisis de tablas $R \times C$ puede considerarse, en principio, una generalización del caso de tablas 2×2 . Ahora bien, en el caso $R \times C$ los test exactos, recomendados en el caso de que $E_{ij} \leq 5$ en más del 20 % de las celdas [Reynolds (1984)]⁶, son un auténtico reto y aún no están disponibles en el software convencional.

Si no se cumple el requisito anterior, una solución es agrupar categorías, con sentido común y coherencia⁷. Si la agrupación de categorías no pudiese hacerse, por carecer de sentido o cualquier otro motivo, lo más honesto sería no realizar el contraste hasta disponer de una base de datos mejor.

A la luz de lo anteriormente expuesto, en el caso de tablas $R \times C$ la atención se centrará en los tests aproximados.

⁶La razón es que, sea cual sea el diseño de muestreo, si la probabilidad de que un elemento de la tabla caiga en una determinada celda $\{i,j\}$ es muy pequeña (se suele utilizar el límite del 5 %), entonces la distribución de probabilidad de N_{ij} es muy asimétrica, y para que se simetrique lo suficiente y se pueda aproximar por una Normal (que después, al cuadrado, participará en una chi-cuadrado), el tamaño muestral, en el tipo 3, y los totales marginales fijados, en los tipos 1 y 2, tienen que ser grandes (se suele fijar el valor de 100), de ahí el límite de $E_{ij} \leq 5$; por eso es recomendable que el tamaño muestral sea grande y los totales marginales no estén desequilibrados. En todo caso, lo ideal es que requisito $E_{ij} \leq 5$ se cumpla en todas las celdas de la tabla.

⁷En tablas 2×2 no se pueden agrupar filas, y la única solución son los tests exactos.

5.3.1. Contrastes aproximados

Cuando el procedimiento de muestreo o el diseño experimental es de tipo 1 o 2 el contraste aproximado de independencia es el denominado contraste Chi-cuadrado. La filosofía de dicho contraste es la siguiente: Parece lógico que el contraste se base en las diferencias (cuadráticas, para que no se compensen las negativas con las positivas) entre las frecuencias observadas y las esperadas bajo la hipótesis de independencia. Si los factores son independientes, dichas diferencias serán pequeñas y atribuibles a fluctuaciones aleatorias. Si están asociados, serán grandes y atribuibles a la asociación existente entre sus niveles. Pearson propuso el siguiente estadístico de contraste:

$$\chi^2 = \sum_{i=1}^R \sum_{j=1}^C \frac{(N_{ij} - \hat{E}_{ij})^2}{\hat{E}_{ij}},$$

que, si no se incumple el requisito expuesto en las primeras líneas de la Sec. 5.3, y bajo la hipótesis de independencia, se distribuye como una $\chi^2_{(R-1)(C-1)}$. En caso de que la probabilidad de que una $\chi^2_{(R-1)(C-1)}$ sea inferior al nivel de significación prefijado, se rechazará la hipótesis de independencia. Téngase en cuenta que en el caso $R \times C$ la hipótesis alternativa es “al menos un nivel de un factor está asociado con un nivel del otro factor”.

La razón de que las diferencias $(N_{ij} - \hat{E}_{ij})^2$ se dividan por \hat{E}_{ij} en el estadístico chi-cuadrado de Pearson es la siguiente: la misma diferencia $N_{ij} - \hat{E}_{ij}$ puede significar cosas bien diferentes. Una diferencia de 5 no es nada si $\hat{E}_{ij} = 1,000$; pero es muchísimo si $\hat{E}_{ij} = 2$. Por eso la diferencia (cuadrática) se pone en relación con la frecuencia esperada.

En el caso de que el procedimiento de muestreo sea del tipo 3, aunque puede aplicarse el contraste chi-cuadrado, es recomendable proceder con el contraste de independencia de razón de verosimilitudes, que compara las frecuencias esperadas bajo la hipótesis de independencia y las observadas por cociente. Se basa en la razón de la verosimilitud de la hipótesis de independencia a la luz de la muestra obtenida y del máximo de la función de verosimilitud, Λ . Bajo el supuesto de independencia la razón será cercana a la unidad, atribuyéndose la diferencia a fluctuaciones aleatorias; el logaritmo neperiano de dicha razón (negativo) estará cercano a cero. En caso contrario, el cociente de verosimilitudes (negativo) disminuye, tanto más cuanto más diferencia hay entre la verosimilitud de la hipótesis de independencia y el máximo de la función de verosimilitud. En Wilks (1935) se demostró que, cuando la hipótesis de independencia es cierta, $G = -2\ln\Lambda$, con $\Lambda = \prod_{i=1}^R \prod_{j=1}^C \left(\frac{\hat{E}_{ij}}{n_{ij}}\right)^{n_{ij}}$ se distribuye como una $\chi^2_{(R-1)(C-1)}$. Ambos estadísticos de contraste, χ^2 y G , son asintóticamente equivalentes.

A modo de ejemplo, se quiere contrastar si en la Comunidad de Madrid la opinión sobre la presidenta Dña. Isabel Díaz Ayuso depende de la zona geográfica o si, por el contrario, es independiente de ella. Para ello se encuestan, por algún procedimiento aleatorio 2795 personas con derecho a voto en la comunidad y se eliminan las respuestas “NS/NC/me es indiferente”. Los resultados obtenidos fueron los siguientes (dataset `ayuso` del paquete CDR):

```
data("ayuso")
tabla_ayuso<-table(ayuso)
```

```
tabla_ayuso
#>           opinion
#> zona      n1_nefasto n2_mala n3_buena n4_excelente
#> n1_mad_muni        25     500      50     1000
#> n2_metropol         10     280      50      460
#> n3_extraradio        5     130      25      260
chisq.test(tabla_ayuso)$expected
#>           opinion
#> zona      n1_nefasto n2_mala n3_buena n4_excelente
#> n1_mad_muni    22.540250 512.7907 70.43828   969.2308
#> n2_metropol     11.449016 260.4651 35.77818   492.3077
#> n3_extraradio    6.010733 136.7442 18.78354   258.4615
chisq.test(tabla_ayuso, correct=FALSE)
#>
#> Pearson's Chi-squared test
#>
#> data: tabla_ayuso
#> X-squared = 19.486, df = 6, p-value = 0.003418
```

```
library("DescTools")
GTest(tabla_ayuso,correct = "none")
#>
#> Log likelihood ratio (G-test) test of independence without correction
#>
#> data: tabla_ayuso
#> G = 19.357, X-squared df = 6, p-value = 0.003602
```

Como puede verse, sea cual sea el estadístico de contraste, la hipótesis de independencia se rechaza para valor de α de los utilizados en la práctica (1 %, 2,5 %, 5 %, 10 %).

5.3.2. Contraste aproximado con corrección de continuidad

Afortunadamente, en la mayoría de las ocasiones, el tamaño muestral es grande y los totales marginales no están muy desequilibrados, con lo que los estadísticos chi-cuadrado y chi-cuadrado corregido de continuidad son prácticamente iguales, sobre todo si el número de niveles de ambos factores es elevado. En caso de utilizar una corrección de continuidad, hay unanimidad en utilizar la de Yates, sea cual sea el procedimiento de muestreo y el test (chi-cuadrado o G), si bien dicha unanimidad tiene mucho que ver con que es la única que está programada en el software convencional sobre tablas de contingencia.

5.4. Medidas de asociación en tablas 2×2

Si no se rechaza la hipótesis de independencia, el análisis de la tabla se puede dar por finalizado. En caso contrario, el nuevo objetivo es determinar la dirección de la asociación detectada (o las

5.4. *Medidas de asociación en tablas 2×2*

99

fuentes de asociación en el caso $R \times C$) y su intensidad, y para ello se utilizan las denominadas medidas de asociación. Igual que en el contraste de independencia, se distinguirán los casos 2×2 y $R \times C$, en esta ocasión no tanto por motivos académicos sino porque las situaciones son bien diferentes.

En el caso 2×2 , los tipos de asociación en función de su dirección (positiva y negativa) ya se definieron en la Sec. 5.2.1. Por lo que se refiere a los límites de su intensidad, se dice que asociación es perfecta cuando al menos uno de los niveles de uno de los factores queda determinado por un nivel del otro factor. Puede ser estricta o implícita de tipo 2⁸:

- Estricta: dado el nivel de un factor, el nivel del otro queda inmediatamente determinado.
- Implícita de tipo 2: dado un nivel de un factor, el nivel del otro queda inmediatamente determinado; dado el otro nivel, no queda determinado el nivel del otro factor.

5.4.1. La \hat{Q} de Yule

En caso de independencia, las frecuencias observadas coinciden con las estimaciones de las esperadas. A medida que las primeras se separan de las segundas, se produce un alejamiento de dicha hipótesis y los niveles de los factores aumentan la intensidad de su asociación. Por consiguiente, las diferencias \hat{D}_{ij} entre las frecuencias observadas y las estimaciones de las esperadas bajo el supuesto de independencia pueden ser la base de una magnífica medida de asociación. A mayores diferencias, mayor asociación. Mas sencillo todavía: una única diferencia, por ejemplo la \hat{D}_{11} , podría servir como medida de asociación porque, como bajo la hipótesis de independencia, $\hat{D}_{ij} = 0$ y $\hat{D}_{11} = \hat{D}_{22} = -\hat{D}_{12} = -\hat{D}_{21}$, entonces se tiene que:

- En caso de independencia: $\hat{D}_{11} = \hat{D}_{22} = 0$ y $\hat{D}_{12} = \hat{D}_{21} = 0$, o simplemente, $\hat{D}_{11} = 0$
- En caso de asociación positiva: $\hat{D}_{11} = \hat{D}_{22} \geq 0$ y $\hat{D}_{12} = \hat{D}_{21} \leq 0$, o simplemente, $\hat{D}_{11} \geq 0$
- En caso de asociación negativa: $\hat{D}_{11} = \hat{D}_{22} \leq 0$ y $\hat{D}_{12} = \hat{D}_{21} \geq 0$, o simplemente, $\hat{D}_{11} \leq 0$

Por tanto, \hat{D}_{11} determina muy fácilmente la dirección de la asociación. Sin embargo, en cuanto a la intensidad de la misma, el campo de variación de \hat{D}_{11} , $[-\frac{n_{12}n_{21}}{n}; \frac{n_{12}n_{21}}{n}]$ depende de los valores de las frecuencias observadas (esto es un problema a la hora de la interpretación) y la máxima intensidad asociativa se da cuando la diagonal descendente o la diagonal ascendente contienen ceros, es decir en caso de asociación perfecta estricta (negativa o positiva).

Para solucionar el problema anterior, se define la \hat{Q} de Yule como:

$$\hat{Q} = \frac{n\hat{D}_{11}}{n_{11}n_{22} - n_{12}n_{21}} = \frac{n_{11}n_{22} - n_{12}n_{21}}{n_{11}n_{22} + n_{12}n_{21}}.$$

El campo de variación de \hat{Q} es $[-1; 1]$ y:

⁸La asociación perfecta implícita de tipo 1, que no puede darse en tablas 2×2 , consiste en que, dado un nivel de un factor, el nivel del otro queda inmediatamente determinado; pero dado el nivel de este último no se puede determinar el nivel del primero.

- En caso de independencia, $\hat{Q} = 0$.
- En caso de asociación positiva $\hat{Q} < 0$.
- En caso de asociación negativa $\hat{Q} > 0$.

Lógicamente, a mayor valor absoluto de \hat{Q} mayor intensidad de la asociación.

En el ejemplo utilizado para ilustrar el diseño experimental de tipo 1, se tiene:

```
YuleQ(tabla_jud)
#> [1] -0.826087
```

A la luz del valor de \hat{Q} se concluye la existencia de una fuerte asociación negativa.

5.4.2. Otras medidas de asociación para tablas 2×2

5.4.2.1. Cuadrado medio de la contingencia de Pearson

La primera medida de asociación que se nos viene a todos a la cabeza es el propio estadístico de contraste χ^2 . Sin embargo, no puede utilizarse como medida de asociación porque es siempre positivo y, sobre todo, porque su valor máximo, $N(K - 1)$, depende del tamaño muestral N y de K , el número más pequeño de filas o columnas. En el caso 2×2 depende únicamente de N porque $K - 1 = 1$. Para eliminar el efecto tamaño muestral, se define el cuadrado medio de la contingencia de Pearson como:

$$\hat{\phi}^2 = \frac{\chi^2}{N} = \frac{(n_{11}n_{22} - n_{12}n_{21})^2}{n_1.n_2.n_{1.}n_{.2}}.$$

Su campo de variación es $[0; 1]$, tomando el valor 0 en caso de independencia y 1 cuando hay asociación perfecta y estricta. Cuanto mayor sea el valor del coeficiente, mayor es intensidad de la asociación.

No proporciona la dirección de la asociación, si bien se puede saber por el signo de $n_{11}n_{22} - n_{12}n_{21}$. Otra consideración importante es que, si se codifican los niveles de los factores como $(0;1)$, $\hat{\phi}^2$ coincide con el coeficiente de determinación lineal entre los factores. Por tanto, la asociación que mide es “lineal” (de ahí que su valor suela ser más bajo que el de \hat{Q}). Su raíz cuadrada es conocida como “la V de Cramer”. En el ejemplo utilizado se tiene que:

```
(Phi(tabla_jud))^2
#> [1] 0.1420701
```

5.4.2.2. Odds ratio o cociente de posibilidades⁹

Se define como $\alpha = \frac{P_{11}/P_{12}}{P_{21}/P_{22}}$ y se estima como $\hat{\alpha} = \frac{n_{11}/n_{12}}{n_{21}/n_{22}} = \frac{n_{11}n_{22}}{n_{12}n_{21}}$.

⁹Para no confundirlo con el riesgo relativo.

5.5. Medidas de asociación en tablas $R \times C$

101

Su campo de variación es $[0; \infty)$, asimétrico, y por consiguiente difícil de interpretar. En todo caso:

- Si $\alpha < 1$, la probabilidad (aquí denominada posibilidad) de pertenecer al nivel 1 del factor B es menor en el nivel 1 del factor A que en el 2.
- Si $\alpha = 1$, la probabilidad de pertenecer al nivel 1 del factor B es la misma en ambos niveles del factor A.
- Si $\alpha > 1$, la probabilidad pertenecer de al nivel 1 del factor B es mayor en el nivel 1 del factor A que en el 2.

Una posible solución a la dificultad de interpretación es definir $\ln\hat{\alpha}$, que es una medida simétrica en $(-\infty; +\infty)$. Sin embargo, su interpretación, dada la gran amplitud del campo de variación, continúa siendo muy difusa.

Una ventaja que tiene respecto a $\hat{\alpha}$ es que no cambia si las filas se convierten en columnas y las columnas en filas.¹⁰ Por ello, la razón de posibilidades se puede utilizar no solo en estudios retrospectivos, sino también en aquéllos prospectivos y transversales. Finalmente, nótese que (i) la razón de posibilidades y el riesgo relativo (P_1/P_2) se relacionan como sigue: $\alpha = \frac{P_1(1-P_2)}{P_2(1-P_1)}$ y que (ii) ambos son similares cuando la probabilidad de éxito P_i está cerca de cero en ambos grupos. En el ejemplo que nos ocupa:

```
library("epiR")
epi.2by2(tabla_jud, method = 'cohort.count')$massoc.summary[2,]
#>      var      est      lower      upper
#> 2 Odds ratio 0.0952381 0.01021365 0.8880565
```

5.5. Medidas de asociación en tablas $R \times C$

En caso de rechazo de la hipótesis de independencia en una tabla $R \times C$, se concluye que al menos un nivel de uno de los factores está asociado con uno del otro factor. En ese caso, se utilizarán las medidas de asociación para determinar la intensidad de la misma. Las asociaciones de determinados niveles del factor A con determinados niveles del factor B que llevan al rechazo de la independencia de ambos se denominan **fuentes de asociación**, y se determinan mediante los residuos estandarizados ajustados.

5.5.1. Medidas derivadas del estadístico Chi-cuadrado

Como se avanzó en la Sec. 5.4.2.1, el estadístico χ^2 no puede utilizarse como medida de asociación porque su valor máximo, $N(K - 1)$, siendo N el tamaño muestral y K el número más pequeño de filas o columnas, depende tanto de N como del número de niveles de los factores.

El cuadrado medio de la contingencia, ϕ^2 , elimina el efecto “tamaño muestral”, pero no el efecto “número de niveles de los factores”. Igual le ocurre al coeficiente de contingencia; y a la T de

¹⁰Es decir, no es necesario identificar la variable respuesta para utilizar esta medida.

Tschuprow, salvo en las tablas cuadradas. La única medida derivada del estadístico χ^2 que corrige ambos efectos es la V de Cramer:

$$V = \sqrt{\frac{\chi^2}{KN}},$$

con $K = \min(R - 1; C - 1)$. Su campo de variación es $[0, 1]$ y alcanza su máximo en caso de asociación perfecta. En tablas cuadradas $V = T$.

En el ejemplo utilizado en la Sec. 5.3.1

```
CramerV(tabla_ayuso)
#> [1] 0.0590406
```

Aunque se rechaza la hipótesis de independencia, la asociación existente entre la opinión sobre la presidente y la zona geográfica es muy pequeña. Y ello, porque, sea cual sea la zona geográfica, aunque hay ligerísimas variaciones, la opinión es muy buena: para alrededor del 60% es excelente, para la tercera parte muy buena y tan solo para el 5% es mala (alrededor del 4%) o muy mala (apenas el 1%).

5.5.2. Medidas basadas en la reducción proporcional del error: λ de Goodman y Kruskal

Al contrario que las medidas basadas en el estadístico Chi-cuadrado, exigen determinar cuál es el factor explicativo y cuál el factor a explicar. Sea A el factor explicativo y B el factor a explicar: supóngase que se selecciona aleatoriamente uno de los elementos de la tabla. Este elemento pertenecerá a un nivel “ i ” de A y a un nivel “ j ” de B . Supóngase que se quiere predecir el nivel de B al que pertenece, (i) sin utilizar el hecho de saber a qué nivel de A pertenece, y (ii) utilizando dicho hecho. Lógicamente, tanto en el caso (i) como en el (ii) se comete un error ($P(i)$ y $P(ii)$, respectivamente). La probabilidad de error será la misma si los factores son independientes. Sin embargo, si están asociados, el conocimiento del nivel de A al que pertenece el elemento seleccionado ayudará en la predicción del nivel de B al que pertenece (tanto más cuanto más asociados estén los factores), y la probabilidad de error disminuirá respecto al caso (i). La reducción proporcional que se opera es:

$$\lambda = \frac{P(i) - P(ii)}{P(i)},$$

con $P(i) = N - \max_j n_{\cdot j}$ y $P(ii) = N - \sum_{j=1}^C \max_j n_{\cdot j}$.

En el caso en que A sea el factor a explicar, $P(i) = N - \max_i n_{\cdot i}$ y $P(ii) = N - \sum_{i=1}^R \max_i n_{\cdot i}$.

En caso de no tener claro cual es el factor a explicar, se utiliza la media agregativa de las dos medidas anteriores:

$$\lambda = \frac{\sum_{j=1}^C \max_i n_{i\cdot} - \max_i n_{i\cdot} + \sum_{i=1}^R \max_j n_{\cdot j} - \max_j n_{\cdot j}}{2N - \max_i n_{i\cdot} \max_j n_{\cdot j}}.$$

Su campo de variación es $[0, 1]$. En caso de independencia $\lambda = 0$. Ahora bien, que $\lambda = 0$ no implica necesariamente que A y B tengan que ser independientes, puesto que también λ vale 0 cuando en uno de los niveles del factor a explicar las frecuencias son superiores a las de los demás niveles, y ello para todos los niveles del factor explicativo, aunque los factores no sean independientes. En caso de asociación, $0 < \lambda \leq 1$, alcanzándose la unidad en caso de asociación perfecta.

Una limitación de λ (además de la anterior y de que exige determinar el factor explicativo y el factor a explicar) es su sensibilidad a totales marginales desequilibrados; en este caso toma valores anormalmente bajos. Tal es el caso del ejemplo que nos ocupa, donde $\lambda = 0$, con factor a explicar la opinión, y los factores no son independientes. Y es que, sea cual sea la zona, las frecuencias de la categoría de opinión “excelente” son siempre las más elevadas.

```
Lambda(tabla_ayuso, direction = "row")
#> [1] 0
```

5.5.3. Determinación de las fuentes de asociación

En el caso de tablas RxC el estadístico, el rechazo la hipótesis de independencia, no indica que cada nivel de uno de los factores esté asociado con uno de los niveles del otro factor, como en las tablas 2×2 . Lo que indica es que al menos uno de los niveles de uno de los factores está asociado con un nivel del otro. Por tanto, puede ser, y así es normalmente, que dicho rechazo se deba a que algunos niveles de uno de los factores (incluso solo uno) están asociados con alguno de los del otro factor. Ya no hay dirección de la asociación. Hay fuentes de asociación.

Para identificar las fuentes de asociación lo lógico es fijarse en cada celda en las diferencias entre la frecuencia observada y la esperada bajo el supuesto de independencia (los también denominados residuos). Pero su interpretación depende del tamaño de la frecuencia esperada, y por ello se estandarizan, es decir, se ponen en relación a la raíz cuadrada de las correspondientes frecuencias esperadas.

Como se necesita una distribución de probabilidad para dichos residuos estandarizados, bajo la hipótesis de independencia, para decidir si son significativamente grandes (asociación) o no (independencia), se dividen por su desviación típica, y así tienen aproximadamente una distribución $N(0;1)$. Estos nuevos residuos se denominan residuos estandarizados ajustados (o de Haberman), y son los que se utilizan para identificar las fuentes de asociación. Por tanto:

$$\hat{R}_{ij} = n_{ij} - \hat{E}_{ij},$$

$$\hat{R}_{ij}(\text{est}) = \frac{n_{ij} - \hat{E}_{ij}}{\sqrt{\hat{E}_{ij}}},$$

$$\hat{R}_{ij}(est; adj) = \frac{\hat{R}_{ij}(est)}{\sqrt{\left(1 - \frac{n_{i\cdot}}{N}\right)\left(1 - \frac{n_{\cdot j}}{N}\right)}}.$$

Habrá una fuente de asociación en cada celda $\{i;j\}$ que verifique: $|\hat{R}_{ij}(est; adj)| \geq k$, con $k = 2, 33; 1, 96; 1, 64$ para $\alpha = 0, 01; 0, 05; 0, 10$, respectivamente.

En el ejemplo que nos ocupa, los residuos estandarizados ajustados son:

```
library(questionr)
chisq.residuals(tabla_ayuso, digits = 2, std = TRUE)
#>           opinion
#> zona      n1_nefasta n2_mala n3_buena n4_excelente
#>   n1_mad_muni      0.79   -1.04    -3.77      2.41
#>   n2_metropol     -0.51    1.74     2.88     -2.78
#>   n3_extraradio    -0.45   -0.76     1.59      0.17
```

Asumiendo $\alpha = 0, 05$, las fuentes de asociación son “Madrid municipio-excelente” a costa de “buena”, y “Madrid metropolitano-buena” a costa de “excelente”.

5.6. Contrastes de independencia en tablas multidimensionales

En tablas con más de dos factores (el objetivo aquí es el caso $R \times C \times M$, por simplicidad), no solo se puede contrastar la hipótesis de independencia global sino que, en caso de ser rechazada, también se pueden contrastar las hipótesis de (i) independencia parcial: dos factores están asociados y el tercero es independiente de ellos, y (ii) condicional: dos de los factores son independientes para cada nivel del tercero pero están asociados con él.

En los tres casos, el estadístico de contraste (contraste aproximado) es

$$\chi^2 = \sum_{i=1}^R \sum_{j=1}^C \sum_{m=1}^M \frac{(n_{ijm} - \hat{E}_{ijm})^2}{\hat{E}_{ijm}},$$

con los siguientes grados de libertad (g.l.) y \hat{E}_{ijm} bajo la correspondiente H_0 :

Independencia global:

$$dl = (R \times C \times M) - (R - 1) - (C - 1) - (M - 1) - 1 \text{ y } \hat{E}_{ijm} = \frac{n_{i\cdot} \cdot n_{\cdot j} \cdot n_{\cdot\cdot m}}{N^2}$$

Independencia parcial:

A y B asociados entre sí pero independientes de C :

- $g.l. = (R \times C \times M) - (R \times C - 1) - (M - 1) - 1$ y $\hat{E}_{ijm} = \frac{n_{ij} \cdot n_{\cdot\cdot m}}{N}$

5.6. Contrastes de independencia en tablas multidimensionales

105

A y *C* asociados entre sí pero independientes de *B*:

- $g.l. = (R \times C \times M) - (R \times M - 1) - (C - 1) - 1$ y $\hat{E}_{ijm} = \frac{n_{i..}n_{.j..}}{N}$

B y *C* asociados entre sí pero independientes de *A*:

- $g.l. = (R \times C \times M) - (C \times M - 1) - (R - 1) - 1$ y $\hat{E}_{ijm} = \frac{n_{.i..}n_{i..}}{N}$

Independencia condicional

A y *B* son independientes entre sí, pero están asociados con *C*:

- $g.l. = (R \times C \times M) - (R \times M - 1) - (C \times M - 1) - 1$ y $\hat{E}_{ijm} = \frac{n_{i..}n_{.jm}}{N}$

A y *C* son independientes entre sí, pero están asociados con *B*:

- $g.l. = (R \times C \times M) - (R \times C - 1) - (M \times C - 1) - 1$ y $\hat{E}_{ijm} = \frac{n_{ij..}n_{.jm}}{N}$

B y *C* son independientes entre sí, pero están asociados con *A*:

- $g.l. = (R \times C \times M) - (C \times R - 1) - (M \times R - 1) - 1$ y $\hat{E}_{ij.} = \frac{n_{i..}n_{.jm}}{N}$

También son interesantes las relaciones de segundo orden o superior (por ejemplo, si la asociación entre dos de los factores difiere en dirección y/o intensidad para distintos niveles del tercero), pero se estudian mediante **modelos logarítmico lineales**.

Resumen

Las tablas de contingencia analizan la relación entre variables categóricas. Su análisis responde preguntas como: los factores involucrados en la tabla, ¿son independientes o están asociados? Si están asociados, ¿qué niveles de dichos factores son los que están asociados?, ¿cuál es la intensidad de dicha asociación? Se aborda ampliamente el caso de tablas bifactoriales y se proponen test exactos y aproximados para el contraste de la hipótesis de independencia (para tres procedimientos de muestreo diferentes) y una selección de medidas de asociación. Finalmente, se hace una breve incursión en el ámbito de las tablas multidimensionales.

Parte I

Machine learning supervisado

Capítulo 6

Árboles de clasificación y regresión

Ramón A. Carrasco^a e Itzcóatl Bueno^{b,a}

^aUniversidad Complutense de Madrid ^bInstituto Nacional de Estadística

6.1. Introducción

Los árboles de decisión son modelos que se utilizan principalmente para la resolución de problemas de clasificación, en los que hay que predecir las distintas categorías de la variable objetivo o dependiente, aunque también son aplicables a la predicción de valores numéricos de dicha variable objetivo, esto es, como modelos de regresión. De ahí que sean conocidos como árboles de clasificación y regresión (CART, Classification and Regression Trees). Algunos ejemplos de árboles de decisión son:

- **Clasificación:** en la medida que la variable objetivo debe ser categórica se podrían usar por ejemplo para tomar la decisión de qué empleados deberían de promocionar (variable con dos categorías: sí promocionar o no promocionar) en base a sus méritos, capacidades, edad, etc. Otro ejemplo podría ser su uso para decidir si se juega o no un partido de tenis en base a la climatología prevista. Este ejemplo se muestra gráficamente en la Fig. 6.1. En este último caso, el algoritmo que se utilice indicará la decisión a tomar en base a los registros climatológicos de los partidos que ya se hayan jugado. Así, si un determinado día se quiere jugar al tenis, se deberán tomar como variables de entrada las previsiones de Tipo de día (soleado, nublado o lluvioso), la fuerza del Viento y la Humedad. En caso de ser un día nublado, el algoritmo sugerirá que se juegue. En caso de ser soleado, comprobará el nivel de Humedad y, si no es muy elevada, recomendará que se juegue el partido. Lo mismo pasará si la previsión es de lluvia pero la fuerza del Viento prevista no es lo suficientemente intensa como para impedir el normal desarrollo del partido.

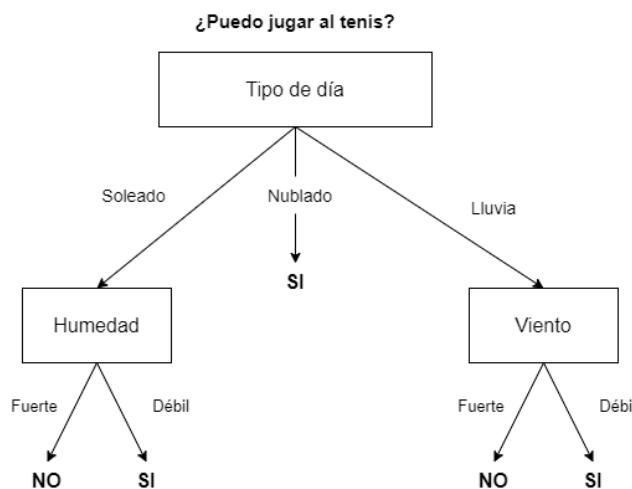


Figura 6.1: Ejemplo de árbol de decisión.

- **Regresión:** siguiendo con el ejemplo del partido de tenis, también se puede aplicar un árbol de decisión para determinar cuántas horas jugar de acuerdo a las condiciones climatológicas. En la Fig. 6.1 se sustituirían la predicciones dicotómicas SI/NO por valores numéricos, como se muestra en la Fig. 6.2. Por ejemplo, el algoritmo puede sugerir jugar 5 horas si el día está soleado pero la Humedad es del 30 % de vapor de agua por m^3 , y 3,5 horas si está soleado pero la Humedad es del 80 %. También puede decidir que si el día está nublado se jueguen 4 horas. O en caso de lluvia, podría decidir que el partido dure 0,75 horas si la fuerza del Viento es de 62km/h y 1,15 horas si es de 27km/h.

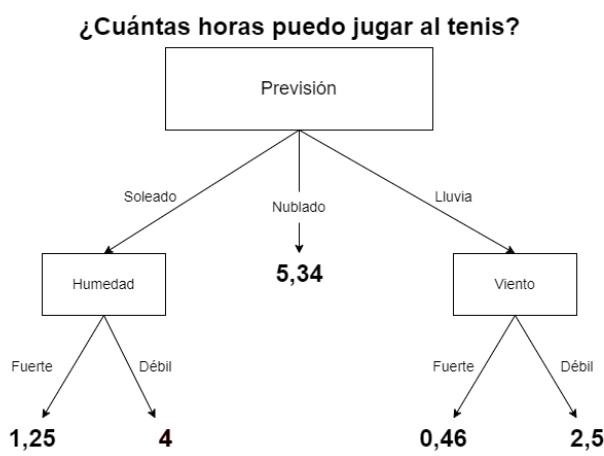


Figura 6.2: Ejemplo de árbol de regresión

6.1. Introducción

111

Como se ha comentado, CART es un término genérico para describir este tipo de algoritmos de árbol y también un nombre específico para el algoritmo original de (Breiman et al., 1984) de construcción de árboles de clasificación y regresión. Sin embargo, existen otros como el ID3 (Induction Decision Trees), o el C4.5 que está basado en el ID3. En la Tabla 6.1 se muestra una pequeña comparativa de estos tres algoritmos:

Tabla 6.1: Características de los principales algoritmos de árboles de decisión.

Algoritmo	Criterio de división	Tipo de variables input	Estrategia de poda
ID3	Ganancia de información	Solo categóricas	No poda
CART	Índice de Gini	Categóricas y numéricas	Poda basada en el coste de complejidad
C4.5	Ratio de ganancia	Categóricas y numéricas	Poda basada en el error

Los árboles de decisión tienen múltiples ventajas. Entre ellas destacan:

- Son fáciles de entender e interpretar. Su visualización clara permite interpretar la salida del modelo y entender su proceso como un conjunto de condicionantes.
- El mismo algoritmo incorporado en R (CART) es válido tanto para problemas de clasificación como de regresión y, por tanto, la variable objetivo puede ser continua o categórica. Respecto al resto de variables de entrada, las independientes, comentar que puede ser tanto categóricas como numéricas. Al contrario que ocurre con otros algoritmos, este último tipo de variables no requieren la estandarización, puesto que se basa en reglas y no en el cálculo de distancias entre observaciones.
- Tratan mejor que otros algoritmos el problema de la no linealidad.
- Respecto a los datos, hacen un tratamiento automático de valores ausentes (en la mayoría de los árboles de clasificación) y no se ven afectados con las observaciones atípicas.

Sin embargo, también tienen ciertas desventajas:

- Son inestables ya que la inclusión de una nueva observación en la fase de entrenamiento obliga a reconstruirlo, pudiendo modificar la estructura del árbol final.
- No son recomendables en caso de grandes conjuntos de datos, puesto que el modelo entrenado puede estar sobreajustado. Este sobreajuste es el principal problema de los árboles de decisión, ya que modelos demasiados complejos pueden ajustar muy bien los datos observados, pero también pueden cometer muchos errores en la fase de predicción. Cuando esta circunstancia se da, el modelo ha aprendido los datos de entrenamiento pero no la generalidad del problema que es lo que normalmente se pretende. El sobreajuste da lugar también a una varianza elevada.

6.2. Procedimiento con R: la función rpart()

En el paquete **rpart** de **R** se encuentra la función **rpart()** que se utiliza para entrenar un árbol de decisión:

```
rpart(formula, data, ...)
```

- **formula**: Refleja la relación entre la variable dependiente Y y los predictores tal que $Y \sim X_1 + \dots + X_p$.
- **data**: Conjunto de datos con el que entrenar el árbol de acuerdo a la fórmula indicada.

6.3. Árboles de clasificación

Formalmente, un árbol de decisión es un grafo acíclico (un grafo sin ciclos, siendo un ciclo un circuito completo) que se inicia en un **nodo raíz**, el cual se divide en **ramas**, también conocidas como **aristas**. De las ramas salen las **hojas**, también denominadas **nodos**. Estos nodos pueden ser nodos finales o **puntos de decisión** (si de ellos no salen nuevas ramas con nuevos nodos) o no (de ellos salen nuevas ramas con nuevas hojas o nodos) y así hasta que todos los nodos sean puntos de decisión. En el ejemplo de la Fig. 6.1 el nodo raíz es la caja *Tipo de día*. Las ramas o aristas, son sus tres niveles o categorías: *Soleado*, *Nublado* o *Lluvia*. Cada una de estas ramas conecta con una nueva hoja o nodo: *Humedad* o *Viento* en los casos de soleado o lluvia, respectivamente. Sin embargo, en ese ejemplo, *Nublado* representa un nodo terminal, puesto que, llegado a ese punto, la salida que proporcionaría el árbol es “*Jugar al tenis*”. Este proceso se repite utilizando el conjunto de datos disponible en cada hoja, generándose una clasificación final cuando una hoja no tenga ramas nuevas, en cuyo caso recibe la denominación de nodo final. El objetivo es que el árbol sea lo más general y pequeño posible. Esto se consigue seleccionando, en cada paso, la variable que optimice la división de los datos en conjuntos homogéneos, de tal forma que se prediga mejor la clase objetivo.

6.3.1. ¿Cómo se va formando el árbol de clasificación?

Como ya se ha mencionado, en la construcción de un árbol de decisión se va dividiendo en nuevas ramas de forma recursiva, es decir, cada división está condicionada por las anteriores. El objetivo en cada hoja, es encontrar la variable más adecuada para dividir los datos de ese nodo en dos nuevas hojas, de tal forma que el error global entre la clase observada y la predicha por el árbol se minimice. Para la construcción de árboles de clasificación, el algoritmo CART utiliza la medida de impureza de Gini para generar las particiones, mientras que los algoritmos ID3 y C4.5 están basados en las de entropía.

6.3.1.1. Impureza de Gini

La **Impureza de Gini**, utilizada por el algoritmo CART, es una medida de la frecuencia con la que una observación elegida aleatoriamente del conjunto se asignaría a la clase errónea si se

6.3. Árboles de clasificación

113

etiqueta al azar en una de las clases que se consideran. Formalmente, sea X un conjunto de datos con κ clases, y sea p_i la probabilidad de que una observación pertenezca a la clase i . La Impureza de Gini para X se define como:

$$Gini(X) = 1 - \sum_{i=1}^{\kappa} p_i^2 \quad (6.1)$$

Como se ha comentado, a la hora de construir el árbol se selecciona el atributo con la menor impureza de Gini para dividir el conjunto de datos en el nodo en dos. Si un conjunto de datos X se divide en un atributo φ en dos subconjuntos X_1 y X_2 con tamaños n_1 y n_2 , respectivamente, la impureza ponderada de Gini se define como:

$$Gini_{\varphi}(X) = \frac{n_1}{n} Gini(X_1) + \frac{n_2}{n} Gini(X_2) \quad (6.2)$$

En el ejemplo de la Fig. 6.1 considérese la siguiente situación:

Tabla 6.2: Datos para decidir si se juega el partido

Día	Tipo de día	Humedad	Viento	Decisión
1	Soleado	Fuerte	Débil	NO
2	Soleado	Fuerte	Fuerte	NO
3	Lluvia	Fuerte	Débil	SI
4	Nublado	Fuerte	Débil	SI
5	Lluvia	Débil	Débil	SI
6	Lluvia	Débil	Fuerte	NO
7	Soleado	Fuerte	Débil	NO
8	Nublado	Débil	Fuerte	SI
9	Soleado	Débil	Débil	SI
10	Lluvia	Débil	Débil	SI
11	Soleado	Débil	Fuerte	SI
12	Nublado	Fuerte	Fuerte	SI
13	Nublado	Débil	Débil	SI
14	Lluvia	Fuerte	Fuerte	SI
15	Soleado	Fuerte	Fuerte	NO

Para el *Tipo de día*, los datos se agruparían como muestra la Tabla 6.3, permitiendo el cálculo de la impureza de Gini para cada una de sus categorías.

Tabla 6.3: Días que se juega o no de acuerdo al *Tipo de día*

Tipo de día	SI	NO	# observaciones
Soleado	2	4	6
Nublado	4	0	4

Tipo de día	SI	NO	# observaciones
Lluvia	4	1	5

$$Gini(Soleado) = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0,45$$

$$Gini(Nublado) = 1 - \left(\frac{4}{4}\right)^2 = 0$$

$$Gini(Lluvia) = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0,32$$

Ahora, se calcula la suma ponderada de la impureza de Gini para la variable *Tipo de día*:

$$Gini(\text{Tipo de día}) = 0,45 \cdot \left(\frac{6}{15}\right) + 0 \cdot \left(\frac{4}{15}\right) + 0,32 \cdot \left(\frac{5}{15}\right) = 0,29$$

Del mismo modo, se puede calcular la impureza de Gini para el resto de variables. La Tabla 6.4 y la Tabla 6.5 presentan los resultados para *Humedad* y *Viento*, respectivamente.

Tabla 6.4: Impureza de Gini para las categorías de Humedad

Humedad	SI	NO	# observaciones	p_{SI}	p_{NO}	Impureza de Gini
Fuerte	4	4	8	0,50	0,50	0,50
Débil	6	1	7	0,86	0,14	0,76

$$Gini(Humedad) = 0,5 \cdot \left(\frac{8}{15}\right) + 0,76 \cdot \left(\frac{7}{15}\right) = 0,62$$

Tabla 6.5: Impureza de Gini para las categorías de Viento

Viento	SI	NO	# observaciones	p_{SI}	p_{NO}	Impureza de Gini
Fuerte	4	3	7	0,57	0,43	0,49
Débil	6	2	8	0,75	0,25	0,38

$$Gini(Viento) = 0,49 \cdot \left(\frac{7}{15}\right) + 0,38 \cdot \left(\frac{8}{15}\right) = 0,43$$

En la Tabla 6.6 se puede ver que la impureza de Gini para las tres variables incluidas en el ejemplo. La variable con la menor impureza de Gini, el *Tipo de día*, es la elegida para ser el nodo raíz del árbol de clasificación.

6.3. Árboles de clasificación

115

Tabla 6.6: Impureza de Gini para las variables de entrada

Variable	Impureza de Gini
Tipo de día	0,29
Humedad	0,62
Viento	0,43

Al entrenar un árbol de decisión, se repite este proceso, y a la hora de dividir cada nodo, se elige el atributo que proporcione el menor $Gini_\varphi(X)$.

Para obtener la ganancia de información para una variable, las impurezas ponderadas de los nodos hijos se restan de la impureza del nodo padre. La ganancia de Gini para la variable X, $\Delta Gini()$, se calcula así:

$$\Delta Gini(\varphi) = Gini(X) - Gini_\varphi(X) \quad (6.3)$$

Siguiendo el ejemplo del árbol de clasificación, para saber si se puede jugar al tenis o no, se tendría que obtener la impureza de Gini para el nodo *Humedad* o el nodo *Viento*. Repitiendo el proceso anteriormente mostrado, dado que el *Tipo de día* sea soleado, se obtienen los resultados de la Tabla 6.7.

Tabla 6.7: Impureza de Gini para las variables en días soleados

Variable	Impureza de Gini
Humedad	0,00
Viento	0,44

Entonces, la ganancia de Gini para cada variable será:

$$\begin{aligned} \Delta Gini(Humedad) &= 0,45 - 0 = 0,45 \\ \Delta Gini(Viento) &= 0,45 - 0,45 = 0 \end{aligned}$$

Puede observarse que la ganancia de información al dividir por *Humedad* es mayor que al hacerlo por *Viento*, por lo que el árbol se dividirá respecto a la *Humedad*, como se observó en la Fig. 6.1.

6.3.1.2. Entropía

La entropía es un concepto matemático que mide la incertidumbre de una fuente de información, es decir, la varianza en los datos entre diferentes clases. Para cada nodo y su partición, la entropía se calcula como:

$$E = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \quad (6.4)$$

donde p_1 y p_2 representan la probabilidad de pertenecer a cada una de las clases en ese nodo. En teoría de la información, la base logarítmica varía dependiendo de la aplicación, y con ella varía la unidad de medida. En este caso, la ganancia de información se obtiene como:

$$IG = E_{\pi} - E_{\pi+1}, \quad (6.5)$$

donde E_{π} representa la entropía en el nodo padre, mientras que $E_{\pi+1}$ es la entropía en el nodo que resulta de dividir el nodo padre. Entonces, siguiendo el ejemplo basado en los datos de la Tabla 6.3 se tendría que la entropía en origen es:

$$E = -\frac{10}{15} \log_2\left(\frac{10}{15}\right) - \frac{5}{15} \log_2\left(\frac{5}{15}\right) = 0,9183$$

Si se obtiene la entropía para cada variable, se determinará el nodo raíz para aquel que aporte una mayor ganancia de información. En el caso de la variable *Tipo de día* se calcula:

$$\begin{aligned} E_{Soleado} &= -\frac{2}{6} \log_2\left(\frac{2}{6}\right) - \frac{4}{6} \log_2\left(\frac{4}{6}\right) = 0,9183 \\ E_{Nublado} &= -\frac{4}{4} \log_2\left(\frac{4}{4}\right) - \frac{0}{4} \log_2\left(\frac{0}{4}\right) = 0 \\ E_{Lluvia} &= -\frac{4}{5} \log_2\left(\frac{4}{5}\right) - \frac{1}{5} \log_2\left(\frac{1}{5}\right) = 0,7219 \end{aligned}$$

Y por tanto:

$$E_{\text{Tipo de día}} = \frac{6}{15} \cdot 0,9183 + \frac{4}{15} \cdot 0 + \frac{5}{15} \cdot 0,7219 = 0,608$$

Repetiendo el mismo procedimiento con las variables *Viento* y *Humedad* se puede comprobar que $E(Viento) = 0,893$ y $E(Humedad) = 0,809$. A partir de esto se puede obtener la ganancia de información como:

$$IG_{\text{Tipo de día}} = E - E_{\text{Tipo de día}} = 0,918 - 0,608 = 0,31$$

$$IG_{\text{Viento}} = E - E_{\text{Viento}} = 0,918 - 0,893 = 0,025$$

$$IG_{\text{Humedad}} = E - E_{\text{Humedad}} = 0,918 - 0,809 = 0,109$$

Se puede comprobar que la disminución de la aleatoriedad, o la ganancia de información, es mayor para la variable *Tipo de día* y por tanto se elige para ser el nodo raíz. Repitiendo este proceso se va construyendo el árbol hasta alcanzar los nodos terminales.

6.3.2. Sobreajuste

Ya se ha comentado en la Sec. @ref(intro_dectree) que una de las principales desventajas de los árboles de decisión es su propensión a sobreajustar el modelo al conjunto de datos de entrenamiento y, por tanto, hay que prestar especial atención a la complejidad del modelo. Basándose en las observaciones utilizadas en la fase de entrenamiento, un árbol de decisión puede extraer los patrones presentes en el conjunto de observaciones de entrenamiento y ser muy preciso en el ajuste de dichas observaciones. Sin embargo, puede ocurrir que el árbol resultante no sea capaz de clasificar correctamente ni el conjunto de validación ni nuevas observaciones. Esta circunstancia puede ocurrir porque haya patrones no observados en los datos de entrenamiento que el modelo no es capaz de detectar, o porque la división de los datos entre entrenamiento y validación no se realizó correctamente siendo los datos de entrenamiento no representativos del conjunto de datos completo. Intentando que el árbol entrenado tenga la capacidad de aprender patrones muy complejos, se puede producir este sobreajuste materializado con árboles muy profundos. La forma de evitar el sobreajuste es controlar el crecimiento del árbol para evitar que se vuelva excesivamente complejo.

6.3.3. ¿Cuánto debe crecer un árbol de clasificación?

En cada paso de construcción del árbol se determina la variable óptima para realizar la división de las observaciones de un nodo padre en sus nodos hijos. La pregunta es: ¿cuándo se detiene?, ¿cuál es el criterio de parada? Por ejemplo, se puede utilizar como criterio de parada que el árbol alcance un tamaño o profundidad determinado, para que no sea excesivamente complejo y así no tengan lugar las consecuencias derivadas del sobreajuste.

En consecuencia, se debe llegar a un equilibrio entre la profundidad y complejidad del árbol para optimizar la predicción de futuras observaciones. Este equilibrio se puede lograr siguiendo alguno de los siguientes enfoques: la parada temprana o la poda.

6.3.3.1. La parada temprana

La parada temprana restringe el crecimiento del árbol, tanto de clasificación como de regresión, de forma explícita. Existen distintas maneras de establecer esta restricción al árbol, pero dos de las técnicas más populares son las de restringir la profundidad a un cierto nivel o la de establecer un número mínimo de observaciones permitidas en un nodo terminal. En el primer caso, el árbol deja de dividirse al llegar a cierta profundidad. Así, cuanto menos profundo sea el árbol, menos variación habrá en las predicciones que proporcione. Sin embargo, existe el riesgo de introducir mucho sesgo al modelo al no ser capaz de captar interacciones y patrones complejos en los datos. El segundo enfoque lo que provoca es que no se dividan nodos intermedios con pocas observaciones. En el caso extremo, si se permite que un nodo terminal sólo contuviese una observación esta actuaría como predicción. De este modo, los resultados probablemente no serían generalizables y tendrían mucha variabilidad. En el otro extremo, si se exigen un gran número de observaciones en el nodo terminal se reduce el número de divisiones y, por lo tanto, se reduce la varianza.

6.3.3.2. La poda

El otro enfoque es el de la poda que consiste en construir un árbol muy profundo y complejo y después podarlo para encontrar el subárbol óptimo. Este subárbol se obtiene utilizando un hiperparámetro de complejidad (ζ) que penaliza la función objetivo de la partición por el número de nodos terminales del árbol (τ), es decir, se busca minimizar:

$$R_\zeta(\tau) = R(\tau) + \zeta|\tau| \quad (6.6)$$

Donde $R(\tau)$ es el error total de entrenamiento de los nodos, $|\tau|$ es el número total de nodos y ζ es el hiperparámetro de complejidad. A medida que ζ aumenta, más ramas del árbol son podadas. Mientras que a valores más bajos, los modelos producidos son más complejos y en consecuencia más grandes. En conclusión, a medida que un árbol crece, el error de entrenamiento debe tener una reducción mayor que la penalización por la complejidad.

6.3.4. Ejemplo: Árbol de clasificación para determinar la intención de compra

A continuación se describe el caso que se va a resolver mediante modelos de clasificación tanto en este como en los siguientes capítulos. Existen diversas aserciones para definir Comercio Electrónico (CE). Entre ellas, la Organización para la Cooperación y el Desarrollo Económico (OCDE) lo define como el proceso de compra, venta o intercambio de bienes, servicios e información a través de redes de comunicación, comúnmente Internet. La clasificación más básica del CE se hace en base al tipo de entes que se relacionan: empresas (businesses, B), consumidores (consumers, C) y entes públicos (governments, G). De esta forma, una empresa de CE convencional suele ser B2B si vende a otras empresas, B2G si su relación comercial es con administraciones o B2C si vende a consumidores finales.

En este caso, se puede considerar que la empresa “Beauty eSheep” lleva a cabo un CE de tipo B2C. Su producto estrella es una crema hidratante unisex, denominada internamente como “Crema Luxury”, con mucho éxito entre su clientela. A partir de este producto inicial, la empresa ha ido ofreciendo un catálogo de productos tanto de belleza como de bienestar y salud.

Hace tiempo la empresa instauró una estrategia relacional, centrada en el cliente, de tal manera que ha ido recabando diversos datos sobre los mismos, incluidas las distintas compras que han realizado.

Basándose en los datos recopilados para cada cliente, la empresa quiere realizar una campaña para impulsar la venta de tensiómetros digitales. La empresa tiene acceso a un stock muy flexible en fechas de envío de estos productos y el precio de los tensiómetros es muy bueno, por lo que se espera una buena rentabilidad en su venta.

Por tanto, en este proyecto hay que identificar el público objetivo susceptible de comprar dicho producto para ofrecérselo a través de la plataforma de CE de la compañía, SMS y/o webmail durante el periodo que dura la campaña.

La tabla con los datos integrados a nivel de cliente, incluyendo el consumo de los distintos productos de la empresa, es **dp_ENTR**, incluida en el paquete **CDR**, y que se resume en la

6.3. Árboles de clasificación

119

Tabla 6.8. Este ejemplo se va a replicar en el resto de capítulos de machine learning supervisado para clasificación.

Tabla 6.8: Descripción de las variables del conjunto de datos **dp_entr**.

COLUMNA	TIPO	DESCRIPCIÓN
CLS_PRO_pro13	Factor	Clase objetivo, es un indicador de si el cliente es consumidor de ese producto “Tensiómetro Digital” (‘S’) o no (‘N’)
ind_pro11	Factor	Indicador de si el cliente es consumidor del producto “Fragancia Luxury” (‘S’) o no (‘N’)
ind_pro12	Factor	Indicador de si el cliente es consumidor del producto “Depiladora Eléctrica” (‘S’) o no (‘N’)
ind_pro14	Factor	Indicador de si el cliente es consumidor del producto “Crema Luxury” (‘S’) o no (‘N’)
ind_pro15	Factor	Indicador de si el cliente es consumidor del producto “Smartwatch Fitness” (‘S’) o no (‘N’)
ind_pro16	Factor	Indicador de si el cliente es consumidor del producto “Kit Pesas Inteligentes” (‘S’) o no (‘N’)
ind_pro17	Factor	Indicador de si el cliente es consumidor del producto “Estimulador Muscular” (‘S’) o no (‘N’)
importe_pro11	Doble	Importe neto global gastado por el cliente en ese producto en euros
importe_pro12	Doble	Importe neto global gastado por el cliente en ese producto en euros
importe_pro14	Doble	Importe neto global gastado por el cliente en ese producto en euros
importe_pro15	Doble	Importe neto global gastado por el cliente en ese producto en euros
importe_pro16	Doble	Importe neto global gastado por el cliente en ese producto en euros
importe_pro17	Doble	Importe neto global gastado por el cliente en ese producto en euros
edad	Entero	Edad del cliente
tamano_fam	Entero	Número de miembros de la unidad familiar a la que pertenece el cliente incluyéndolo a él mismo
anos_exp	Entero	Años de trabajo del cliente

COLUMNA	TIPO	DESCRIPCIÓN
ingresos_ano	Doble	Ingresos anuales del cliente en euros
des_nivel_edu	Factor	Descripción del nivel de educación del cliente

Se construye un árbol de clasificación utilizando el conjunto de entrenamiento, como se ha comentado, sin transformar (en su escala original) mediante el algoritmo CART implementado en el paquete `rpart` con Árboles de Regresión y Partición Recursiva (Recursive Partitioning and Regression Trees, RPART) que se puede usar tanto para regresión como para clasificación.

```
library("CDR")
library("reshape")
library("caret")
library("rpart")
library("rpart.plot")
library("ggplot2")

data("dp_entr")
head(dp_entr)

  ind_pro11 ind_pro12 ind_pro14 ind_pro15 ind_pro16 ind_pro17 importe_pro11
1      S       N       S       S       S       N      157
497    N       N       S       N       S       N       0
265    N       N       S       S       S       S       0
534    N       S       S       N       N       N       0
415    N       S       S       N       S       N       0
298    S       N       S       N       N       N      115
  importe_pro12 importe_pro14 importe_pro15 importe_pro16 importe_pro17 edad
1          0        40      200      180      0     49
497        0       240       0      180      0     38
265        0       425      200      180     300     61
534       120       60       0       0      0     47
415       120      133       0      180      0     34
298        0       220       0       0      0     43
  tamano_fam anos_exp ingresos_ano des_nivel_edu CLS_PRO_pro13
1         4       24     30000      MEDIO       S
497       2       12     53000      MEDIO       N
265       4       37    172000     BASICO       S
534       3       21     38000      MEDIO       N
415       1       10     38000     BASICO       N
298       2       18     60000      ALTO       N

trControl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)
```

6.3. Árboles de clasificación

121

En primer lugar, se carga la librería necesaria para entrenar el modelo, así como los datos de compras de los clientes. En este caso se usa el método de remuestreo de validación cruzada con 10 folds, visto en el Cap. ???. A continuación, se determina la semilla aleatoria para que los resultados sean replicables, y se entrena el modelo.

```
# se fija una semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13 ~ ., # . equivale a incluir todas las variables
                data=dp_entr,
                method="rpart",
                metric="ROC",
                trControl=trControl)
```

```
model
CART

558 samples
17 predictor
 2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:

      cp        ROC        Sens        Spec
0.05017921  0.8172123  0.9214286  0.7026455
0.10394265  0.7559406  0.8386243  0.6914021
0.51971326  0.6347222  0.8564815  0.4129630

ROC was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.05017921.
```

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

Los resultados de validación cruzada quedan recogidos en los boxplot, por lo que se puede ver los valores entre los que oscilan las principales medidas en los 10 folds del proceso de validación. Estas medidas (ROC, sensibilidad y especificidad) se definieron en el Cap. ???, y en el caso de árboles de clasificación se utilizan para medir la precisión del modelo. A continuación se

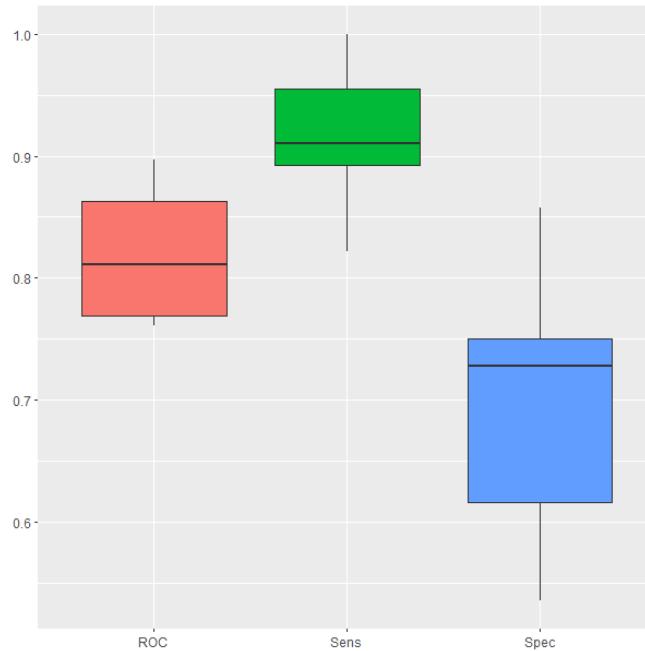


Figura 6.3: Resultados del modelo durante la validación cruzada.

muestra el árbol generado. Se puede observar que este árbol es muy sencillo, y por tanto es fácil obtener su interpretación. En primer lugar decide si un cliente que compra el *smartchwatch fitness* comprará el nuevo producto. En caso de no comprar el *smartchwatch fitness* (No a `ind_pro15S=1`), pero sí compra la *depiladora eléctrica* (Yes a `ind_12S=1`) sí comprará el *tensiómetro digital*. Si no compra ninguno de esos dos productos no comprará el nuevo producto.

```
# Gráfico del árbol obtenido
rpart.plot(model$finalModel)
```

Este modelo se puede mejorar ajustando automáticamente el hiperparámetro incluido en `rpart` para el entrenamiento de árboles de decisión. Los hiperparámetros son los valores utilizadas durante el proceso de entrenamiento en la configuración del modelo. Por consiguiente, primero es necesario conocer el hiperparámetro a optimizar en el algoritmo implementado en R que estemos usando. Esto se consigue mediante la siguiente instrucción incluida en el paquete `caret`:

```
modelLookup("rpart")
model parameter          label forReg forClass probModel
1 rpart      cp Complexity Parameter   TRUE    TRUE    TRUE
```

El hiperparámetro a optimizar es la complejidad del árbol, `cp`, que es un hiperparámetro que se aplica en la fase de parada durante la construcción del árbol. Según se ha comentado, esta

6.3. Árboles de clasificación

123

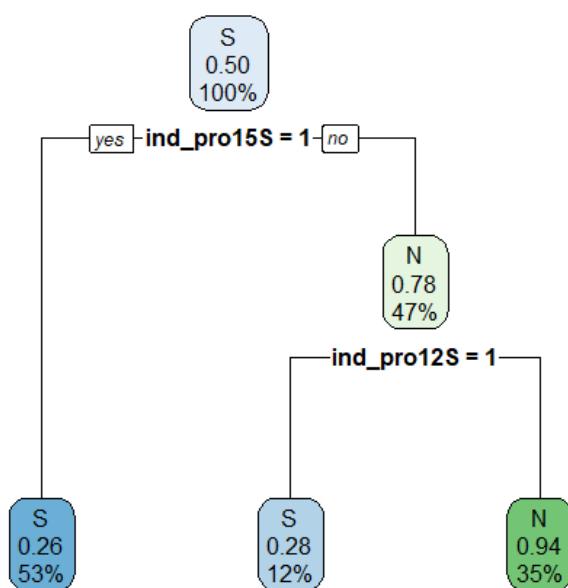


Figura 6.4: Árbol de clasificación sin ajuste automático de hiperparámetros.

fase tiene como función principal evitar desarrollar divisiones que no valgan la pena. Se puede entender `cp` como la mejora mínima necesaria en cada nodo del modelo. Es necesario definir los valores de `cp` que se quieren evaluar con el objetivo de obtener su valor óptimo.

```
# Se especifica un rango de valores típicos para el hiperparámetro
tuneGrid <- expand.grid(cp = seq(0.01,0.05,0.01))
```

```
# se entrena el modelo
set.seed(101)

model <- train(CLSPRO_pro13 ~ .,
  data=dp_entr,
  method="rpart",
  metric="ROC",
  trControl=trControl,
  tuneGrid=tuneGrid)
```

```
model
CART

558 samples
 17 predictor
  2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:
```

cp	ROC	Sens	Spec
0.01	0.8962254	0.8678571	0.8167989
0.02	0.8663454	0.9000000	0.7667989
0.03	0.8458097	0.9392857	0.7310847
0.04	0.8449381	0.9214286	0.7383598
0.05	0.8172123	0.9214286	0.7026455

```
ROC was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01.
```

De forma automática se construyen diversos árboles para cada uno de los valores explicitados del parámetro `cp`. Para cada uno de esos árboles se obtienen las correspondientes métricas de precisión: el área bajo la curva (denotada como ROC, por las siglas en inglés de *Receiver Operating Characteristic*), sensibilidad (Sens) y especificidad (Spec), todas ellas definidas en el Cap. ???. El valor ROC es el utilizado para la elección del valor óptimo de `cp`, por lo que se determina que finalmente el óptimo es $cp = 0,01$ al maximizar el valor ROC alcanzando un 89,6 %. Por tanto, ajustando el hiperparámetro se ha aumentado la precisión del modelo en casi un 8 % respecto al 81,7 % que tenía el modelo sin ajustar automáticamente el valor de `cp`.

6.3. Árboles de clasificación

125

En la Fig. 6.5 se puede ver el rendimiento de cada una de las métricas del árbol entrenado utilizando validación cruzada. Dicha figura se obtiene con la siguiente instrucción:

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_violin(show.legend=FALSE) +
  xlab(NULL) +
  ylab(NULL)
```

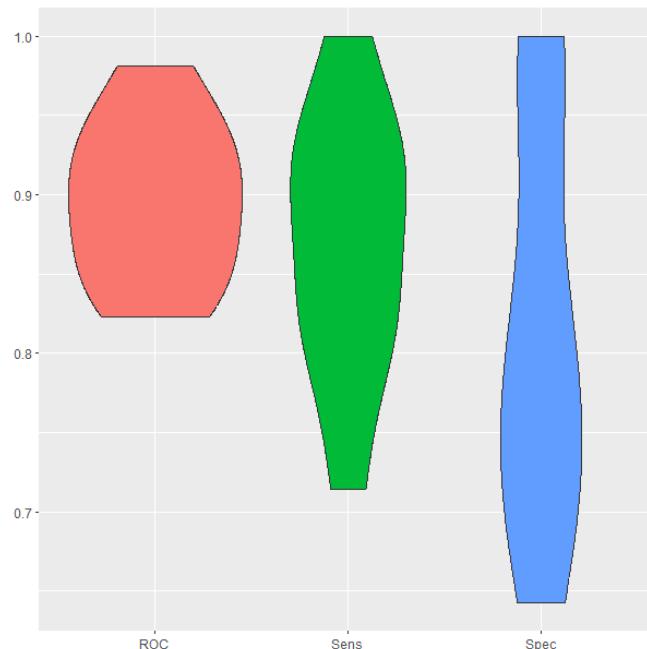


Figura 6.5: Resultados del modelo con ajuste automático durante la validación cruzada

En la Fig. 6.6 se muestra el árbol generado. Dicha visualización se ha obtenido con el siguiente código:

```
# Gráfico del árbol obtenido
rpart.plot(model$finalModel)
```

Con el objetivo de aumentar la generalidad del árbol y facilitar su interpretación, se procede a reducir su tamaño podándolo. Para ello se establece que un nodo terminal tenga como mínimo 50 observaciones, dando lugar al árbol que se muestra en la Fig. 6.7.

```
set.seed(101)
prunedtree <- rpart(CLSPRO_pro13 ~ ., data=dp_entr,
                      cp= 0.01, control = rpart.control(minbucket = 50))
```

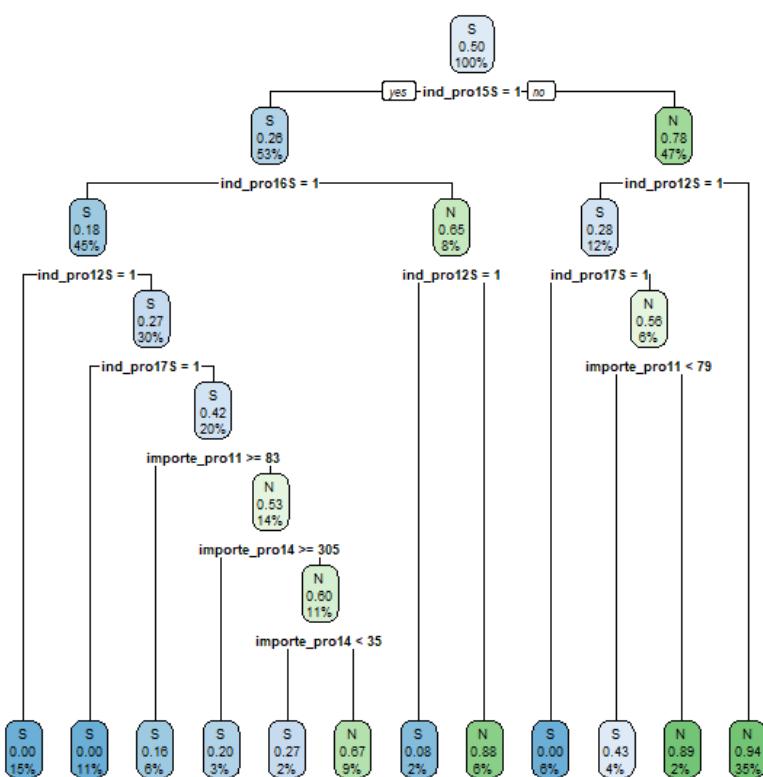


Figura 6.6: Árbol de clasificación con ajuste automático.

6.4. Árboles de regresión

127

```
rpart.plot(prunedtree)
```

El árbol ha reducido el número de nodos terminales, en tres de ellos el árbol predice que un cliente comprará el nuevo producto si:

1. Compra el *smartwatch fitness* (*ind_pro15* = S - Yes) y la *depiladora eléctrica* (*ind_pro12* = S - Yes).
2. Compra el *smartwatch fitness* (*ind_pro15* = S - Yes) y el *estimulador muscular* (*ind_pro17* = S - Yes), pero no la *depiladora eléctrica* (*ind_pro12* = S - No).
3. No compra el *smartwatch fitness* (*ind_pro15* = S - No), pero si la *depiladora eléctrica* (*ind_pro12* = S - Yes).

Sin embargo, dos nodos terminales predicen que el cliente no compra el nuevo producto si:

1. Compra el *smartwatch fitness* (*ind_pro15* = S - Yes), pero no la *depiladora eléctrica* (*ind_pro12* = S - No) ni el *estimulador muscular* (*ind_pro17* = S - No).
2. No compra el *smartwatch fitness* (*ind_pro15* = S - No) ni la *depiladora eléctrica* (*ind_pro12* = S - No).

6.4. Árboles de regresión

Como se ha comentado, los árboles de decisión también pueden ser usados para resolver problemas de regresión. En este caso, la idea es que la predicción dada en cada hoja sea un valor numérico en lugar de un valor de una categoría. En la Tabla 6.9 se muestran los datos para un problema de regresión equivalente al presentado en secciones anteriores para clasificación. Como ya se ha mencionado, la variable objetivo (*Horas jugadas*) ahora es continua en lugar de categórica, como ocurría en el ejemplo anterior con la variable *Decisión*.

Tabla 6.9: Datos de Horas jugadas dada la climatología del día

Día	Tipo de día	Humedad	Viento	Horas jugadas
1	Soleado	Fuerte	Débil	2,3
2	Soleado	Fuerte	Fuerte	1,5
3	Lluvia	Fuerte	Débil	1,3
4	Nublado	Fuerte	Débil	2,4
5	Lluvia	Débil	Débil	1,9
6	Lluvia	Débil	Fuerte	2,4
7	Soleado	Fuerte	Débil	2,3
8	Nublado	Débil	Fuerte	2,2

Día	Tipo de día	Humedad	Viento	Horas jugadas
9	Soleado	Débil	Débil	1,3
10	Lluvia	Débil	Débil	1,8
11	Soleado	Débil	Fuerte	1,2
12	Nublado	Fuerte	Fuerte	2,9
13	Nublado	Débil	Débil	2,2
14	Lluvia	Fuerte	Fuerte	1,5
15	Soleado	Fuerte	Fuerte	1,5

Se pueden calcular medidas descriptivas de la variable respuesta, *Horas jugadas*, como la media, varianza, desviación típica y coeficiente de variación siendo estas:

$$\bar{x}_{\text{Horas jugadas}} = \frac{1}{n} \sum x = 1,91 \quad (6.7)$$

$$\sigma_{\text{Horas jugadas}}^2 = \frac{\sum (x - \bar{x})^2}{n} = 0,25 \quad (6.8)$$

$$\sigma_{\text{Horas jugadas}} = \sqrt{\sigma^2} = 0,50 \quad (6.9)$$

$$CV_{\text{Horas jugadas}} = \frac{\sigma}{\bar{x}} = 0,26 \quad (6.10)$$

6.4.1. ¿Cómo se va formando el árbol de regresión?

Mientras que en los árboles de clasificación se utilizaba la entropía o la impureza de Gini para medir la homogeneidad de un nodo, en los árboles de regresión se utiliza como métrica la desviación típica (σ). Por tanto, cuando se selecciona una variable para hacer la división, se calcula la desviación típica para cada una de las ramas, y se obtiene una media ponderada en función del número de elementos de cada una de ellas. Esta media ponderada se calcula del siguiente modo:

$$\sigma_X = \sum_{r \in X} P(r) \cdot \sigma_r \quad (6.11)$$

Donde X es la variable de la cual se quiere obtener la desviación típica y r son las ramas que crecen desde este nodo. Para llevar a cabo la división, en primer lugar se debe calcular para cada nodo su desviación típica. A continuación, se seleccionan posibles variables para hacer la división y se obtiene su desviación típica. Para cada una de estas variables se calcula el decremento de la desviación, y se selecciona aquel que introduzca la mayor reducción.

Por ejemplo, para los datos mostrados en la Tabla 6.9, la desviación típica es 0,50 Horas jugadas, como se calculó en la ecuación (6.9), y el árbol se construiría como se muestra a continuación.

6.4. Árboles de regresión

129

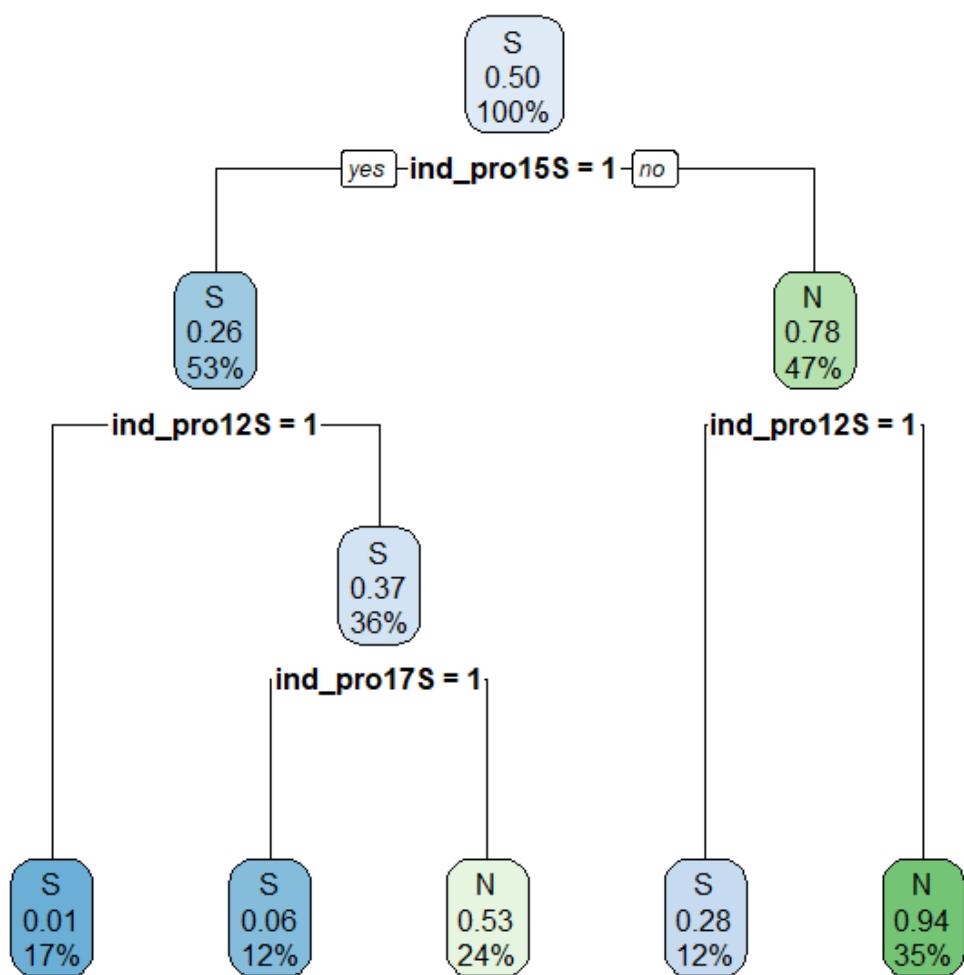


Figura 6.7: Árbol de clasificación con ajuste automático y podado.

En primer lugar, se selecciona *Tipo de día*, *Humedad* y *Viento* como candidatos a nodo raíz y se obtiene su desviación típica tal que:

Tabla 6.10: Desviación típica en las ramas de la variable *Tipo de día*

Tipo de día	# observaciones	$\sigma_{\text{Horas jugadas}}$
Soleado	6	0,45
Nublado	4	0,29
Lluvia	5	0,38

Tabla 6.11: Desviación típica en las ramas de la variable *Humedad*

Humedad	# observaciones	$\sigma_{\text{Horas jugadas}}$
Fuerte	8	0,55
Débil	7	0,43

Tabla 6.12: Desviación típica en las ramas de la variable *Viento*

Viento	# observaciones	$\sigma_{\text{Horas jugadas}}$
Fuerte	7	0,57
Débil	8	0,42

A partir de las desviaciones típicas en las ramas de cada variable, se puede obtener la desviación típica de cada variable de acuerdo a la ecuación (6.11). Además, se calcula la reducción de desviación típica como la diferencia entre la desviación de la variable respuesta y la desviación si se divide el conjunto de datos en base a alguna de las variables. Tanto la desviación típica de cada variable como el decremento en la desviación que producen se muestran en la Tabla 6.13.

Tabla 6.13: Desviación típica y decremento de desviación de cada variable

Variable	$\sigma_{\text{Horas jugadas}}$	Decremento
Tipo de día	0,38	0,12
Humedad	0,49	0,00
Viento	0,49	0,00

Dado que la variable *Tipo de día* es la que produce una mayor reducción en la desviación típica, resulta elegida como nodo raíz. El árbol seguiría creciendo repitiendo este proceso. Por ejemplo, se muestra cuál sería la siguiente división desde la rama soleado que crece desde nodo *Tipo de día*

6.4. Árboles de regresión

131

Tabla 6.14: Desviación típica en las ramas de la variable *Humedad* en días soleados

Humedad	# observaciones	$\sigma_{\text{Horas jugadas}}$
Fuerte	4	0,4
Débil	2	0,05

Tabla 6.15: Desviación típica en las ramas de la variable *Viento*

Viento	# observaciones	$\sigma_{\text{Horas jugadas}}$
Fuerte	3	0,14
Débil	3	0,47

En la Tabla 6.16 se muestra la desviación típica para cada variable así como la reducción de desviación que produce. Por tanto, la siguiente división se realizaría con la variable *Humedad*.

Tabla 6.16: Desviación típica y decremento de desviación de cada variable en la rama soleado

Variable	$\sigma_{\text{Horas jugadas}}$	Decremento
Humedad	0,28	0,17
Viento	0,31	0,14

6.4.2. ¿Cuánto debe crecer el árbol de regresión?

Como ocurría en los árboles de clasificación, es necesario establecer reglas que pongan fin al proceso de crecimiento del árbol. Además de los criterios de parada que se utilizan en árboles de clasificación (número de elementos mínimos en un nodo y nivel máximo del árbol), en árboles de regresión se detiene su crecimiento estableciendo un *threshold* (umbral de decisión) sobre el coeficiente de variación del nodo. En el ejemplo expuesto sobre Horas jugadas, se puede ver qué nodos podrían seguir creciendo si se establece que el árbol continúe creciendo en nodos con un coeficiente de variación de un 15 % o más, y que tenga al menos 5 observaciones en el nodo.

Tabla 6.17: Medidas para decidir si el árbol sigue creciendo

Nodo padre	Rama	CV en nodo hijo	# observaciones
Tipo de día	Nublado	11,80 %	4
Tipo de día	Lluvia	21,14 %	5
Humedad	Fuerte	21,04 %	4
Humedad	Débil	4,04 %	2

En este ejemplo, el árbol seguiría creciendo por la rama *Lluvia* donde habría que seleccionar la siguiente variable de división. En el resto de ramas, no se supera el número mínimo establecido de observaciones en el nodo, y en ocasiones tampoco se alcanza el coeficiente de variación mínimo. Por otra parte, en los árboles de regresión la poda se lleva a cabo del mismo modo que para árboles de clasificación. En la ecuación (6.6) se mediría el error de entrenamiento a través de la suma de los cuadrados de los errores (en inglés *Sum of Squared Estimate of Errors*, SSE), es decir:

$$SSE_\zeta(\tau) = SSE(\tau) + \zeta|\tau| \quad (6.12)$$

6.4.3. Árbol de regresión para estimar el número de días de hospitalización

En este ejemplo se utilizan los datos *cleveland*, contenidos en el paquete CDR, y que han sido utilizados en el Cap. ?? para estimar la variable *dhosp*. El conjunto de datos contiene información sobre pacientes que llegan a un hospital con dolor de pecho y de los cuales se han recogido distintas características. Se pretende predecir el número de días de hospitalización que necesitará un paciente en base al resto de características observadas: si el paciente está diagnosticado de accidente coronario, su edad, su sexo, el tipo de dolor que padece y la depresión en el segmento ST inducida por ejercicio en relación al reposo.

```
# se cargan los datos
data("cleveland")

# se entrena el modelo
set.seed(101)
model <- rpart(dhosp ~ diag + edad + sexo + tdolor + dep,
                data=cleveland, method="anova")
```

```
model$cptable

      CP nsplit rel error     xerror      xstd
1 0.37275022    0 1.0000000 1.0128283 0.09213359
2 0.01674747    1 0.6272498 0.6427926 0.06048143
3 0.01132433    4 0.5770074 0.6788431 0.06681871
4 0.01007684    6 0.5543587 0.6825792 0.06505426
5 0.01000000    7 0.5442819 0.6843192 0.06514439
```

Se observa que para valores muy altos del hiperparámetro de complejidad, el SSE es muy elevado. Esto es, produce modelos muy sencillos pero con nula potencia predictiva. En el otro extremo, para $\zeta = 0,01$ el SSE se minimiza hasta llegar a $SSE = 0,54$, por lo que el árbol se poda de acuerdo a la ecuación (6.12) con dicho valor de ζ . El resultado del modelo se muestra en el árbol de la Fig. 6.8. La interpretación de este árbol sería:

6.4. Árboles de regresión

133

1. Si el paciente no tiene diagnóstico de accidente coronario, solo necesitará un día de hospitalización.
2. En el caso de tener este diagnóstico, y una depresión mayor o igual a dos en el segmento ST inducida por ejercicio en relación al reposo, necesitará 2,8 días de hospitalización.
3. En un último ejemplo, si la depresión en el segmento ST inducida por ejercicio en relación al reposo está entre 0,35 y 2 entonces el paciente necesitará 3,8 días de hospitalización. Si por el contrario, la depresión en el segmento ST inducida por ejercicio en relación al reposo es menor a 0,35, el número de días de hospitalización depende del sexo del paciente: los hombres necesitarán 3,2 días y las mujeres tan solo 1,9 días.

```
# se pinta el árbol obtenido
rpart.plot(model)
```

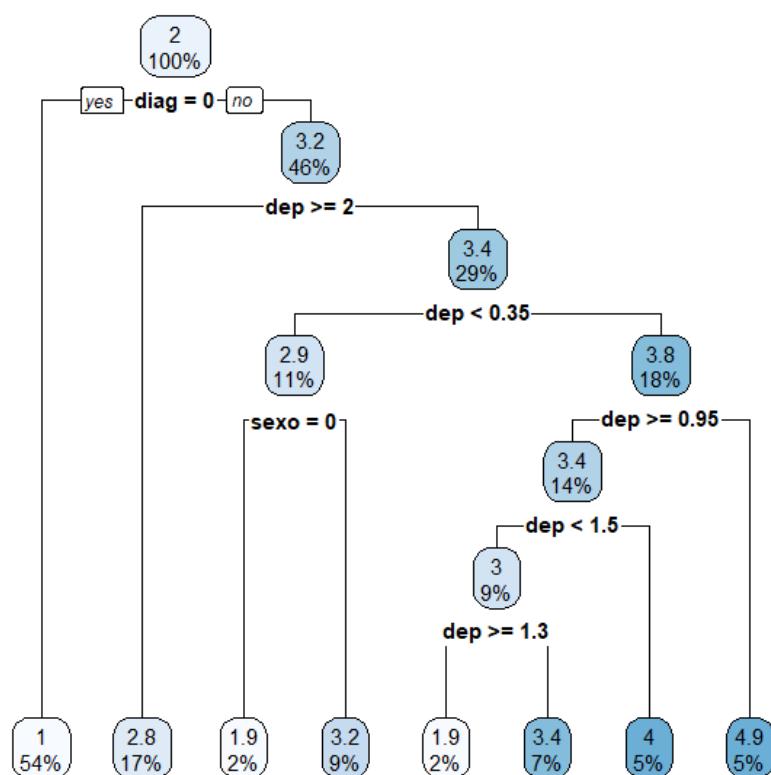


Figura 6.8: Árbol de regresión para predecir el número de días de hospitalización.

6.4.4. Árbol de regresión para la predicción del precio unitario de la vivienda en Madrid

En este ejemplo se va a entrenar un árbol de regresión para predecir el precio unitario de la vivienda en Madrid. Para ello, se van a utilizar los datos de viviendas a la venta en Madrid publicadas en Idealista durante el año 2018. Estos datos están incluidos en el paquete `idealista18`. Para facilitar la interpretación del modelo, sólo se van a utilizar 8 de las variables incluidas en el conjunto de datos: superficie construida, número de dormitorios, número de baños, si tiene terraza, si tiene ascensor, si el precio incluye el parking, distancia al centro de Madrid y distancia a una parada de metro.

```
library("idealista18")
data("Madrid_Sale")

Madrid_Sale <- Madrid_Sale |>
  dplyr::select(UNITPRICE, CONSTRUCTEDAREA, ROOMNUMBER, BATHNUMBER,
    HASTERRACE, HASLIFT, ISPARKINGSPACEINCLUDEDINPRICE,
    DISTANCE_TO_CITY_CENTER, DISTANCE_TO_METRO)

head(Madrid_Sale)

  UNITPRICE CONSTRUCTEDAREA ROOMNUMBER BATHNUMBER HASTERRACE HASLIFT
1 2680.851        47         1         1         0         1
2 4351.852        54         1         1         0         0
3 4973.333        75         2         1         0         0
4 5916.667        48         1         1         0         1
5 4560.000        50         0         1         0         0
6 3921.260       127         3         2         0         1
  ISPARKINGSPACEINCLUDEDINPRICE DISTANCE_TO_CITY_CENTER DISTANCE_TO_METRO
1                         0             8.0584293          0.8720746
2                         0             0.8763693          0.1163821
3                         0             0.9074793          0.1391088
4                         0             0.8454622          0.1442990
5                         0             1.2502313          0.3370982
6                         0             0.5417727          0.1614363
```

```
# Se entrena el modelo
library("rpart")
set.seed(101)
model <- rpart(UNITPRICE ~ ., Madrid_Sale, method = "anova")
```

Como en el ejemplo anterior, para $\zeta = 0,01$ el SSE se minimiza hasta llegar a $SSE = 0,56$, por lo que el árbol se poda de acuerdo a la ecuación (6.12) con dicho valor de ζ . El resultado del modelo se muestra en el árbol de la Fig. 6.9. La interpretación de este árbol sería:

1. Si una vivienda con ascensor se encuentra a menos de 3,2km del centro de Madrid y a menos de 0,46km de una estación de Metro, el precio unitario predicho para esa vivienda será de $5.248\text{€}/m^2$.

6.4. Árboles de regresión

135

2. Si una vivienda se encuentra a más de 3,2km del centro de Madrid y no tiene ascensor, el precio unitario predicho será de $2.160\text{€}/m^2$.
3. Si una vivienda se encuentra a menos de 3,2km del centro de Madrid y a más de 0,46km de una estación de Metro, el precio unitario predicho para esa vivienda será de $3.873\text{€}/m^2$.

```
# se pinta el árbol obtenido
rpart.plot(model)
```

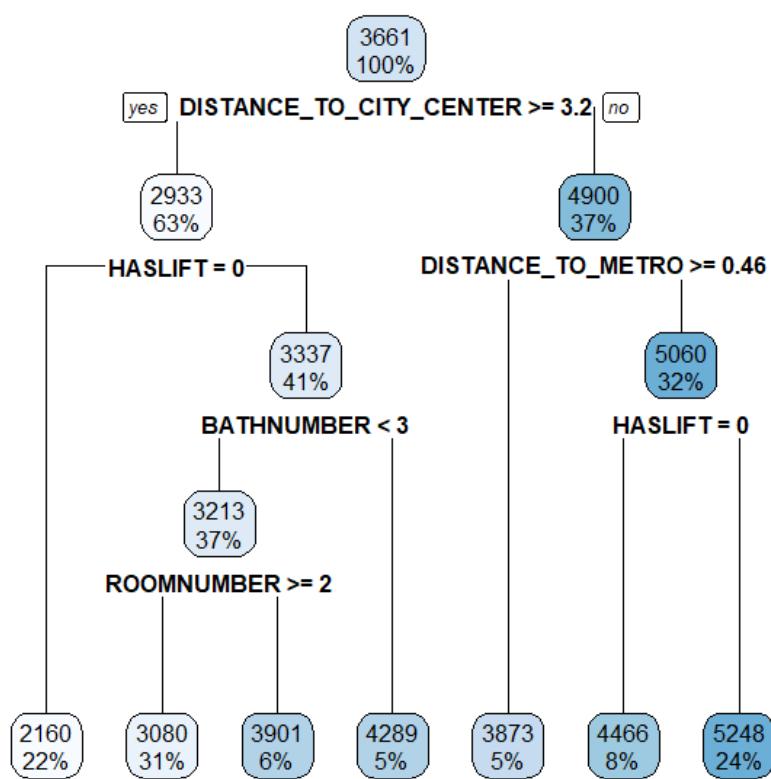


Figura 6.9: Árbol de regresión para predecir el precio unitario de las viviendas en Madrid.

Resumen

En este capítulo se introduce al lector en los árboles de decisión para clasificación y regresión, en particular:

- Se presenta la lógica para la construcción de árboles de decisión, ya sean de regresión o clasificación.
- Se contemplan diferentes medidas con las que el árbol decide avanzar hacia un nuevo punto de decisión.
- Se presentan los conceptos de sobreajuste y complejidad del árbol, así como la forma de controlarlos.
- Se muestra el uso de **R** para la clasificación de clases binarias y para la predicción de variables respuesta numéricas en casos aplicados.

Capítulo 7

Máquinas de vector soporte

Ramón A. Carrasco^a e Itzcóatl Bueno^{b,a}

^aUniversidad Complutense de Madrid ^bInstituto Nacional de Estadística

7.1. Introducción

Aunque las máquinas de vector soporte se desarrollaron en los años 90 dentro de la comunidad informática ((Boser et al., 1992), (Cortes and Vapnik, 1995)) como un método de clasificación binaria, su aplicación se ha extendido a problemas de clasificación múltiple y regresión. Como técnica de clasificación, las máquinas de vector soporte (SVM por sus siglas inglés *Support Vector Machines*) son similares a la regresión logística pero la SVM enfatiza en un margen de error aceptable en torno a la frontera de decisión.

En la Fig. 7.1 se muestra que la regresión logística divide las observaciones en dos clases de tal forma que se minimice la distancia entre los puntos y la **frontera de decisión** (A). Por otro lado, la frontera de decisión (B) de la SVM separa los datos en dos clases, pero maximizando la distancia entre esta y los puntos de ambas clases. El **margen** es la distancia entre la frontera de decisión y los puntos más cercanos. El margen es una parte clave del SVM, puesto que evita clasificaciones erróneas de casos futuros como podría pasar en el caso de la regresión logística y como se ilustra en la Fig. 7.2.

En resumen, los nuevos datos pueden ser clasificados dentro del margen. Cuanto mayor sea este margen, mayor será la capacidad para clasificar correctamente estos puntos. Por tanto, para obtener una clasificación errónea en la SVM es necesario que una observación se clasifique aún más allá del margen que en cualquier otro discriminante lineal. En problemas reales, es difícil que los discriminantes lineales, vistos en el Cap. 3, logren una línea que divida perfectamente las categorías a clasificar. Sin embargo, en la SVM, se incluye en la función objetivo (que mide la calidad del ajuste de los datos de entrenamiento) una penalización a los puntos que queden del lado equivocado del límite de decisión. En caso de que los datos puedan ser divididos linealmente, no se cometerá ninguna penalización y se maximizará el margen. Mientras que si

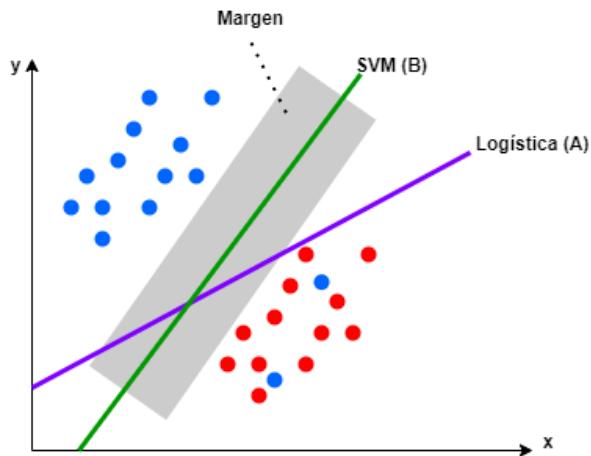


Figura 7.1: SVM vs Regresión logística.

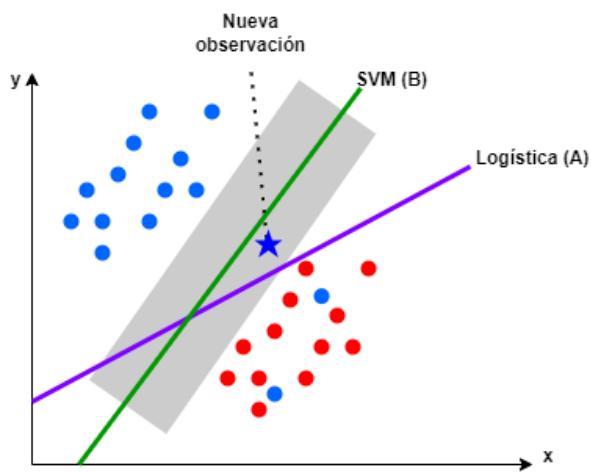


Figura 7.2: Nueva observación clasificada en SVM vs Regresión logística.

los datos no son linealmente separables, el mejor ajuste vendrá dado por el equilibrio entre una penalización del error total bajo y un margen de decisión grande. La penalización a una observación mal clasificada es proporcional a la distancia desde la frontera de decisión.

Sin embargo, la SVM también tiene desventajas reseñables. En primer lugar, la SVM no es adecuada en conjuntos de datos grandes porque la complejidad de entrenamiento es elevada. Además, la SVM no funciona bien cuando los datos tienen mucho ruido, es decir, cuando las clases se superponen. Finalmente, si el conjunto de datos de entrenamiento tiene más variables que observaciones, el rendimiento del modelo disminuirá.

7.2. Algoritmo SVM para clasificación binaria

El algoritmo por el que se obtiene un modelo SVM ([Vapnik, 1997](#)) se basa en la ecuación del hiperplano compuesta por dos hiperparámetros: un vector de números reales ω de la misma dimensión que el vector de variables de entrada x , y un número real b tal que:

$$\omega x - b = 0 \quad (7.1)$$

Donde ωx es $\omega^{(1)}x^{(1)} + \omega^{(2)}x^{(2)} + \dots + \omega^{(p)}x^{(p)}$ siendo p el número de variables incluidas en x . De este modo, la predicción para una instancia de x viene dada por:

$$y = sign(\omega x - b) \quad (7.2)$$

Siendo *sign* el operador que devuelve +1 para cualquier valor positivo y -1 para los valores negativos. Por tanto, el objetivo es ajustar los valores óptimos de ω y b para el algoritmo. Estos hiperparámetros se obtienen resolviendo un problema de optimización sujeto a las siguientes restricciones:

$$\omega x_i - b \geq 1 \text{ si } y_i = +1 \text{ y} \quad (7.3)$$

$$\omega x_i - b \leq 1 \text{ si } y_i = -1 \quad (7.4)$$

Además, el objetivo del problema de optimización es maximizar el margen en torno a la frontera de decisión. Para conseguir esto es necesario minimizar la norma euclídea, y, por tanto, el problema a resolver es:

$$\min \|\omega\| \text{ sujeto a}$$

$$y_i(\omega x_i - b) \geq 1 \text{ para } i = 1, \dots, N$$

7.3. ¿Y si tengo más de dos clases?

Hasta ahora se ha presentado la SVM como un algoritmo solo aplicable a la clasificación de dos clases pero ¿y si se tienen más de dos clases? En general, hay dos enfoques para resolver esto: **uno contra todos** (OVA, por *One Vs All*) y **uno contra uno** (OVO, por *One Vs One*). En el enfoque OVA, se ajusta una SVM para cada clase, es decir una clase contra las demás y se clasifica a la clase para la cual el margen es mayor. En cambio, en el enfoque OVO se ajustan todas las SVM por pares y se clasifica a la clase que gane las competiciones por pares.

7.4. Truco del kernel: tratando con la no linealidad

Las SVM funcionan muy bien si la separación entre clases es lineal. Sin embargo, si la separación es más compleja se intenta transformar el espacio en otro de mayor dimensionalidad donde las clases sí sean separables linealmente. Para ello, el modelo SVM se extiende incluyendo la función de pérdida (ℓ) “hinge” (([Gentile and Warmuth, 1998](#)),([Lee and Lin, 2013](#))) definida como:

$$\ell(y_i) = \max(0, 1 - y_i(\omega x_i - b)) \quad (7.5)$$

En machine learning, esta función de pérdida se utiliza para entrenar clasificadores, más concretamente para la clasificación por el margen máximo (métodos de clasificación binaria que se utiliza cuando hay una frontera lineal que separa perfectamente los datos de entrenamiento de una categoría de los de la otra), sobre todo para las SVM. La función de pérdida es cero cuando se cumplen las restricciones, es decir, si ωx_i es clasificado en el lado correcto de la frontera de decisión. Por otro lado, si un dato es mal clasificado, el valor obtenido con la función de pérdida es proporcional a la distancia hasta la frontera de decisión. Por tanto, el objetivo es minimizar la función de coste:

$$C\|\omega\|^2 + \frac{1}{N} \sum_{i=1}^N \max(0, 1 - y_i(\omega x_i - b)) \quad (7.6)$$

Donde C es un hiperparámetro que controla la compensación entre incrementar el tamaño de la frontera de decisión y asegurar que cada x_i sea clasificado en el lado correcto de la frontera de decisión.

Un modelo SVM que optimiza la función de pérdida se denomina SVM *soft-margin* mientras que el modelo original es conocido como SVM *hard-margin*. La ecuación (7.6) muestra que para valores grandes de C el segundo término es despreciable, por lo que el algoritmo ignorará por completo la clasificación errónea y tratará de obtener el mayor margen posible. Si se reduce el valor de C , se penaliza más cada error de clasificación, por lo que se cometerán menos errores sacrificando amplitud del margen.

A veces no es posible separar los datos por un hiperplano en su espacio original. Sin embargo, el **truco del kernel** utiliza una función que implícitamente transforma el espacio original a un espacio de mayor dimensión durante la optimización de la función de coste, como se muestra

7.4. Truco del kernel: tratando con la no linealidad

141

en la Fig. 7.3. Así, es posible transformar un espacio de datos bidimensional no separable linealmente en un espacio de datos tridimensional linealmente separable usando un mapeo específico definido por $\phi : x \rightarrow \phi(x)$ donde $\phi(x)$ es un vector de mayor dimensión que x . Sin embargo, no se conoce la función de mapeo que funcionará en los datos. Si se prueban todas las transformaciones posibles, podría ser ineficiente y no llegar a la resolución del problema de clasificación planteado.

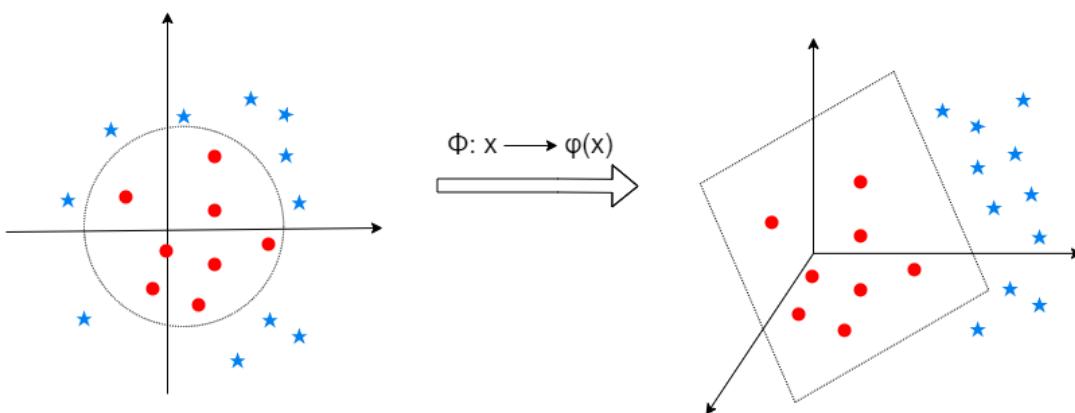


Figura 7.3: Izquierda: Las dos clases en el espacio original (2-D). Derecha: Las dos clases en el espacio de sobredimensionado (3-D).

Se puede trabajar eficientemente en espacios de mayor dimensión sin necesidad de hacer las transformaciones explícitamente. Utilizando el truco del *kernel* se puede evitar este proceso costoso de transformación de tal manera que se evita calcular el producto escalar reemplazándolo por una operación más simple con las variables originales que proporciona el mismo resultado. A continuación se explican algunos de estos operadores especiales, llamados *kernels*, que permiten llevar a cabo dicha transformación.

7.4.1. Algunos *kernels* populares

Los *kernels* ((Schölkopf et al., 1997), (Scholkopf et al., 1997)) más populares en el entrenamiento de SVM están incluidos dentro de la función `svm()` del paquete `e1071` donde se puede especificar en el hiperparámetro `kernel`. Estos kernel son:

- lineal: $K(u, v) = \langle u, v \rangle$
- polinomial de grado δ : $K(u, v) = \gamma(k_1 + \langle u, v \rangle)^\delta$
- base radial: $K(u, v) = e^{\gamma \|u-v\|^2}$
- sigmoidal: $K(u, v) = \tanh(\gamma \langle u, v \rangle + k_1)$

Donde $\langle u, v \rangle = \sum_{i=1}^n u_i v_i$ es el producto escalar. Cada uno de estos kernels tiene sus propios hiperparámetros, como δ o γ , que es necesario tunear para optimizar el rendimiento de la SVM.

En machine learning, el término **tunear**, hace referencia al hecho de ajustar automáticamente tratando de optimizar los hiperparámetros del algoritmo. A la hora de ajustar en **R** un modelo SVM se puede conocer los hiperparámetros a ajustar utilizando la función `modelLookup` de `caret`. Por ejemplo, para una SVM con kernel de base lineal se usaría así:

```
modelLookup("svmLinear")
  model parameter label forReg forClass probModel
1 svmLinear          C    Cost    TRUE    TRUE
```

El hiperparámetro que se puede ajustar en un modelo SVM con kernel de base lineal en **R** es el coste (C), el cual representa a la constante C en la ecuación (7.6).

7.5. Procedimiento con R: la función `svm()`

En el paquete `e1071` de R se encuentra la función `svm()` que se utiliza para entrenar un modelo máquinas vector soporte:

```
svm(x, y, scale = TRUE, type = NULL, kernel = ..., ...)
```

- `x`: conjunto de datos de entrenamiento que contiene los predictores
- `y`: vector respuesta con las clases o valores de la variable respuesta.
- `scale`: booleano que indica si es necesario escalar las variables.
- `type`: indica si se pretende resolver un problema de clasificación o de regresión.
- `kernel`: *kernel* utilizado durante el entrenamiento y la predicción.

7.6. Aplicación de un modelo SVM Radial con ajuste automático en R

Los datos utilizados para entrenar el modelo SVM en este capítulo se cargan desde la librería `CDR`. Además, para su entrenamiento se requieren las librerías `caret` y `e1071`.

```
library("CDR")
library("caret")
library("e1071")
library("reshape")
library("ggplot2")

data(dp_entr_NUM)
```

Se entrena un modelo SVM con kernel radial utilizando el conjunto de entrenamiento con todas las variables numéricas. Previamente, se aplica una normalización z-score, presentadas en el

7.6. Aplicación de un modelo SVM Radial con ajuste automático en R

143

Cap. ??, al conjunto de entrenamiento. De este modo, las variables que inicialmente tenían distintas escalas de medida, ahora todas se miden en la misma escala. Además, se ajustan automáticamente los hiperparámetros de dicho algoritmo durante el proceso de entrenamiento.

```
trControl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  preProcOptions = list("center"),
  summaryFunction = twoClassSummary
)

# Se especifica un rango de valores para los hiperparámetros
tuneGrid <- expand.grid(sigma = seq(from=0.1, to=0.2, by=0.05),
                           C = 10**(-2:4))
```

Se define como procedimiento de muestreo una validación cruzada, como la presentada en el Cap. ??, de 10 folds. Además, se le indica al modelo que debe calcular las probabilidades de clase en cada remuestreo en caso de estar entrenando un modelo de clasificación. Con el argumento `summaryFunction = twoClassSummary` se le indica al modelo que para resumir los resultados se calculen la sensibilidad, especificidad y el área bajo la curva ROC. Como se ha comentado, conviene estandarizar los datos, esto se le indica a la función a través del argumento `preProcOptions` con la opción `center`. A su vez, se define una red de hiperparametros a optimizar. A través de la función `train()` se ajusta automáticamente el modelo con los hiperparametros óptimos.

```
# Se fija la semilla aleatoria
set.seed(101)

# Se entrena el modelo
model <- train(CLSPRO_pro13 ~ .,
                 data=dp_entr_NUM,
                 method="svmRadial",
                 metric="ROC",
                 trControl=trControl,
                 tuneGrid=tuneGrid)

model

Support Vector Machines with Radial Basis Function Kernel

558 samples
 19 predictor
  2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
```

```
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:
```

sigma	C	ROC	Sens	Spec
0.10	1e-02	0.9553241	0.8785714	0.7071429
0.10	1e-01	0.9566327	0.8924603	0.8247354
0.10	1e+00	0.9434902	0.8604497	0.8496032
0.10	1e+01	0.9227230	0.8460317	0.8423280
0.10	1e+02	0.8804894	0.8567460	0.8279101
0.10	1e+03	0.8645692	0.8674603	0.8206349
0.10	1e+04	0.8548469	0.8423280	0.8242063
0.15	1e-02	0.9527636	0.8535714	0.6642857
0.15	1e-01	0.9513653	0.9105820	0.8105820
0.15	1e+00	0.9310091	0.8783069	0.8494709
0.15	1e+01	0.8941421	0.8531746	0.8387566
0.15	1e+02	0.8602088	0.8781746	0.8242063
0.15	1e+03	0.8369331	0.8458995	0.8134921
0.15	1e+04	0.8369284	0.8637566	0.8064815
0.20	1e-02	0.9443925	0.8535714	0.6321429
0.20	1e-01	0.9440098	0.9250000	0.7384921
0.20	1e+00	0.9199310	0.8818783	0.8387566
0.20	1e+01	0.8752031	0.8674603	0.8207672
0.20	1e+02	0.8477324	0.8674603	0.8063492
0.20	1e+03	0.8308296	0.8638889	0.8134921
0.20	1e+04	0.8308296	0.8638889	0.8099206

```
ROC was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1 and C = 0.1.
```

Los argumentos que requiere la función son la `formula`, es decir, indicar la variable respuesta y qué predictores intervienen en el modelo. Los datos que se van a utilizar, así como el algoritmo a entrenar, en este caso la SVM con kernel de base radial. Además, se indica una métrica para el rendimiento del modelo, en caso de no indicarlo **R** asigna la más acorde de acuerdo a la variable respuesta. Finalmente, se incluyen las opciones de entrenamiento y la red de hiperparámetros a probar para determinar la combinación óptima. Los hiperparámetros del modelo entrenado son $\sigma = 0,1$ y $C = 0,1$. Este resultado queda definido en la salida del modelo, pero también es representable como en la Fig. 7.4. En este gráfico el eje y mide el rendimiento del modelo para ciertos valores de los hiperparámetros. Cada línea representa un valor para el hiperparámetro `sigma`, y se mide su rendimiento variando distintos niveles del parámetro coste (`C`), que queda representado en el eje x. Así, se observa que la línea roja (`sigma=0,1`) alcanza el mayor nivel de precisión en el valor $C=0,1$.

```
ggplot(model)
```

Los boxplot de la Fig. 7.5 muestran un resumen del rendimiento del modelo en las distintas repeticiones del proceso de validación cruzada. De esta manera, se observa como la sensibilidad es superior al 75 % y la especificidad supera valores del 70 %, esto indica que el modelo entrenado

7.6. Aplicación de un modelo SVM Radial con ajuste automático en R

145

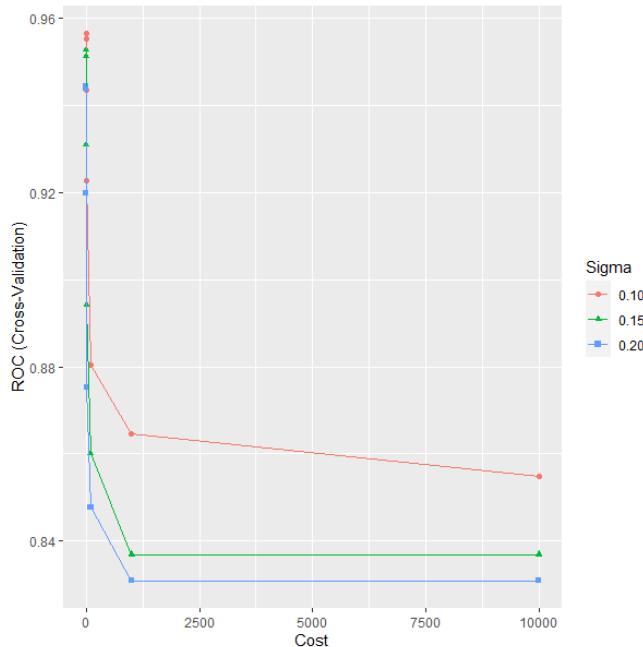


Figura 7.4: Optimización de los parámetros C y sigma de una SVM.

es capaz de predecir correctamente tanto a los clientes que van a comprar el nuevo producto como los que no lo van a hacer.

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

7.6.1. Importancia de las variables

En machine learning, muchos de los algoritmos de caja negra (definidos en el Cap. ??), no proporcionan información sobre la importancia que tiene cada variable en el modelo. Este es el caso de las máquinas de vector soporte. Tanto para la SVM como para otros algoritmos, es posible cuantificar la importancia de cada variable utilizando paquetes de **R** como DALEX, iml o vip.

Este último paquete incluye una función con el mismo nombre `vip()`. Para medir la importancia, se indica qué métrica se utilizó en el proceso de entrenamiento del modelo, en el caso de la SVM se indicará que fue el área bajo la curva (`metric=.auc`). En el argumento `pred_wrapper` se indica una función de medida que contenga tanto los valores observados como los valores predichos. Dado que la SVM entrenada utiliza AUC para medir el rendimiento del modelo ajustado, la función de medida indicada en `pred_wrapper` deberá devolver la probabilidad

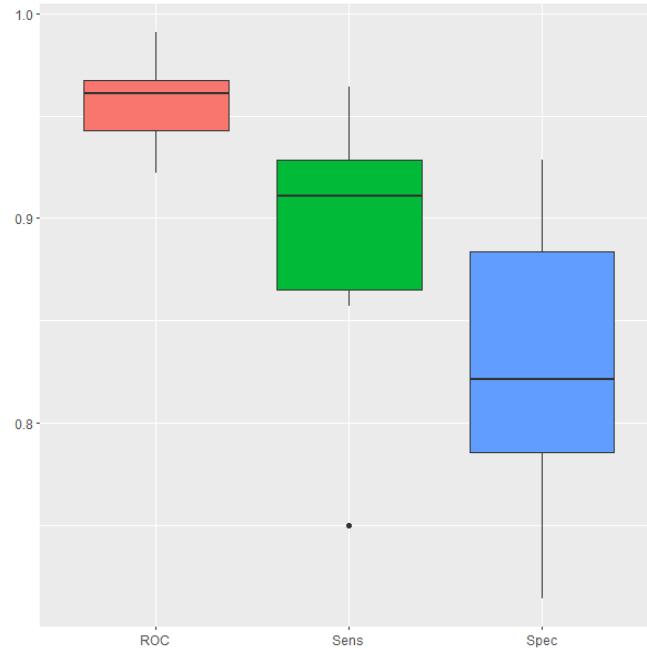


Figura 7.5: Resultados del modelo obtenidos durante la validación cruzada.

de que el modelo asigne una observación a la clase de referencia. En este ejemplo, la clase de referencia es “SI”, puesto que interesa saber si un cliente comprará el nuevo producto. Entonces, la función de predicción se define como:

```
prob_si <- function(object, newdata) {
  predict(object, newdata = newdata, type = "prob")[, "S"]
}
```

Ejecutando la función `vip()` con los argumentos mencionados se genera la Fig. 7.6. Este gráfico indica la importancia de cada variable en el modelo de más a menos importante. En este caso, la variable más importante es el importe gastado en el *smartchwatch fitness*, seguida muy de cerca por la variable que indica si el cliente compra o no el *smartchwatch fitness*. En el otro extremo, se observa que las variables que indican si el cliente tiene un nivel de educación básico o no, no son muy relevantes en la SVM entrenada.

```
library("vip")

set.seed(101)
vip(model, train = dp_entr_NUM, target = "CLS_PRO_pro13", metric = "auc",
  reference_class = "S", pred_wrapper = prob_si, method="permute",
  aesthetics = list(color = "steelblue2", fill = "steelblue2"))
```

7.6. Aplicación de un modelo SVM Radial con ajuste automático en R

147

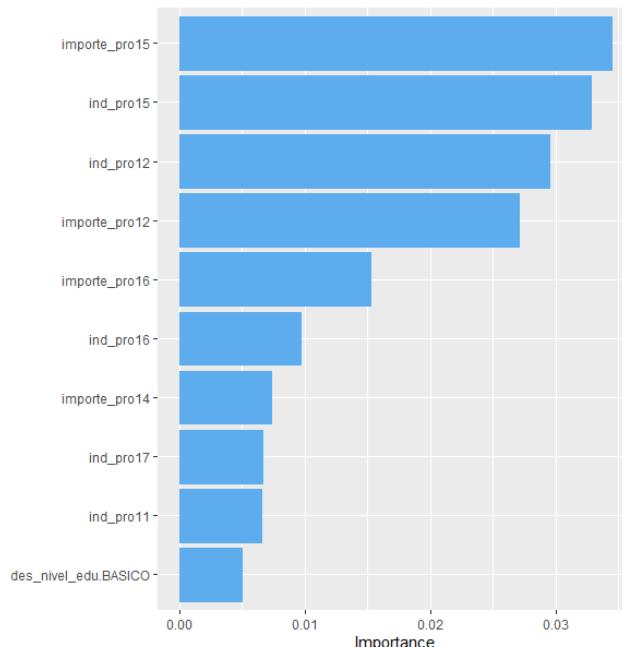


Figura 7.6: Importancia de las variables incluidas en la SVM.

A partir de la Fig. 7.6 se puede concluir que para predecir si un cliente comprará o no el *tensiómetro digital* las variables que más importancia tienen: son el importe que gastó en el *smartwatch fitness*, si compró o no el *smartwatch fitness*, el importe que gastó en la *depiladora eléctrica* y si compró o no la *depiladora eléctrica*.

De forma similar se podrían probar el resto de *kernels* disponibles para el algoritmo SVM.

Resumen

En este capítulo se introduce al lector en el algoritmo de máquinas vector soporte, en particular:

- Se presenta el concepto de margen de decisión, y las ventajas de la SVM respecto a otras técnicas de clasificación.
- Se explica el truco del kernel cuando los datos no son separables por un hiperplano en su espacio original
- Se da un repaso a los kernels más utilizados.
- Se presenta la aplicación de una SVM con kernel radial en **R** para la clasificación de datos con respuesta binaria, en el que se ajustan automáticamente los hiperparámetros.
- Se obtiene la importancia de las variables del modelo final.

Capítulo 8

Clasificador k-vecinos más próximos

Ramón A. Carrasco^a e Itzcóatl Bueno^{b,a}

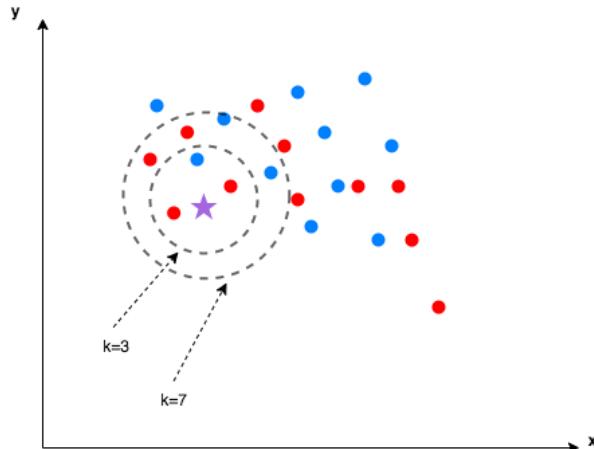
^aUniversidad Complutense de Madrid ^bInstituto Nacional de Estadística

8.1. Introducción

El k-vecinos más próximos (KNN, por sus siglas en inglés *k-Nearest Neighbors*) es un algoritmo de aprendizaje no paramétrico. Un algoritmo no paramétrico no presupone la forma concreta del modelo a entrenar, siendo más flexible. Sin embargo, esto se consigue a costa de necesitar más datos de entrenamiento y siendo más lentos que los algoritmos paramétricos. Al contrario que otros algoritmos de aprendizaje que permiten deshacerse de los datos de entrenamiento una vez se entrena el modelo, el modelo KNN guarda las observaciones de entrenamiento en memoria. Esto es, al incorporar una nueva observación x , el algoritmo KNN encuentra las k observaciones del conjunto de datos de entrenamiento más similares a la nueva y proporciona la clase mayoritaria (en el caso de clasificación) o el valor medio (en el caso de regresión).

El número de casos (k) a utilizar para clasificar las nuevas observaciones es un parámetro crucial para este algoritmo (James et al., 2013). Si, por ejemplo, $k = 3$, el modelo KNN utilizará las tres observaciones más similares (vecinos) al nuevo caso para clasificarlo. Es recomendable probar distintos valores de k para conseguir el mejor ajuste del modelo, y por tanto, es conveniente evitar valores extremos de k . Si se establece un valor muy bajo de k aumentará el sesgo y llevará a clasificaciones erróneas. Mientras que valores muy elevados de k harán que el algoritmo sea computacionalmente costoso y además tampoco será un buen clasificador. Además, también se recomienda establecer valores impares de k para evitar puntos muertos estadísticos (empate entre categorías) y un resultado no válido.

La escala de las variables puede impactar en el resultado del modelo KNN. Por ello, el conjunto de datos debe escalarse para que aquellas variables con unidades de medida grandes no tengan

Figura 8.1: Ejemplo de k -vecinos más próximos.

más importancia en el cálculo que otras con magnitudes menores. Así se reduce la importancia de las variables debido a sus unidades de medida y se estandariza la varianza.

Pese a que el modelo KNN es fácil de entender y generalmente preciso, almacenar el conjunto de datos de entrenamiento, así como calcular la distancia entre cada nueva observación a clasificar y las observaciones del conjunto de datos, supone la necesidad de recursos computacionales altos. Esto implica que cuanto mayor es la cantidad de observaciones en el conjunto de datos, mayor es el tiempo para la ejecución de una sola predicción, y, por tanto, esto puede dar lugar a tiempos de procesamiento lentos. Por este motivo, no se recomienda el uso del algoritmo KNN cuando se dispone de conjuntos de datos muy grandes. Otra desventaja a tener en cuenta es la dificultad de aplicar KNN a conjuntos de datos con un gran número de variables, puesto que calcular las distancias entre observaciones con múltiples dimensiones también incrementará la necesidad de recursos computacionales y podría dificultar que se clasifique de forma precisa.

8.2. Decisiones a tener en cuenta

La elección de la función de distancia, así como el número de vecinos k son decisiones que debe tomar el investigador antes de ejecutar el algoritmo. Siendo este último el hiperparámetro del modelo que la función `modelLookup()` de `caret` nos devuelve considerando que es primordial ajustarlo:

```
modelLookup("knn")
  model parameter      label forReg forClass probModel
  1   knn             k #Neighbors    TRUE     TRUE      TRUE
```

8.2.1. Función de distancia a utilizar

El modelo KNN determina la cercanía entre dos observaciones a través de una función de distancia. Generalmente, se utilizan la distancia euclídea, mostrada en la ecuación (8.1), y la distancia de Manhattan, mostrada en la ecuación (8.2). Otras funciones de distancia, como las presentadas en el Cap. ??, también pueden ser utilizadas para el entrenamiento de este algoritmo.

$$d(x_i, x_k) = \sqrt{\sum_{j=1}^p (x_i^{(j)} - x_k^{(j)})^2} \quad (8.1)$$

$$d(x_i, x_k) = \sum_{j=1}^p |x_i^{(j)} - x_k^{(j)}| \quad (8.2)$$

En el caso de querer incluir tanto variables cuantitativas como variables cualitativas en el cálculo de la distancia, el coeficiente de disimilitud de Gower es la función de distancia más popular para esta situación. El coeficiente de disimilitud de Gower se define como:

$$d(i, j) = \frac{\sum_{k=1}^p \omega_k \delta_{ij}^{(k)} d^{(k)}_{ij}}{\omega_k \delta_{ij}^{(k)}} \quad (8.3)$$

El coeficiente de Gower es una media ponderada de las distancias $d_{ij}^{(k)}$ con ponderaciones $\omega_k \delta_{ij}^{(k)}$.

8.2.2. Número de vecinos (k) seleccionados

Como se ha reiterado, la elección de cuántos vecinos (k) intervienen en el ajuste del algoritmo es determinante para su rendimiento. Si se escogen demasiado pocos vecinos, se producirá sobreajuste en el modelo. En el extremo en el que sólo se utilizará un vecino ($k = 1$), la predicción se basará en la observación con la menor distancia al elemento a clasificar. Por otro lado, un número alto de vecinos (k) hace que el modelo no ajuste bien al tener en cuenta un vecindario más grande. En este sentido, en el caso extremo de elegir todas las observaciones como vecinos más próximos ($k = n$), se obtendrá el valor medio (en el caso de la regresión) o la clase mayoritaria (en el caso de la clasificación) como valor predicho para todas las observaciones del conjunto de entrenamiento.

No existe una regla general para la elección óptima de k , puesto que en gran medida dependerá del conjunto de datos utilizado. Cuando el conjunto de datos tiene pocas variables que no aporten información, valores pequeños de k tienden a funcionar mejor. Cuantas más variables sin importancia se incluyen en el conjunto de datos, mayor deberá ser el valor de k para suavizar su efecto.

8.3. Procedimiento con R: la función knn()

En el paquete `class` de R se encuentra la función `knn()` que se utiliza para entrenar el modelo k -vecinos más próximos:

```
knn(train, test, cl, k = 1, ...)
```

- `train`: conjunto de datos con las observaciones de entrenamiento.
- `test`: conjunto de datos con las observaciones de validación. Un vector se interpreta como una única observación a validar.
- `cl`: clases de las observaciones de entrenamiento.
- `k`: número de vecinos a considerar

8.4. Aplicación del modelo KNN en R

En este ejemplo se entrena un modelo KNN para clasificar qué clientes comprarán el *tensiómetro digital* teniendo en cuenta sus características y el resto de sus compras. Este conjunto de datos está incluido en el paquete CDR con el nombre `dp_entr_NUM`. En este conjunto de datos todas las variables son cuantitativas (excepto la clase objetivo) pero dichas variables tienen distintas escalas de medida (euros, años, unidades, etc.) por lo que es necesario indicar en la función `trainControl()` que se haga un preprocessamiento para estandarizar las variables. Además, se define como método de remuestreo la validación cruzada, como la presentada en el Cap. ??, con 10 folds.

```
library("CDR")
library("class")
library("caret")
library("reshape")
library("ggplot2")

data(dp_entr_NUM)

head(dp_entr_NUM)

  ind_pro11 ind_pro12 ind_pro14 ind_pro15 ind_pro16 ind_pro17 des_nivel_edu.ALTO
1       1       0       1       1       1       0       0
2       0       0       1       0       1       0       0
3       0       0       1       1       1       1       0
4       0       1       1       0       0       0       0
5       0       1       1       0       1       0       0
6       1       0       1       0       0       0       1
  des_nivel_edu.BASICO des_nivel_edu.MEDIO importe_pro11 importe_pro12 importe_pro14
1             0             1         157            0            40
2             0             1            0            0            240
3             1             0            0            0            425
```

8.4. Aplicación del modelo KNN en R

153

```

4          0          1          0        120        60
5          1          0          0        120      133
6          0          0        115        0      220
    importe_pro15 importe_pro16 importe_pro17 edad tamano_fam anos_exp ingresos_ano
    ← CLS_PRO_pro13
1          200        180        0     49       4     24     30000
    ← S
2          0        180        0     38       2     12     53000
    ← N
3          200        180      300     61       4     37   172000
    ← S
4          0          0        0     47       3     21     38000
    ← N
5          0        180        0     34       1     10     38000
    ← N
6          0          0        0     43       2     18     60000
    ← N

# Definimos un método de remuestreo
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5,
  classProbs = TRUE,
  preProcOptions = list("center"),
  summaryFunction = twoClassSummary
)

```

Antes de obtener el modelo definitivo, es necesario la selección del número óptimo de vecinos k entrenando distintos modelos variando dicho hiperparámetro. Para facilitar este trabajo arduo, en el paquete `caret` de R se puede definir una red de posibles valores sobre los que evaluar el modelo KNN y que de forma automática se determine el valor que mejor rendimiento proporcione. A continuación se definen los posibles valores de k que se quieren evaluar.

```
# Definimos la red de posibles valores del hiperparámetro
hyper_grid <- expand.grid(k = c(1:10,15,20,30,50,75,100))
```

Una vez se que se ha definido tanto el método de remuestreo como la red de posibles valores del hiperparámetro se puede entrenar el modelo:

```

set.seed(101)
# Se entrena el modelo ajustando el hiperparámetro óptimo
model <- train(
  CLS_PRO_pro13 ~ .,
  data = dp_entr_NUM,

```

```

method = "knn",
trControl = cv,
tuneGrid = hyper_grid,
metric = "ROC"
)

model

k-Nearest Neighbors

558 samples
 17 predictor
 2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:

      k    ROC      Sens      Spec
1  0.6584524  0.6466402  0.6702646
2  0.6769109  0.6179101  0.6131217
3  0.6828893  0.6216402  0.6496561
4  0.6851087  0.6404233  0.6394709
5  0.6951129  0.6540212  0.6666138
6  0.6914664  0.6216402  0.6543386
7  0.6982592  0.6252381  0.6953439
8  0.6974556  0.6281481  0.6960053
9  0.6992229  0.6159524  0.7117725
10 0.6994133  0.6037037  0.7052910
15 0.6875879  0.5749206  0.7232011
20 0.6731477  0.5722751  0.7010582
30 0.6752986  0.5529630  0.7024603
50 0.6890259  0.5163757  0.7605556
75 0.6852886  0.5092593  0.7670106
100 0.6719378  0.5049471  0.7820106

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 10.

```

En la Fig. 8.2 se observa que el número óptimo de vecinos es $k = 10$, donde se alcanza el rendimiento óptimo del modelo.

```

ggplot(model) + geom_vline(xintercept =
  unlist(model$bestTune), col="red", linetype="dashed") + theme_light()

```

El boxplot de los resultados obtenidos durante el proceso de validación cruzada muestra que el AUC del modelo oscila entre un 60 % y un 85 % aproximadamente.

8.4. Aplicación del modelo KNN en R

155

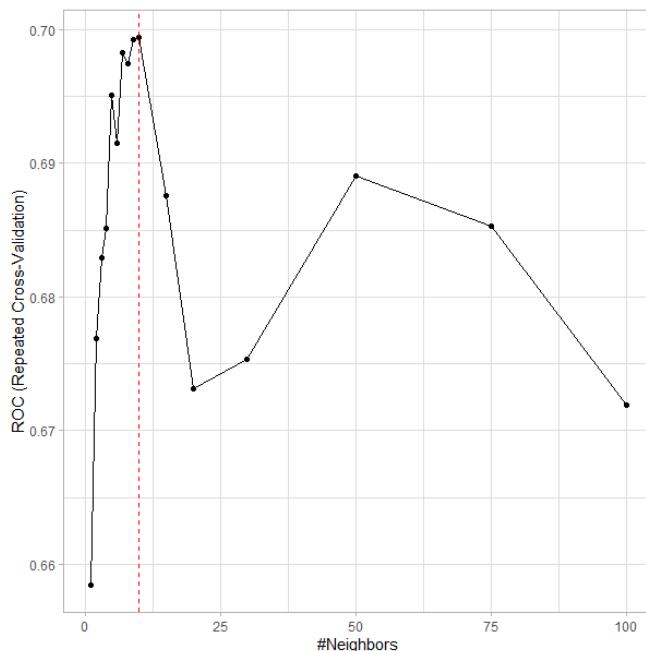


Figura 8.2: Número óptimo de vecinos (k).

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

El modelo se resiente en su rendimiento al tener dificultades en predecir correctamente la clase positiva, más concretamente se puede observar en la Fig. 8.3 que la sensibilidad oscila entre el 40 % y el 75 %; resultados ligeramente peores que los que obtiene al predecir observaciones de la clase negativa, los cuales oscilan entre el 50 % y el 85 %.

Resumen

En este capítulo se introduce al lector en el algoritmo de aprendizaje supervisado conocido como k-vecinos más próximos, destacando:

- Las decisiones a tener en cuenta antes de entrenar el modelo de k-vecinos más próximos.
- Se exponen algunas de las distancias más utilizadas para el entrenamiento de este modelo.
- Se especifican las ventajas y desventajas del número de vecinos a tener en cuenta.

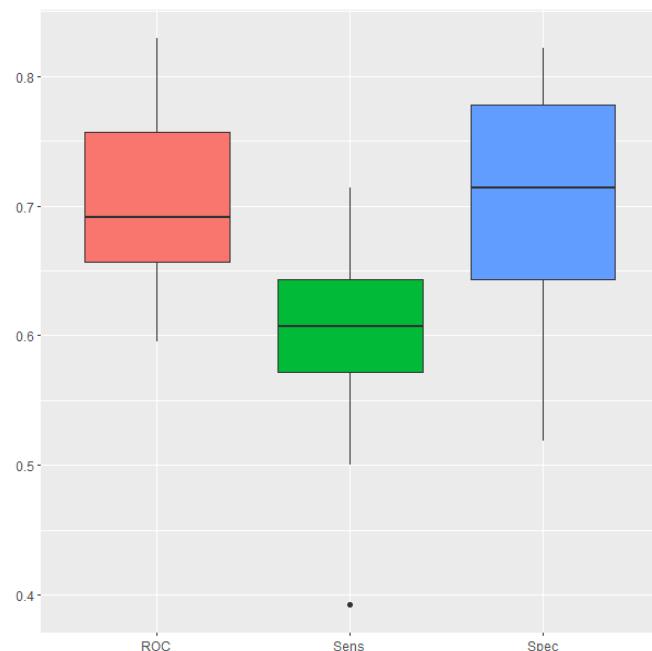


Figura 8.3: Resultados obtenidos durante el proceso de validación cruzada.

Capítulo 9

Naive Bayes

Ramón A. Carrasco^a e Itzcóatl Bueno^{b,a}

^aUniversidad Complutense de Madrid ^bInstituto Nacional de Estadística

9.1. Introducción

Naive Bayes es un algoritmo de aprendizaje supervisado que se utiliza principalmente para la clasificación. Como otros algoritmos de aprendizaje supervisado, este algoritmo se entrena con variables de entrada y la categoría asociada a cada observación y que el modelo debe predecir. Sin embargo, se denomina ‘naive’ dado que asume que las variables de entrada que se incluyen en el modelo son independientes entre sí. Por lo tanto, si se cambia una de las variables de entrada, las demás no se verán afectadas.

Aunque el algoritmo *Naive Bayes* es sencillo, destaca por su facilidad de implementación y su potencia predictiva. Su ventaja principal es que utiliza un enfoque probabilístico, lo que implica que todos los cálculos se realizan en tiempo real y, por tanto, los resultados se obtienen inmediatamente, como se detalla más adelante. Además, cuando el conjunto de datos tiene un gran número de observaciones, el algoritmo *Naive Bayes* es ventajoso respecto a algoritmos como la SVM (Cap. 7) o el Random Forest (Cap. 10) debido a su mejor tiempo de computación.

Al utilizar un enfoque probabilístico, el algoritmo *Naive Bayes* está construido sobre conceptos de probabilidad, presentados en el Cap. ??, y en especial, este algoritmo hace uso del **Teorema de Bayes**. A continuación se repasan los conceptos fundamentales en los que está basado el algoritmo.

9.2. Teorema de Bayes

Sean dos eventos A y B definidos en un espacio muestral, se puede definir la probabilidad condicional de que ocurra el evento A dado que previamente se haya observado B como:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (9.1)$$

Siempre que $P(B) \neq 0$ y donde $P(A \cap B)$ es la probabilidad de que ocurran ambos eventos a la vez. Los eventos son intercambiables de tal forma que $P(A \cap B) = P(B|A)P(A)$ y si se reemplaza en la primera ecuación tenemos:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (9.2)$$

Esta fórmula es la definición del teorema de Bayes. El algoritmo de clasificación *Naive Bayes* (NB) está basado en este teorema. Para ampliar los conceptos estadísticos aquí presentados pueden consultarse en más detalle en el Cap. ??.

9.3. El algoritmo *naive* Bayes

Si se adapta el teorema de Bayes a un problema de clasificación, se tendrá:

$$P(C = c|\ell) = \frac{P(\ell|C = c) \cdot P(C = c)}{P(\ell)} \quad (9.3)$$

En este caso, $P(C = c|\ell)$ representa el objetivo de estimación en un problema de clasificación, es decir, la probabilidad de que un individuo pertenezca a la clase c después de haber observado la evidencia ℓ (incluida en las variables del modelo). Esta es la denominada **probabilidad a posteriori**. El resto de elementos de la fórmula, se definen como:

- $P(C = c)$ es la **probabilidad a priori** de pertenecer a la clase c , es decir, la probabilidad que un individuo tiene de ser asignado a esa clase sin observar sus características previamente.
- $P(\ell|C = c)$ es la verosimilitud de observar una instancia particular de las variables incluidas en el modelo cuando el individuo pertenece a la clase c .
- $P(\ell)$ es la verosimilitud de observar una instancia particular de las variables incluidas en el modelo, independientemente de a qué clase pertenezca el individuo.

Sin embargo, una gran dificultad para aplicar esta ecuación es la necesidad de conocer que $P(\ell|c)$ es igual a $P(\ell_1 \cap \ell_2 \cap \dots \cap \ell_k|c)$. La existencia de un ejemplo concreto en el conjunto de datos de entrenamiento que coincida a la perfección con ℓ es complicado, y en el caso de existir, no se tendrían suficientes ejemplos para poder estimar una probabilidad de forma fiable. La forma de solucionar este problema es incluir una suposición de independencia particularmente fuerte, que como ya se mencionó en la Sec. 9.1, es lo que aporta la denominación de ‘naive’ al algoritmo.

9.3. El algoritmo naive Bayes

159

La *independencia condicional* implica que conocer un evento no aporta información sobre otro evento. Esto es equivalente a:

$$P(AB|C) = P(A|C) \cdot P(B|C) \quad (9.4)$$

De este modo, el problema de clasificación en el que era difícil estimar $P(\ell_1 \cap \ell_2 \cap \dots \cap \ell_k|c)$, ahora se tendría:

$$P(\ell|c) = P(\ell_1|c) \cdot P(\ell_2|c) \cdots P(\ell_k|c) \quad (9.5)$$

Y cada uno de los elementos $P(\ell_i|c)$ puede obtenerse directamente de los datos. Combinando este resultado con la regla de Bayes aplicada a un problema de decisión, se obtiene la ecuación dada por el algoritmo *Naive Bayes*:

$$P(c|\ell) = P(\ell_1|c) \cdot P(\ell_2|c) \cdots P(\ell_k|c)P(c) \quad (9.6)$$

El algoritmo *Naive Bayes* clasifica una nueva observación estimando la probabilidad de que pertenezca a cada clase y asignándole a aquella que tenga la mayor probabilidad.

En definitiva, el clasificador *Naive Bayes* es muy eficiente en términos de espacio de almacenamiento necesario, así como tiempos de procesamiento. Además, a pesar de ser muy simple, tiene en cuenta las características observadas. Otra de las ventajas de este clasificador es que es un modelo de aprendizaje incremental. Esto quiere decir que es una técnica de inducción que se actualiza con cada nueva observación de entrenamiento, es decir, no es necesario volver a procesar todo el conjunto de entrenamiento cuando se dispone de nuevas observaciones.

El ejemplo presentado en el Cap. 6 en el que se buscaba predecir si se podría jugar o no al tenis bajo unas condiciones meteorológicas determinadas, puede desarrollarse utilizando el modelo *Naive Bayes*. En este caso, el procedimiento puede resumirse en tres pasos:

- Resumir los datos en una tabla de frecuencias.
- Generar una tabla de verosimilitud obteniendo las probabilidades de las variables.
- Aplicar el teorema de Bayes para calcular la probabilidad a posteriori.

De este modo, las 15 observaciones registradas con el tipo de día (soleado, nublado, lluvioso) y si ese día se jugó, deben resumirse en una tabla de frecuencias como la Tabla 9.1. En este primer paso no se tiene en cuenta la información sobre humedad o viento.

Tabla 9.1: Tabla de frecuencias - Tipo de día vs Jugar partido

	SI	NO	Total
Soleado	2	4	6
Nublado	4	0	4
Lluvia	4	1	5

	SI	NO	Total
Total	10	5	15

En un segundo paso, se obtienen las probabilidades de cada categoría a partir de la Tabla 9.1 resultando en la Tabla 9.2.

Tabla 9.2: Tabla de verosimilitud - Tipo de día vs Jugar partido

	SI	NO	$P(\text{Tipo de día}_i)$
Soleado	2	4	$\frac{6}{15} = 0,40$
Nublado	4	0	$\frac{4}{15} = 0,27$
Lluvia	4	1	$\frac{5}{15} = 0,33$
$P(\text{Jugar})$	$\frac{10}{15} = 0,67$	$\frac{5}{15} = 0,33$	

A partir de la Tabla 9.2 se obtiene la probabilidad de cada tipo de día dado que con esa climatología se jugó o no, es decir, $P(\text{Tipo de día} | \text{Jugar})$. Obteniendo las probabilidades mostradas en la Tabla 9.3

Tabla 9.3: Probabilidad de Tipo de día sabiendo si se jugó el partido

	$c = \text{SI}$	$c = \text{NO}$
$P(\text{Soleado} C=c)$	$\frac{2}{15}$	$\frac{4}{15}$
$P(\text{Nublado} C=c)$	$\frac{4}{15}$	$\frac{1}{15}$
$P(\text{Lluvia} C=c)$	$\frac{4}{15}$	$\frac{1}{15}$

Este proceso se repite de forma independiente para las variables *viento* y *humedad* obteniendo la Tabla 9.4 y la Tabla 9.5 respectivamente.

Tabla 9.4: Probabilidad fuerza del Viento sabiendo si se jugó el partido

	$c = \text{SI}$	$c = \text{NO}$
$P(\text{Débil} C=c)$	$\frac{6}{15}$	$\frac{2}{15}$
$P(\text{Fuerte} C=c)$	$\frac{4}{15}$	$\frac{3}{15}$

9.4. Procedimiento con R: la función `naive_bayes()`

161

Tabla 9.5: Probabilidad nivel de Humedad sabiendo si se jugó el partido

	c = SI	c = NO
P(Débil C=c)	$\frac{6}{15}$	$\frac{1}{15}$
P(Fuerte C=c)	$\frac{4}{15}$	$\frac{4}{15}$

Finalmente, aplicando el teorema de Bayes se podría predecir si se juega o no el partido ante la previsión de un nuevo día. Por ejemplo, ¿cuál es la probabilidad de no jugar al tenis si el día se espera soleado, con fuertes rachas de viento y escasa humedad? Esto es, $\ell=[\text{Soleado}, \text{Fuerte}, \text{Débil}]$ y, de acuerdo al teorema de Bayes, esta pregunta se respondería a través de:

$$P(c|\ell) = \frac{P(\ell|c) \cdot P(c)}{P(\ell)}$$

A partir de las probabilidades previamente obtenidas y de la asunción de independencia entre las variables, se puede calcular la probabilidad de jugar como:

$$P(Si|\ell) = P(\text{Soleado}|Si) \cdot P(\text{Fuerte}|Si) \cdot P(\text{Débil}|Si) \cdot P(Si) = \frac{2}{15} \frac{4}{15} \frac{2}{15} \frac{10}{15} = 0,0032 \quad (9.7)$$

$$P(No|\ell) = P(\text{Soleado}|No) \cdot P(\text{Fuerte}|No) \cdot P(\text{Débil}|No) \cdot P(No) = \frac{4}{15} \frac{3}{15} \frac{1}{15} \frac{5}{15} = 0,0012 \quad (9.8)$$

La probabilidad de jugar es superior a la probabilidad de no jugar y, por tanto, dado un día con esas condiciones climáticas se clasificará como un día en el que se puede jugar.

9.4. Procedimiento con R: la función `naive_bayes()`

En el paquete `naivebayes` de R se encuentra la función `naive_bayes()` que se utiliza para entrenar un modelo *Naive Bayes*:

```
naive_bayes(formula, data, prior = ..., ...)
```

- **formula**: refleja la relación lineal entre la variable dependiente y los predictores $Y \sim X_1 + \dots + X_p$.
- **data**: conjunto de datos con el que se entrena el modelo.
- **prior**: vector con las probabilidades a priori de las clases.

9.5. Clasificación de clientes utilizando el modelo *Naive Bayes*

Como en los capítulos precedentes, en este ejemplo se pretende entrenar un modelo *Naive Bayes* utilizando el conjunto de datos de compras realizadas por clientes incluido en el paquete CDR. Este conjunto de datos cuenta con unas variables predictoras que indican qué productos han comprado los clientes, el importe que han gastado y otras características como su edad y su nivel educativo. Se utiliza el conjunto de datos sin transformar (`dp_entr`), es decir, en su escala original y con las variables categóricas sin codificar. La variable objetivo indica si un cliente comprará o no el nuevo producto (*tensiómetro digital*).

```
library("caret")
library("naivebayes")
library("reshape")
library("ggplot2")
library("CDR")

data("dp_entr")

# se fija la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13 ~ .,
                 data=dp_entr,
                 method="nb",
                 metric="Accuracy",
                 trControl=trainControl(classProbs = TRUE,
                                         method = "cv",
                                         number = 10))

# se muestra la salida del modelo
model
```

```
Naive Bayes

558 samples
17 predictor
2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:

usekernel Accuracy Kappa
FALSE      0.8512662 0.7026716
TRUE       0.8512338 0.7025165
```

9.5. Clasificación de clientes utilizando el modelo Naive Bayes

163

```
Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter
  'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```

Los resultados del proceso de entrenamiento muestran que, en este caso, es indiferente indicar el argumento `usekernel` como FALSE o TRUE, los resultados de precisión son equivalentes. El resumen del modelo muestra que la precisión media obtenida durante la validación cruzada alcanza el 85,1 %, lo cual indica que el modelo ajusta bastante bien la intención de compra de nuevos clientes.

```
confusionMatrix(model)
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

      Reference
Prediction   S     N
          S 41.8  6.6
          N  8.2 43.4

Accuracy (average) : 0.8513
```

En la matriz de confusión del modelo se observa para cada celda el promedio porcentual entre remuestreos. Así, se observa que en media el modelo predice mejor cuando un cliente no va a comprar el nuevo producto que cuando sí lo hace, aunque no con mucha diferencia (menos de un 2 %). En ambos casos, las clasificaciones erróneas no suponen ni el 10 %.

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

Se puede observar como la precisión oscila entre el 75 % y el 95 %, aunque en uno de los resultados se obtuvo un 96 % de precisión, el cual se marca como un resultado atípico.

Resumen

En este capítulo se introduce al lector en el algoritmo de *Naive Bayes*, en concreto:

- Se presentan los fundamentos del algoritmo bayesiano, particularmente el Teorema de Bayes.
- Se explica el funcionamiento del algoritmo *Naive Bayes* y su relación con dicho Teorema de Bayes.
- Se demuestra su aplicabilidad a un caso real de clasificación a través de R.

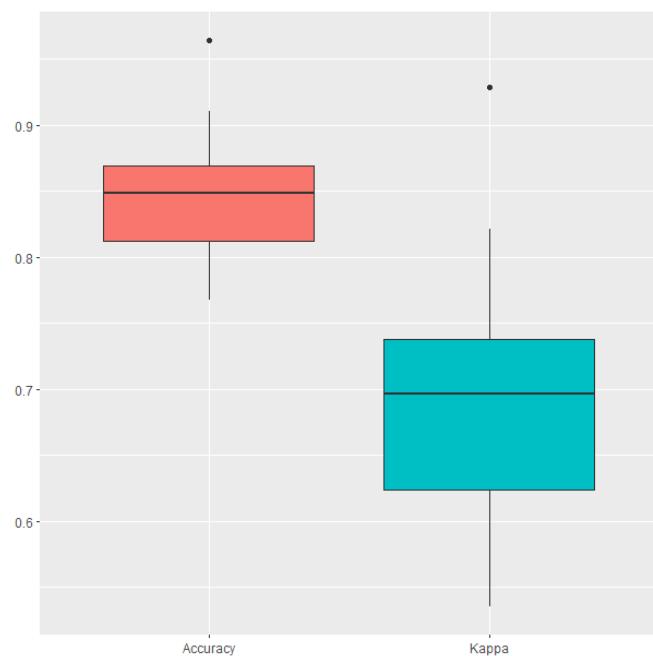


Figura 9.1: Resultados del modelo Naive Bayes obtenidos durante el proceso de validación cruzada.

Capítulo 10

Métodos ensamblados: bagging y random forest

Ramón A. Carrasco^a e Itzcóatl Bueno^{b,a}

^aUniversidad Complutense de Madrid ^bInstituto Nacional de Estadística

10.1. Introducción a los métodos ensamblados

Puede ocurrir que ninguno de los algoritmos hasta ahora presentados (Caps. 6, 7, 8 y 9) proporcionen resultados convincentes para el problema que se quiere modelar. El *aprendizaje ensamblado* (Zhou, 2012) es un paradigma que, como muestra la Fig. 10.1, en lugar de entrenar un modelo muy preciso, se centra en entrenar un gran número de modelos con menor precisión, y después combinar sus predicciones para obtener un metamodelo de una precisión más alta.

A los modelos de menor precisión se les suele nombrar como algoritmos “débiles”, es decir, algoritmos con menor capacidad de aprender patrones complejos en los datos. Por tanto, generalmente, son rápidos tanto en tiempo de entrenamiento como de procesamiento. Existen dos paradigmas de aprendizaje ensamblado: el *bagging* y el *boosting* (Cap. 11).

10.2. Bagging

En lugar de buscar la división más eficiente en cada capa, como ocurre en el árbol de decisión, una alternativa sería construir un metamodelo combinando los resultados de múltiples árboles de decisión. Esta técnica se conoce como *bagging* y consiste en construir varios árboles utilizando una selección aleatoria de los datos que se utilizan para cada árbol y, finalmente, combinar la predicción de cada uno de ellos a través de la media (en el caso de regresión) o mediante un sistema de votación (en el caso de un problema de clasificación).

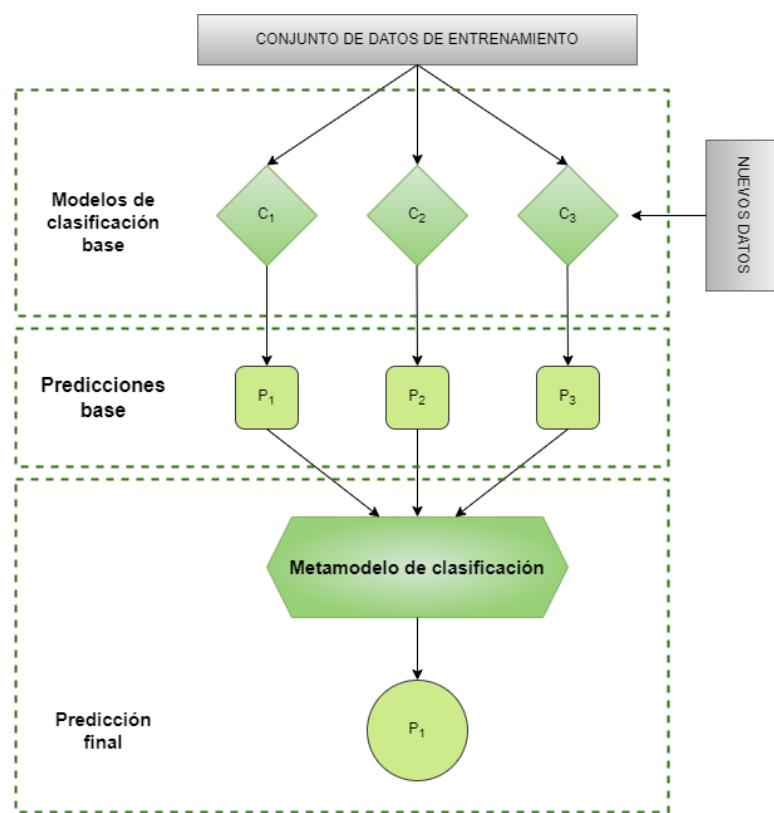


Figura 10.1: Esquema de un metamodelo.

La principal característica del *bagging* es el llamado muestreo bootstrap. La idea básica del bootstrap es que la inferencia sobre una población se haga a partir de una muestra, tomando el papel de población y se remuestree, permitiendo comparar valor poblacional y el valor muestral. En el *bagging*, el objetivo de este remuestreo es que cada árbol esté entrenado con una muestra única, y por tanto, generen respuestas únicas, esto es modelos débiles distintos. Para ello, debe existir aleatoriedad y variación en cada árbol que conforme el modelo final, puesto que no tendría sentido construir varios árboles idénticos. Como se ha comentado, este problema queda resuelto por el muestreo bootstrap, el cual extrae una muestra aleatoria de los datos en cada ronda. En el caso del *bagging*, se extraen distintas muestras de datos para el entrenamiento de cada árbol. Aunque esto no elimina la problemática del sobreajuste, los patrones presentes en el conjunto de datos aparecerán en la mayoría de los árboles entrenados y, por tanto, en la predicción final. Es por ello que el *bagging* es una técnica de gran eficacia para el tratamiento de los valores atípicos y para la reducción de la varianza que generalmente afecta a un modelo compuesto por un único árbol de decisión.

10.2.1. Procedimiento con R: la función bagging()

En el paquete **ipred** de R se encuentra la función **bagging()** que se utiliza para entrenar un modelo *bagging*:

```
bagging(formula, data, ...)
```

- **formula:** Refleja la relación lineal entre la variable dependiente y los predictores $Y \sim X_1 + \dots + X_p$.
- **data:** Conjunto de datos con el que se entrena el modelo.
- **nbagg:** Número de replicaciones bootstrap.
- **coob:** Indica si se debe calcular una estimación del ratio de error de predicción.

10.2.2. Implementando *bagging* en R

Es posible la implementación de un modelo de predicción de agregación bootstrap en R. Para ello, se pueden utilizar múltiples funciones como la ya mencionada en la Sec. 10.2.1 **bagging()**. En este ejemplo se utilizan los datos sobre compras de clientes **dp_entr** del paquete **CDR**, cuyo objetivo es clasificar a los clientes entre quienes comprarían un nuevo producto y quienes no.

```
library("CDR")
library("ipred")
library("caret")
library("reshape")
library("ggplot2")

data("dp_entr")
```

```
# se fija la semilla aleatoria
set.seed(101)

# Se entrena el modelo
bag_model <- bagging(
  formula = CLS_PRO_pro13 ~ .,
  data = dp_entr,
  nbagg = 100,
  coob = TRUE,
  control = rpart.control(minsplit = 2, cp = 0)
)

bag_model

Bagging classification trees with 100 bootstrap replications

Call: bagging.data.frame(formula = CLS_PRO_pro13 ~ ., data = dp_entr,
  nbagg = 100, coob = TRUE, control = rpart.control(minsplit = 2,
  cp = 0))

Out-of-bag estimate of misclassification error:  0.1416
```

El error de clasificación de este modelo es del 14,16 %, o lo que es equivalente, el modelo tiene una precisión del 85,84 %. Desafortunadamente, `bagging()` no selecciona el número óptimo de replicaciones reduciendo el error de clasificación. Para seleccionar el número de replicaciones que minimice el error, se puede graficar la curva de error por número de replicaciones como en la Fig. 10.2. Se itera el modelo variando los valores del hiperparámetro `nbagg` (en este ejemplo entre 10 y 150, incrementándose de cinco en cinco). Se observa que el error mínimo (13,79 %) se obtiene al establecer el hiperparámetro igual a 60.

```
missclass <- c() # vector vacío para recopilar el error en cada iteración
for (n in seq(10,150,5)) { # valores a probar para nbagg
  # se establece la semilla aleatoria
  set.seed(101)
  # se entrena el modelo
  bag_model <- bagging(
    formula = CLS_PRO_pro13 ~ .,
    data = dp_entr,
    nbagg = n,
    coob = TRUE,
    control = rpart.control(minsplit = 2, cp = 0)
  )
  # se agrega el error de esta iteración
  missclass <- c(missclass, bag_model$err) # se agrega el error de esta iteración
}
```

10.2. Bagging

169

```
plot(seq(10,150,5),missclass,type = "l",xlab = "Número de árboles",
  ylab="Missclassification error")
```

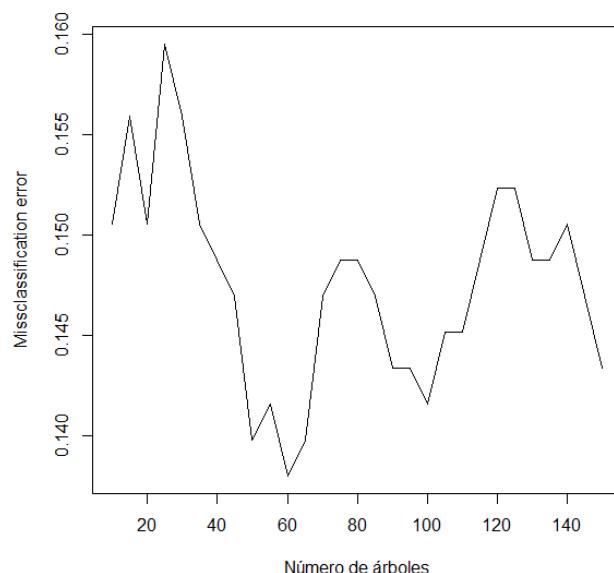


Figura 10.2: Número de replicaciones vs Error de clasificación.

La función `train()` del paquete `caret` es otro método para entrenar un algoritmo de *bagging* en R. Para ello, el argumento `method` debe tomar el valor “`treebag`”. Sin embargo, este algoritmo no incluye hiperparámetros a optimizar. Dado que se ha obtenido recursivamente el número óptimo de replicaciones, se puede entrenar el modelo con el valor obtenido y comprobar que el error dado coincide. Se observa que si se entrena un modelo *bagging* con 60 replicaciones, la precisión del modelo es del 86,93 %. Esto es aproximadamente el resultado obtenido anteriormente en el que para 60 replicaciones el modelo tenía un error de clasificación del 13,79 %.

```
set.seed(101)
model_bag <- train(
  CLS_PRO_pro13 ~ .,
  data = dp_entr,
  method = "treebag",
  trControl = trainControl(method = "cv", number = 10),
  nbagg = 60,
  control = rpart.control(minsplit = 2, cp = 0)
)
```

```

model_bag

Bagged CART

558 samples
17 predictor
2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results:

Accuracy   Kappa
0.8692532  0.7385449

```

10.2.3. Interpretación de variables en el *bagging*

Una de las principales desventajas de los algoritmos ensamblados (incluido el *bagging*) es que mientras que los modelos base son interpretables, el metamodelo resultante no lo es. Pese a esto, aún es posible hacer inferencia de cómo cada una de las variables influye en el modelo entrenado. La manera de medir la importancia de las variables incluidas en un árbol es registrar para cada variable la reducción de la función de pérdida que se le atribuye en cada partición. Dado que una variable puede utilizarse varias veces para dividir el árbol, la importancia total de esa variable será la suma de la reducción de la función de pérdida que se le atribuya por todas las particiones en las que intervenga. Este proceso es similar para el *bagging*. En este caso, para cada árbol se calcula la reducción de la función de pérdida en todas las divisiones. Tras esto, se agrega esta medida en todos los árboles que forman el metamodelo. El paquete `ipred`, en el que se encuentra la función `bagging()`, no captura la información requerida para calcular la importancia de las variables. Sin embargo, el paquete `caret` sí lo hace y se puede construir un gráfico de importancia utilizando la función `vip()` del paquete `vip`.

```

library("vip")
vip(model_bag, num_features = 15,
    aesthetics = list(color = "skyblue", fill = "skyblue"))

```

La Fig. 10.3 muestra que las variables más importantes en el modelo *bagging* entrenado para predecir si un cliente comprará o no el *tensiómetro digital* son: si ha comprado la *depiladora eléctrica*, cuánto importe ha gastado en ese producto, si ha comprado el *estimulador muscular* y si ha comprado el *smartwatch fitness*.

10.3. Random Forest

171

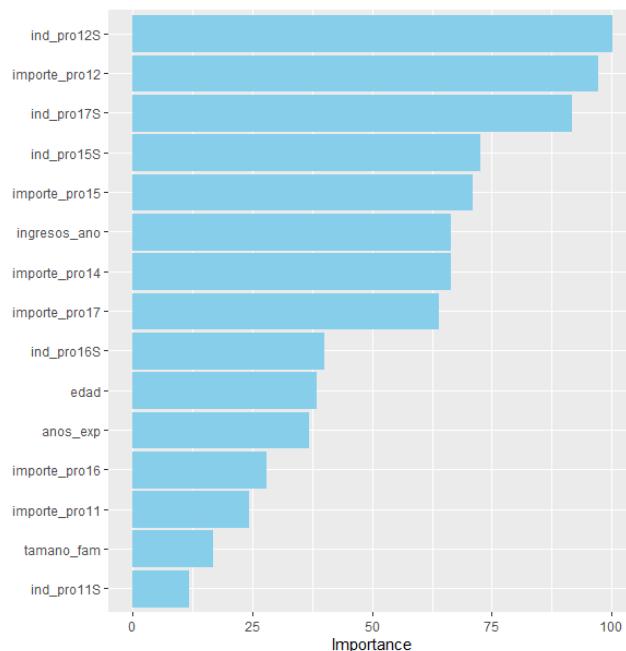


Figura 10.3: Importancia de las variables incluidas en el modelo bagging.

10.3. Random Forest

El *bagging* es el paradigma tras el algoritmo de *random forest*. Este algoritmo fue desarrollado por primera vez por (Ho, 1995). Sin embargo, fueron (Cutler and Zhao, 1999) y (Breiman, 2001) quienes desarrollaron una versión extendida del modelo y registraron **Random Forest** como marca comercial. Este algoritmo básico de *bagging* funciona del siguiente modo: a partir del conjunto de datos de entrenamiento se generan K muestras aleatorias \mathbb{S}_k , se entrena un modelo de árbol de decisión (f_k) utilizando la muestra \mathbb{S}_k como conjunto de entrenamiento. Tras el entrenamiento, se dispone de K árboles de decisión, como se observa en la Fig. 10.4. La predicción de una nueva observación x se obtiene como la media de las K predicciones:

$$y \leftarrow \hat{f}(x) = \frac{1}{K} \sum_{k=1}^K f_k(x) \quad (10.1)$$

En el caso de regresión, o por la votación por mayoría en el caso de clasificación.

Tanto el *bagging* como el *random forest* desarrollan múltiples árboles y utilizan el muestreo bootstrap para la aleatorización de los datos. Sin embargo, el *random forest* establece una limitación artificial a la selección de variables al no considerar todas en cada árbol.

El *bagging* considera las mismas variables para construir cada árbol con el objetivo de minimizar su entropía, y, por tanto, todos los árboles suelen tener un aspecto similar. Esto lleva a que las

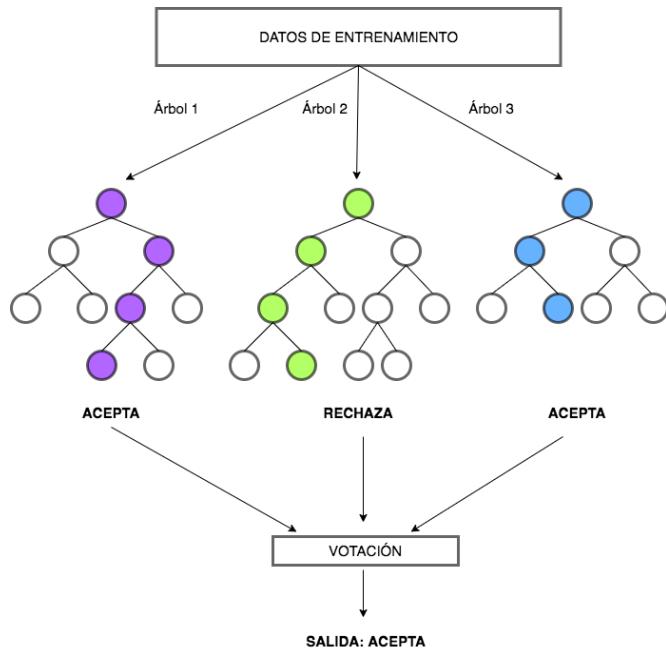


Figura 10.4: Ejemplo de Random Forest.

predicciones dadas por los árboles estén altamente correlacionadas. El modelo *random forest* evita este problema estableciendo la obligación, en cada división, de utilizar un subconjunto de las variables. Esto proporciona a algunas variables mayor probabilidad de ser seleccionadas, y al generar árboles únicos y no correlacionados se consigue una estructura de decisión final más fiable.

En general, es mejor que el *random forest* esté formado por una gran cantidad de árboles (por lo menos 100) para suavizar el impacto de valores atípicos. Sin embargo, la tasa de efectividad disminuye a medida que se incorporan más árboles. Llegado a cierto punto, los nuevos árboles no aportan una mejora significativa al modelo, pero si incrementan los tiempos de procesamiento.

El modelo *random forest* es rápido de entrenar y es una buena técnica para obtener un modelo de referencia. Aunque estos modelos funcionan bien en la interpretación de patrones complejos y son versátiles, otras técnicas, como por ejemplo el *gradient boosting* (Cap. 11), proporcionan una mayor precisión en las predicciones en muchos casos.

Estos modelos se han vuelto populares porque tienden a proporcionar un muy buen rendimiento con los parámetros predeterminados en las distintas implementaciones. En efecto, a pesar de tener muchos hiperparámetros que pueden ser ajustados, los valores por defecto de dichos hiperparámetros tienden a ofrecer buenos resultados en la predicción. Los hiperparámetros más importantes que hay que ajustar al entrenar un modelo *random forest* son: el número de árboles (K), el número de variables incluidos en el subconjunto aleatorio en cada división (`mtry`), la complejidad de cada árbol, el esquema de muestreo y la regla de división a utilizar durante la construcción del árbol.

10.3.1. Número de árboles (K)

El primer hiperparámetro a ajustar es el número de árboles que componen el modelo de *random forest*. Su valor debe ser lo suficientemente grande como para que la tasa de error se estabilice. La regla general es que el valor mínimo de árboles sea igual a 10 veces el número de variables incluidas en el modelo. Sin embargo, cuando se tienen en cuenta otros hiperparámetros para optimizar, es posible que el número de árboles se vea afectado. El tiempo de procesamiento aumenta linealmente con la cantidad de árboles incluidos, pero cuantos más se incluyan, se obtendrán estimaciones de error más estables.

10.3.2. Número de variables a considerar (`mtry`)

`mtry` se refiere al hiperparámetro encargado de controlar la aleatorización de variables utilizadas para las particiones de los árboles. Este hiperparámetro ayuda a equilibrar la baja correlación del árbol con los demás, y una razonable fuerza predictiva. Existe un valor predeterminado para este hiperparámetro el cual se puede utilizar en caso de no querer o no poder ajustarlo. En el caso de la regresión, se determina que $mtry = \frac{p}{3}$ siendo p el número de variables incluidas en el modelo. Y en problemas de clasificación, el valor predeterminado es $mtry = \sqrt{p}$. Cuando hay pocas variables relevantes, es decir, los datos son muy ruidosos, tiende a funcionar mejor que el valor de `mtry` sea alto, pues hace que sea más probable seleccionar esas variables. En cambio, cuando muchas variables son importantes, funciona mejor un valor bajo de `mtry`.

10.3.3. Complejidad de los árboles

Un modelo *random forest* se construye con árboles de decisión a los que se les puede controlar su profundidad y su complejidad como se vio en el Cap. 6. Esto se puede hacer ajustando los hiperparámetros de profundidad máxima permitida, tamaño del nodo o la cantidad máxima de nodos terminales.

El tamaño del nodo es el hiperparámetro más común para controlar la complejidad del árbol y la mayoría de las implementaciones usan los valores predeterminados de 1 para árboles de clasificación y 5 para los árboles de regresión, dado que estos valores tienden a producir buenos resultados. Si se quiere controlar el tiempo de procesamiento, se pueden conseguir reducciones significativas del tiempo aumentando el tamaño del nodo impactando de manera marginal en la estimación del error.

10.3.4. Esquema de muestreo

Por defecto, el *random forest* tiene como esquema de muestreo el bootstrapping, explicado anteriormente, en el cual todas las observaciones se muestran con reemplazo. Todas las repeticiones de bootstrap tienen el mismo tamaño que el conjunto de datos de entrenamiento. Sin embargo, el esquema de muestreo se puede ajustar tanto en el tamaño de la muestra como en el diseño muestral (con o sin reposición). El hiperparámetro de tamaño de muestra determina cuántas observaciones se extraen para el entrenamiento de cada árbol. Cuanto menor sea el

tamaño muestral, menor será la correlación entre los árboles, lo cual puede llevar a mejores resultados de precisión en la predicción. La forma de determinar el tamaño muestral óptimo puede hallarse evaluando algunos valores que oscilen entre el 25 % y el 100 %, y en el caso de que haya variables no balanceadas respecto a los valores de las categóricas se puede intentar muestrear sin reposición.

10.3.5. Regla de división

Por defecto, la regla de división que utilizan los árboles de decisión que conforman un *random forest* es la que se presentó en el Cap. 6. Esto es, en el caso de regresión seleccionar la división que minimiza la desviación típica (σ); y en el caso de clasificación la división que minimiza la impureza de Gini o la entropía.

10.3.6. Procedimiento con R: la función `randomForest()`

En el paquete `randomForest` de R se encuentra la función `randomForest()` que se utiliza para entrenar un modelo de este tipo:

```
randomForest(formula, data=..., ...)
randomForest(x, y, xtest, ytest, ntree=500, mtry, ...)
```

- **formula:** Refleja la relación entre la variable dependiente Y y los predictores tal que $Y \sim X_1 + \dots + X_p$.
- **data:** Conjunto de datos con el que entrenar el árbol de acuerdo a la fórmula indicada.
- **x:** Conjunto de datos de entrenamiento que contiene los predictores
- **y:** Vector respuesta con las clases o valores de la variable respuesta.
- **xtest:** Conjunto de datos que contiene los predictores del conjunto de datos de validación.
- **ytest:** Variable respuesta del conjunto de datos de validación.
- **ntree:** Número de árboles a construir en el modelo.
- **mtry:** Número de variables muestreadas aleatoriamente como candidatas en cada partición.

10.3.7. Aplicación del modelo *random forest* en R

En esta sección se aplica el modelo *random forest* al ejemplo de datos de retail incluido en el paquete CDR. Se carga el paquete y con ello, los datos `dp_entr`. Se busca predecir si un cliente va a comprar o no el nuevo producto de acuerdo a los productos que ha consumido, el importe que gasta en ellos y otras características como, por ejemplo, su nivel educativo.

```
library("CDR")
library("randomForest")
library("caret")
library("reshape")
```

10.3. Random Forest

175

```
library("ggplot2")
data(dp_entr)
```

Este algoritmo al estar basado en árboles de clasificación tiene los mismos requisitos para el entrenamiento que tenían dichos árboles, así se construye el modelo usando el conjunto de datos de entrenamiento.

```
# se fija la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13~, data=dp_entr_NUM,
                 method="rf", metric="Accuracy", ntree=500,
                 trControl=trainControl(method="cv",
                                         number=10,
                                         classProbs = TRUE))
```

```
model
Random Forest
558 samples
19 predictor
 2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2    0.8602922  0.7206238
   10   0.8620455  0.7241029
   19   0.8620130  0.7240248

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 10.
```

Los resultados de la validación cruzada se pueden ver en el siguiente boxplot. Se observa como la precisión oscila entre el 80 % y el 95 %. Además, se puede ver en el resultado del modelo que el hiperparámetro *mtry* se ha ajustado a 10 variables.

Finalmente, aunque el *random forest* generado está compuesto por 500 árboles, se puede acceder a cualquiera de ellos para estudiarlos en profundidad. Para ello, es necesario instalar el paquete `reptrree` desde el repositorio <https://github.com/araastat/reptrree>.

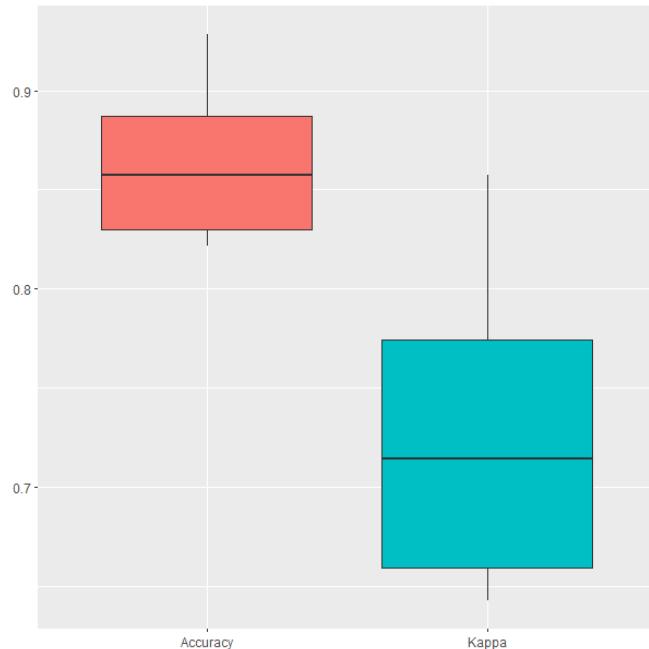


Figura 10.5: Resultados del modelo random forest durante el proceso de validación cruzada.

```
library("devtools")
if(!(`reprtree` %in% installed.packages())){
  devtools::install_github('araastat/reprtree')
}
```

Se pueden observar las decisiones que se toman en el árbol de forma tabulada, indicando qué variable se utiliza para la partición, cuál es el valor que decide la división, indicando si es un nodo terminal (-1) o no (1) y la predicción del nodo, el cual es NA si no es un nodo terminal.

```
set.seed(101)
rf <- randomForest(CLSPRO_pro13~, data = dp_entr_NUM, ntree=500,
                     mtry=unlist(model$bestTune))

# se observa el árbol número 205
tree205 <- getTree(rf, 205, labelVar=TRUE)

head(tree205[,-c(1,2)])
  split var split point status prediction
1 importe_pro15      100     1      <NA>
2 importe_pro12       60      1      <NA>
3 importe_pro16       90      1      <NA>
4 ingresos_ano    156500     1      <NA>
```

10.3. Random Forest

177

```

5 importe_pro17      150     1      <NA>
6     anos_exp       33      1      <NA>

tail(tree205[,-c(1,2)])
  split var split point status prediction
120      <NA>      0.0    -1        N
121      <NA>      0.0    -1        S
122 des_nivel_edu.BASICO  0.5     1      <NA>
123      <NA>      0.0    -1        S
124      <NA>      0.0    -1        S
125      <NA>      0.0    -1        N

```

Este árbol se muestra en la Fig. 10.6.

```

library("rpart")
plot.getTree(rf, k=205)

```

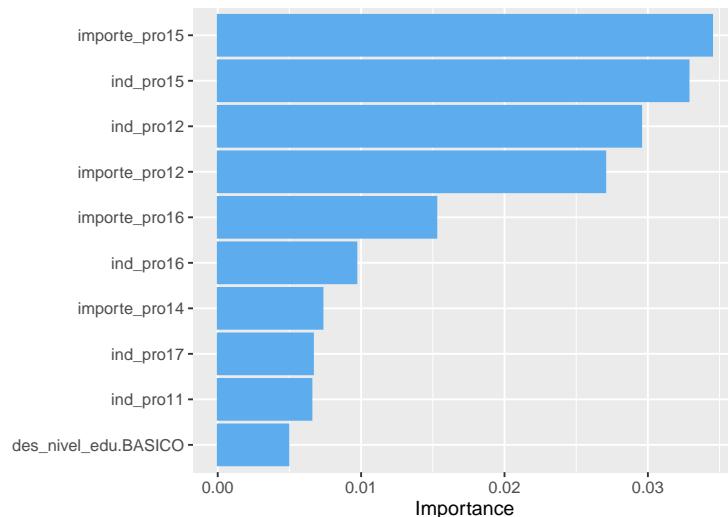


Figura 10.6: Árbol número 205 del random forest entrenado.

Sin embargo, el método por el que se representa gráficamente no es muy claro y puede llevar a confusión o dificultar la interpretación del árbol. Si se desea estudiar hasta cierto nivel del árbol, se puede incluir el argumento `depth` como en el ejemplo abajo mostrado, y que representa el mismo árbol con una profundidad de 5 ramas en la Fig. 10.7.

```

plot.getTree(rf, k=205, depth = 5)

```

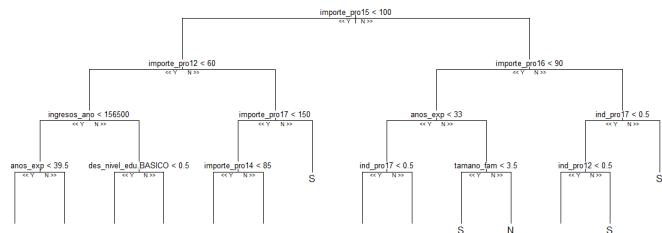


Figura 10.7: Árbol número 205 del random forest entrenado hasta la capa 5.

10.3.7.1. Aplicación del modelo *random forest* con ajuste automático

En este segundo ejemplo, se pretende mejorar la precisión del modelo anterior. Para ello, se ajusta de forma automática los hiperparámetros de dicho algoritmo. De los mencionados anteriormente, solo se va a ajustar automáticamente `mtry`, que es el único incluido el método `rf`.

```
modelLookup("rf")
  model parameter           label forReg forClass probModel
1   rf      mtry #Randomly Selected Predictors  TRUE     TRUE     TRUE
```

Para ajustar el número de árboles y el resto de hiperparámetros, se puede iterar el modelo y probar distintos valores. En una red de opciones se incluyen los valores a probar para el hiperparámetro `mtry`.

```
# Se especifica un rango de valores posibles de mtry
tuneGrid <- expand.grid(mtry = 1:18)
```

A continuación, se entrena el modelo para que se ajuste al valor de `mtry` que maximice el rendimiento predictivo del modelo.

```
# se fija la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13 ~ ., data=dp_entr_NUM,
                 method = "rf", metric = "Accuracy",
                 tuneGrid = tuneGrid,
                 trControl = trainControl(classProbs = TRUE))
```

10.3. Random Forest

179

```

model

Random Forest

558 samples
19 predictor
2 classes: 'S', 'N'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 558, 558, 558, 558, 558, 558, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.8641354  0.7283098
  2     0.8650087  0.7298376
  3     0.8629614  0.7256812
  4     0.8635609  0.7268514
  5     0.8639559  0.7276250
  6     0.8612659  0.7222420
  7     0.8604934  0.7206476
  8     0.8610116  0.7216937
  9     0.8590645  0.7177882
  10    0.8589073  0.7174718
  11    0.8607248  0.7211179
  12    0.8583609  0.7163903
  13    0.8587296  0.7170933
  14    0.8587384  0.7171642
  15    0.8583195  0.7163106
  16    0.8585407  0.7167355
  17    0.8573597  0.7144030
  18    0.8581404  0.7159558

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.

```

Mientras que en el ejemplo anterior el algoritmo sólo probó tres valores de `mtry`, esta vez se realiza una prueba exhaustiva de valores. En el primer ejemplo, el valor del hiperparámetro era `mtry=10`, pero ahora se ha reajustado a `mtry=2`. Esto es equivalente a decir que 2 variables seleccionadas en cada partición son suficientes, y que no son necesarias 10 como en el ejemplo anterior. Finalmente, se puede observar en la Fig. 10.8 los resultados obtenidos durante la validación cruzada. Se observa cómo no sólo la precisión es mayor que en el ejemplo anterior, sino que además los resultados tienen menos dispersión.

```

ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)

```

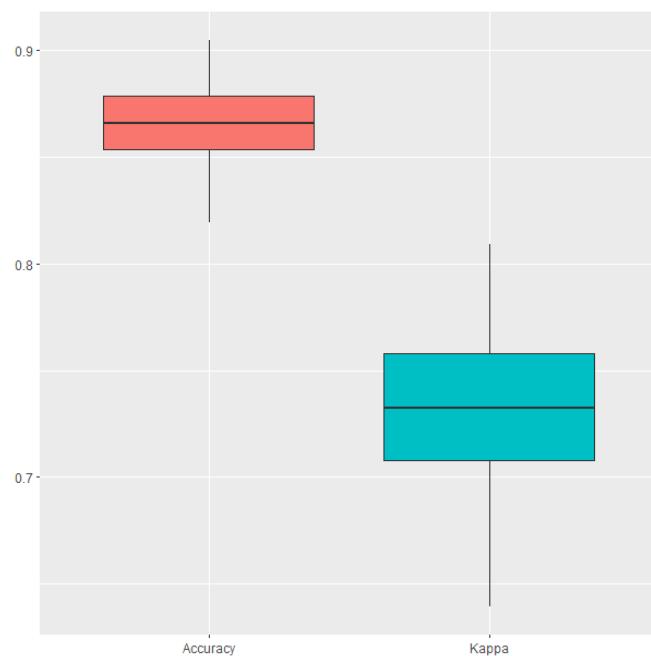


Figura 10.8: Resultados obtenidos por el random forest con ajuste automático durante el proceso de validación cruzada.

Resumen

En este capítulo se introduce al lector en el *bagging* y el algoritmo de aprendizaje supervisado conocido como *random forest*, en concreto:

- Se presenta el concepto de aprendizaje ensamblado, y se profundiza en uno de sus paradigmas: el *bagging*.
- Se implementa el *bagging* en R a través de un caso de clasificación binaria.
- Se expone cómo medir la importancia de las variables incluidas en un modelo *bagging* para facilitar su interpretación.
- Se explica el modelo *random forest*, fundamentado en los árboles decisión y en el *bagging*. Así como los hiperparámetros más importantes para ajustar el modelo de mayor precisión.
- Se presenta un ejemplo de clasificación binaria utilizando el modelo *random forest* en R.

Capítulo 11

Boosting y el algoritmo XGBoost

Ramón A. Carrasco^a e Itzcóatl Bueno^{b,a}

^aUniversidad Complutense de Madrid ^bInstituto Nacional de Estadística

11.1. Métodos ensamblados: bagging vs boosting

En el Cap. 10 se presentó la idea de métodos ensamblados. Una serie de modelos útiles cuando ningún modelo de aprendizaje supervisado es capaz de explicar bien la variable dependiente de interés. En el aprendizaje ensamblado se entrena un gran número de modelos con menor precisión, y se combinan sus predicciones para obtener un metamodelo de una precisión más alta. Los dos modelos de aprendizaje ensamblado más populares son el *bagging* (Cap. 10) y el *boosting*.

La principal diferencia entre ellos radica en cómo se combinan los modelos individuales para obtener una predicción final. Por ejemplo, si se quisiera organizar una fiesta y se tuviese que tomar una decisión sobre el tema para la decoración, se podría hacer tanto con *bagging* como con *boosting*. Si se utilizase el *bagging*, se pediría opinión a distintos grupos de amigos. Después, se combinarán todas sus ideas para tomar una decisión final sobre la decoración de la fiesta. Por otro lado, si se utiliza *boosting*, en lugar de preguntarle a diferentes grupos de amigos al mismo tiempo, se pediría a un amigo en particular su opinión. Si su respuesta no es convincente, entonces se busca la ayuda de otro amigo, quien se enfocará en mejorar la respuesta anterior. Este proceso continúa secuencialmente, solicitando la ayuda de diferentes amigos y construyendo sobre las respuestas anteriores para obtener una decisión final más precisa y refinada sobre la decoración de la fiesta.

11.2. ¿Qué es el boosting?

El *boosting* (Schapire and Freund, 2012) es el otro de los paradigmas de aprendizaje ensamblado, presentado en el Cap. 10. Como el *bagging*, el *boosting* agrega múltiples modelos con menor precisión (débiles) combinando sus predicciones para obtener un metamodelo con una precisión más alta. Los árboles de decisión son los modelos base o débiles que se usan más frecuentemente. En este caso, para llegar al metamodelo a partir de los modelos base, es necesario introducir ponderaciones a los árboles basándose en las clasificaciones erróneas del árbol entrenado previamente a dicho árbol.

El *boosting* reduce el problema del sobreajuste utilizando menos árboles que un modelo *random forest*. Mientras que agregar más árboles al *random forest* ayuda a compensar el sobreajuste, también puede llevar a un aumento del mismo y, por ello, hay que ser cauteloso a la hora de agregar nuevos árboles. Sin embargo, el *boosting* aprende iterativamente de los errores en árboles anteriores, pudiendo llevar a sobreajustar el modelo. Aunque este enfoque produce predicciones más precisas, muchas veces mejores a la mayoría de algoritmos, puede llevar a ajustar las observaciones atípicas. Es por esto que el *random forest* es una técnica más recomendada cuando se trabaja con conjuntos de datos muy complejos con un gran número de observaciones atípicas.

Otra de las grandes desventajas del *boosting* es que su tiempo de procesamiento es muy elevado, puesto que su entrenamiento sigue una lógica secuencial. En el proceso de entrenamiento, un árbol debe esperar a que el inmediatamente anterior sea entrenado, para iniciar su entrenamiento, y esto limita la escalabilidad del modelo. Mientras tanto, un *random forest* entrena los árboles en paralelo, lo que hace que su tiempo de procesamiento sea más rápido.

Tanto los algoritmos de *boosting* como a los de *bagging* presentan el inconveniente de la dificultad de interpretación que tienen respecto a los árboles de decisión. En este aspecto estos algoritmos de ensamblado se pueden considerar como de caja negra.

11.3. Gradient Boosting (GB)

Uno de los algoritmos de *boosting* más conocidos es el **gradient boosting**. Mientras que el *random forest* seleccionaba combinaciones aleatorias de variables en cada proceso de construcción de un árbol, el *gradient boosting* selecciona variables que mejoren la precisión con cada nuevo árbol. Por lo tanto, la construcción del modelo es secuencial, puesto que cada nuevo árbol se construye utilizando información derivada del árbol anterior y, en consecuencia, la construcción de estos árboles no son independientes. En cada iteración se registran los errores cometidos en los datos de entrenamiento y se tienen en cuenta para la siguiente ronda de entrenamiento. Además, se incorporan ponderaciones, como se observa en la Fig. 11.1 a los datos basándose en los resultados de la iteración anterior. Las ponderaciones más altas se aplicarán a las observaciones que fueron erróneamente clasificadas, y no se dará tanta atención a las bien clasificadas. Este proceso se repite hasta que se llega a un nivel bajo de error. El resultado final se obtiene a través de la media ponderada de las predicciones de los árboles de decisión.

Matemáticamente, un algoritmo *gradient boosting* para clasificación sigue los pasos que a continuación se detallan. Sea un problema de clasificación binaria y, asumiendo que se tienen K

11.3. Gradient Boosting (GB)

185

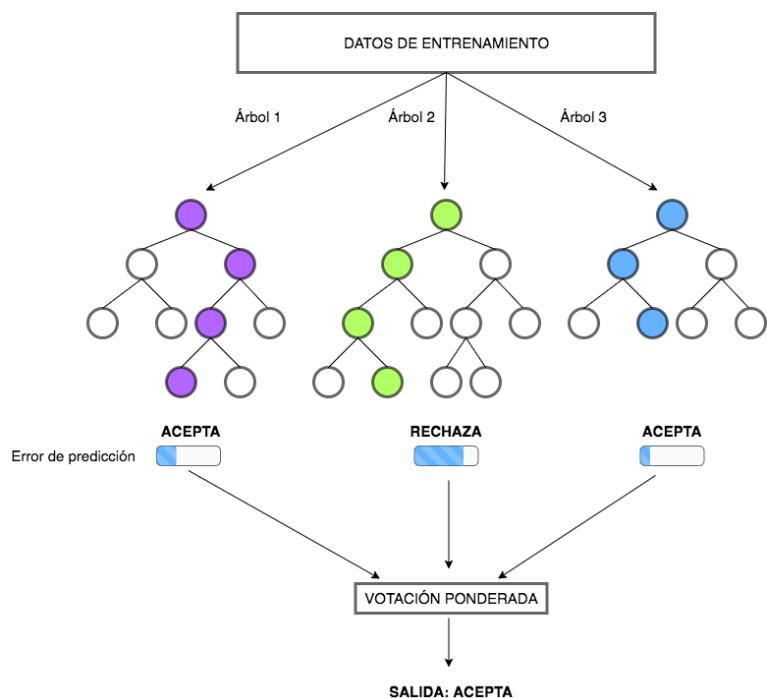


Figura 11.1: Ejemplo de Boosting.

árboles de decisión de clasificación, la predicción del modelo ensamblado se obtiene utilizando la función sigmoidal, como en la regresión logística (Cap. ??), tal que:

$$P(y = 1|x, f) = \frac{1}{1 + e^{-f(x)}} \quad (11.1)$$

Donde $f(x) = \sum_{\kappa=1}^K f_\kappa(x)$ y f_m es un árbol de decisión. De nuevo, como en la regresión logística, se aplica el principio de máxima verosimilitud tratando de hallar una f que maximice $\mathcal{L}_f = \sum_{i=1}^N \ln(P(y_i = 1|x_i, f))$

El algoritmo, en origen, es un modelo constante de la forma $f = f_0 = \frac{p}{1-p}$ donde $p = \frac{1}{N} \sum_{i=1}^N y_i$. Tras cada iteración se añade un nuevo árbol f_κ al modelo. Para encontrar el mejor árbol f_κ , la primera derivada parcial g_i del modelo actual se obtiene para $i = 1, \dots, N$:

$$g_i = \frac{\delta \mathcal{L}_f}{\delta f} \quad (11.2)$$

Donde f es el modelo de clasificación ensamblado construido en la iteración previa. Se necesita obtener las derivadas de $\ln(P(y_i = 1|x_i, f))$ con respecto a f para todo i para poder calcular g_i . Nótese que:

$$\ln(P(y_i = 1|x_i, f)) = \ln\left(\frac{1}{1 + e^{-f(x_i)}}\right) \quad (11.3)$$

Y, por tanto, la derivada respecto a f es igual a:

$$\frac{\delta \ln(P(y_i = 1|x_i, f))}{\delta f} = \frac{1}{e^{f(x_i)} + 1} \quad (11.4)$$

Después, se reemplaza en el conjunto de entrenamiento la categoría original y_i por su correspondiente derivada parcial g_i y se construye un nuevo modelo f_κ utilizando el conjunto de entrenamiento transformado. Tras esto, se obtiene la actualización óptima (ρ_κ) como:

$$\rho_\kappa = \arg \max_{\rho} \mathcal{L}_{f+\rho f_\kappa} \quad (11.5)$$

Al terminar la iteración κ , se actualiza el modelo ensamblado f añadiendo el nuevo árbol f_κ :

$$f \leftarrow f + \alpha \rho_\kappa f_\kappa \quad (11.6)$$

Se itera hasta que $\kappa = K$, entonces el proceso se detiene y se obtiene el modelo ensamblado final f .

11.3.1. Hiperparámetros del modelo *gradient boosting*

Un modelo de *gradient boosting* tiene dos tipos de hiperparámetros:

- Hiperparámetros de *boosting*.
- Hiperparámetros del árbol.

11.3.1.1. Hiperparámetros de *boosting*

Los hiperparámetros de *boosting* son principalmente dos: el número de árboles y la tasa de aprendizaje.

El primero indica el número de árboles a construir y que, como se ha comentado, es importante optimizar para evitar el sobreajuste del modelo. A diferencia de los modelos *random forest* o *bagging*, en el *boosting* los árboles crecen en secuencia para que cada árbol corrija los errores del anterior. El número de árboles necesarios para que el modelo sea buen predictor puede verse incrementado en función de los valores que tomen los otros hiperparámetros.

La tasa de aprendizaje es el hiperparámetro con el que se determina la contribución de cada árbol en el resultado final y controla la rapidez con la que el algoritmo avanza por el descenso del gradiente, es decir, la velocidad a la que aprende. Este hiperparámetro toma valores entre 0 y 1, aunque los valores habituales oscilan entre 0,001 y 0,3. El modelo es más robusto a las características específicas de cada árbol, permitiendo una buena generalización, cuando la tasa de aprendizaje toma valores bajos. Estos valores también facilitan la parada temprana antes del sobreajuste del modelo. Sin embargo, utilizar estos valores vuelve al modelo más exigente computacionalmente y dificulta alcanzar el modelo óptimo con un número fijo de árboles. En resumen, cuanto menor sea este valor, más preciso puede ser el modelo, pero también requerirá más árboles en la secuencia.

11.3.1.2. Hiperparámetros de árbol

Los principales hiperparámetros de árbol son: la profundidad del árbol y el número mínimo de observaciones en nodos terminales, como se vio en el Cap. 6.

El primer hiperparámetro controla la profundidad de los árboles individuales. Los valores habituales de profundidad oscilan entre 3 y 8. Los árboles de menor profundidad son eficientes computacionalmente, pero menos precisos. Sin embargo, los árboles de mayor profundidad permiten que el algoritmo capture interacciones únicas, aunque aumentan el riesgo de sobreajuste.

El segundo hiperparámetro, además de controlar el número mínimo de observaciones en los nodos terminales, controla la complejidad de cada árbol. Los valores típicos de este hiperparámetro suelen estar entre 5 y 15. Los valores más altos ayudan a evitar que un modelo aprenda relaciones que pueden ser muy específicas de la muestra particular seleccionada para entrenar el árbol, evitando así el sobreajuste. Sin embargo, los valores más pequeños pueden ayudar con clases desbalanceadas en problemas de clasificación.

11.3.2. Estrategia de ajuste de hiperparámetros

A diferencia del *random forest*, los modelos *gradient boosting* pueden variar mucho en su precisión de acuerdo a su configuración de hiperparámetros. Por ello, el ajuste puede requerir seguir una estrategia. Un buen enfoque para esto es:

- Elegir una tasa de aprendizaje relativamente alta. El valor predeterminado es 0,1 y generalmente funciona. Sin embargo, para la mayoría de problemas funcionan valores entre 0,05 y 0,2.
- Determinar el número óptimo de árboles para la tasa de aprendizaje elegida.
- Ajustar los hiperparámetros del árbol y la tasa de aprendizaje y evaluar la velocidad frente al rendimiento.
- Ajustar los hiperparámetros específicos del árbol para determinar la tasa de aprendizaje.
- Una vez que se ajustan los parámetros específicos del árbol, se reduce la tasa de aprendizaje para evaluar cualquier mejora en la precisión.
- Utilizar la configuración final de hiperparámetros y aumentar los procedimientos de validación cruzada para obtener estimaciones más robustas. Si se utiliza validación cruzada en los pasos anteriores, entonces este paso no es necesario.

11.3.3. Procedimiento con R: la función gbm()

En el paquete **gbm** de **R** se encuentra la función con el mismo nombre **gbm()** que se utiliza para entrenar un modelo *gradient boosting*:

```
gbm(formula, data=..., ...)
```

- **formula**: Refleja la relación entre la variable dependiente Y y los predictores tal que $Y \sim X_1 + \dots + X_p$.
- **data**: Conjunto de datos con el que entrenar el árbol de acuerdo a la fórmula indicada.

11.3.4. Aplicación del modelo *gradient boosting* en R

A través de los datos de compras **dp_entr** incluidos en el paquete CDR se va a aplicar el modelo *gradient boosting* para clasificar qué clientes van a comprar un nuevo producto (*tensiómetro digital*) y quienes no. Se entrena el modelo utilizando el conjunto de datos de entrenamiento sin transformar (en su escala original). Así, en lugar de tener las variables categóricas transformadas mediante one-hot-encoding se usan en su escala original, como ocurre con el caso de la variable que mide el nivel educativo.

```
library("CDR")
library("caret")
library("gbm")
library("reshape")
```

11.3. Gradient Boosting (GB)

189

```

library("ggplot2")
data(dp_entr)

# se determina la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13 ~.,
                 data=dp_entr,
                 method="gbm",
                 metric="Accuracy",
                 trControl = trainControl(classProbs = TRUE,
                                           method = "cv", number = 10)
)

```

model

Stochastic Gradient Boosting

558 samples
17 predictor
2 classes: 'S', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 502, 502, 502, 503, 503, 502, ...
Resampling results across tuning parameters:

interaction.depth	n.trees	Accuracy	Kappa
1	50	0.8564610	0.7130031
1	100	0.8690909	0.7383556
1	150	0.8762338	0.7526413
2	50	0.8690909	0.7382344
2	100	0.8762338	0.7526413
2	150	0.8799026	0.7599227
3	50	0.8763636	0.7528004
3	100	0.8781494	0.7563575
3	150	0.8835390	0.7671499

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

El modelo resultante del proceso de entrenamiento es un *gradient boosting* que ha ajustado los hiperparámetros a 150 árboles y una profundidad igual a 3. Además, los valores tanto del

número mínimo de observaciones en nodos como de la tasa de aprendizaje, toman los valores por defecto de 10 y 0,1, respectivamente. Los resultados en el proceso de validación cruzada se muestran en la Fig. 11.2, en el que se observa como la precisión oscila entre el 84% y el 93% en las iteraciones.

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

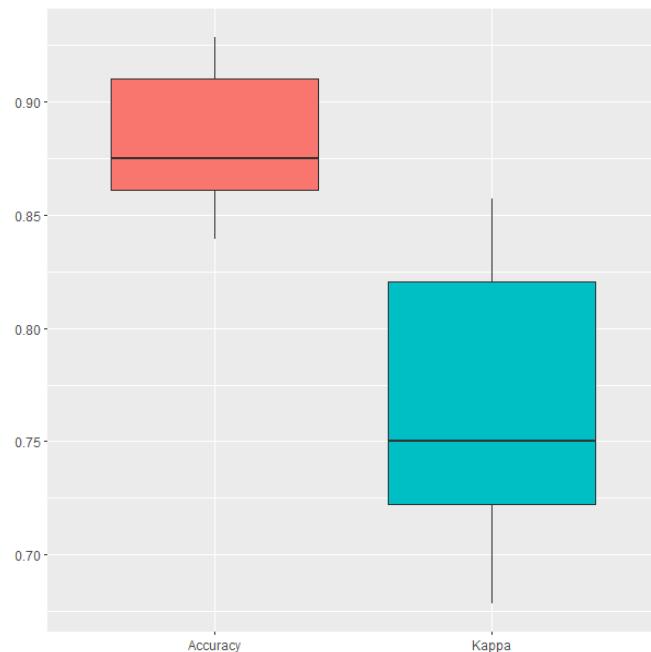


Figura 11.2: Resultados del modelo GB obtenidos durante el proceso de validación cruzada.

11.3.5. Gradient Boosting con ajuste automático

Se repite el procedimiento para el ejemplo anterior. Sin embargo, en este ejemplo se ajustan de forma automática los hiperparámetros más relevantes de dicho algoritmo para mejorar los resultados respecto al modelo anterior. Se observa como los hiperparámetros a ajustar para el método `gbm` son: el número de árboles, la profundidad, la tasa de aprendizaje y el número de observaciones en un nodo.

```
modelLookup("gbm")
model      parameter          label forReg forClass probModel
1  gbm        n.trees    # Boosting Iterations   TRUE     TRUE     TRUE
```

11.3. Gradient Boosting (GB)

191

2	gbm	interaction.depth	Max Tree Depth	TRUE	TRUE	TRUE
3	gbm	shrinkage	Shrinkage	TRUE	TRUE	TRUE
4	gbm	n.minobsinnode	Min. Terminal Node Size	TRUE	TRUE	TRUE

Siguiendo la estrategia descrita se definen rangos de posibles valores para los principales hiperparámetros a optimizar.

```
# Se especifica un rango de valores posibles para los hiperparámetros
tuneGrid <- expand.grid(interaction.depth = c(4,6,8),
                         n.trees = c(10*ncol(dp_entr),300,500),
                         shrinkage = c(0.05,0.1,0.2),
                         n.minobsinnode = c(5,10,15))
```

Esta red de posibles valores para los hiperparámetros del modelo se incorporan a la función de entrenamiento. Cuanto más exhaustivo sea el ajuste de estos valores, mayor será el tiempo de ajuste del modelo. La red presentada está formada por 81 combinaciones de los posibles cuatro hiperparámetros.

```
# se fija la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13~.,
                 data=dp_entr,
                 method="gbm",
                 metric="Accuracy",
                 trControl=trainControl(classProbs = TRUE,
                                         method="cv", number=10),
                 tuneGrid=tuneGrid)
```

El modelo que mejores resultados proporciona es aquel que ajusta los hiperparámetros a los siguientes valores: 180 árboles, una profundidad igual a 6, una tasa de aprendizaje de 0.05 y un tamaño mínimo de los nodos de 10 observaciones.

```
model$bestTune
  n.trees interaction.depth shrinkage n.minobsinnode
13      180              6       0.05        10
```

En la Fig. 11.3 se muestran los resultados obtenidos durante el proceso de validación cruzada. Se puede ver que los resultados son similares a los del modelo anterior, aunque hay diferencias importantes. En primer lugar, se alcanza un valor máximo de precisión mayor al anterior, pues en este caso la precisión oscila entre el 84 % y el 95 %. En segundo lugar, vemos que el valor mediano de la precisión ha subido del 87.5 % del modelo anterior hasta el 90 % de este modelo. Por último, que el rendimiento haya variado tan poco desde el modelo por defecto a un modelo

en el que se ha intentado ajustar los hiperparámetros, confirma lo ya expuesto sobre el buen rendimiento de un modelo de *gradient boosting* con los parámetros por defecto.

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

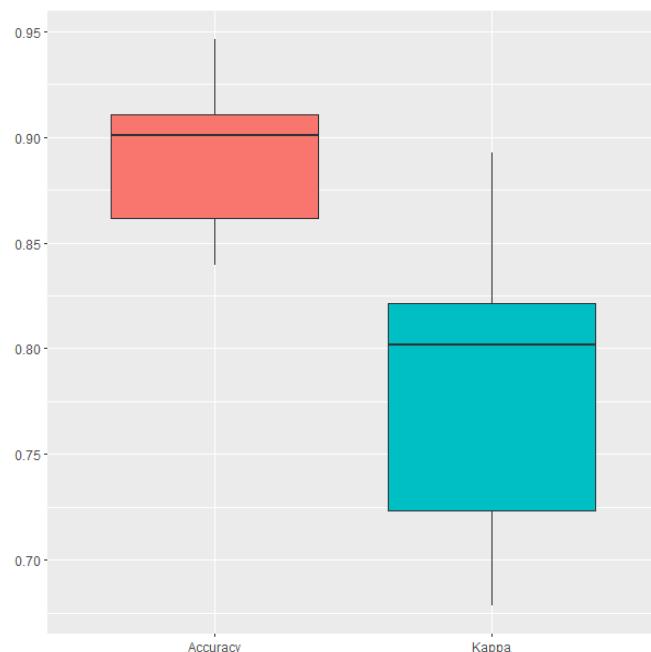


Figura 11.3: Resultados del modelo GB con ajuste automático obtenidos durante el proceso de validación cruzada.

11.4. eXtreme Gradient Boosting (XGB)

El *eXtreme Gradient Boosting* es una implementación eficiente y escalable del modelo *gradient boosting*. Este modelo, abreviado como XGBoost, es un paquete de código abierto en C++, Java, Python ([Wade, 2020](#)), R, Julia, Perl y Scala. En R el modelo se incluye dentro del paquete `xgboost` ([Chen et al., 2015](#)). El paquete incluye un procedimiento para la solución eficiente de modelos lineales y un algoritmo de aprendizaje de árboles.

El paquete es compatible con funciones objetivo de regresión, clasificación y ranking. Además, tiene varias características importantes:

1. Velocidad: `xgboost` puede realizar automáticamente cálculos paralelos. Por lo general, es 10 veces más rápido que el modelo *gradient boosting*.

2. Tipo de entrada: `xgboost` toma varios tipos de datos de entrada en R:

- Matriz densa (`matrix`)
- Matriz dispersa (`Matrix::dgCMatrix`)
- Archivo de datos locales
- Un tipo de datos propio del paquete: `xgb.DMatrix`

3. Dispersion: `xgboost` acepta datos de entrada dispersos para los modelos incluidos.

4. Personalización: `xgboost` admite tanto funciones de objetivo y funciones de evaluación personalizadas.

5. Rendimiento: `xgboost` alcanza generalmente una mayor precisión.

11.4.1. Hiperparámetros del modelo XGBoost

El modelo XGBoost proporciona los hiperparámetros que ya incluía el modelo *gradient boosting* referentes tanto al *boosting* como a los árboles. Sin embargo, `xgboost` también proporciona hiperparámetros adicionales que pueden ayudar a reducir las posibilidades de sobreajuste, lo que lleva a una menor variabilidad de predicción y, por lo tanto, a una mayor precisión. Estos hiperparámetros son: la regularización y el dropout.

Los parámetros de regularización se incluyen para ayudar a evitar el sobreajuste y reducir la complejidad del modelo. Existen tres hiperparámetros que tienen esta funcionalidad: γ (γ), α (α) y λ (λ). γ es un hiperparámetro de pseudo-regularización conocido como multiplicador Lagrangiano y controla la complejidad de un árbol dado. Este hiperparámetro establece que para hacer una partición adicional en un nodo es necesaria una reducción de pérdida mínima especificada por γ . Al especificarlo, el modelo XGBoost hace crecer los árboles hasta una profundidad máxima establecida, pero en un paso de poda eliminará las divisiones que no cumplen con la regularización γ . Este hiperparámetro toma valores entre 0 e infinito (∞), siguiendo la regla de que a mayor valor, mayor será la regularización. Los otros hiperparámetros de regularización, α y λ , son más clásicos. Mientras que α proporciona una regularización L_1 , λ proporciona una regularización L_2 . Estos parámetros de regularización establecen un límite a cómo de extremos pueden llegar a ser los pesos de los nodos en un árbol. Sus valores se encuentran, al igual que los de γ , entre 0 y ∞ .

El dropout es un enfoque alternativo para reducir el sobreajuste. Cuando se entrena un modelo de *gradient boosting*, los primeros árboles tienden a dominar el rendimiento del modelo, mientras que los que se agregan después suelen mejorar la predicción solo para un pequeño grupo de variables. Esto puede llevar a que se incremente el riesgo de sobreajuste. Con el dropout, se descartan árboles aleatoriamente en el proceso de entrenamiento.

En su implementación en R, el modelo XGBoost incluye principalmente los siguientes parámetros para ser optimizados: número de iteraciones, profundidad máxima de los árboles, tasa de aprendizaje y la regularización γ .

```
head(modelLookup("xgbTree"),4)
  model parameter           label forReg forClass probModel
1 xgbTree    nrounds  # Boosting Iterations  TRUE   TRUE   TRUE
2 xgbTree  max_depth      Max Tree Depth  TRUE   TRUE   TRUE
3 xgbTree      eta        Shrinkage  TRUE   TRUE   TRUE
4 xgbTree     gamma Minimum Loss Reduction  TRUE   TRUE   TRUE
```

11.4.2. Procedimiento con R: la función xgboost()

En el paquete `xgboost` de **R** se encuentra la función `xgboost()` que se utiliza para entrenar un modelo *extreme gradient boosting*:

```
xgboost(data = ..., label = ..., ...)
```

- `data`: Conjunto de datos con el que entrenar el modelo.
- `label`: Vector con la variable respuesta.

11.4.3. Aplicación del modelo XGBoost en R

Se entrena este modelo utilizando el conjunto de entrenamiento sin transformar (en su escala original). Se continúa así el ejemplo expuesto durante la aplicación del modelo *gradient boosting* sin y con ajuste automático de sus hiperparámetros. Se repite el procedimiento de entrenar el modelo para los hiperparámetros por defecto que proporciona R.

```
# se determina la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13~.,
                 data=dp_entr,
                 method="xgbTree",
                 metric="Accuracy",
                 trControl=trainControl(classProbs = TRUE,
                                         method = "cv",
                                         number=10))
```

Por defecto, el entrenamiento establece valores constantes para la regularización γ (0) y para el tamaño mínimo del nodo (1). En cambio, ajusta los hiperparámetros del modelo dentro de los valores por defecto de la función. Así, el modelo XGBoost resultante tiene 50 iteraciones, una profundidad máxima igual a 2 y una tasa de aprendizaje de 0,3. Los resultados de la validación cruzada muestran que la precisión obtenida oscila entre el 85 % y el 95 %, resultado similar al del *gradient boosting* con hiperparámetros ajustados. Sin embargo, el valor mediano de la precisión es del 88 %, ligeramente inferior a la observada en el modelo *gradient boosting* con ajuste automático.

11.4. eXtreme Gradient Boosting (XGB)

195

```
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

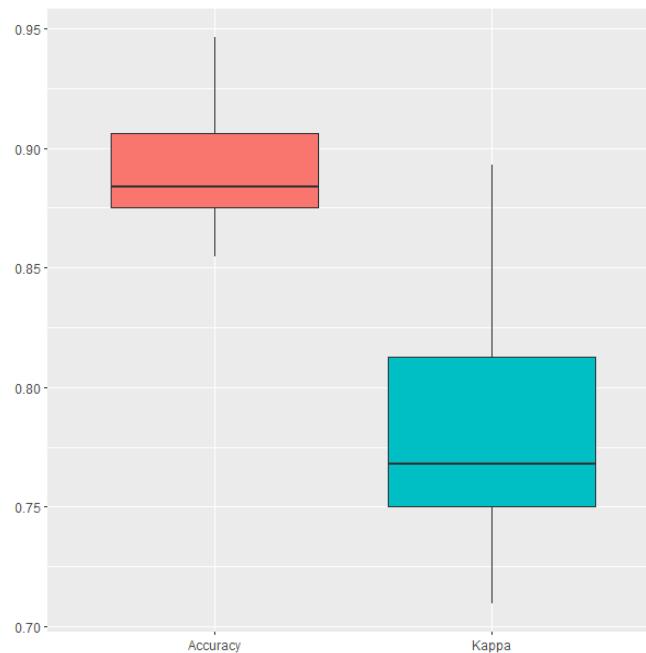


Figura 11.4: Resultados del modelo durante la validación cruzada.

11.4.4. XGBoost con ajuste automático

Se continúa el ejemplo aplicado a los datos sobre compra de un nuevo producto por parte de los clientes utilizando un modelo XGBoost en R. Sin embargo, se quieren mejorar los resultados obtenidos, y por ello ajustan automáticamente los hiperparámetros más relevantes de dicho algoritmo generando una red de posibles valores para dichos hiperparámetros. Por motivos computacionales, ésta no se hace excesivamente exhaustiva para evitar largos tiempos de entrenamiento. Si se dispone de tiempo suficiente para el entrenamiento, es aconsejable tratar de estudiar más valores para los hiperparámetros a optimizar.

```
# Se especifica un rango de valores típicos para los hiperparámetros
tuneGrid <- expand.grid (nrounds=c(50,100,500),
                         max_depth = c(3,4,8),
                         eta =c(0.05,0.1,0.2,0.3),
                         gamma=c(0,0.5,5),
                         colsample_bytree=c(0.8),
```

```

    min_child_weight=c(5),
    subsample=c(0.5))

# se determina la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLSPRO_pro13~.,
                 data=dp_entr,
                 method="xgbTree",
                 metric="Accuracy",
                 trControl=trainControl(classProbs = TRUE,
                                         method = "cv",
                                         number = 10),
                 tuneGrid=tuneGrid)

model$bestTune[,1:4]
  nrounds max_depth eta gamma
71      100          4 0.2     5

```

El modelo resultante establece que se utilicen 100 iteraciones, que los árboles tengan una profundidad máxima de 4, una tasa de aprendizaje del 0,2 y que la regularización γ tome el valor 5.

Los resultados obtenidos durante la validación cruzada muestran que la precisión es muy similar a la del modelo por defecto, al encontrarse entre el 85 % y el 95 %. Sin embargo, se observa en el valor mediano de la precisión una ligera mejoría, al aumentar hasta el 90 %.

Resumen

En este capítulo se introduce al lector en el algoritmo de aprendizaje supervisado conocido como *gradient boosting*, en concreto:

- Se presenta el otro paradigma principal de aprendizaje ensamblado: el *boosting*.
- Se explica el modelo basado en este paradigma, el *gradient boosting*, así como sus diferencias con el *random forest* (basado en *bagging*).
- Se exponen los hiperparámetros más relevantes a la hora de optimizar un modelo de *gradient boosting*.
- Se presenta el *eXtreme gradient boosting*, una implementación eficiente y escalable del modelo *gradient boosting*. Así como los hiperparámetros de regularización y otros parámetros importantes en esta implementación.
- Se aplican ambos algoritmos en R en un caso práctico para la clasificación binaria de datos.

11.4. *eXtreme Gradient Boosting (XGB)*

197

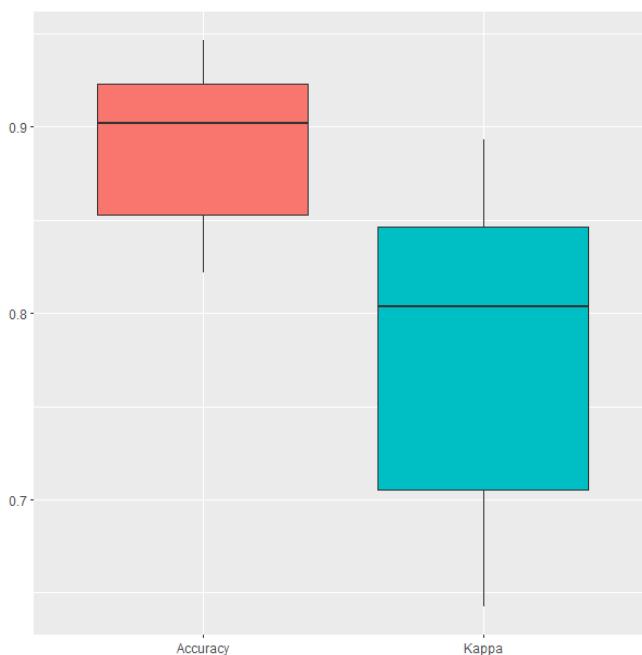


Figura 11.5: Resultados del modelo durante la validación cruzada.

Parte II

Machine learning no supervisado

Capítulo 12

Análisis cluster: clusterización jerárquica

José-María Montero^a y Gema Fernández-Avilés^a

^aUniversidad de Castilla-La Mancha

12.1. Introducción

El origen de la actividad agrupatoria, hoy en día conocida como análisis cluster o de conglomerados (AC), taxonomía numérica o reconocimiento de patrones, entre otras denominaciones, se remonta a tiempos de Aristóteles y su discípulo Teofrasto. Por tanto, tiene unas profundas raíces y hoy en día se aplica en todos los campos del saber. Se ha evitado la palabra “clasificación” porque existe una pequeña diferencia entre agrupación y clasificación. En la actividad clasificatoria se conoce el número de grupos y qué observaciones del conjunto de datos pertenecen a cada uno, siendo el objetivo clasificar nuevas observaciones en los grupos ya existentes. En la actividad agrupatoria, el número de grupos puede ser conocido (normalmente no lo es), pero no las observaciones que pertenecen a cada uno de ellos, siendo el objetivo la asignación de dichas observaciones a diferentes grupos. Este y el siguiente capítulo se centran en este último problema, al cual se hará referencia por su denominación más popular: AC.

El AC está orientado a la síntesis de la información contenida en un conjunto de datos, normalmente una muestra relativa a objetos, individuos o, en general, elementos, definidos por una serie de características, con vistas a establecer una agrupación de los mismos en función de su mayor o menor homogeneidad. En otros términos, el AC trata de agrupar dichos elementos en grupos mutuamente excluyentes, de tal forma que los elementos de cada grupo sean lo más parecidos posible entre sí y lo más diferentes posible de los pertenecientes a otros grupos (Fig. 12.1).

Para llevar a cabo un AC, se deben tomar una serie de decisiones:

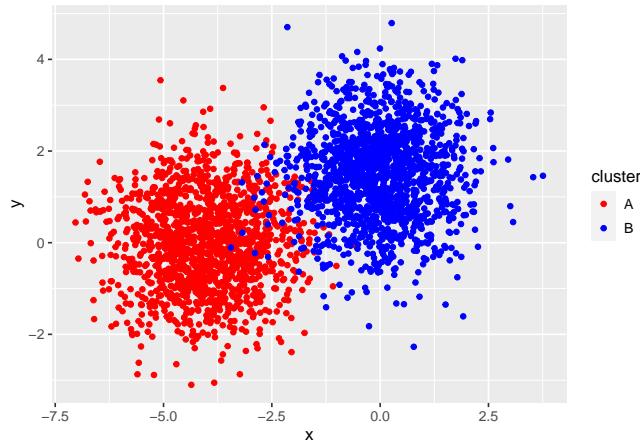


Figura 12.1: Datos simulados que presentan clusters

- Selección de las variables en función de las cuales se van a agrupar los elementos.
- Elección del tipo de distancia o medida de similitud que se va a utilizar para medir la disimilitud entre los elementos objeto de clasificación.
- Elección de la técnica para formar los grupos o conglomerados.
- Determinación del número óptimo de clusters (si no se determina a priori).

En este capítulo se abordarán la primera y, sobre todo, la segunda cuestión, dejando las otras dos para el capítulo siguiente.

Como ilustración práctica, se utilizará la base de datos TIC2021 del paquete CDR, relativa a las estadísticas de uso de las TIC en la Unión Europea en 2021.

12.2. Selección de las variables

La selección de las p variables o características, $\{X_1, X_2, \dots, X_p\}$, en función de las cuales se va a proceder a la agrupación de los n elementos disponibles es crucial, ya que determina la agrupación final, independientemente de los procedimientos técnicos utilizados. Una vez determinadas éstas, la información disponible, para los elementos objeto de agrupación, será:

Tabla 12.1: Información muestral

	X_1	X_2	X_3	...	X_p
Elemento 1	x_{11}	x_{12}	x_{13}	...	x_{1p}
Elemento 2	x_{21}	x_{22}	x_{23}	...	x_{2p}
Elemento 3	x_{31}	x_{32}	x_{33}	...	x_{3p}
...

12.3. Elección de la distancia entre elementos

203

	X_1	X_2	X_3	\cdots	X_p
Elemento n	x_{n1}	x_{n2}	x_{n3}	\cdots	x_{np}

En definitiva, la información de partida es una matriz $\mathbf{X}_{n \times p}$, donde cada elemento viene representado por un punto en el espacio p -dimensional de variables, es decir, una matriz que proporciona los valores de las variables para cada elemento.¹

Una cuestión a tener en cuenta es el número de variables a considerar en el AC. La exclusión de variables relevantes generará una agrupación deficiente. La inclusión de variables irrelevantes complicará el proceso de agrupamiento sin procurar ganancias sustantivas. Dado que el miedo del investigador vendrá por el lado de la exclusión de variables relevantes, tenderá a incluir un número excesivo de variables (muchas de ellas correlacionadas). Por ello, se recomienda realizar previamente un ACP (véase Cap. 14), lo que reduce la dimensionalidad del problema, y llevar a cabo el AC a partir de las componentes principales retenidas (incorreladas, evitando así redundancias). La eliminación de información redundante es una cuestión importante en el proceso de clusterización, porque dicha información estaría sobreponderada en el resultado obtenido. Una solución menos drástica a este problema es la utilización de la distancia de Mahalanobis, que, como se verá posteriormente, corrige estas redundancias.

Otra cuestión importante en este momento es decidir si las variables (o componentes principales en su caso) seleccionadas se utilizarán estandarizadas o no. No existe consenso sobre la cuestión, si bien se suele recomendar su estandarización para evitar consecuencias no deseadas derivadas de la distinta escala y/o unidades de medida. No obstante, autores tan relevantes como Edelbrock (1979) y Brian (1993), están en contra y proponen las siguientes alternativas: (i) recategorizar todas las variables en variables binarias, y aplicar a éstas una distancia apropiada para ese tipo de medidas; (ii) realizar distintos AC con grupos de variables homogéneas (en cuanto a su métrica) y sintetizar después los diferentes resultados; y (iii) utilizar la distancia de Gower, que es aplicable con cualquier tipo de métrica.

12.3. Elección de la distancia entre elementos

Una vez se dispone de la matriz de información $\mathbf{X}_{n \times p}$, la segunda etapa en el AC consiste en la creación de una nueva matriz $\mathbf{D}_{n \times n}$ cuyos elementos $\{d_{ij}\}$ sean las distancias o disimilaridades entre los elementos objeto de agrupamiento.

En caso de variables cuantitativas, la distancia entre dos elementos en un espacio de p dimensiones, $d(\mathbf{x}_i; \mathbf{x}_j)$, se define como una función que a cada dos puntos de \mathbb{R}^p le asocia un número real y que verifica:²

- $d(\mathbf{x}_i; \mathbf{x}_j) \geq 0$,

¹También podría observarse \mathbf{X} por columnas (la j -ésima columna muestra los valores de la j -ésima variable para cada elemento de la muestra). Aparentemente, no hay razón alguna para no poder agrupar las variables que describen cada elemento (cluster de variables en vez de elementos).

²Si se cumple el cuarto requisito la función distancia suele llamarse distancia métrica.

- $d(\mathbf{x}_i; \mathbf{x}_j) = 0$ si y sólo si $\mathbf{x}_i = \mathbf{x}_j$,
- $d(\mathbf{x}_i; \mathbf{x}_j) = d(\mathbf{x}_j; \mathbf{x}_i)$,
- $d(\mathbf{x}_i; \mathbf{x}_j) + d(\mathbf{x}_j; \mathbf{x}_k) \geq d(\mathbf{x}_i; \mathbf{x}_k), \quad \forall \mathbf{x}_k \in \mathbb{R}^p$,

Con variables cualitativas, la similitud entre dos elementos, $s(\mathbf{x}_i; \mathbf{x}_j)$, es una función que a cada dos puntos de \mathbb{R}^p le asocia un número real, y que verifica:

- $s(\mathbf{x}_i; \mathbf{x}_j) \leq s_0$, donde s_0 es un número real finito arbitrario (normalmente 1).
- $s(\mathbf{x}_i; \mathbf{x}_j) = s_0$ si y sólo si $\mathbf{x}_i = \mathbf{x}_j$,
- $s(\mathbf{x}_i; \mathbf{x}_j) = s(\mathbf{x}_j; \mathbf{x}_i)$,
- $|s(\mathbf{x}_i; \mathbf{x}_j) + s(\mathbf{x}_j; \mathbf{x}_k)|s(\mathbf{x}_i; \mathbf{x}_k) \geq d(\mathbf{x}_i; \mathbf{x}_j)s(\mathbf{x}_j; \mathbf{x}_k) \quad \forall \mathbf{x}_k \in \mathbb{R}^p$.

Son numerosas las formas de medir las distancias o similaridades entre dos elementos que satisfacen las condiciones expuestas. Las más populares son las siguientes:

Variables cuantitativas

- **Distancia euclídea.** Se define como:

$$d_e(\mathbf{x}_i; \mathbf{x}_j) = \sqrt{\sum_{k=1}^p (x_{ik} - x_{jk})^2}. \quad (12.1)$$

Ignora las unidades de medida de las variables y, en consecuencia, aunque es invariante a los cambios de origen, no lo es a los cambios de escala. También ignora las relaciones entre ellas. Resulta de utilidad con variables cuantitativas incorreladas y medidas en las mismas unidades. El cuadrado de la distancia euclídea también suele utilizarse como distancia. Para el conjunto de datos TIC2021, la distancia euclídea se obtiene ejecutando el siguiente código:

```
library("CDR")
data("TIC2021")
library("factoextra")

tic <- scale(TIC2021) # estandariza las variables
d_euclidea <- get_dist(x = tic, method = "euclidean")
as.matrix(d_euclidea)[1:5, 1:5]
#>   BE      BG      CZ      DK      DE
#> BE 0.000000 6.421631 2.417212 1.870962 2.304686
#> BG 6.421631 0.000000 4.616177 7.988106 4.871235
#> CZ 2.417212 4.616177 0.000000 3.765714 1.366011
#> DK 1.870962 7.988106 3.765714 0.000000 3.607589
#> DE 2.304686 4.871235 1.366011 3.607589 0.000000
```

12.3. Elección de la distancia entre elementos

205

La Fig. 12.2 muestra un *heatmap*³ de distancias euclídeas entre los países de la UE27 a partir de las estadísticas de uso de las TIC en 2021.

```
fviz_dist(dist.obj = d_euclidea, lab_size = 10)
```

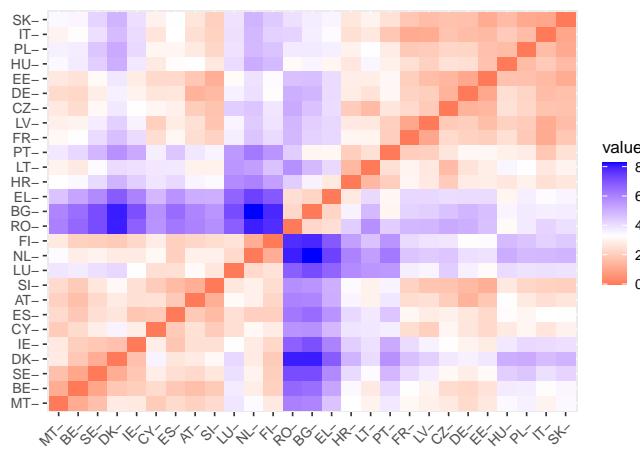


Figura 12.2: *Heatmap* de distancias euclídeas: datos ‘TIC2021’ del paquete ‘CDR’

- **Distancia Manhattan o city block.** Se define como:

$$d_{MAN}(\mathbf{x}_i; \mathbf{x}_j) = \sum_{k=1}^p |x_{ik} - x_{jk}| . \quad (12.2)$$

Viene afectada por los cambios de escala en alguna de las variables y es menos sensible que la distancia euclídea a los valores extremos. Por ello, es recomendable cuando las variables son cuantitativas, con las mismas unidades de medida, sin relaciones entre ellas y con valores extremos.

- **Distancia de Minkowski.** Se define como:

$$d_{MIN}(\mathbf{x}_i; \mathbf{x}_j) = \left(\sum_{k=1}^p |x_{ik} - x_{jk}|^\lambda \right)^{\frac{1}{\lambda}} . \quad (12.3)$$

³Un *heatmap* es una representación visual de fácil lectura e interpretación basada en un código de colores, típica del análisis de páginas web. Normalmente proporciona un patrón visual en forma de “F”, que es el del seguimiento ocular de un sitio o plataforma tecnológica.

Las distancias euclídea y Manhattan son casos particulares de la distancia de Minkowski. En la distancia euclídea $\lambda = 2$ y en la Manhattan $\lambda = 1$.

- **Norma del supremo o distancia de Chebychev.** Su expresión es:

$$d_{CHE}(\mathbf{x}_i; \mathbf{x}_j) = \max_{1 \leq k \leq p} \sum_{k=1}^p |x_{ik} - x_{jk}|. \quad (12.4)$$

Únicamente influye en ella la variable con los valores más extremos y, en este sentido, es muy sensible a los cambios de escala en una de las variables.

- **Distancia de Mahalanobis.** Se define como:

$$d_{MAH} = (\mathbf{x}_i; \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)' \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j) \quad (12.5)$$

Coincide con la distancia euclídea calculada sobre las componentes principales. Es invariante a cambios de origen y de escala (por tanto, la matriz de covarianzas entre las variables agrupadoras, \mathbf{S} , se puede sustituir por su homónima de correlaciones, \mathbf{R}). Además, tiene en cuenta, explícitamente, las correlaciones lineales que puedan existir entre las variables, corrigiendo así el efecto redundancia. Es, por tanto, una distancia apropiada cuando se trabaja con variables cuantitativas con relaciones aproximadamente lineales. Su principal desventaja es que \mathbf{S} involucra, conjuntamente, a todos los elementos, y no únicamente, y de forma separada, a los elementos de cada cluster.

- **Coeficiente de correlación de Pearson.** Se define como:

$$d_P(\mathbf{x}_i; \mathbf{x}_j) = \frac{\sum_{k=1}^p (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)}{\sqrt{\sum_{k=1}^p (x_{ik} - \bar{x}_i)^2 \sum_{k=1}^p (x_{jk} - \bar{x}_j)^2}}. \quad (12.6)$$

No es una distancia sino un indicador de similitud. Por tanto, valores altos indican elementos similares y valores bajos elementos distintos.

Su campo de variación es $[-1, 1]$, por lo que se toma su valor absoluto. Cuando las variables están centradas, se denomina coeficiente de congruencia o distancia coseno, puesto que coincide con el coseno formado por los vectores representativos de cada pareja de elementos. Tiene un inconveniente importante: un valor unitario no significa que los dos elementos sean iguales, puesto que también pueden obtenerse valores unitarios cuando los valores de las p variables en uno de los elementos sean combinación lineal de los valores de las p variables del otro.

Se utiliza, en ocasiones, preferentemente con datos cuantitativos y con el algoritmo de distancia mínima. Los coeficientes de correlación por rangos de Kendall y Spearman se utilizan, también, en casos de variables ordinales.

A efectos prácticos, cambiando el argumento `method` de la función `get_dist` (`euclidean`, `maximum`, `manhattan`, `minkowski`, `pearson`, `spearman`, `kendall`) se obtienen distintas matrices de distancias entre los elementos.

Variables cualitativas (dicotómicas)

En este caso, se pueden establecer distintas medidas de similaridad en base a la siguiente tabla de contingencia 2×2 :

		Elem. j		Total
		Presencia	Ausencia	
Elem. i	Presencia	n_{11}	n_{12}	$n_{1\cdot}$
	Ausencia	n_{21}	n_{22}	$n_{2\cdot}$
	Total	$n_{\cdot 1}$	$n_{\cdot 2}$	p

A partir de la tabla anterior, la similaridad entre dos elementos se puede medir a partir de las coincidencias, ya sea de presencias y ausencias como de solo presencias.

Entre las medidas de similaridad que involucran tanto presencias como ausencias comunes están:

- El **coeficiente de coincidencias simple**: $c_{cs} = \frac{(n_{11} + n_{22})}{2}$
- El **coeficiente de Rogers-Tanimoto**: $c_{RT} = \frac{(n_{11} + n_{22})}{2(n_{11} + n_{22}) + n_{12} + n_{21}}$

Estos dos coeficientes tienen una relación monotónica (si la distancia entre dos elementos es igual o superior a la distancia entre otros dos con una de las medidas, también lo es con la otra). Esto es importante, dado que algunos procedimientos de agrupación no se ven afectados por la medida utilizada siempre y cuando el ordenamiento establecido por ellas sea el mismo.

Entre aquellas que identifican similaridad con presencias destacan:

- El **coeficiente de Jaccard**: $c_J = \frac{n_{11}}{n_{11} + n_{12} + n_{21}}$
- El **coeficiente de Czekanowski**: $c_C = \frac{2n_{11}}{2n_{11} + n_{12} + n_{21}}$
- El **coeficiente de Sokal y Sneath**: $c_{SS} = \frac{n_{11}}{n_{11} + 2(n_{12} + n_{21})}$
- El **coeficiente de Russell y Rao**: $c_{RR} = \frac{n_{11}}{p}$

Los tres primeros coeficientes disfrutan de la relación de monotonía en el sentido anteriormente apuntado, siendo los dos primeros las más utilizados en la práctica.

También se usan como indicadores de similitud las medidas de asociación para tablas 2×2 , sobre todo Q y ϕ (Sec. ref(mejoradas)).

Variables cualitativas (polítómicas)

Cuando todas las variables sean cualitativas y alguna sea polítómica, se generan para estas últimas tantas variables dicotómicas como categorías tienen, denotando con 1 la presencia y con 0 la ausencia.

Variables cuantitativas y cualitativas

Si las variables no son del mismo tipo, se utiliza la medida de similaridad de Gower:

$$S_{ij}(\mathbf{x}_i; \mathbf{x}_j) = \frac{\sum_{k=1}^p s_{ij}}{\sum_{k=1}^p w_{ij}} \quad (12.7)$$

donde w_{ij} vale siempre la unidad, salvo para variables binarias si los dos elementos presentan el valor cero. En cuanto al valor de S_{ij} , se distinguen tres casos:

- Variables cualitativas de más de dos niveles: 1 si ambos elementos son iguales en la k -ésima variable; 0 si son diferentes.
- Variables dicotómicas: 1 si la variable considerada está presente en ambos elementos; 0 en los demás casos.
- Variables cuantitativas: $1 - \frac{|x_{ik} - x_{jk}|}{R_k}$, donde R es el rango de la variable k .

No es recomendable cuando las variables cuantitativas sean muy asimétricas. En este caso, hay dos procedimientos aproximados: (i) calcular medidas separadas para las variables cuantitativas y cualitativas y combinarlas estableciendo algún tipo de ponderación; (ii) pasar las variables cuantitativas a cualitativas y utilizar las medidas propuestas para este tipo de variables.

12.4. Técnicas de agrupación jerárquicas

12.4.1. Introducción

Una vez se han seleccionado las variables en función de las cuales se van a agrupar los elementos disponibles en clusters o conglomerados, así como se ha decidido qué distancia utilizar para tal propósito, el siguiente paso del AC es la selección de un criterio o técnica de agrupamiento para formar los conglomerados. Dichas técnicas se pueden clasificar en (i) jerárquicas y (ii) no jerárquicas.

12.4. Técnicas de agrupación jerárquicas

209

TÉCNICAS DE CLUSTERIZACIÓN:

1. Jerárquicas:

■ *Aglomerativas:*

- Vecino más cercano o encadenamiento simple
- Vecino más lejano o encadenamiento completo
- Método de la distancia media
- Método de la distancia entre centroides
- Método de la mediana
- Método de Ward
- Encadenamiento intra-grupos
- Método flexible de Lance y Williams

■ *Divisivas:*

- Vecino más cercano o encadenamiento simple
- Vecino más lejano o encadenamiento completo
- Método de la distancia media
- Método de la distancia entre centroides
- Método de la mediana
- Método de Ward
- Encadenamiento intra-grupos
- Análisis de la asociación
- Detector automático de interacciones

2. No jerárquicas:

■ *Técnicas de reasignación:*

- Basadas en centroides: Método de Forgy, k -medias
- Basadas en medoides: k -medoides, PAM, CLARA, CLARANS
- Basadas en medianas: k -medianas

■ *Técnicas basadas en la densidad de elementos (mode-seeking):*

- Aproximación tipológica: Análisis modal, métodos TaxMap, de Fortin, de Gitman y Levine, de Catel y Coulter
- Aproximación probabilística: método de Wolf
- DBSCAN

■ *Otras técnicas no jerárquicas*

- Métodos directos: *block-*; *bi*; *co-*; *two – mode clustering*
- Métodos de reducción de la dimensionalidad: modelos Q - y R -factorial
- Clustering difuso
- Métodos basados en mixturas de modelos

Los procedimientos jerárquicos no partitionan el conjunto de elementos de una sola vez, sino que realizan particiones sucesivas a distintos niveles de agrupamiento; es decir, establecen una jerarquía de clusters, de ahí su nombre. Forman los conglomerados, bien agrupando los elementos

en grupos cada vez más grandes, fusionando grupos en cada paso, (jerárquicos aglomerativos), o bien desagregándolos en conglomerados cada vez más pequeños (jerárquicos divisivos).

Las técnicas no jerárquicas se caracterizan porque (i) el número de clusters se suele determinar a priori; (ii) utilizan directamente los datos originales, si necesidad de calcular una matriz de distancias o similaridades; y (iii) los clusters resultantes no están anidados unos en otros, sino que están separados. La caja informativa proporciona un detalle mayor de la tipología de técnicas de agrupación que aborda el presente capítulo⁴. En lo que sigue, el objetivo son las técnicas jerárquicas, abordando las no jerárquicas en el Cap. 13.

12.4.2. Técnicas jerárquicas aglomerativas

Las técnicas jerárquicas aglomerativas, de amplia utilización, parten de tantos conglomerados como elementos y llegan a un único conglomerado final.

Se parte de un conglomerado constituido por los dos elementos más próximos, de tal manera que en la segunda etapa el conglomerado formado actuará a modo de elemento (como si se tuvieran $n - 1$ elementos). En la segunda etapa, de nuevo se agrupan de nuevo los dos elementos más cercanos, que pueden ser dos elementos simples o uno simple y otro compuesto (el conglomerado anterior); en el primer caso, se tendrían dos conglomerados (cada uno de ellos formado por dos elementos) y en el segundo, un conglomerado con tres elementos y otro con uno. Sea cual sea el caso, al final de la segunda etapa se tienen $n - 2$ elementos, dos de los cuales son conglomerados. En las etapas siguientes se procede de idéntica manera: agrupación de los dos elementos (sean elementos simples o conglomerados formados en las etapas anteriores) más cercanos, y así sucesivamente hasta formar un único conglomerado integrado por todos los elementos. Es importante resaltar que un elemento, una vez forma parte de un conglomerado, ya no sale de él.

La pregunta que surge en este momento es: en el proceso de agrupamiento descrito, ¿cómo se mide la distancia de un elemento a un conglomerado, o entre dos conglomerados?⁵ Los métodos más populares son los siguientes:

- **Método del encadenamiento simple o vecino más cercano.**

Utiliza el criterio de “*la distancia mas cercana*”. Por tanto, (i) la distancia entre un elemento y un conglomerado es la menor de las distancias entre dicho elemento y cada uno de los elementos del conglomerado; (ii) la distancia entre dos conglomerados viene dada por la distancia entre sus dos elementos más cercanos. Una vez computada la matriz de distancias se seleccionan los conglomerados más cercanos.

- **Método del encadenamiento completo o vecino más lejano.**

Funciona igual que el anterior, pero ahora el criterio es “*la distancia más lejana*”.

⁴Elaboración propia en base a Kassambara (2017).

⁵El criterio de inclusión de los elementos en dichos conglomerados citado en 12.1.

12.4. Técnicas de agrupación jerárquicas

211

Nótese que, mientras que con el método del vecino más cercano la distancia entre los elementos más próximos de un cluster es siempre menor que la distancia entre elementos de distintos clusters, con el criterio del vecino más lejano la distancia entre los dos elementos más alejados de un cluster es siempre menor que la distancia entre cualquiera de sus elementos y los elementos más alejados de los demás clusters. Nótese también que, mientras que el método del vecino más cercano tiende a separar a los individuos en menor medida que la indicada por sus disimilitudes iniciales (es espacio-contrativo), el criterio del vecino más lejano es espacio-dilatante, es decir, tiende a separar a los individuos en mayor medida que la indicada por sus disimilitudes iniciales (Gallardo-San Salvador and Vera-Vera, 2004).

- **Método de la distancia media.**

Surge como una solución a la constrección o dilatación del espacio que provocan los dos métodos anteriores (por eso se dice que es espacio-conservativo y es muy utilizado), utilizando “*la distancia promedio*”, es decir, la distancia entre un elemento y un conglomerado es la media aritmética de las distancias de dicho elemento a cada uno de los elementos del conglomerado. En caso de dos conglomerados, la distancia entre ellos viene dada por el promedio aritmético de las distancias, dos a dos, tomándose un elemento de cada conglomerado. Igual que los dos métodos precedentes, es invariante a transformaciones monótonas de la distancia utilizada.

En la Fig. 12.3 se puede ver la constrección, dilatación y conservación del espacio que producen los métodos del vecino más cercano, más lejano y de la distancia media, respectivamente. En este caso se utiliza como representación gráfica el dendrograma (diagrama de árbol). En figuras posteriores se utilizarán otras alternativas al dendrograma, con el objetivo de mostrar las más populares.

```
hc_simple <- hcut(tic, k = 3, hc_method = "single")
hc_completo <- hcut(tic, k = 3, hc_method = "complete")
hc_promedio <- hcut(tic, k = 3, hc_method = "average")

library("patchwork")
d1 <- fviz_dend(hc_simple, cex = 0.5, k = 3, main = "Vecino más cercano")
d2 <- fviz_dend(hc_completo, cex = 0.5, k = 3, main = "Vecino más lejano")
d3 <- fviz_dend(hc_promedio, cex = 0.5, k = 3, main = "Distancia promedio")

d1 + d2 + d3
```

- **Método de la distancia entre centroides.**

Según este método, la distancia entre dos grupos o conglomerados es la distancia entre sus centroides, entendiendo por centroide del grupo g : $c_g = (\bar{x}_{1g}, \bar{x}_{2g}, \dots, \bar{x}_{pg})$, donde \bar{x}_{jg} es la media de la j -ésima variable en dicho grupo.

Igual que el método de la media, este método es también espacio-conservativo. Sin embargo, tiene la limitación de que cuando se agrupan dos conglomerados de diferente tamaño, el conglomerado resultante queda más cerca del conglomerado mayor y más alejado del menor, de

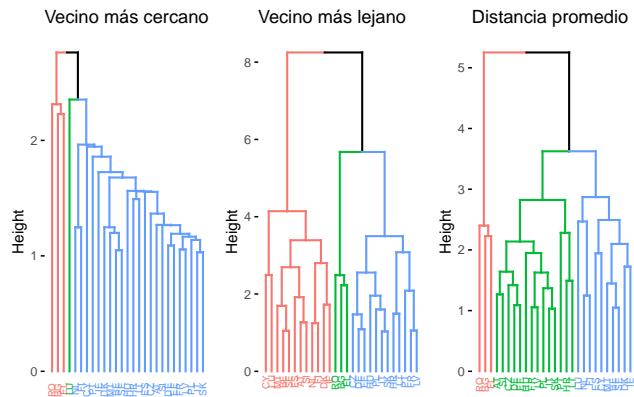


Figura 12.3: Clusterización jerárquica con distancias euclídeas (dendrograma): métodos del vecino más cercano, vecino más lejano y distancia media

forma proporcional a la diferencia de tamaños, lo que lleva a que a lo largo del proceso de clusterización se vayan perdiendo las propiedades de los conglomerados pequeños ([Gallardo San-Salvador, 2022](#)).

- **Método de la mediana.**

Viene a superar la limitación del método del centroide. Para ello, la estrategia natural es suponer que los grupos son de igual tamaño. Dicha estrategia se plasma en suponer que la distancia entre un elemento (o un conglomerado, k) y el conglomerado formado por la agrupación de los conglomerados i y j viene dada por la mediana del triángulo formado por sus centroides (de ahí su nombre). Se trata de un método espacio conservativo, pero, igual que el método del centroide, no es invariante a transformaciones monótonas de la distancia utilizada.

La Fig. 12.4, un tanglegrama o diagrama de laberinto, muestra las agrupaciones producidas por los métodos del centroide y la mediana. En ella se puede observar como el método de la mediana corrige la limitación del método del centroide. `index{método! de la mediana}` `index{método! del centroide}`

```
library("dendextend")
library("cluster")
hc_cent_dend <- as.dendrogram(hclust(d_euclidea, method = "centroid"))
hc_med_dend <- as.dendrogram(hclust(d_euclidea, method = "median"))
tanglegram(hc_cent_dend, hc_med_dend)
```

- **Método de Ward.**

12.4. Técnicas de agrupación jerárquicas

213

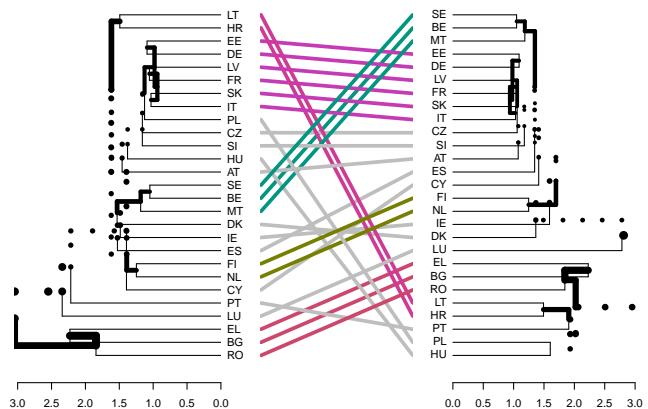


Figura 12.4: Clusterización jerárquica con distancias euclídeas (tanglegrama): método del centroide vs. método de la mediana

El método de Ward agrupa, en cada etapa, los dos clusters que producen el menor incremento de la varianza total intra-cluster: $W = \sum_g \sum_{i \in g} (x_{ig} - \bar{x}_g)'(x_{ig} - \bar{x}_g)$, donde \bar{x}_g es el centroide del grupo g . Así, los grupos formados no distorsionan los datos originales.⁶

Es muy utilizado en la práctica, dado que tiene casi todas las ventajas del método de la media y suele ser más discriminatorio en la determinación de los niveles de agrupación. También suele crear conglomerados muy compactos de tamaño similar. Dado que el menor incremento de W es proporcional a la distancia euclídea al cuadrado entre los centroides de los grupos fusionados, W no es decreciente, solventándose los problemas de los otros métodos basados en centroides.

La Fig. 12.5, muestra el filograma, diagrama filético en forma de árbol filogenético, generado por la librería `igraph` con el método de agrupación de Ward.

```
library("igraph")
set.seed(5665)
hc_ward <- hcut(tic, k = 3, hc_method = "ward.D2")
fviz_dend(
  x = hc_ward,
  k = 3,
  type = "phylogenetic"
)
```

- **Método del encadenamiento intra-grupos.**

Según el método de la distancia promedio (o vinculación entre grupos) la distancia entre dos conglomerados se obtenía calculando las distancias de cada elemento de uno de los grupos con

⁶Específicamente, la propuesta de Ward es que la pérdida de información que se produce al integrar los distintos individuos en clusters sea la mínima posible.

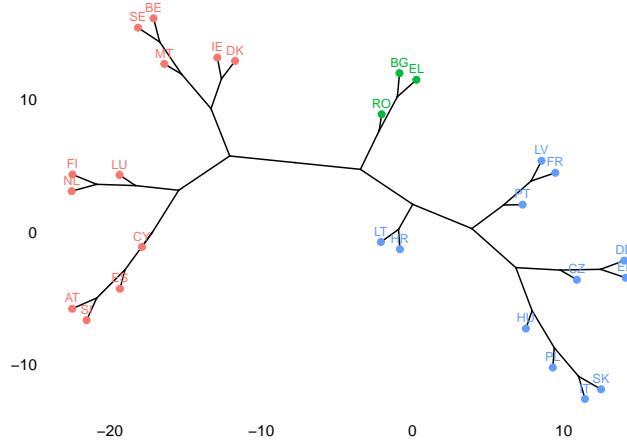


Figura 12.5: Clusterización jerárquica con distancias euclídeas al cuadrado (filograma): método de Ward

todos los del otro y computando, posteriormente, la media aritmética de dichas distancias. Con el método de la vinculación intra-grupos se computa la distancia media entre la totalidad de los elementos de los conglomerados susceptibles de agrupación, con independencia de si pertenecen al mismo conglomerado inicial o a distinto conglomerado. Por ejemplo: si un conglomerado está formado por los elementos a y b , y otro por los elementos c y d , la distancia inter-grupos entre los dos conglomerados es:

$$d_{inter-grupos} = \frac{d_{(a;c)} + d_{(a;d)} + d_{(b;c)} + d_{(b;d)}}{4}$$

mientras que la distancia intra-grupos vendrá dada por la media de las distancias entre los elementos a, b, c y d :

$$d_{intra-grupos} = \frac{d_{(a;b)} + d_{(a;c)} + d_{(a;d)} + d_{(b;c)} + d_{(b;d)} + d_{(-c;d)}}{6}$$

■ Método flexible de Lance y Williams.

Calcula la distancia entre dos conglomerados (el primero formado por la unión de otros dos en la etapa previa) a partir de la siguiente expresión:

$$d_{(g_1 \cup g_2);g_3} = \alpha_1 d_{(g_1;g_3)} + \alpha_2 d_{(g_2;g_3)} + \beta d_{(g_1;g_2)} + \gamma |d_{(g_1;g_2)} - d_{(g_2;g_3)}|,$$

donde $\alpha_1 + \alpha_2 + \beta = 1$; $\alpha_1 = \alpha_2$; $\beta < 1$; $\gamma = 0$, si bien Lance y Williams sugieren adicionalmente un pequeño valor negativo de β . Por ejemplo $\beta = -0,25$.

Los métodos anteriormente expuestos son casos particulares de éste. Denominando n_1 , n_2 y n_3 a los tamaños de los grupos g_1 , g_2 y g_3 , respectivamente, se tiene:

12.4. Técnicas de agrupación jerárquicas

215

Tabla 12.3: Valores de α_1 , α_2 , β y γ para distintos procedimientos de agrupación

Método	α_1	α_2	β	γ
Vecino más cercano	0,5	0,5	0	-0,5
Vecino más lejano	0,5	0,5	0	0,5
Distancia media	$\frac{n_1}{n_1+n_2}$	$\frac{n_2}{n_1+n_2}$	0	0
Distancia entre centroides	$\frac{n_1}{n_1+n_2}$	$\frac{n_2}{n_1+n_2}$	$\frac{-n_1 n_2}{(n_1+n_2)^2}$	0
Método de la mediana	0,5	0,5	-0,25	0
Ward	$\frac{n_1+n_3}{n_1+n_2+n_3}$	$\frac{n_2+n_3}{n_1+n_2+n_3}$	$\frac{-n_3}{n_1+n_2+n_3}$	0
Flexible	$0,5(1-\beta)$	$0,5(1-\beta)$	β	0

12.4.3. Técnicas jerárquicas divisivas

En este caso, la secuencia de acontecimientos es justo la inversa. Se parte de un único conglomerado formado por todos los elementos y se llega a n conglomerados formados, cada uno de ellos, por un único elemento (a veces el proceso termina cuando se llega a un número de grupos preestablecido). Ahora bien, dado que ahora se trata de subdividir conglomerados, es decir, de identificar los elementos más distantes, o menos similares, para separarlos del resto del conglomerado, la estrategia a seguir estará basada en maximizar las distancias (o minimizar las similitudes). En el proceso disociativo surge una cuestión importante: cuándo debe dejar de dividirse un cluster determinado y pasar a dividir otro, cuestión que se resuelve por el procedimiento propuesto por MacNaughton-Smith et al. (1964). Las técnicas divisivas (también llamadas partitivas o disociativas), pueden ser monotéticas o politéticas. En el primer caso, las divisiones se basan en una sola característica o atributo. En el segundo, se tienen en cuenta todas.

Las técnicas divisivas son menos populares que las aglomerativas. Sin embargo, la probabilidad de que lleven a decisiones equivocadas (debido a la variabilidad estadística de los datos) en las etapas iniciales del proceso, lo cual distorsionaría el resultado final del mismo, es menor que en las aglomerativas. En este sentido, los métodos partitivos, al partir del total de elementos, se consideran más seguros que los aglomerativos. Los métodos disociativos más populares son los siguientes:

- **Método de la distancia promedio**

Dentro de las técnicas politéticas, entre las que se cuentan todas las vistas en la clusterización jerárquica aglomerativa, quizás la más popular es la que utiliza para la partición el método de la distancia promedio. Para ilustrarla, supóngase que se tienen 5 elementos y que su matriz de distancias es la siguiente:

$$\mathbf{X} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ 8 & \cdot & \cdot & \cdot & \cdot \\ 7 & 4 & \cdot & \cdot & \cdot \\ 6 & 1 & 4 & \cdot & \cdot \\ 3 & 4 & 5 & 4 & \cdot \end{pmatrix}$$

En la primera etapa hay que dividir el grupo de cinco elementos en dos conglomerados. Hay $2^{2n-1} - 1$ posibilidades, pero según el método de la distancia promedio, se calcula la distancia de cada elemento a los demás y se promedia, desgajándose el elemento con distancia promedio máxima. En este caso, se desgajaría el primer elemento, y en la segunda etapa se partiría de dos grupos: $\{e_1\}$ y $\{e_2, e_3, e_4, e_5\}$.

A partir de la segunda etapa, se procede como sigue (véase Tabla 12.4):

- (i) Se calculan las (4) distancias promedio de cada elemento del conglomerado principal al elemento desgajado;
- (ii) Se calculan las (4) distancias promedio de cada elemento del conglomerado principal al resto de elementos del mismo;
- (iii) Se computan las diferencias (i) – (ii) para cada uno de los 4 elementos del conglomerado principal;
- (iv) De entre aquellos elementos del grupo principal en los que (i) – (ii) < 0 se selecciona aquel para el cual es máxima. Tras esta segunda etapa los conglomerados son $\{e_1, e_5\}$ y $\{e_2, e_3, e_4\}$.

Tabla 12.4: Distancias entre conglomerados: segunda etapa

Elemento	Distancia promedio al grupo desgajado $\{e_1\}$	Distancia promedio al grupo principal	Diferencia
$\{e_2\}$	8	3	5
$\{e_3\}$	7	3	4
$\{e_4\}$	6	3	3
$\{e_5\}$	3	4,33	-1,33

En las siguientes etapas se procede de igual manera, hasta que todas las diferencias sean positivas (en el caso que se considera, esto ocurre en la tercera etapa; véase Tabla 12.5).

Tabla 12.5: Distancias entre conglomerados: tercera etapa

Elemento	Distancia promedio al grupo desgajado $\{e_1, e_5\}$	Distancia promedio al grupo principal	Diferencia
$\{e_2\}$	6	2,5	3,5
$\{e_3\}$	6	4	2

12.4. Técnicas de agrupación jerárquicas

217

Elemento	Distancia promedio al grupo desgajado $\{e_1, e_5\}$	Distancia promedio al grupo principal	Diferencia
$\{e_4\}$	5	2,5	2,5

Cuando esto ocurre, es decir, cuando todos los elementos del conglomerado principal están más cerca de los demás que lo componen que de los del conglomerado disociado, se vuelve a iniciar el algoritmo, pero esta vez para cada uno de los dos conglomerados generados ([MacNaughton-Smith et al., 1964](#)). En caso que nos ocupa, en $\{e_1, e_5\}$ la única partición posible es $\{e_1\}, \{e_5\}$. En $\{e_2, e_3, e_4\}$ se desgaja el elemento con mayor distancia promedio a los demás del grupo. Como $\frac{d_{(2,3)}+d_{(2,4)}}{2} = 2,5$, $\frac{d_{(3,2)}+d_{(3,4)}}{2} = 4$ y $\frac{d_{(4,2)}+d_{(4,3)}}{2} = 2,5$, se desgaja $\{e_3\}$.

A continuación se aplica el algoritmo anteriormente expuesto a cada elemento del grupo principal $\{e_2, e_4\}$ y $\{e_3\}$ (Tabla 12.6) y, como todas las distancias son positivas, se divide $\{e_2, e_4\}$ en $\{e_2\}$ y $\{e_4\}$.

Tabla 12.6: Distancia entre conglomerados: etapa final

Elemento	Distancia promedio al grupo desgajado $\{e_3\}$	Distancia promedio al grupo principal	Diferencia
$\{e_2\}$	4	1	3
$\{e_4\}$	4	1	3

El algoritmo DIvisive ANAlysis (DIANA) permite llevar a cabo la partición anterior utilizando el diámetro de los clusters para decidir el orden de partición clusters cuando se tienen varios con más de un elemento (véase capítulo 6 de [Kaufman and Rousseeuw \(1990\)](#)). Proporciona (i) el coeficiente divisivo (véase `?diana.object`), que mide la cantidad de estructura de agrupamiento encontrada; y (ii) la pancarta, una novedosa presentación gráfica (véase `?plot.diana`).

Para el ejemplo de los datos TIC, DIANA proporciona el coeficiente divisivo (valores cercanos a 1 sugieren una estructura de agrupación fuerte), y el dendrograma, en este caso circular, representado en la Fig. 12.6.

```
hc_diana <- diana(tic, metric = "euclidean")
hc_diana$dc
#> [1] 0.8043393
fviz_dend(
  x = hc_diana,
  k = 3,
  type = "circular",
  ggtheme = theme_minimal()
)
```

- Análisis de la asociación

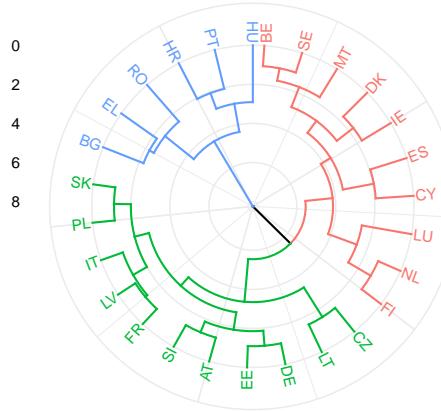


Figura 12.6: Clusterización jerárquica divisiva con DIANA

En caso de que los elementos vengan caracterizados por variables cualitativas o factores dicotómicos, F_1, F_2, \dots, F_n (si alguno fuese politómico, cada una de sus categorías se consideraría como un factor dicotómico), el método del análisis de la asociación (o suma de estadísticos chi-cuadrado) es una técnica monotética muy utilizada que procede como sigue:

- (i) Considérese F_1 y divídase el conjunto de elementos en dos grupos o categorías: uno con los elementos en los que F_1 esté presente y otro con aquellos en los que esté ausente. Hágase lo mismo con los demás factores.
- (ii) Constrúyanse las $n \times (n - 1)$ tablas de contingencia 2×2 que cruzan cada factor con cada uno de los demás (véase Sec. 5.1.1).

		Presencia	Factor j		Total
			SI	NO	
Presencia	SI	n_{11}	n_{21}	$n_{1\cdot}$	$n_{1\cdot}$
	NO	n_{21}	n_{22}	$n_{2\cdot}$	
Total		$n_{\cdot 1}$	$n_{\cdot 2}$	n	

donde $i \neq j$.

- (iii) Calcúlese el estadístico chi-cuadrado ($\chi^2_{ij} = \frac{n(n_{11}n_{22} - n_{12}n_{21})^2}{n_{1\cdot}n_{2\cdot}n_{\cdot 1}n_{\cdot 2}}$) para una de dichas tablas (véase eígrafe 5.2.4) y compútese $\sum_{i \neq j} \chi^2_{ij}$.
- (iii) Desgájese del conglomerado inicial en dos: uno con los elementos que contienen el factor con la máxima $\sum_{i \neq j} \chi^2_{ij}$ y otro con el resto de los elementos (donde dicho factor está ausente).
- (iv) Procédase así iterativamente.

■ **Método del detector automático de interacciones (AID)**

No es propiamente un método de AC, sino de la esfera de los modelos lineales de rango no completo. Sin embargo, se menciona, siquiera mínimamente, porque se utiliza en algunas ocasiones para combinar categorías de los factores utilizados con la finalidad de generar grupos que difieran lo más posible entre sí respecto de los valores de una variable dependiente medida en una escala métrica (con una escala proporcional o de intervalo) o ficticia (dicotómica con valores 0 y 1). Específicamente, el AID realiza divisiones secuenciales dicotómicas de la variable a explicar mediante un ANOVA, dividiendo inicialmente el conjunto de elementos objeto de agrupación en dos grupos según la variable que mejor explica las diferencias en el comportamiento a estudiar (en cada etapa se busca la partición que maximiza la varianza inter-grupos y minimiza la varianza intra-grupos); cada uno de los dos grupos formados se vuelve a subdividir de acuerdo con la variable que mejor explica las diferencias entre ellos; este proceso continúa hasta que el tamaño de los grupos dicotómicos alcanza un mínimo pre-establecido o hasta que las diferencias entre los valores medios de los grupos sean no significativas.

En este algoritmo, el proceso de subdivisión del conjunto de elementos en grupos dicotómicos continúa hasta que se verifica algún criterio de parada.

Las limitaciones más importantes del AID son las siguientes:

- Tiende a seleccionar como más explicativas las variables con mayor número de categorías. Por eso no conviene utilizarlo cuando las variables explicativas difieren mucho en el número de categorías.
- Las particiones resultantes dependen de la variable elegida en primer lugar, condicionando las sucesivas particiones.
- Su naturaleza exclusivamente dicotómica también es una limitación importante. Si se llevasen a cabo particiones con tres o más ramas producirían una mayor reducción de la varianza residual y, además, permitirían una mejor selección de otras variables.

El AID basado en tablas de contingencia y el estadístico chi-cuadrado (CHAID) corrige la mayoría de estas limitaciones. Aunque inicialmente fue diseñado para variables categóricas, posteriormente se incluyó la posibilidad de trabajar con variables categóricas nominales, categóricas ordinales y continuas, permitiendo generar tanto árboles de decisión, para resolver problemas de clasificación, como árboles de regresión. Además, los nodos se pueden dividir en más de dos ramas.

12.5. Calidad de la agrupación y número de clusters

12.5.1. El coeficiente de correlación lineal cofenético

Dado que las técnicas jerárquicas imponen una estructura sobre los datos y pueden producir distorsiones significativas en las relaciones entre los datos originales, una vez realizada la jerarquización de los elementos objeto de clusterización, surge la siguiente pregunta: ¿en qué medida la estructura final obtenida representa las similitudes o diferencias entre dichos objetos? En

otros términos, ¿en qué medida el dendrograma representa la matriz de distancias o similitudes original?

El coeficiente de correlación lineal cofenético da respuesta a dichas preguntas. Se define como el coeficiente de correlación lineal entre los $n(n - 1)$ elementos del triángulo superior de la matriz de distancias o similitudes y sus homónimos en la matriz cofenética, \mathbf{C} , cuyos elementos $\{c_{ij}\}$ son las distancias o similitudes entre los elementos (i, j) tras la aplicación de la técnica de jerarquización. Obviamente, se utilizará la técnica jerárquica que origine el mayor coeficiente.

En el ejemplo TIC, el mayor coeficiente cofenético corresponde al método del promedio o del centroide, si bien el de las otras técnicas de agregación es bastante parecido.

```
# comparación con la distancia euclídea: d_euclidea
cof_simp <- cophenetic(hc_simple)
cof_comp <- cophenetic(hc_completo)
cof_prom <- cophenetic(hc_promedio)
cof_ward <- cophenetic(hc_ward)
cof_dia <- cophenetic(hc_diana)
coef_cofeneticos <- cbind(d_euclidea, cof_simp, cof_comp, cof_prom, cof_ward, cof_dia)

round(cor(coef_cofeneticos)[1, ], 2)
#> d_euclidea   cof_simp   cof_comp   cof_prom   cof_ward   cof_dia
#>      1.00     0.71     0.61     0.77     0.60     0.65
```

12.5.2. Número óptimo de clusters

Acabado el procedimiento de clusterización de los n elementos disponibles, sea por un procedimiento jerárquico aglomerativo o divisivo, hay que tomar una decisión sobre el número de óptimo de clusters, k . Esta decisión es ardua y requiere un delicado equilibrio. Valores grandes de k pueden mejorar la homogeneidad de los clusters; sin embargo, se corre el riesgo de sobreajuste. Lo contrario ocurre con un k pequeño.

Para tomar esta decisión, además del sentido común y el conocimiento que se tenga del fenómeno en estudio, se puede echar mano de distintos procedimientos heurísticos:

- El primero se basa en el **dendrograma** y, en concreto, en la representación de las distintas etapas del algoritmo y las distancias a la que se producen las agrupaciones o particiones de los clusters. Para cada distancia, el dendrograma produce un número determinado de clusters que aumenta (o disminuye) con la misma. Por tanto, el número de clusters dependerá de la distancia a la que se corte el dendrograma (eje de ordenadas del dendrograma, *height*). Dicha distancia debería elegirse de tal forma que los conglomerados estuviesen bien determinados y fuesen interpretables. En las primeras etapas del proceso las distancias no varían mucho, pero en las etapas intermedias y, sobre todo, finales, las distancias aumentan mucho entre dos etapas consecutivas. Por ello, se suele cortar el dendrograma a la distancia a la cual las distancias entre dos etapas consecutivas del proceso empiecen a ser muy grandes, indicador de que los grupos empiezan a ser muy distintos.

12.5. Calidad de la agrupación y número de clusters

221

- Otra posibilidad es utilizar el **gráfico de sedimentación** (Sec. 14.4), que relaciona la variabilidad entre clusters (eje de ordenadas) con el el número de clusters (eje de abscisas). Normalmente, decrece bruscamente al principio, y posteriormente más despacio, hasta llegar a la parte de sedimentación (el codo del gráfico), donde el decrecimiento es muy lento. Pues bien, el número óptimo de conglomerados es el correspondiente al codo o comienzo del área de sedimentación del gráfico.

El algoritmo del gráfico de sedimentación es como sigue:

1. Clusterícese variando el número de grupos, k , por ejemplo, de 1 a 10.
2. Para cada valor de k , compítese la suma de cuadrados intra-grupo (WSS).
3. Trácese la gráfica de WSS vs. k .
4. Determínese el número óptimo de grupos.

Con conjuntos de datos de tamaño pequeño a moderado, este proceso se puede realizar convenientemente con `factoextra::fviz_nbclust()`.

- Otra opción es el *ancho de silueta promedio*. El coeficiente o ancho de silueta compara, por cociente, la distancia media a elementos en el mismo grupo con la distancia media a elementos en otros grupos.

Este método calcula el ancho de silueta promedio (`avg.sil.wid.`) de los elementos objeto de agrupación para diferentes valores de k . Como un valor alto del ancho promedio indica una buena agrupación, el número óptimo de conglomerados es el que lo maximiza. El campo de variación del ancho de silueta es $[-1, 1]$, donde 1 significa que los elementos están muy cerca de su propio cluster y lejos de otros clusters, mientras que -1 indica que están cerca de los clusters vecinos.

- El **criterio del gap (brecha)**, similar al método del codo, tiene como finalidad encontrar la mayor diferencia o distancia entre los diferentes grupos de elementos que se van formando en el proceso de clusterización y que se representan normalmente en un dendrograma. Se computan las distancias de cada uno de los enlaces que forman el dendrograma y se observa cuál es la mayor de ellas. El máximo del gráfico de estas diferencias vs. el número de clusters indica el número óptimo de clusters.

```
p1 <- fviz_nbclust(tic,
  FUN = hcut, method = "wss",
  k.max = 10
) +
  ggtitle("Elbow")
p2 <- fviz_nbclust(tic,
  FUN = hcut, method = "silhouette",
  k.max = 10
) +
  ggtitle("Silhouette")
p3 <- fviz_nbclust(tic,
  FUN = hcut, method = "gap_stat",
  k.max = 10
) +
```

```
ggttitle("Gap")
p1 + p2 + p3
```

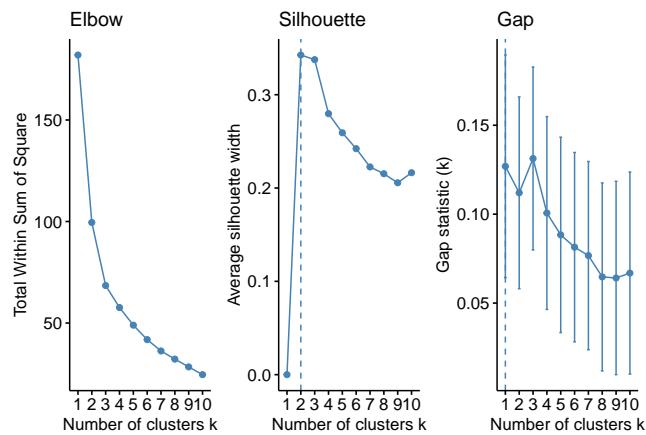


Figura 12.7: Métodos heurísticos para la determinación del número óptimo de clusters

- Finalmente, el **índice de Dunn** es el cociente entre la mínima distancia inter-grupos y la máxima distancia intra-grupos. A mayor índice, mayor calidad de clusterización.

```
library("clValid")
cut2_hc_prom <- cutree(hc_promedio, k = 2)
cut3_hc_prom <- cutree(hc_promedio, k = 3)
cut4_hc_prom <- cutree(hc_promedio, k = 4)
cut5_hc_prom <- cutree(hc_promedio, k = 5)

dunn(d_euclidea, cut2_hc_prom)
#> [1] 0.4465593
dunn(d_euclidea, cut3_hc_prom)
#> [1] 0.3751942
dunn(d_euclidea, cut4_hc_prom)
#> [1] 0.4074884
dunn(d_euclidea, cut5_hc_prom)
#> [1] 0.4366356
```

En el ejemplo TIC, el gráfico de sedimentación y criterio del *gap* indican un número óptimo de clusters de 3. El ancho de silueta alcanza su máximo con dos clusters, si bien la altura del gráfico para tres clusters es prácticamente la misma. Por ello, se opta por 3 clusters a pesar de que el índice de Dunn también se decanta por dos. El primero lo forman Rumanía, Bulgaria y Grecia, la franja sudeste de la UE27, que se caracteriza por tener los peores guarismos en

12.5. *Calidad de la agrupación y número de clusters*

223

dotación y uso de las TIC, tanto a nivel de hogar como de empresa. El segundo lo integran el resto de la franja este más las tres primeras economías de la Unión y Portugal. Tienen unos elevados porcentajes en todas las variables, pero no los mayores, que corresponden a los demás países de la UE27, el tercer conglomerado.

Además de los procedimientos anteriores, hay otros, no tan populares, (*i*) basados en el contraste de hipótesis, suponiendo que los datos siguen alguna distribución multivariante (casi siempre la normal) o (*ii*) procedentes de la abstracción de procedimientos inherentes al análisis multivariante paramétrico; los detalles pueden verse en [Gallardo San-Salvador \(2022\)](#). El paquete **NbClust** de **R** contiene la función **NbClust()**, que calcula 30 índices para valorar el número óptimo de clusters.

RESUMEN El análisis cluster está orientado a la agrupación de un conjunto de elementos en grupos, en función de una serie de características, tal que los elementos de cada grupo sean lo más parecidos posible entre sí y lo más diferentes posible de los de otros grupos. Este proceso implica (*i*) la selección de las variables en función de las cuales se van a agrupar; (*ii*) la elección de la distancia o medida de similitud entre ellos; (*iii*) la elección de la técnica para formar los grupos; y (*iv*) la determinación del número óptimo de clusters, cuando sea menester. Estas son las cuestiones que se estudian en este capítulo, si bien, por cuestiones de espacio, en (*iii*) solo se abordan las técnicas de clusterización jerárquicas, estudiándose las no jerárquicas en el siguiente capítulo.

Capítulo 13

Análisis cluster: clusterización no jerárquica

José-María Montero^a y Gema Fernández-Avilés^a

^aUniversidad de Castilla-La Mancha

Como se avanzó en 12.4.1, aunque las técnicas de agrupación jerárquicas son muy utilizadas, existen otras, también muy populares, que se aglutan bajo la denominación de no jerárquicas y que se pueden clasificar, sin ánimo de exhaustividad, en (i) **de optimización o reasignación**; (ii) **basadas en la densidad de elementos**; y (iii) **otras**, como los *métodos directos* (por ejemplo, el *block-*; *bi-*; *co-*; *two-mode cluster*), los *de reducción de la dimensionalidad* (como el *Q-* y el *R-factorial*), los *métodos de clusterización difusa*, o los *basados en mixturas de modelos*.

Las técnicas no jerárquicas proceden con el criterio de la inercia, maximizando la varianza inter-grupos y minimizando la intra-grupos. Se caracterizan porque:

- El número de clusters se suele determinar a priori.
- Utilizan directamente los datos originales, si necesidad de computo de una matriz de distancias o similaridades.
- Los elementos pueden cambiar de cluster.
- Los clusters resultantes no están anidados unos en otros.

13.1. Métodos de reasignación

Los métodos de reasignación permiten que un elemento asignado a un grupo en una determinada etapa del proceso de clusterización sea reasignado a otro grupo, en una etapa posterior, si dicha reasignación implica la optimización del criterio de selección. El proceso finaliza cuando no hay ningún elemento cuya reasignación permita optimizar el resultado conseguido. Estas técnicas suelen asumir un número determinado de clusters a priori y se diferencian entre sí en la manera

de obtener la partición inicial y en la medida a optimizar en el proceso. Respecto a esta última cuestión, los procedimientos más populares son: (i) la minimización de la traza de la matriz de covarianzas intra-grupos; (ii) la minimización de su determinante; (iii) la maximización de la traza del producto de las matrices de covarianzas inter-grupos e intra-grupos; (iv) medidas de información o de estabilidad.

13.1.1. Técnicas basadas en centroides: métodos de Forgy y k-medias

Los algoritmos de reasignación más populares son el de Forgy y, sobre todo, el *k*-medias. La literatura sobre este tipo de técnicas no es clara y, frecuentemente, se confunden el método de Forgy y el *k*-medias, así como el *k*-medias con algunas de sus otras denominaciones (dándose a entender que son técnicas distintas). Sin embargo, la historia es la siguiente: originalmente, [Forgy \(1965\)](#) propuso un algoritmo consistente en la iteración sucesiva, hasta obtener convergencia, de las dos operaciones siguientes: (i) representación de los grupos por sus centroides; y (ii) asignación de los elementos al grupo con el centroide más cercano. Posteriormente, [Diday \(1971\)](#), [Diday \(1973\)](#), [Anderberg \(1973\)](#), [Bock \(1974\)](#) y [Späth \(1975\)](#) desarrollaron una variante del método de Forgy, que solo se diferencia de él en que los centroides se recalculan después de asignar cada elemento (con la técnica de Forgy primero se llevan a cabo todas las asignaciones y posteriormente se recalculan los centroides). Diday la llamó método de las nubes dinámicas o clusters dinámicos, Anderberg se refirió a ella como el criterio de inclusión en el grupo del centroide más cercano, Bock la denominó particionamiento iterativo basado en la mínima distancia, y Späth la llamó HMEANS, una versión por lotes del procedimiento de los autores anteriores. Sin embargo, fue [MacQueen \(1967\)](#) quien previamente acuñó la denominación de “*k*-medias” que se usa hasta la fecha.

K-medias¹ requiere la especificación previa del número de grupos, *k*, en los que se va a dividir el conjunto de elementos. El algoritmo (i) selecciona *k* elementos por algún procedimiento; (ii) asigna los restantes elementos al elemento más cercano de los previamente seleccionados; (iii) sustituye los elementos seleccionados en (i) por los centroides de los grupos que se han formado; (iv) asigna el conjunto de elementos al centroide más cercano del punto (iii); (v) repite iterativamente los dos últimos pasos hasta que la asignación de elementos a los centroides no cambia. Los grupos entonces formados maximizan la distancia inter-grupos y minimizan la distancia intra-grupos. Recuérdese que con el método de Forgy la etapa (iii) no comienza hasta que no se hayan asignado todos los elementos a un cluster en la etapa (ii), mientras que en “*k*-medias” los centroides se recomputan cada vez que un elemento es asignado a un grupo.

La partición que se obtiene es un óptimo local (pequeños cambios en la reasignación de elementos no lo mejoran), pero no se puede asegurar que sea el global, pues se trata de un método heurístico. Sí se puede asegurar que la partición es de calidad.

K-medias es eficiente y sencillo de implementar, pero tiene algunas desventajas: (i) necesita conocer a priori el número de grupos; (ii) la agrupación resultante puede depender de la asignación inicial (normalmente aleatoria) de los centroides, pudiendo converger a mínimos locales, por lo que se recomienda repetir la clusterización 25-50 veces y seleccionar la que tenga menor varianza intra-grupos; (iii) no es robusto a valores extremos; y (iv) no trabaja con datos nominales.

¹Recuérdese que el centro de un conglomerado viene dado por el centroide, vector de medias.

13.1. Métodos de reasignación

227

En el ejemplo TIC, se ha usado el algoritmo AS 136 de [Hartigan and Wong \(1979\)](#), una versión eficiente del de [Hartigan \(1975\)](#) que no busca óptimos locales (varianza intra-grupos mínima en cada grupo), sino soluciones tales que ninguna reasignación de elementos reduzca la varianza (global) intra-grupos (véase Fig. 13.1).

```
set.seed(123)
kmeans_tic <- eclust(tic, "kmeans", k = 3)
```

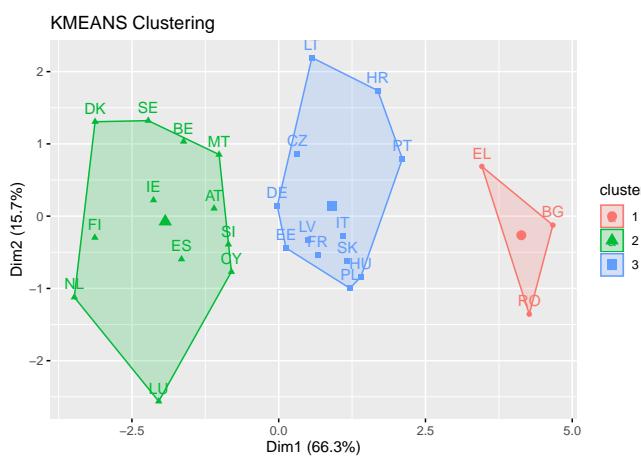


Figura 13.1: Clusterización no jerárquica con k -medias

Algunas versiones del k -medias como el k -medias difuso, el k -medias recortadas, el k -medias armónicas, el k -medias sparse y el k -medias sparse robusto pueden verse en [Carrasco-Oberto \(2020\)](#).

13.1.2. Técnicas basadas en medoides

13.1.2.1. K-medoides (PAM)

Es un método de clusterización similar al k -medias que también requiere la especificación a priori del número de grupos. La diferencia es que, en k -medoides, cada grupo está representado por uno de sus elementos, denominados medoides (o centrotipos)². En el k -medias están representados por sus centroides, que no tienen por qué coincidir con ninguno de los elementos a agrupar. Se trata pues, de formar grupos *particionando el conjunto de elementos alrededor de los medoides* (PAM).

El algoritmo k -medoides es más robusto al ruido y a valores grandes de los datos (de hecho es invariante a los outliers) que el k -medias, ya que minimiza la suma de diferencias por parejas (utiliza la distancia Manhattan) en lugar de la suma de los cuadrados de las distancias

²El medoide de un conglomerado es el elemento del conglomerado con la menor disimilitud promedio entre él y todos los demás miembros del grupo. Es el elemento más céntrico del grupo.

euclídeas³. Además, sus agrupaciones no dependen del orden en que han sido introducidos los elementos, cosa que puede ocurrir con otras técnicas no-jerárquicas, y, como se avanzó anteriormente, propone como centro del cluster un elemento del mismo.

PAM funciona muy bien con conjuntos de datos pequeños (por ejemplo 100 elementos en 5 grupos) y permite un análisis detallado de la partición realizada, puesto que proporciona las características del agrupamiento y un gráfico de silueta, así como un índice de validez propio para determinar el número óptimo de clusters. El algoritmo PAM puede verse al completo en [Kaufman and Rousseeuw \(1990\)](#); para un muy buen resumen véase [Amat Rodrigo \(2017\)](#).

```
set.seed(123)
pam_tic <- eclust(tic, "pam", k = 3)
```

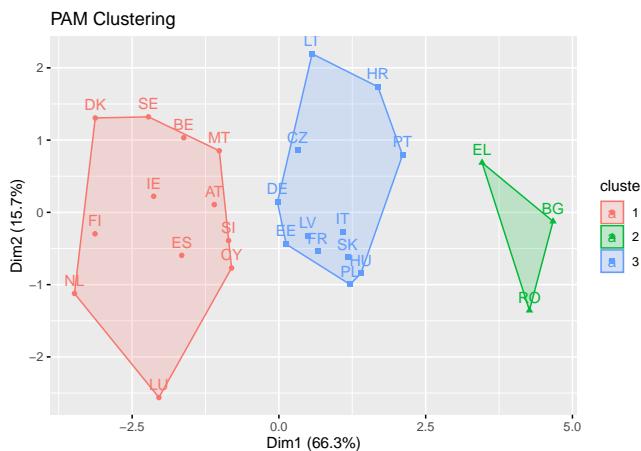


Figura 13.2: Clusterización no jerárquica con PAM

13.1.2.2. CLARA

La ineeficiencia de PAM para bases de datos grandes, junto con su complejidad computacional, llevó al desarrollo de CLARA (clustering Large Applications)⁴

La diferencia entre PAM y CLARA es que el segundo se basa en muestrazos. Solo una pequeña porción de los datos totales es seleccionada como representativa de los datos y los medoides son escogidos (en la muestra) usando PAM. CLARA, pues, combina la idea de k -medoides con el remuestreo para que pueda aplicarse a grandes volúmenes de datos. De acuerdo con [Amat Rodrigo](#)

³Las técnicas basadas en la minimización de promedios de distancias o de residuos en valor absoluto son más robustas que las basadas en sumas de cuadrados.

⁴No obstante, hay una interesante modificación de PAM cambiando el orden de anidamiento de los bucles. La idea es encontrar el mejor intercambio de elementos para cada medoide y ejecutar tantos como sea posible en cada iteración, lo que reduce el número de iteraciones necesarias para la convergencia sin pérdida de calidad ([Schubert and Rousseeuw \(2021\)](#)). También se puede aplicar a los algoritmos CLARA y CLARANS que se verán a continuación.

13.1. Métodos de reasignación

229

(2017) (una descripción completa puede verse en [Kaufman and Rousseeuw \(1990\)](#)), CLARA selecciona una muestra aleatoria y le aplica el algoritmo de PAM para encontrar los clusters óptimos dada esa muestra. Alrededor de esos medoides se agrupan los elementos de todo el conjunto de datos. La calidad de los medoides resultantes se cuantifica con la suma total de distancias intra-grupos. CLARA repite este proceso un número predeterminado de veces con el objetivo de reducir el sesgo de muestreo. Por último, se seleccionan como clusters finales los obtenidos con los medoides que minimizaron la suma total de distancias intra-grupo.

```
set.seed(123)
clara_tic <- eclust(tic, "clara", k = 3)
```



Figura 13.3: Clusterización no jerárquica con CLARA

13.1.2.3. CLARANS

CLARANS (Clustering Large Applications based upon Randomized Search) es una mezcla de PAM y CLARA. Como CLARA puede dar lugar a una mala clusterización si uno de los medoides de la muestra está lejos de los mejores medoides, CLARANS trata de superar esta limitación. El algoritmo puede verse en [Ng and Han \(2002\)](#).

13.1.3. Técnicas basadas en medianas: k-medianas

Igual que el k -medoides, es una variante del k -medias que utiliza como centros las medianas, para que no le afecten ni el ruido ni los valores atípicos. La diferencia con el k -medoides es que la mediana de un grupo no tiene por qué ser una de las observaciones. K -medianas utiliza la distancia Manhattan.

Volviendo al ejemplo TIC, como se ha podido comprobar, la clusterización de los países de la UE27 en función del uso de las TIC es prácticamente la misma con técnicas jerárquicas

aglomerativas como el vecino más lejano, el método de Ward, el del centroide, o algoritmos divisivos como DIANA, que con las técnicas no jerárquicas con pre-selección de 3 grupos (k -medias, PAM y CLARA).

13.2. Métodos basados en la densidad de elementos

Utilizan indicadores de frecuencia, construyendo grupos mediante la detección de aquellas zonas del espacio de las variables (que caracterizan a los elementos) densamente pobladas (clusters naturales) y de aquellas otras con un escasa densidad de elementos. Los elementos que no forman parte de un conglomerado se consideran ruido. Emulan, pues, el funcionamiento del cerebro humano.

La identificación de los grupos (y los parámetros que los caracterizan, cuando se manejan modelos probabilísticos) se lleva a cabo haciéndolos crecer hasta que la densidad del grupo más próximo sobrepase un cierto umbral. Por tanto, imponen reglas para evitar el problema de obtener un solo grupo cuando existen puntos intermedios. Se suele suponer que la densidad de elementos en los grupos es Gaussiana si las variables son cuantitativas, y Multinomial si son cualitativas.

Se suelen clasificar en:

(i) Las que tienen un **enfoque tipológico**: los grupos se construyen buscando las zonas con mayor concentración de elementos. Pertenece a este tipo el *análisis modal de Wishart*, que supone clusters esféricos y dada la complejidad de su algoritmo no tuvo mucho éxito, el *método TaxMap*, que introduce un valor de corte en caso de que los grupos no estén claramente aislados (ello lleva a que los resultados tengan un cierto grado de subjetividad), y el *método de Fortin*, también con muy escasa difusión en la literatura.

(ii) Las que tienen un **enfoque probabilístico**: las variables que caracterizan los elementos siguen una distribución de probabilidad cuyos parámetros cambian de un grupo a otro. Se trata, pues, de agrupar los elementos que pertenecen a la misma distribución. Un ejemplo es el *método de las combinaciones de Wolf*.

No obstante, estos algoritmos, y otros como, por ejemplo, los de Gitman y Levine, y Catel y Coulter, aunque muy citados en la literatura en español, tuvieron poco éxito.

Mayor éxito han tenido otros algoritmos como *expectation-maximization* (EM), *model based clustering* (MCLUST), *density-based spatial clustering of applications with noise* (DBSCAN), *ordering point to identify clustering structure clustering* (OPTICS), que es una generalización de DBSCAN, *wavelet-based cluster* (WAVECLUSTER) y *density-based clustering* (DENCLUE), entre otros.

DBSCAN⁵ es, quizás, el más popular. Incluso ha recibido premios por sus numerosísimas aplicaciones a lo largo del tiempo. Soluciona los problemas de los métodos de reasignación, que son

⁵La densidad se refiere al número de elementos en una misma zona. Sin embargo, es un concepto subjetivo, porque (i) ¿cuántos puntos son necesarios para considerar a una zona como densa? y (ii) ¿cómo de distantes pueden estar dichos elementos entre sí? Por ello, ambos (número de puntos y distancia) son los dos hiperparámetros del modelo.

buenos para clusters con forma esférica o convexa que no tengan demasiados *outliers* o ruido, pero que fallan cuando los clusters tienen formas arbitrarias. De acuerdo con [Amat Rodrigo \(2017\)](#), DBSCAN evita este problema siguiendo la idea de que, (i) para que una observación forme parte de un cluster, tiene que haber un mínimo de observaciones vecinas dentro de un radio de proximidad y (ii) que los clusters están separados por regiones vacías o con pocas observaciones. Consecuentemente, DBSCAN necesita dos parámetros: el radio (ϵ) que define la región vecina a una observación (ϵ -neighborhood); y el número mínimo de puntos (minPts) u observaciones en ella.⁶

Los elementos objeto de agrupación se pueden clasificar, en función de su ϵ -neighborhood y minPts, como: (i) *elementos centrales*, si el número de elementos en su ϵ -neighborhood es igual o mayor que minPts; (ii) *elementos frontera*, si no son elementos centrales pero pertenecen al ϵ -neighborhood de otro elemento que sí es central; y (iii) *elementos atípicos o de ruido*, si no verifican ni (i) ni (ii).

A partir de la clasificación anterior, y para ϵ -neighborhood y minPts dados, se origina otra: (i) un elemento Q es *denso-alcanzable directamente* desde el elemento P si Q está en el ϵ -neighborhood de P y P es un elemento central; Q es *denso-alcanzable* desde P si existe una cadena de objetos $\{Q_1 = P, Q_2, Q_3, \dots, Q_n\}$ tal que Q_{i+1} es denso-alcanzable directamente desde Q_i , $\forall 1 \leq i \leq n$; (iii) Q está *denso-conectado* con P si hay un elemento R desde el cual P y Q son denso-alcanzables.

Los pasos del algoritmo DBSCAN son los siguientes ([Amat Rodrigo \(2017\)](#)):

- Para cada elemento x_i calcúlese su distancia con el resto de observaciones. Márquese como central si lo es y como visitado si no lo es.
- Para cada observación marcada como elemento central, si aún no ha sido asignada a ningún grupo, créese un grupo nuevo y asígnesele a él. Búsquese, recursivamente, todas las observaciones denso-conectadas con ella y asígnense al mismo grupo.
- Itérese el mismo proceso para todas las observaciones no visitadas.
- Aquellas observaciones que tras haber sido visitadas no pertenecen a ningún cluster se marcan como *outliers*.

Como resultado del algoritmo DBSCAN se generan clusters que verifican: (i) todos los elementos que forman parte de un mismo cluster están denso conectados entre ellos; y (ii) si un elemento P es denso-alcanzable desde cualquier otro de un cluster, entonces P también pertenece al cluster.

El éxito de DBSCAN se debe a sus importantes ventajas. De nuevo siguiendo a [Amat Rodrigo \(2017\)](#), no requiere la especificación previa del número de clusters; no requiere esfericidad (ni ninguna forma determinada) en los grupos; y puede identificar valores atípicos, por lo que la clusterización resultante no vendrá influenciada por ellos. También tiene algunas desventajas, como que no es un método totalmente determinista puesto que (i) los puntos frontera que son denso-alcanzables desde más de un cluster pueden asignarse a uno u otro dependiendo del orden

⁶Véase [Amat Rodrigo \(2017\)](#) para una discusión sobre el valor de ambos hiperparámetros.

en el que se procesen los datos; y (ii) no genera buenos resultados cuando la densidad de los grupos es muy distinta, ya que no es posible encontrar un ϵ -neighborhood y un minPts válidos para todos a la vez.

Dado el escaso número de datos de la base de datos TIC2021 del paquete CDR no se puede utilizar DBSCAN. Sin embargo, para ilustrar su utilización, la Fig. 13.4 muestra la agrupación en 5 clusters de la base de datos `multishapes` de la librería `factoextra` mediante DBSCAN (función `dbSCAN`) y k -medias. Se trata de una base de datos que contiene observaciones pertenecientes a 5 grupos distintos y con cierto ruido (*outliers*); en consecuencia, los grupos no deberían ser esféricos y DBSCAN sería un algoritmo de agrupación adecuado.

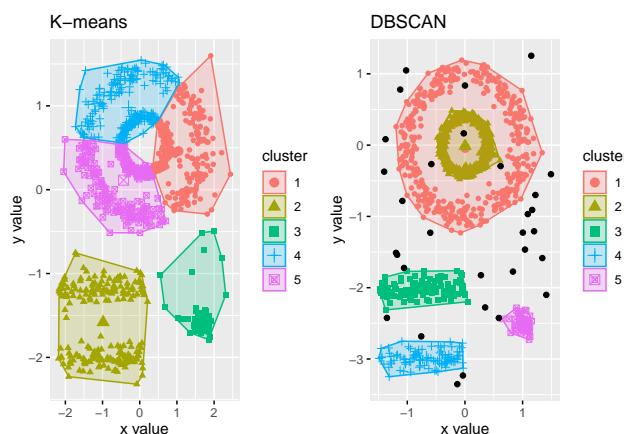


Figura 13.4: Comparación entre los algoritmos k -means y DBSCAN para el conjunto de datos simulado ‘multishapes’

13.3. Otros métodos

Por último, en el cajón de sastre de otras técnicas de clusterización no jerárquicas, merece la pena siquiera mencionar los métodos directos, los de reducción de la dimensionalidad, los de clustering difuso y la clusterización basada en modelos.

Los **métodos directos** agrupan simultáneamente los elementos y las variables. El más conocido es el cluster por bloques (*block-*; *bi-*; *co-*; o *two mode clustering*). El paquete `bicluster` proporciona varios algoritmos para encontrar clusters en dos dimensiones. Además, es muy recomendable para el pre-procesamiento de los datos y para la visualización y validación de los resultados.

Las **técnicas de reducción de la dimensionalidad** buscan factores en el espacio de los elementos (modelo *Q*-factorial) o de las variables (modelo *R*-factorial) haciendo corresponder un cluster a cada factor. Centrándonos en el modelo *Q*-factorial, el método parte de la matriz de correlaciones entre los elementos y rota ortogonalmente los factores encontrados. Dado que

los elementos pueden pertenecer a varios clusters y, por tanto, los clusters pueden solaparse, su interpretación se hace muy difícil.

Las técnicas de clustering difuso precisamente permiten la pertenencia de un elemento a varios clusters, estableciendo un grado de pertenencia a cada uno de ellos. El algoritmo de clustering difuso más popular es fuzzy c -medias, muy similar al k -medias pero que calcula los centroides como una media ponderada (la ponderación es la probabilidad de pertenencia) y, lógicamente, proporciona la probabilidad de pertenencia a cada grupo.

La **clusterización basada en modelos** tiene un enfoque estadístico y consiste en la utilización de una mixtura finita de modelos estocásticos para la construcción de los grupos. - Un vector aleatorio \mathbf{X} procede de una mixtura finita de distribuciones paramétricas si, $\forall \mathbf{x} \subset \mathbf{X}$ su función de densidad conjunta se puede escribir como $f(\mathbf{x}|\psi) = \sum_{g=1}^G \pi_g f_g(\mathbf{x}|\theta_g)$, donde π_g son las proporciones asignadas a cada grupo en la mixtura, tal que $\sum_{g=1}^G \pi_g = 1$; $f_g(\mathbf{x}|\theta_g)$ es la función de densidad correspondiente al g -ésimo grupo y $\psi = (\pi_1, \pi_2, \dots, \pi_G, \theta_1, \theta_2, \dots, \theta_G)$. Las funciones de densidad $f_g(\mathbf{x}|\theta_g)$ suelen ser idénticas para todos los grupos.

En términos menos formales, el clustering basado en modelos considera que los datos observados (multivariantes) han sido generados a partir de una combinación finita de modelos componentes (distribuciones de probabilidad, normalmente paramétricas). A modo de ejemplo, en un modelo resultante de una mixtura de normales multivariantes (el caso habitual), cada componente (cluster) es una normal multivariante y el componente responsable de la generación de una observación específica determina el grupo al que pertenece dicha observación. Para la estimación de la media y matriz de covarianzas, se suele recurrir al algoritmo *expectation-maximization*, una extensión del k -medias⁷. El paquete `mclust` utiliza la estimación máximo verosímil para estimar dichos modelos con distintos número de clusters, utilizando el *Bayesian information criterion* (BIC) para la selección del mejor.

Sus limitaciones fundamentales son (*i*) considerar que las características de los elementos son independientes y (*ii*) que no es recomendable para grandes bases de datos o distribuciones de probabilidad que impliquen un elevado coste computacional.

Una revisión de la evolución de la clusterización basada en modelos desde sus orígenes en 1965 puede verse en [McNicholas \(2016\)](#). Para una idea intuitiva, véase [Amat Rodrigo \(2017\)](#).

13.4. Nota final

La elección de que técnica de clusterización, jerárquica o no, es una decisión del investigador, y dependerá de cómo quiera realizar la agrupación, la métrica de las variables y la distancia o medida de similaridad elegida. No obstante, ambos tipos de técnicas tienen sus ventajas y desventajas, y deberán ser tenidas en cuenta a la hora de decidir. Las jerárquicas adolecen de cierta inestabilidad, lo que plantea dudas sobre la fiabilidad de sus resultados. Además, a veces es difícil decidir cuántos grupos deben seleccionarse. Suelen recomendarse en caso de conjuntos de datos pequeños. En caso de grandes conjuntos de datos, la literatura suele recomendar las no jerárquicas; además, tienen una gran fiabilidad, ya que al permitir la reasignación de los elementos, una incorrecta asignación puede ser corregida posteriormente.

⁷Esta es la razón para incluir este tipo de clusterización entre las técnicas no jerárquicas.

Resumen

En este capítulo se pasa revista a las principales técnicas y algoritmos de agrupación no jerárquicas. Primeramente, se abordan los principales métodos de reasignación, y en particular los basados en centroides (método de Forgy y k -medias), medoides (k -medoides, PAM, CLARA, CLARANS) y medianas (k -medianas). Posteriormente, se exponen las técnicas basadas en la densidad de puntos desde las perspectivas tipológica (análisis modal, métodos TaxMap, de Fortin, de Gitman y Levine, y de Catel y Coulter) y probabilística (método de Wolf), así como se estudia el DBSCAN. Finalmente, se muestran otras técnicas de agrupación no jerárquicas como los métodos directos (*block-; bi; co-; two mode-\$ clustering*), los de reducción de la dimensionalidad (modelos Q - y R -factorial), el clustering difuso y los métodos basados en mixturas de modelos.

Capítulo 14

Análisis de componentes principales

José-María Montero^a y José Luis Alfaro Navarro^a

^aUniversidad de Castilla-La Mancha

14.1. Introducción

En el estudio de cualquier problema de interés, lo ideal es tomar información del mayor número de variables posible, lo cual, actualmente, no es un impedimento. Sin embargo, trabajar con muchas variables es incómodo (por ejemplo, si fueran 30 y se estuviese interesado en su correlación dos a dos, habría que calcular 435 coeficientes). Además, tener muchas variables no implica necesariamente tener mucha información. Si están correlacionadas entre ellas (que suele ser el caso en la realidad), parte de la información que proporcionan es redundante. Por consiguiente, el reto es reducir la dimensionalidad del problema sin reducir la cantidad de información proporcionada por las variables originales, midiéndose dicha cantidad de información a través de su variabilidad, en consonancia con el concepto de entropía. En concreto, se adopta como medida de la variabilidad de las variables originales la suma de sus varianzas.

Pues bien, el análisis de componentes principales (ACP, perteneciente al ámbito del aprendizaje no supervisado) es una técnica de reducción de la dimensionalidad, un problema importante en ciencia de datos, tanto en el aprendizaje supervisado como no supervisado. ACP opera sustituyendo las variables originales por un número reducido de combinaciones lineales de ellas, incorreladas, denominadas **componentes principales** (c.p.), que capturan un elevado porcentaje de la variabilidad de las variables originales (Hothorn and Everitt, 2014; Boehmke, 2020). ACP es el primer intento de reducción de la dimensionalidad y el único utilizado a tal fin hasta el advenimiento del escalamiento multidimensional (aunque no es su función principal) y otras técnicas más complicadas pertenecientes al ámbito del aprendizaje múltiple (*manifold learning*).

La reducción de la dimensionalidad no solo es útil en el estudio de fenómenos complejos con un elevado número de dimensiones, sino también para facilitar la implementación de otros métodos de aprendizaje no supervisado, como el análisis cluster¹ (reduciendo el número de dimensiones a utilizar para configurar los clusters), o supervisado, como, por ejemplo, la regresión (reduciendo el número de regresores y haciéndolos incorrelados, evitando así información redundante y la multicolinealidad); o la técnica de *partial least squares* (PLS, similar a la regresión con c.p. pero que, en vez de ignorar la variable respuesta en la determinación las combinaciones lineales, busca aquellas que, además de explicar la varianza de las variables originales, predicen la variable respuesta lo mejor posible).² También es muy útil para representar gráficamente relaciones multivariantes.

En **R** hay varias opciones para la realización de un ACP: `princomp()`, `prcomp()` y `PCA()`, de la librería `FactoMineR` (Lê et al., 2008), entre otras. Se ha optado por la última porque (i) incorpora notables mejoras gráficas; (ii) permite el ACP con *missing values*, imputando dichos valores (paquete `missMDA`); (iii) proporciona una descripción e interpretación automática de los resultados, seleccionando los mejores gráficos, mediante el paquete `FactoInvestigate`; (iv) permite la implementación de técnicas híbridas (por ejemplo, clusterización con c.p.); y (vi) posibilita la predicción de las coordenadas de individuos y variables adicionales utilizando únicamente inputs del ACP previo.

Como ilustración práctica del ACP, se abordará la reducción de la dimensionalidad de un problema del ámbito de la sociedad de la información en la UE-27, en 2021. Se dispone, para 2021 y a nivel de país, de información sobre 7 variables: 4 relacionadas con el uso de las TIC por parte de las empresas y 3 relativas al uso de dichas tecnologías por parte de las personas y a la equipación TIC de los hogares. Dicha información, así como la descripción de las variables, puede consultarse en la base de datos `TIC2021` del paquete `CDR`.

14.2. Obtención de las componentes principales

14.2.1. Descripción formal del proceso

Sea $\mathbf{X}' = (X_1, \dots, X_p)$ un vector p -dimensional de variables aleatorias con vector de medias $\boldsymbol{\mu}$ y matriz de covarianzas conocida $\boldsymbol{\Sigma}$. Puesto que los cambios de origen no afectan a la covarianza, las variables originales se consideran centradas, de tal manera que $\boldsymbol{\mu} = \mathbf{0}$ y $\boldsymbol{\Sigma} = E(\mathbf{X}'\mathbf{X})$. Se trata de encontrar un conjunto de p combinaciones lineales incorreladas de dichas variables, $Y_j = a_{1j}X_1 + a_{2j}X_2 + \dots + a_{pj}X_p = \mathbf{a}'_j\mathbf{X}$, $j = 1, 2, \dots, p$, denominadas c.p., que recojan la variabilidad existente en los datos. La idea es ordenar las c.p. tal que $V(Y_1) > V(Y_2) > \dots > V(Y_p)$, y seleccionar m de ellas (las m primeras), $m < p$, que capturen un elevado porcentaje de la variabilidad de los datos.

¹El término “customer churn” se suele traducir como perdida de clientes o rotación de clientes. Se compone de las palabras inglesas “change” (en castellano cambio) y “turn” (en castellano abandonar)

²Como señala Amat Rodrigo (2017), PLS puede considerarse como una versión supervisada de la regresión con c.p.

Nota

Geométricamente, las c.p. representan un nuevo sistema de coordenadas obtenido mediante la rotación de los ejes originales. Los nuevos ejes representan las direcciones de máxima variabilidad y proporcionan una descripción más simple de la estructura de covarianza.

La varianza de cada componente y la covarianza entre ellas vienen dadas por:

$$\begin{aligned} \text{Var}(Y_j) &= \mathbf{a}'_j \boldsymbol{\Sigma} \mathbf{a}_j, \quad \forall j = 1, 2, \dots, p, \\ \text{Cov}(Y_j, Y_k) &= \mathbf{a}'_j \boldsymbol{\Sigma} \mathbf{a}_k, \quad \forall j, k \{j \neq k\} = 1, 2, \dots, p. \end{aligned} \quad (14.1)$$

Obtención de la primera componente principal

La primera c.p., Y_1 , se obtiene seleccionando el vector \mathbf{a}_1 que maximice su varianza. Sin embargo, dado que la varianza de cada c.p. puede incrementarse arbitrariamente multiplicando \mathbf{a}_1 por una constante, se impone la condición $\mathbf{a}'_1 \mathbf{a}_1 = 1$; es decir, se normalizan los vectores, de tal forma que tengan longitud unitaria. Por tanto, se trata de encontrar el vector \mathbf{a}_1 que maximiza $\text{Var}(Y_1) = \mathbf{a}'_1 \boldsymbol{\Sigma} \mathbf{a}_1$ sujeto a que $\mathbf{a}'_1 \mathbf{a}_1 = 1$. En otros términos, se selecciona el vector \mathbf{a}_1 que maximiza el lagrangiano:

$$\mathcal{L}(\mathbf{a}_1) = \mathbf{a}'_1 \boldsymbol{\Sigma} \mathbf{a}_1 - \lambda(\mathbf{a}'_1 \mathbf{a}_1 - 1). \quad (14.2)$$

Para ello, se deriva respecto a \mathbf{a}_1 y λ , y se igualan a cero dichas derivadas:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{a}_1)}{\partial \mathbf{a}_1} &= 2\boldsymbol{\Sigma} \mathbf{a}_1 - 2\lambda \mathbf{a}_1 = (\boldsymbol{\Sigma} - \lambda \mathbf{I}) \mathbf{a}_1 = \mathbf{0}, \\ \frac{\partial \mathcal{L}(\mathbf{a}_1)}{\partial \lambda} &= \mathbf{a}'_1 \mathbf{a}_1 - 1 = 0. \end{aligned} \quad (14.3)$$

La primera ecuación tendrá solución distinta del vector nulo cuando $(\boldsymbol{\Sigma} - \lambda \mathbf{I})$ sea singular. Es decir, cuando $|\boldsymbol{\Sigma} - \lambda \mathbf{I}| = 0$, o en otros términos, cuando λ sea un autovalor de $\boldsymbol{\Sigma}$. Dado que,

- $\boldsymbol{\Sigma}$ es semidefinida positiva y, en general, tendrá p autovalores no negativos,
- y que en el proceso de optimización, premultiplicando $(\boldsymbol{\Sigma} - \lambda \mathbf{I}) \mathbf{a}_1 = \mathbf{0}$ por \mathbf{a}'_1 y teniendo en cuenta que $\mathbf{a}'_1 \mathbf{a}_1 = 1$, resulta que $\lambda = \mathbf{a}'_1 \boldsymbol{\Sigma} \mathbf{a}_1 = V(Y_1)$,³

se seleccionará el mayor de los autovalores de $\boldsymbol{\Sigma}$, obteniéndose el autovector \mathbf{a}_1 de tal forma que cumpla la condición $\mathbf{a}'_1 \mathbf{a}_1 = 1$.

Obtención de la segunda componente principal

$Y_2 = \mathbf{a}'_2 \mathbf{X}$ se obtiene igual que Y_1 , pero añadiendo la condición de incorrelación con Y_1 : $\text{Cov}(Y_1, Y_2) = \mathbf{a}'_2 \boldsymbol{\Sigma} \mathbf{a}_1 = 0$, o equivalentemente, $\mathbf{a}'_2 \mathbf{a}_1 = 0$ (\mathbf{a}_1 y \mathbf{a}_2 ortogonales).⁴

Por tanto, el lagrangiano a maximizar es:

³La condición $\mathbf{a}'_i \mathbf{a}_i = 1$ lleva a que los autovalores de $\boldsymbol{\Sigma}$ coincidan con las varianzas de las c.p.

⁴Si todos los autovalores de $\boldsymbol{\Sigma}$ son distintos, los autovectores son ortogonales. En caso contrario, los autovectores asociados a autovalores comunes se eligen de forma que sean ortogonales.

$$\mathcal{L}(\mathbf{a}_2) = \mathbf{a}'_2 \Sigma \mathbf{a}_2 - \lambda(\mathbf{a}'_2 \mathbf{a}_2 - 1) - \gamma(\mathbf{a}'_2 \mathbf{a}_1 - 0). \quad (14.4)$$

Derivando respecto a \mathbf{a}_2 e igualando a cero:

$$\frac{\partial \mathcal{L}(\mathbf{a}_2)}{\partial \mathbf{a}_2} = 2\Sigma \mathbf{a}_2 - 2\lambda \mathbf{a}_2 - \gamma \mathbf{a}_1 = 2(\Sigma - \lambda \mathbf{I})\mathbf{a}_2 - \gamma \mathbf{a}_1 = \mathbf{0}. \quad (14.5)$$

Premultiplicando por \mathbf{a}'_1 y considerando la condición de ortogonalidad, se tiene que $\gamma = 2Cov(Y_1, Y_2) = 0$, con lo que $\frac{\partial \mathcal{L}(\mathbf{a}_2)}{\partial \mathbf{a}_2} = 2\Sigma \mathbf{a}_2 - 2\lambda \mathbf{a}_2 = 0$, que implica que $(\Sigma - \lambda \mathbf{I})\mathbf{a}_2 = 0$.

Siguiendo el mismo razonamiento que en la obtención de la primera componente, se elige el segundo mayor autovalor de Σ ⁵, λ_2 , siendo \mathbf{a}_2 el autovector asociado a él.

Obtención del resto de las componentes principales

Repetiendo este procedimiento, se obtienen las p c.p., siendo los coeficientes de la j -ésima los componentes del autovector asociado al j -ésimo mayor autovalor de Σ .

El vector de c.p. se puede expresar como $\mathbf{Y} = \mathbf{A}'\mathbf{X}$, donde $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_p]$ es la matriz de autovectores (ortogonales) obtenidos.

14.2.2. Cuestiones importantes en el análisis de componentes principales

14.2.2.1. Varianza de las variables originales y las componentes principales

La matriz de covarianzas de las c.p., $\mathbf{V}(\mathbf{Y}) = \mathbf{A}'\Sigma\mathbf{A}$, coincide con Λ , que es una matriz diagonal, puesto que las c.p. están incorreladas y sus varianzas (los valores de la diagonal principal) son los autovalores de Σ . En consecuencia:

$$\sum_{i=1}^p Var(Y_i) = tr(\Lambda) = tr(\mathbf{A}'\Sigma\mathbf{A}) = tr(\Sigma\mathbf{A}\mathbf{A}') = tr(\Sigma) = \sum_{i=1}^p Var(X_i), \quad (14.6)$$

pudiéndose comprobar que la suma de las varianzas de las variables originales⁶ coincide con la suma de las varianzas de las c.p.

Por tanto, la j -ésima c.p. captura un porcentaje de la variabilidad de las variables originales cifrado en $\frac{\lambda_j}{\sum_{j=1}^p \lambda_j} 100$, siendo $\frac{\sum_{j=1}^m \lambda_j}{\sum_{j=1}^p \lambda_j} 100$ la proporción capturada por las m primeras componentes.

⁵El mayor no puede ser, ya que coincidirían \mathbf{a}_1 y \mathbf{a}_2 , en cuyo caso Y_1 e Y_2 no estarían incorrelacionadas.

⁶Recuérdese que se adopta como medida de la variabilidad de las variables originales la suma de sus varianzas.

14.2.2.2. Componentes principales a partir de variables estandarizadas

A menudo, no solo se centran las variables originales sino que también se estandarizan, para que tengan varianza unitaria.⁷ La razón es que, si las variables originales presentan grandes diferencias en sus escalas de medida o en los rangos de las unidades de medida (edad en años, altura en metros, longitud en kilómetros...), sus combinaciones lineales tendrán poco significado, porque las variables que las conforman no son “igualmente importantes” y en la primera componente tendrá un gran peso la variable original con mayor magnitud (Chatfield and Collins, 1980). Si no fuera el caso, es mejor partir de Σ ; además, la teoría muestral de las c.p. es mucho más compleja cuando las variables están estandarizadas que cuando no lo están (Morrison, 1976).

El mecanismo de obtención de las c.p. no cambia en absoluto, pero su punto de arranque ya no es Σ sino \mathbf{P} , la matriz de correlaciones de dichas variables. Los autovectores de \mathbf{P} son, en general, distintos a los de Σ . Además, la suma de los autovalores, como coincide con la suma de las varianzas de las variables originales, es p , luego el porcentaje de la variación total capturada por la componente j -ésima es $\frac{\lambda_j}{p} 100$, siendo $\frac{\sum_{j=1}^m \lambda_j}{p} 100$ la proporción capturada por las m primeras componentes .

14.2.2.3. Correlación entre las variables originales y las componentes principales

Considérese la variable original X_i y la c.p. $Y_j = a_{1j}X_1 + a_{2j}X_2 + \dots + a_{pj}X_p = \mathbf{a}'_j \mathbf{X}$. Dado que $\mathbf{X}' = [X_1, \dots, X_p]$, entonces $X_i = \mathbf{e}'_i \mathbf{X}$, donde \mathbf{e}_i es un vector de ceros excepto un 1 en la i -ésima posición.

Entonces, como $(\Sigma - \lambda_j \mathbf{I})\mathbf{a}_j = 0$, se tiene que $\Sigma \mathbf{a}_j = \lambda_j \mathbf{a}_j$ y que:

$$Cov(X_i, Y_j) = Cov(\mathbf{e}'_i \mathbf{X}, \mathbf{a}'_j \mathbf{X}) = \mathbf{e}'_i \Sigma \mathbf{a}_j = \mathbf{e}'_i \lambda_j \mathbf{a}_j = \lambda_j a_{ij}, \quad (14.7)$$

$$r_{X_i, Y_j} = \frac{Cov(X_i, Y_j)}{\sqrt{Var(X_i)} \sqrt{Var(Y_j)}} = \frac{\lambda_j a_{ij}}{\sqrt{\sigma_{ii}} \sqrt{\lambda_j}} = \frac{\sqrt{\lambda_j} a_{ij}}{\sigma_{ii}}, \quad (14.8)$$

donde σ_{ii} es el elemento i -ésimo de la diagonal principal de Σ .

Si se parte de variables estandarizadas, entonces se tiene que $r_{Z_i, Y_j} = \sqrt{\lambda_j} a_{ij}$, donde ahora λ_j es el j -ésimo autovalor de \mathbf{P} y a_{ij} es el elemento i -ésimo de su autovector asociado. Sin embargo, el coeficiente de correlación lineal no varía por el hecho de haber estandarizado las variables originales.

Como se verá posteriormente, estos dos coeficientes, r_{X_i, Y_j} y r_{Z_i, Y_j} , serán de gran utilidad en la interpretación de las c.p. Además, como r_{X_i, Y_j} coincide con el coseno del ángulo que forma X_i con Y_j (que es la proyección o coordenada de X_i en el eje de Y_j), resulta de gran ayuda para representar las variables originales en el espacio de las componentes y, por consiguiente, para la interpretación de estas últimas. A mayor coseno, mayor correlación lineal entre X_i e Y_j . Matricialmente, y denominando \mathbf{A}^* a la matriz de coeficientes de correlación lineal entre las

⁷Las variables estandarizadas se denotan por Z_1, Z_2, \dots, Z_p .

variables originales estandarizadas y las c.p., se tiene que $\mathbf{A}^* = \mathbf{A}\Lambda^{\frac{1}{2}}$. \mathbf{A}^* (imprescindible en la interpretación de las c.p.) no cambia por el hecho de estandarizar también las c.p.

Los cuadrados de los elementos de \mathbf{A}^* expresan la proporción de varianza de la variable X_i explicada por la componente Y_j . Por tanto, la suma de los cuadrados de las filas de \mathbf{A}^* será la unidad. Se denomina *contribución* (individual) de X_i a la componente Y_j a la cantidad

$$\frac{r_{X_i, Y_j}^2}{\sum_{i=1}^p r_{X_i, Y_j}^2} = \frac{\cos(X_i, Y_j)}{\sum_{i=1}^p \cos(X_i, Y_j)}.$$

Otras dos expresiones interesantes que involucran \mathbf{A}^* son $\mathbf{A}^*\mathbf{A}^{*\prime} = \mathbf{R}$ y $\mathbf{A}^{*\prime}\mathbf{A}^* = \Lambda$.

14.3. Estimación de las componentes principales

Hasta el momento, se han derivado las c.p. suponiendo conocida la matriz de covarianzas poblacional Σ (o la de correlaciones \mathbf{P}). Sin embargo, este no suele ser el caso en la práctica, por lo que se sustituyen por sus homónimas muestrales $\mathbf{S} = \frac{1}{N}\mathbf{X}'\mathbf{X}$ (o \mathbf{R}). Nada cambia en el proceso de obtención de las c.p., salvo que el punto de partida es \mathbf{S} (o \mathbf{R}) y que los valores de los autovalores y autovectores asociados son estimaciones.

En el ejemplo que nos ocupa, \mathbf{R} puede verse en la Fig. 14.1. Puede apreciarse que la correlación entre las variables es notable en la mayoría de los casos, lo que invita a analizar el problema con menos variables e incorreladas, es decir, mediante ACP.

```
library("CDR")
data("TIC2021")
TIC <- TIC2021

library("corrplot")
corrplot.mixed(cor(TIC))
```

```
library("FactoMineR")
acp <- PCA(TIC, ncp = 7, graph = FALSE)
```

14.4. Número de componentes a retener

Dado que la finalidad de la técnica de componentes principales es la reducción de la dimensionalidad, una decisión clave es el número m de componentes a retener. Los criterios más populares para tomar esta decisión son:

- a) **Seleccionar un número de componentes que capturen, entre todas, un porcentaje de la variabilidad total determinado**

Dicho porcentaje suele estar alrededor del 80 %, si bien, si el número de c.p. es elevado, su interpretación es muy difícil.

14.4. Número de componentes a retener

241

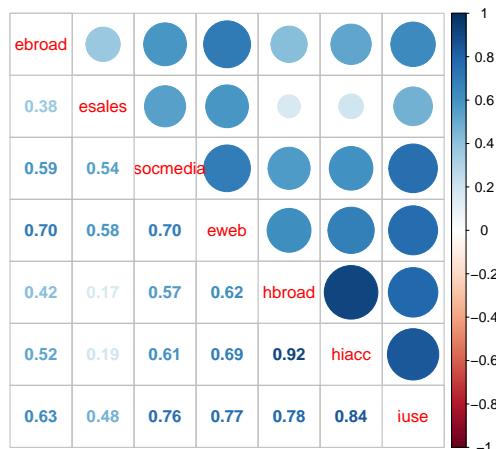


Figura 14.1: Matriz de correlaciones

b) Criterio de la media aritmética o criterio de Kaiser

Dado que la variabilidad total coincide con la suma de los autovalores, se seleccionan aquéllas c.p. cuya varianza exceda la varianza media. Es decir, se selecciona la componente j -ésima si $\lambda_j > \bar{\lambda}$ (si se parte de Σ) o si $\lambda_j > 1$ (si se parte de \mathbf{R}). En caso de valores anómalos (*outliers*) es recomendable utilizar la media geométrica en vez de la aritmética.

c) Criterio de Catell

Se basa en la representación gráfica de los autovalores vs. su número de orden, que se denomina **gráfico de sedimentación** porque se asemeja a la ladera de una montaña con su correspondiente zona de sedimentación. Se seleccionan las c.p. asociadas a los autovalores previos a la zona de sedimentación. En general, el criterio de Catell tiende a incluir demasiadas c.p., al contrario que el de la media, que tiende a incluir demasiado pocas (sobre todo si $p < 20$) (Mardia et al., 1979b).

d) Otros criterios

Otros criterios menos populares son la validación cruzada, el test de esfericidad o igualdad de autovalores de Anderson (requiere normalidad multivariante) y el criterio del bastón roto (véase Cuadras (2007) para los dos últimos). Para grandes conjuntos de datos, Jobson (1992) propone un criterio basado en la partición de la muestra en submuestras mutuamente excluyentes, similar a la validación cruzada.

La Fig. 14.2 muestra el gráfico de sedimentación en el ejemplo que nos ocupa. Puede apreciarse que con tan solo las dos primeras c.p. se captura el 82,07 % de la variabilidad total de las siete variables originales.

```
round(acp$eig[1:7, 1:2], 3)
#>      eigenvalue percentage of variance
#> comp 1      4.644      66.341
#> comp 2      1.101      15.731
#> comp 3      0.547       7.814
#> comp 4      0.328       4.679
#> comp 5      0.191       2.731
#> comp 6      0.124       1.768
#> comp 7      0.066       0.937
library("factoextra")
fviz_eig(acp, addlables = TRUE)
```

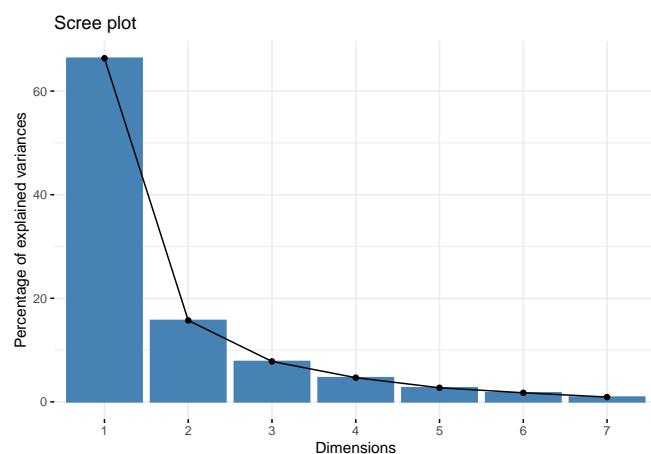


Figura 14.2: Gráfico de sedimentación

14.5. Interpretación de las componentes principales

Una primera vía consiste en analizar el signo y la magnitud de los coeficientes (cargas o *loadings*) de cada variable original en cada componente.

```
loadings <- sweep(acp$var$coord, 2, sqrt(acp$eig[1:7, 1]))
round(loadings, 3)
#>           Dim.1 Dim.2 Dim.3 Dim.4 Dim.5 Dim.6 Dim.7
#> ebroad     -1.410 -0.854 -1.358 -0.561 -0.303 -0.271 -0.259
#> esales     -1.604 -0.318 -0.411 -0.404 -0.310 -0.262 -0.225
#> esocmedia   -1.317 -0.858 -0.645 -1.062 -0.536 -0.311 -0.244
#> eweb        -1.264 -0.865 -0.825 -0.356 -0.773 -0.422 -0.284
#> hbroad     -1.343 -1.550 -0.570 -0.495 -0.413 -0.159 -0.386
#> hiacc       -1.290 -1.496 -0.675 -0.495 -0.413 -0.348 -0.053
#> iuse        -1.217 -1.135 -0.652 -0.587 -0.255 -0.608 -0.331
```

14.5. Interpretación de las componentes principales

243

- Una segunda vía es el análisis de los $r_{X_i, Y_j}, \forall i, j$.

```
round(acp$var$cor, 3)
#>           Dim.1  Dim.2  Dim.3  Dim.4  Dim.5  Dim.6  Dim.7
#> ebroad     0.745  0.195 -0.618  0.012  0.134  0.081 -0.003
#> esales      0.551  0.731  0.328  0.169  0.128  0.090  0.031
#> esocmedia   0.838  0.191  0.095 -0.490 -0.099  0.040  0.012
#> eweb        0.891  0.185 -0.085  0.217 -0.336 -0.070 -0.028
#> hbroad      0.812 -0.501  0.170  0.077  0.024  0.193 -0.129
#> hiacc       0.865 -0.446  0.065  0.077  0.024  0.003  0.203
#> iuse         0.938 -0.086  0.087 -0.014  0.183 -0.256 -0.075
fviz_pca_var(acp,
  col.var = "contrib",
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = TRUE
)
```

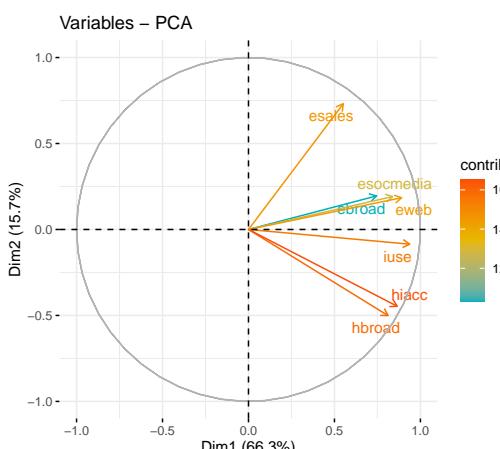


Figura 14.3: Gráfico de cosenos o coeficientes de correlación variables-componentes

Se puede utilizar cualquiera de las dos vías, pues los resultados no serán contradictorios. Sin embargo, algunos autores recomiendan no utilizar sólo la segunda, pues los r_{X_i, Y_j} sólo tienen en cuenta la variable original considerada y no el resto; es decir, se estarían interpretando las componentes desde una perspectiva univariante.

Nota

También es de interés la siguiente consideración: cuando las variables originales están correlacionadas positivamente, la primera c.p. tiene todas sus coordenadas del mismo signo y puede interpretarse como un promedio ponderado de todas ellas.

En la matriz de cargas (o *loadings*) se aprecia que la primera c.p. es una media ponderada (con ponderaciones similares) de las variables originales, mientras que *esales*, *hbroad* y *hiacc*

cargan fuertemente en la segunda (*esales* positivamente y las otras dos de forma negativa). La interpretación desde la perspectiva univariante de los coeficientes de correlación lineal es prácticamente la misma. Por ello, cabe interpretar la primera c.p. como un indicador general del uso de las TIC, mientras que la segunda, positivamente relacionada con la dotación TIC de las empresas pero con una fuerte relación negativa con la de los individuos y hogares, pudiera estar relacionada con las ayudas públicas a la implantación de TICs en el tejido empresarial.

```
acp1 <- fviz_contrib(acp, choice = "var", axes = 1, top = 10)
acp2 <- fviz_contrib(acp, choice = "var", axes = 2, top = 10)
library(patchwork)
acp1 + acp2
```

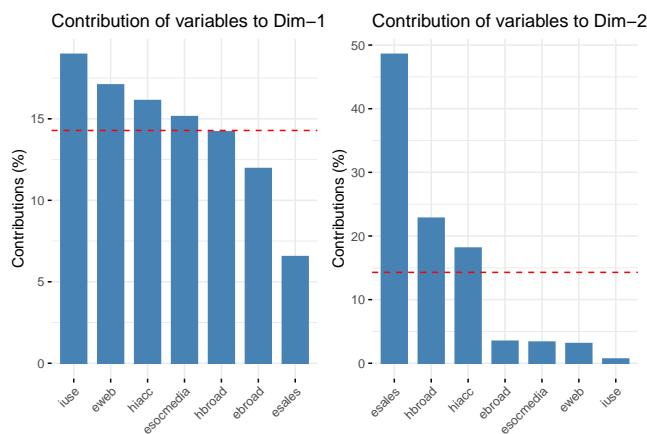


Figura 14.4: Contribución de las variables originales a las componentes retenidas

14.6. Reproducción de los datos tipificados y de la matriz de

En la práctica, el punto de partida del ACP es la matriz **R**, y en tal caso se suelen estandarizar también las c.p. Pues bien, se tiene que:

$$\mathbf{Y}^* = \mathbf{X}\mathbf{A}\Lambda^{-\frac{1}{2}} = \mathbf{X}\mathbf{A}\Lambda^{\frac{1}{2}}\Lambda^{-1} = \mathbf{X}\mathbf{A}^*\Lambda^{-1} = \mathbf{XF},$$

donde **F** es la matriz de puntuaciones de las c.p. La expresión anterior proporciona las coordenadas de los N elementos en el espacio de las c.p. y, por tanto, sirve de ayuda en la interpretación de éstas. La estandarización de las c.p. asegura que los m ejes (componentes) tengan una métrica homogénea que facilitará la visualización e interpretación. Dichas coordenadas también constituyen el input de técnicas híbridas como, por ejemplo, regresión con c.p., cluster o PLS.

En este caso:

14.7. Limitaciones del análisis de componentes principales

245

```
puntuaciones <- acp$ind$coord
round(puntuaciones[1:5, ], 3)
#> Dim.1 Dim.2 Dim.3 Dim.4 Dim.5 Dim.6 Dim.7
#> BE 1.651 1.053 0.310 -0.238 -0.196 0.296 -0.196
#> BG -4.759 -0.127 -0.128 -0.223 0.023 -0.013 -0.106
#> CZ -0.324 0.875 -0.564 0.870 -0.082 -0.032 -0.345
#> DK 3.188 1.331 0.497 0.262 0.343 -0.300 0.319
#> DE 0.024 0.144 -0.183 0.560 -0.871 -0.485 0.115
```

De la expresión anterior se deduce que $\mathbf{X} = \mathbf{Y}\Lambda^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}\mathbf{A}' = \mathbf{Y}^*_{N \times m}\mathbf{A}_{m \times p}^{*'}\mathbf{Y}_{N \times m}\mathbf{A}_{m \times p}$, expresión que permite reproducir la matriz \mathbf{X} a partir de las m primeras c.p. estandarizadas. En consecuencia, la reproducción de \mathbf{R} a partir de las m primeras c.p. estandarizadas se lleva a cabo como sigue:

$$\begin{aligned} \mathbf{R}_{p \times p} &= \frac{1}{N} \mathbf{X}'_{p \times N} \mathbf{X}_{N \times p} = \mathbf{A}_{p \times m}^* \frac{1}{N} \mathbf{Y}_{m \times N}^{*'} \mathbf{Y}_{N \times m}^* \mathbf{A}_{m \times p}^* \\ &= \mathbf{A}_{p \times m}^* \mathbf{I}_{m \times m} \mathbf{A}_{m \times p}^{*'} = \mathbf{A}_{p \times m}^* \mathbf{A}_{m \times p}^{*'}. \end{aligned} \quad (14.9)$$

Debajo se muestran las tres primeras filas de la reproducción de \mathbf{R} a partir de las dos primeras c.p. De la comparación de sus valores con los de la verdadera \mathbf{R} (Fig. 14.1) se concluye que se trata de una buena reproducción.⁸

```
matrix <- acp$var$coord[, 1:2] %*% t(acp$var$coord)[1:2, ]
round(matrix[1:3, ], 3)
#> ebroad esales esocmedia eweb hbroad hiacc iuse
#> ebroad 0.593 0.553 0.662 0.700 0.507 0.557 0.682
#> esales 0.553 0.839 0.602 0.626 0.081 0.151 0.455
#> esocmedia 0.662 0.602 0.740 0.782 0.585 0.640 0.770
```

14.7. Limitaciones del análisis de componentes principales

Una primera limitación es que su implementación sólo es posible si todas las variables se trabajan bajo un nivel de análisis numérico. Otra limitación importante es el supuesto subyacente de que los datos observados son combinación lineal de una cierta base. Es decir, sólo se consideran las combinaciones lineales de las variables originales. Otros métodos de reducción de la dimensionalidad como, por ejemplo, el *t-distributed stochastic neighbor embedding* (t-SNE), o la versión Kernel de la técnica, que también funcionan con no linealidad, superan esta limitación.

Además, el hecho que todas las c.p. sean combinaciones lineales de todas las variables originales dificulta su interpretación. Para superar esta limitación, han surgido algunas alternativas, como el *sparse PCA*, que obtiene las c.p. como un problema minimización del error de reconstrucción forzando a que los autovectores tengan una gran parte de sus componentes nula.

⁸Con tres c.p. la reproducción es casi perfecta. Se han retenido sólo dos para poder mostrar representaciones bidimensionales y para evitar la interpretación de una tercera componente.

El t-SNE no es la única alternativa no lineal procedente de la comunidad de *machine learning*. Otras, denominadas actualmente “aprendizaje múltiple” (*manifold learning*) , incluyen el *Sammon's mapping*, el *curvilinear component analysis* (CCA) y sus variantes: los *Laplacian eigenmaps* y el *maximum variance unfolding* (MVU); véase [Wismüller et al. \(2010\)](#).

Finalmente, señalar que el ACP es una técnica matemática que no requiere que las variables originales sigan una distribución normal multivariante, aunque, si así fuera, se podría dar una interpretación más profunda de las c.p.

Resumen

El ACP es una técnica de reducción de la dimensionalidad que captura un gran porcentaje de la variabilidad de un conjunto de variables correladas a partir de un número mucho menor de componentes latentes (las componentes principales) incorreladas. La piedra angular de la construcción de estas componentes son los autovalores de la matriz de covarianzas (o de correlaciones) de las variables originales. En el ACP son cuestiones importantes, entre otras, (i) la determinación del número de componentes a retener, (ii) su interpretación y (iii) la cuantificación del valor de las componentes para cada observación (puntuaciones), que constituyen el input de técnicas híbridas como, por ejemplo, regresión con componentes principales, cluster o *partial least squares*.

Capítulo 15

Análisis factorial

José-María Montero^a y José Luis Alfaro Navarro^a

^aUniversidad de Castilla-La Mancha

15.1. Introducción

Según el trabajo pionero de Harman (1976), el objeto del **Análisis Factorial** (AF) es la representación de una variable X_j en términos de varios factores subyacentes no observables¹. En el marco lineal, y considerando p variables², hay varias alternativas dependiendo del objetivo que se pretenda:

- La captura de la mayor cantidad posible de la varianza de dichas variables (o “explicación” de su varianza).
- La mejor reproducción (o “explicación”) de sus correlaciones observadas.

A modo introductorio, supónganse dos variables polítómicas que surgen de las respuestas de N futbolistas profesionales a dos preguntas: (1) ¿Está usted a gusto en el club? y (2) ¿Se quedaría en el club la siguiente temporada? Las posibles respuestas son: 1, 2, 3, 4, 5 (1 “en total desacuerdo” y 5 “totalmente de acuerdo”).

Cada variable tiene su varianza (nula si todos los futbolistas opinan igual y máxima si la mitad marcase el 1 y la otra mitad el 5). Esta varianza puede ser *común* o *compartida* por las dos variables, o no. Lo normal es que cuanto más a gusto estén los futbolistas en su club mayor sea su deseo de permanecer en él la siguiente temporada, por lo que gran parte de la variabilidad de cada una de las variables sería compartida (ya que la relación -lineal- entre ellas es positiva). El resto de la variabilidad sería *específica* de cada variable (puede que un futbolista

¹Se asumirá la representación lineal, por sencillez, pero puede ser cualquier otra.

²Por las mismas razones que en el análisis de componentes principales (ACP), se trabaja con las variables estandarizadas; véase Cap. 14.

esté muy bien en el club, pero quiera ir a otro más prestigioso; o que esté mal, pero a su familia le encante la ciudad) o *residual* (normalmente debida a factores de medida). El porcentaje de *varianza compartida* se mide a través del coeficiente de determinación lineal, r^2 . El resto, hasta la varianza unidad, o el 100 %, es *varianza única* de cada variable, que incluye tanto la específica como la residual.

De acuerdo con [De la Fuente \(2011\)](#), en el AF caben dos enfoques:³

- El análisis de toda la varianza (común y no común).
- El análisis, únicamente, de la varianza común.

Ambos caben bajo el paraguas genérico del AF; ambos se basan en las relaciones entre las variables para identificar grupos de ellas asociadas entre sí. Sin embargo, del primero se ocupa el ACP (Cap. 14) y, si se parte de la matriz de correlaciones (cuyas entradas fuera de la diagonal principal, al cuadrado, indican la proporción de varianza compartida por las variables que se cruzan en dicha entrada), ésta lleva unos en la diagonal principal. Al segundo se le aplica la denominación de AF y en la matriz de correlaciones se sustituyen los unos de la diagonal principal por la varianza que cada variable comparte con las demás (su **comunalidad**). Por eso se dice que el objetivo del AF es la explicación de la varianza compartida o común de las variables en estudio mediante una serie de **factores comunes** latentes.⁴

El AF puede ser exploratorio o confirmatorio. En el primero no se establecen consideraciones a priori sobre el número de factores comunes a extraer, sino que éste se determina a lo largo del análisis. Por el contrario, en el segundo se trata de contrastar hipótesis relativas al número de factores comunes, así como sobre qué variables serán agrupadas o tendrán más peso en cada factor. Una práctica habitual es validar mediante el *análisis factorial confirmatorio* los modelos teóricos basados en los resultados del *análisis factorial exploratorio*. Sin embargo, [Pérez-Gil et al. \(2000\)](#) alertan de los peligros de esta práctica. Este capítulo considera la versión exploratoria del AF.

A efectos prácticos, se utilizará la base de datos TIC2021 del paquete CDR, ya trabajada en Cap. 14 para el ACP, relativa al uso (por empresas e individuos) y equipación (de los hogares) de las TIC en los países de la UE-27, así como la librería psych ([Revelle, 2022](#)) de R.

```
library(psych)
library("CDR")
data("TIC2021")
```

15.2. Elementos teóricos del análisis factorial

³No son los únicos.

⁴Ambos enfoques dan resultados similares cuando hay más de 30 variables y las communalidades (véase Sec. 15.2.1) son superiores a 0,70, y se interpretan de manera casi idéntica.

15.2.1. Modelo básico y terminología

Considérense p variables $\{X_1, X_2, \dots, X_p\}$ y N elementos, objetos o individuos, siendo las matrices de datos, \mathbf{X} , y datos estandarizados, \mathbf{Z} , las siguientes:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pN} \end{pmatrix}, \quad \mathbf{Z} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1N} \\ z_{21} & z_{22} & \cdots & z_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ z_{p1} & z_{p2} & \cdots & z_{pN} \end{pmatrix},$$

donde el primer subíndice indica la variable y el segundo el elemento.

Mientras que el enfoque de componentes principales está representado por:

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jp}F_p, \quad j = 1, 2, \dots, p, \quad (15.1)$$

en el enfoque AF clásico el modelo teórico es:

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jk}F_k + b_jSP_j + c_jE_j, \quad j = 1, 2, \dots, p, \quad (15.2)$$

donde Z_j , $j = 1, 2, \dots, p$, se modeliza, linealmente, en términos de (i) $k \ll p$ **factores comunes**, F_m , $m = 1, 2, \dots, k$, que dan cuenta de la correlaciones entre las variables Z_j , $j = 1, 2, \dots, p$, y (ii) un **factor específico**, SP_j , $j = 1, 2, \dots, p$, y un término de error, E_j , $j = 1, 2, \dots, p$, que dan cuenta de la **varianza no compartida** (específica y residual, respectivamente). Los coeficientes a_{jm} se denominan **cargas factoriales** y, aunque su notación es igual que en el modelo de componentes principales, no tienen por qué coincidir; el problema básico del AF es precisamente la estimación de dichas cargas. En lo que sigue, se aunará el factor específico y el término de error de Z_j en un **factor único**, U_j , con lo que:

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jk}F_k + d_jU_j, \quad j = 1, 2, \dots, p, \quad (15.3)$$

Los supuestos del modelo (15.3) son los siguientes:

- Como en la práctica los factores comunes y únicos son desconocidos, sin pérdida de generalidad pueden suponerse con media cero y varianza unitaria;
- Los factores únicos se suponen independientes entre sí y de los factores comunes;
- Y dado que los factores involucrados en el modelo se consideran variables aleatorias, si se asume normalidad, e independencia de los factores comunes, $\{F_1, F_2, \dots, F_k\}$ sigue una distribución normal multivariante y Z_j , $j = 1, 2, \dots, p$, una distribución normal.

En términos de valores observados, el modelo AF (15.3) viene dado por:⁵

$$z_{ji} = a_{j1}f_{1i} + a_{j2}f_{2i} + \cdots + a_{jk}f_{ki} + d_ju_{ji}, \quad i = 1, 2, \dots, N; \quad j = 1, 2, \dots, p, \quad (15.4)$$

⁵Aunque en el modelo figuran explícitamente los valores de los factores, en la práctica hay que estimarlos. En otras versiones del AF estos valores se estiman conjuntamente con los parámetros.

El modelo AF es muy parecido al de regresión lineal: una variable se describe como una combinación lineal de otro conjunto de variables más un residuo. Sin embargo, en el análisis de regresión las variables son observables, mientras que en el AF son construcciones hipotéticas que sólo pueden estimarse a partir de los datos observados. Los propios factores se estiman en una etapa posterior del análisis.

En términos matriciales, y considerando:

$$\mathbf{z} = \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_p \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_k \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_p \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pk} \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_p \end{pmatrix},$$

el modelo (15.3) puede expresarse como $\mathbf{z} = \mathbf{Af} + \mathbf{Du}$.

Centrándonos en el modelo (15.3), la varianza de Z_j viene dada por:

$$V(Z_j) = 1 = \sum_{m=1}^k a_{jm}^2 + 2 \sum_{m < q} a_{jm} a_{jq} r_{(F_{mi}, F_{qi})} + d_j^2, \quad (15.5)$$

y si los factores comunes están incorrelados, $V(Z_j) = 1 = \sum_{m=1}^k a_{jm}^2 + d_j^2$.

De la expresión (15.5) surgen las siguientes definiciones:

- a_{jm}^2 es la contribución de F_m a la varianza de Z_j .
- $V_m = \sum_{j=1}^p a_{jm}^2$ es la contribución de F_m a suma de las varianzas de todas las variables Z_j , $j = 1, 2, \dots, p$.
- $V = \sum_{m=1}^k V_m$ es la contribución de todos los factores comunes a la varianza de todas las variables Z_j , $j = 1, 2, \dots, p$.
- $\frac{V}{p}$ es un indicador de la *completitud* del análisis factorial.
- $h_j^2 = a_{j1}^2 + a_{j2}^2 + \dots + a_{jk}^2$ es la communalidad de Z_j , $j = 1, 2, \dots, p$, es decir la contribución de los factores comunes a la variabilidad de Z_j .
- d_j^2 es la *unicidad* (o varianza única) de Z_j , $j = 1, 2, \dots, p$, o contribución de U_j a la varianza de Z_j . Es un indicador de la medida en la que los factores comunes fracasan a la hora de representar la varianza (unitaria) de Z_j .
- Cuando se descompone el factor único en sus dos componentes (modelo (15.2)), b_j^2 se denomina *especificidad* (o varianza específica) de Z_j y es la varianza de Z_j debida a la particular selección de las variables en el estudio, mientras que c_j^2 es la que se debe al error (normalmente de medida), que mide la “falta de fiabilidad”.

15.2.2. Patrón y estructura factoriales

Se denomina **patrón factorial** a la siguiente expansión del modelo (15.3),

$$\begin{aligned} Z_1 &= a_{11}F_1 + a_{12}F_2 + \cdots + a_{1k}F_k + d_1U_1 \\ Z_2 &= a_{21}F_1 + a_{22}F_2 + \cdots + a_{2k}F_k + d_2U_2 \\ &\vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \\ Z_p &= a_{p1}F_1 + a_{p2}F_2 + \cdots + a_{pk}F_k + d_pU_p \end{aligned} \tag{15.6}$$

o simplemente a la tabla, o matriz, con los coeficientes a_{jm} y d_j (o únicamente, caso habitual, a la *matriz de cargas A*). k determina la “complejidad del modelo”.

Se denomina **estructura factorial** al siguiente conglomerado de $k+1$ conjuntos de p ecuaciones lineales, en $\{a_{jm}\}$, los k primeros, y en $\{d_j\}$, el último, $j = 1, 2, \dots, p$; $m = 1, 2, \dots, k$:⁶

$$\begin{aligned} r_{Z_j F_1} &= a_{j1}r_{F_1 F_1} + a_{j2}r_{F_1 F_2} + \cdots + a_{jm}r_{F_1 F_m} + \cdots + a_{jk}r_{F_1 F_k} \\ &\vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ r_{Z_j F_m} &= a_{j1}r_{F_m F_1} + a_{j2}r_{F_m F_2} + \cdots + a_{jm}r_{F_m F_m} + \cdots + a_{jk}r_{F_m F_k} \\ &\vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ r_{Z_j F_k} &= a_{j1}r_{F_k F_1} + a_{j2}r_{F_k F_2} + \cdots + a_{jm}r_{F_k F_m} + \cdots + a_{jk}r_{F_k F_k} \\ &\vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ r_{Z_j U_j} &= d_j \\ &\vdots \end{aligned} \tag{15.7}$$

En la práctica, viene dada por una tabla, o matriz, Γ , con los coeficientes $\{r_{jm}\}$. Cuando todos los factores están incorrelados el patrón y la estructura coinciden.

El conjunto patrón factorial más estructura factorial se denomina **solución factorial completa**. El patrón factorial muestra la relación lineal de las variables en términos de los factores, como si de una regresión lineal se tratase, y puede usarse para reproducir la correlación entre las variables (y, por tanto, para determinar la bondad de la solución). La estructura factorial es útil para la identificación de los factores y la posterior estimación de las **puntuaciones factoriales**.

En términos matriciales, denominando

$$\mathbf{F} = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \cdots & f_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{k1} & f_{k2} & \cdots & f_{kN} \end{pmatrix}, \quad \boldsymbol{\Gamma} = \begin{pmatrix} r_{Z_1 F_1} & r_{Z_1 F_2} & \cdots & r_{Z_1 F_k} \\ r_{Z_2 F_1} & r_{Z_2 F_2} & \cdots & r_{Z_2 F_k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{Z_p F_1} & r_{X_p^* Z_p F_2} & \cdots & r_{Z_p F_k} \end{pmatrix},$$

⁶En caso de variables dicotómicas se utiliza el coeficiente ϕ (véase 5.4) como medida de correlación momento-producto.

el patrón factorial viene dado por $\mathbf{Z} = \mathbf{AF} + \mathbf{DU}$. Multiplicando por \mathbf{F}' y realizando simples operaciones se tiene que $\mathbf{\Gamma} = \mathbf{A}\Phi$, donde Φ es la matriz de correlaciones entre los factores comunes. Si los factores comunes están incorrelados, $\mathbf{\Gamma} = \mathbf{A}$.

Por último, resaltar que el AF es indeterminado, es decir, dado un conjunto de correlaciones, los coeficientes del patrón factorial no son únicos (dado \mathbf{R} , se pueden encontrar infinitos sistemas de factores incorrelados u ortogonales)⁷ consistentes con ella. Por ello, normalmente, tras obtener una solución que ajuste bien los datos originales, se lleva a cabo una rotación de la misma (que ajusta igual de bien dichos datos) que facilite la *interpretación de los factores*.⁸

15.3. El análisis factorial en la práctica

15.3.1. Pre-análisis factorial

15.3.1.1. ¿Procede la realización de un análisis factorial?

Antes de comenzar con el AF, conviene determinar si procede o no; es decir, si las variables se encuentran fuertemente intercorrelacionadas o no. En caso negativo, el AF no tendría sentido. Para ello, se pueden utilizar procedimientos sencillos como observar si el determinante de \mathbf{R} es bajo (correlaciones altas) o elevado (correlaciones bajas); o calcular la *matriz de correlaciones anti-imagen*, cuyos elementos son los coeficientes de correlación parcial cambiados de signo. En la diagonal muestra la *medida de adecuación muestral* para esa variable, MSA_j . Para que se den las condiciones de realización del AF, la mayoría de los elementos no diagonales deben ser pequeños y los diagonales deben estar próximos a la unidad.

Otras alternativas más sofisticadas incluyen las dos siguientes:

Contraste de esfericidad de Bartlett

Exige normalidad multivariante. Contrastá la incorrelación de las variables, es decir, $H_0 : \mathbf{R} = \mathbf{I}$ frente a $H_1 : \mathbf{R} \neq \mathbf{I}$ (o $H_0 : |\mathbf{R}| = 1$ frente a $H_1 : |\mathbf{R}| \neq 1$). El estadístico de contraste es $d_{\mathbf{R}} = -(N - 1 - \frac{1}{6}(2p + 5)) \ln|\mathbf{R}|$ y, bajo H_0 , sigue una $\chi^2_{\frac{p(p-1)}{2}}$, siendo nulo en caso de incorrelación.

```
n <- dim(TIC2021)[1]
cortest.bartlett(cor(TIC2021),n)$chisq
#> [1] 149.7113
cortest.bartlett(cor(TIC2021),n)$p.value
#> [1] 1.992514e-21
```

⁷La historia es más larga: el AF es indeterminado porque dada una matriz $\mathbf{C}_{k \times k}$ no singular, si se define otro vector de factores comunes $\mathbf{f}^* = \mathbf{C}^{-1}\mathbf{f}$ y otra matriz $\mathbf{A}^* = \mathbf{AC}$, entonces $\mathbf{Z} = \mathbf{Af} + \mathbf{Du} = \mathbf{A}^*\mathbf{C}^{-1}\mathbf{C}\mathbf{f}^* + \mathbf{Du} = \mathbf{A}^*\mathbf{f}^* + \mathbf{Du}$ y ambos son equivalentes. Una solución es exigir la incorrelación de los factores comunes ($\Phi = \mathbf{I}$), con lo que la indeterminación se reduciría sólo a cuando \mathbf{C} sea ortogonal. En este caso, la solución será única salvo rotaciones ortogonales.

⁸Una solución determina el espacio k -dimensional que contiene los k factores comunes, pero no su posición exacta.

Medida de adecuación muestral de Kaiser, Meyer y Olkin (KMO)

Se basa en la idea de que, entre cada par de variables, el coeficiente de correlación parcial (que mide la correlación existente entre cada par de ellas eliminando el efecto que el resto de variables tiene sobre las dos consideradas), debe ser cercano a cero, puesto que es una estimación de la correlación entre sus factores específicos, que se suponen incorrelados. Por tanto, si el número de coeficientes de correlación parcial no nulos es elevado, la solución factorial no es compatible con los datos.

En otros términos, cuando las variables incluidas en el análisis comparten gran cantidad de información debido a la presencia de factores comunes, la correlación parcial entre cualquier par de variables debe ser reducida. Por el contrario, cuando dos variables comparten gran cantidad de información entre ellas, pero no la comparten con las restantes variables (ni, consecuentemente, con los factores comunes), la correlación parcial entre ellas será elevada, lo cual es un mal síntoma de cara a la idoneidad del AF.

El índice KMO se define como $KMO = \frac{\sum_{j \neq i} r_{ji}^2}{\sum_{j \neq i} r_{ji}^2 + \sum_{j \neq i} r_{ji}^{*2}}$, donde r_{ji}^* es el coeficiente de

correlación parcial entre las variables Z_j y Z_i . Se considera que valores por encima 0,9 implican elevadísimas correlaciones en **R**; entre 0,5 y 0,9 permiten el AF; y por debajo de 0,5 resultan inaceptables para el AF.

Las MSA_j mencionadas anteriormente, son la versión del índice KMO limitado a cada variable:

$$MSA_j = \frac{\sum_{i \neq j} r_{ji}^2}{\sum_{i \neq j} r_{ji}^2 + \sum_{i \neq j} r_{ji}^{*2}}.$$

La interpretación es similar a la de KMO, pero mide la adecuación de cada variable para realizar un AF, lo que permite no considerar aquellas variables con menor MSA de cara a mejorar la KMO. No obstante, para eliminar una variable del estudio es aconsejable tener en cuenta también las comunidades de cada variable, los residuos del modelo e interpretar los factores obtenidos.

```
round(KMO(TIC2021)$MSA,3)
#> [1] 0.83
round(KMO(TIC2021)$MSAi,3)
#>   ebroad   esales esocmedia      eweb    hbroad     hiacc      iuse
#>   0.850    0.671    0.934    0.856    0.808    0.764    0.875
```

Como puede apreciarse, en nuestro ejemplo TIC, tanto el test de Barlett como el índice *KMO* y las MSA_j indican que el AF se puede llevar a cabo con garantías.

15.3.1.2. El problema de la communalidad y/o del número de factores comunes

El objetivo del AF es encontrar una *matriz de correlaciones reproducida* a partir de los resultados obtenidos, \mathbf{R}^{rep} , con menor rango que la original, \mathbf{R} , tal que su diferencia, la *matriz de correlaciones de los residuos*, \mathbf{R}^{res} , se atribuya únicamente a errores muestrales. \mathbf{R} es una matriz gramiana: simétrica de números reales y de diagonal principal dominante, con lo cual es semidefinida positiva y sus autovalores son nulos o positivos. Por tanto, el número de factores comunes será igual al de autovalores positivos ($k \leq p$). Si el punto de partida en el análisis es \mathbf{R} , rara vez se obtienen menos factores comunes que variables originales, con lo cual el AF realmente es un ACP. Ahora bien, como el número de factores comunes coincide con el rango de \mathbf{R}^{rep} , y éste se ve afectado por los valores de la diagonal principal, al sustituir los unos por las estimaciones de las communalidades (en este caso se está realizando un AF), \mathbf{R}^{rep} no será, en general, gramiana y $k < p$. En conclusión: como la solución factorial ($k < p$) pasa por el conocimiento del rango de \mathbf{R} o de las communalidades, o se hipotetiza sobre dicho rango o se hipotetizan o estiman las communalidades. Normalmente se sigue uno de estos dos caminos:

- (1) Se parte de un k prefijado, se lleva a cabo el AF y se contrasta la hipótesis H_0 : número de factores comunes = k .
- (2) Se estiman las communalidades y se obtienen los factores comunes .

En cuanto a prefijar un número k de factores, se pueden seguir los criterios expuestos en el Cap. 14 para determinar el número de componentes principales a retener (criterio de Kaiser, gráfico de sedimentación, porcentaje mínimo de varianza explicada, ...).

En cuanto a la estimación de las communalidades, de las múltiples posibilidades existentes, las siguientes son interesantes por su sencillez y buenos resultados:

- Una de las más sencillas, si el número de variables es grande, es aproximar la communalidad de una variable por su correlación más alta con las demás variables: $\hat{h}_j^2 = \max(r_{j1}, r_{j2}, \dots, r_{j(j-1)}, r_{j(j+1)}, \dots, r_{jp})$.
- Otra posibilidad es $\hat{h}_j^2 = \frac{r_{js}r_{jt}}{r_{ts}}$, donde Z_s y Z_t son, por este orden, las dos variables más correlacionadas con Z_j . Este procedimiento modera el efecto que tendría en el anterior una correlación excepcionalmente elevada.
- En la misma línea, otra posibilidad es el promedio de los coeficientes de correlación entre la variable en cuestión y las restantes: $\hat{h}_j^2 = \frac{\sum_{j \neq s} r_{js}}{p-1}$.
- Otra alternativa es realizar un ACP y tomar como communalidad de cada variable la varianza explicada por los factores retenidos con el criterio de autovalor mayor que la unidad.
- También se puede utilizar el coeficiente de determinación lineal múltiple de cada variable con las demás como estimación de la cota inferior de sus communalidades: $\hat{h}_j^2 \geq r_{Z_j;(Z_{j1}, \dots, Z_{j-1}, Z_{j+1}, \dots, Z_p)}^2 = 1 - \frac{1}{r^{jj}}$, donde r^{jj} es el j -ésimo elemento de la diagonal de \mathbf{R}^{-1} .

Un valor alto de la communalidad, próximo a $V(X_j)$, significa que dicha variable está bien representada en el espacio de factores.

15.3.2. Análisis factorial

15.3.2.1. Métodos de extracción de los factores

Método de componentes principales

Su objetivo es el análisis de toda la varianza, común y no común, (modelo (15.1)). Por consiguiente, las entradas de la diagonal de \mathbf{R}^9 son unitarias y no se requiere la estimación a priori de las communalidades ; tampoco se requiere la estimación a priori del numero de factores comunes, que se determinan a posteriori. Para la exposición del método, así como para su ejemplificación con la base de datos TIC2021 del paquete CDR, se remite al lector al Cap. 14. Aunque en el Cap. 14 se utilizó la función PCA de la librería FactoMineR, también se puede utilizar la función `principal` de la librería psych.

Este método tiene la ventaja de que siempre proporciona una solución. Sin embargo, al no estar basado en el modelo (15.3), puede dar estimaciones de las cargas factoriales muy sesgadas, sobre todo cuando hay variables con communalidades pequeñas.

Método de los factores principales

Es la aplicación del método de componentes principales a la *matriz de correlaciones reducida*, \mathbf{R}^* , es decir, con communalidades en la diagonal en vez de unos. Exige, por tanto, la estimación previa de las communalidades y su objetivo es el análisis de la varianza compartida por todas las variables (modelo (15.3)). Se trata de un procedimiento iterativo que consta de las siguientes etapas:

1.- Cálculo de la matriz de correlaciones muestrales.

2.- Estimación inicial de las communalidades utilizando el coeficiente de determinación lineal múltiple de cada variable con las demás.¹⁰

3.- Cálculo de la matriz de correlaciones reducida:

$$\mathbf{R}^* = \begin{pmatrix} \hat{h}_{1(0)}^2 & r_{12} & \cdots & r_{1p} \\ r_{21} & \hat{h}_{2(0)}^2 & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \cdots & \hat{h}_{p(0)}^2 \end{pmatrix}$$

4.- Cálculo de los autovalores y autovectores asociados a \mathbf{R}^* (matriz no necesariamente semidefinida positiva) y, a partir de ellos, obtención de las estimaciones de la matriz de cargas factoriales $\mathbf{A}_{(0)}$. En este paso hay que determinar el número de factores utilizando los criterios del ACP.

⁹Cuando se utiliza este método, se debe utilizar \mathbf{R} , porque, de otra forma, los factores comunes no tendrían media cero y varianza unitaria.

¹⁰Una estimación inicial de las communalidades equivale a una estimación inicial de las varianzas únicas: $\hat{d}_{j(0)}^2 = \hat{\sigma}_j^2 - \hat{h}_{j(0)}^2$; y si las variables originales están tipificadas: $\hat{d}_{j(0)}^2 = 1 - \hat{h}_{j(0)}^2$.

5.- A partir de la estimación de $\mathbf{A}_{(0)}$, obtención de una nueva estimación de las comunidades: $\hat{h}_{j(1)}^2 = \hat{a}_{j1(1)}^2 + \hat{a}_{j2(1)}^2 + \cdots + \hat{a}_{jk(1)}^2$ y, por tanto, de una nueva estimación de la varianza única (o unicidad) $\hat{d}_{j(1)}^2 = 1 - \hat{h}_{j(1)}^2$.

6.- Comparación de $\hat{h}_{j(1)}^2$ con $\hat{h}_{j(0)}^2$, $j = 1, 2, \dots, p$. Si hay diferencia significativa se vuelve al paso 3, y si la discrepancia no supera una cantidad prefijada se aceptan como válidas las últimas estimaciones disponibles.

En el software **R**, el método de los factores principales se implementa con la función **fa** de la librería **psych**, que parte de **R*** (véase Harman (1976) para el procedimiento iterativo y Revelle (2022) para los detalles sobre la manera cómo **fa** parte de **R*** y lleva a cabo la extracción de los factores).

```
af_facprin <- fa(cor(TIC2021), nfactors=2, rotate="none", fm='pa')
```

```
print(af_facprin, digits=3)
#> Factor Analysis using method = pa
#> Call: fa(r = cor(TIC2021), nfactors = 2, rotate = "none", fm = "pa")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>          PA1      PA2    h2   u2   com
#> ebroad    0.678   0.189 0.495 0.5050 1.16
#> esales    0.503   0.547 0.553 0.4474 1.99
#> esocmedia 0.796   0.212 0.678 0.3218 1.14
#> eweb      0.872   0.239 0.818 0.1822 1.15
#> hbroad    0.816   -0.452 0.869 0.1306 1.56
#> hiacc     0.888   -0.439 0.982 0.0181 1.46
#> iuse      0.935   -0.023 0.875 0.1248 1.00
#>
#>          PA1      PA2
#> SS loadings  4.435 0.835
#> Proportion Var 0.634 0.119
#> Cumulative Var 0.634 0.753
```

En la salida anterior, *SS loadings* son los autovalores de **R***, que coinciden con la suma de los cuadrados de las cargas de las variables en cada factor (suma de las cargas al cuadrado por columnas). *h2* son las comunidades (suma de las cargas al cuadrado por filas; sólo se muestran las cargas para los dos primeros factores puesto que entre ambos ya acumulan una varianza explicada de más del 75%). *u2* son las varianzas de los factores únicos; finalmente, *com_j* (en la salida *com*) es el número de factores comunes necesarios para explicar la variable Z_j : $com_j = \frac{(\sum_{j=1}^p a_{jm}^2)^2}{\sum_{j=1}^p a_{jm}^4}$. Cuanto mayor es *com_j* mejor es la calidad de la variable para participar en la extracción factorial. El promedio de los *com_j* se denomina *índice de complejidad de Hofmann*. Una solución de estructura simple perfecta tiene una complejidad de uno (cada variable carga solo en un factor); una solución con elementos distribuidos uniformemente tiene una complejidad mayor que 1. Interesa que la estructura no sea simple y perfecta porque entonces no tendría sentido la reducción dimensional. Por tanto, el índice de Hofmann deberá ser superior a la unidad.

Las communalidades y unicidades son:

```
round(af_facprin2$communality,3)
#> ebroad esales esocmedia eweb hbroad hiacc iuse
#> 0.495 0.553 0.678 0.818 0.869 0.982 0.875
round(af_facprin2$uniquenesses,3)
#> ebroad esales esocmedia eweb hbroad hiacc iuse
#> 0.505 0.447 0.322 0.182 0.131 0.018 0.125
```

Nótese que con el método de los factores principales, al aplicar ACP sobre \mathbf{R}^* , los factores obtenidos están incorrelados y la estructura coincide con el patrón.

Los resultados son, en signo, aunque no tanto en valor, similares a los obtenidos por el método de componentes principales. Además, como era de esperar, no permiten una interpretación clara de los factores comunes. Para facilitar dicha interpretación, dichos factores deberán ser rotados (véase Sec. 15.3.2.2).

Método de máxima verosimilitud

Exige normalidad multivariante y la determinación a priori del número de factores comunes, pero no la estimación de las communalidades. Obedece al modelo (15.3) y consiste en obtener las estimaciones máximo verosímiles de \mathbf{A} y \mathbf{D} . Dado que cualquier transformación ortogonal de \mathbf{A} puede ser una solución, se impone la condición de que $\mathbf{A}'(\mathbf{DD}')^{-1}\mathbf{A}$ sea diagonal. La log-verosimilitud viene dada por $l = -\frac{N}{2} (\log|2\pi\Sigma| + \text{tr}\Sigma^{-1}\mathbf{S})$, donde $\Sigma = \mathbf{AA}' + \mathbf{DD}'$ y \mathbf{S} son las matrices de covarianzas poblacional y muestral, respectivamente, de las variables X_j , $j = 1, 2, \dots, p$. \mathbf{DD}' es la matriz de covarianzas (diagonal) de los factores únicos, donde los elementos de la diagonal representan la parte de la varianza única de cada variable, y en la literatura sobre AF es conocida como Ψ (por ello, $d_j^2 = \psi_{jj}$).

La decisión sobre el número de factores comunes, k , que en este método debe hacerse al principio, es muy importante, pues dos soluciones, una con k factores y otra con $k+1$, pueden tener matrices de cargas factoriales muy diferentes, al contrario que en el método de componentes principales, donde los primeros k componentes son siempre iguales. Pues bien, una ventaja del método de máxima verosimilitud es que lleva asociado un test estadístico secuencial para determinar el número de factores (véase Sec. 15.3.3). Otra ventaja es que las estimaciones máximo verosímiles son invariantes ante cambios de escala; en consecuencia, las matrices de covarianzas teórica y muestral de la log-verosimilitud pueden ser sustituidas por sus homónimas de correlación sin variación alguna en los resultados. Una desventaja es que puede haber un problema de grados de libertad; o, en otros términos, el número de factores k debe ser compatible con un número de grados de libertad positivo.

El método máximo verosímil se puede implementar en **R** con la librería **pysch** y la función **fa** (con **fm=m1**). Otra posibilidad es utilizar la función **factanal**. En ambos casos hay comprobar el cumplimiento de la hipótesis de normalidad. En el ejemplo TIC no procede su implementación al no cumplirse tal hipótesis.

Otros métodos

Razones de espacio impiden comentar otros procedimientos de extracción de los factores. No obstante, hay que señalar que la función **fa** de la librería **psych** también permite implementar

los métodos (*i*) minres (mínimo residuo), que estima las cargas factoriales minimizando (sin ponderaciones) los cuadrados de los residuos no diagonales de \mathbf{R}^{res} ; parte de una estimación de k y, como el método máximo verosímil, no precisa estimar las comunidades, que se obtienen como subproducto tras la estimación de las cargas; y (*ii*) alpha, que maximiza el alpha de Cronbach para los factores. Aunque `fa` también proporciona el método del centroide o la descomposición triangular (que exigen la estimación de las comunidades, o el análisis imagen (que requiere el número de factores), en la actualidad están en desuso.

Otros métodos de extracción de los factores son los métodos de mínimos cuadrados no ponderados y mínimos cuadrados generalizados, que minimizan la suma de las diferencias cuadráticas entre las matrices de correlación observada y reproducida, en el último caso ponderando los coeficientes de correlación inversamente a la unicidad de las variables (alta unicidad supone baja communalidad). Ambos son proporcionados por `fa` y `FAiR`, que también es una librería muy recomendable.

15.3.2.2. Rotaciones en el modelo de análisis factorial

La interpretación de los factores se lleva a cabo a través de la estructura factorial , que, si los factores comunes están incorrelados, coincide con el patrón factorial. Sin embargo, aunque el modelo obtenido sea representativo de la realidad, en ocasiones la interpretación de los factores es harto difícil, porque presentan correlaciones similares con un gran número de variables. Como la solución AF no es única (si \mathbf{A} es una solución factorial, también lo es cualquier transformación ortogonal), con la rotación se trata de que cada variable tenga una correlación próxima a 1 con un factor y a 0 con el resto, facilitando la interpretación de los factores.

Geométricamente, la j -ésima fila de la matriz de cargas contiene las coordenadas de un punto (elemento, observación) en el espacio de las cargas correspondientes a X_j . Al realizar la rotación, se obtienen las coordenadas respecto a unos nuevos ejes, siendo el objetivo situarlos lo más cerca posible del mayor número de puntos. Esto asociaría cada grupo de variables con un sólo factor, haciendo la interpretación más objetiva y sencilla.

Sea \mathbf{T} una matriz ortogonal ($\mathbf{T}'\mathbf{T} = \mathbf{T}\mathbf{T}' = \mathbf{I}$), denominada *matriz de transformación*. Entonces, el modelo (15.3) puede escribirse como $\mathbf{Z} = \mathbf{ATT}'\mathbf{F} + \mathbf{U} = \mathbf{BT}'\mathbf{F} + \mathbf{U}$. Se trata de llegar a una *estructura simple*, que se caracteriza porque en \mathbf{B} :

- Cada fila tiene al menos un cero.
- Cada columna tiene, al menos, tantos ceros como factores comunes (k).
- Cada par de columnas debe ser tal que, para varias variables, una tenga cargas despreciables y la otra no.
- Si $k \geq 4$, cada par de columnas debe tener un número elevado de variables cuyas cargas sean nulas en ambas variables.
- Para cada par de columnas, el número de variables con cargas no nulas en ambas columnas debe ser muy pequeño.

Como se avanzó, se trata de que las variables se aglomeren lo más posible en torno a los factores comunes, y de la manera más discriminatoria posible. Así mejora la interpretación de éstos y, por lo general, aumenta su significado teórico.

Las rotaciones pueden ser ortogonales u oblicuas, dependiendo de si los nuevos factores siguen estando incorrelacionados (ejes perpendiculares) o no (ejes oblicuos).

15.3.2.2.1. Rotaciones ortogonales

Preservan la perpendicularidad de los ejes y no varían las comunidades, pues $\mathbf{B}\mathbf{B}' = \mathbf{A}\mathbf{T}'\mathbf{A}' = \mathbf{A}\mathbf{A}'$. Tampoco modifican los cuadrados de las comunidades ni, por tanto, la suma de sus cuadrados (para todas las variables): $\sum_{j=1}^p \sum_{m=1}^k b_{jm}^4 + 2 \sum_{m < r=1}^k \sum_{m=1}^k b_{jm}^2 b_{jr}^2$. Y como esta expresión se mantiene invariante, minimizar el segundo término implica maximizar el primero.

Las rotaciones ortogonales más usadas son:

Rotación VARIMAX

Se define *simplicidad* del factor m -ésimo como la varianza de los cuadrados de las cargas factoriales (rotadas) b_{ji} , $j = 1, 2, \dots, p$: $SMPL_m = \frac{\sum_{j=1}^p b_{jm}^4}{p} - \left(\frac{\sum_{j=1}^p b_{jm}^2}{p} \right)^2$. Cuanto mayor es la simplicidad de los factores, más sencilla es su interpretación. Por ello, el objetivo es que \mathbf{T} sea tal que se maximice la varianza del cuadrado de las cargas en cada columna del patrón factorial, es decir, en cada factor.

Dicho lo anterior, la rotación VARIMAX consiste en la obtención de una \mathbf{T} que maximice la suma de las simplicidades de todos los factores, $V = \sum_{m=1}^k SMPL_m$.¹¹

Sin embargo, las variables con mayor communalidad, y por tanto con mayores cargas factoriales, tendrán mayor influencia en la solución final, porque la communalidad no se ve afectada por la rotación ortogonal. Para evitar esto, Kaiser propuso la rotación VARIMAX normalizada¹², donde las cargas se dividen entre la raíz cuadrada de la communalidad de la variable correspondiente. Los valores obtenidos son los elementos de \mathbf{B} .

El procedimiento de cálculo de las cargas de los factores rotados es iterativo, rotándose los factores por parejas hasta que se consigue maximizar la suma de simplicidades normalizadas.

La rotación VARIMAX es muy popular por la robustez de sus resultados, si bien se recomienda para un número no muy elevado de factores comunes .

Rotación QUARTIMAX

Su objetivo es maximizar la varianza de los cuadrados de todas las cargas factoriales, es decir, maximizar $Q = \frac{\sum_{j=1}^p \sum_{m=1}^k b_{jm}^4}{pk} - \left(\frac{\sum_{j=1}^p \sum_{m=1}^k b_{jm}^2}{pk} \right)^2$.

Nótese que, como la rotación ortogonal no modifica las comunidades, $h_j^2 = \sum_{m=1}^k b_{jm}^2$, el segundo término de la expresión anterior no se verá modificado, por lo que el criterio anterior equivale a maximizar $\frac{\sum_{j=1}^p \sum_{m=1}^k b_{jm}^4}{pk}$.

¹¹Este criterio es compatible con el hecho de que, en cada fila, uno de los elementos esté próximo a cero y los demás a uno, porque la suma de los cuadrados de los elementos de una fila, es la communalidad fija de la variable correspondiente.

¹²La normalización Kaiser se aplica también en los demás tipos de rotación.

QUARTIMAX es recomendable cuando el número de factores es elevado. Tiende a generar un factor general, el primero, sobre el que la mayor parte de las variables tienen cargas elevadas, lo cual contradice los objetivos que persigue la rotación.

Rotación ORTOMAX

Es una clase general de los métodos de rotación ortogonal que se construye como una composición ponderada de las dos rotaciones anteriores: $\alpha Q + \beta V$, donde V se multiplica por p por conveniencia, ya que una constante multiplicativa no afecta a la solución. Por tanto, su objetivo es maximizar la expresión: $ORT = \sum_{m=1}^k \left(\sum_{j=1}^p b_{ji}^4 - \left(\frac{\theta}{p} \sum_{j=1}^p b_{ji}^2 \right)^2 \right)$, $0 < \theta = \frac{\alpha}{\alpha+\beta} < 1$.

Si $\theta = 1$, se tiene el criterio VARIMAX; si $\theta = 0$, se tiene el criterio QUARTIMAX; si $\theta = 0,5$, se tiene un criterio igualmente ponderado denominado BIQUARTIMAX; y si $\theta = \frac{k}{2}$, se tiene el criterio EQUAMAX, recomendado por parte de la literatura.

Nótese que QUARTIMAX pone el énfasis en la simplificación de la descripción por filas (variables) de la matriz factorial, mientras que VARIMAX lo pone en la simplificación por columnas (factores), para satisfacer los requisitos de *estructura simple*; así, aunque se pueda conseguir la simplicidad de cada variable y que, a la vez, las cargas respecto del mismo factor sean grandes, tal factor queda excluido por la restricción impuesta por la simplificación sobre cada factor (Harman, 1976).

15.3.2.2. Rotaciones oblicuas

Superan la incorrelación u ortogonalidad de los factores y se suelen aplicar cuando: (i) se sospecha que, en la población, los factores tienen una fuerte correlación y/o (ii) cierta correlación entre los factores permite una gran ganancia en la interpretación de los mismos. Podrían aplicarse siempre, como norma general, puesto que, en realidad, la ortogonalidad es un caso particular de la oblicuidad.

Los procedimientos que proporcionan soluciones con estructura simple oblicua emanan de los mismos criterios objetivos que los que proporcionan soluciones con estructura simple ortogonal. De hecho, si se relajan las condiciones de ortogonalidad, algunos procedimientos de rotación ortogonal pueden adaptarse al caso oblicuo (tal es el caso, por ejemplo, del método OBLIMAX, a partir del criterio QUARTIMAX). Por otra parte, los métodos de rotación oblicua no solo son directos, sino que también pueden introducir los principios de estructura simple que se requieren para la solución factorial primaria de forma indirecta (métodos indirectos). Las rotaciones oblicuas exigen nuevos conceptos y nueva nomenclatura:

- *Factores de referencia*, G_m , $m = 1, 2, \dots, k$: para cada factor rotado se puede encontrar un nuevo factor incorrelado con los rotados. A esos nuevos factores les llama factores de referencia. En caso de rotación ortogonal, los factores de referencia coinciden con los primeros.
- *Estructura factorial de referencia*: hasta ahora, se denominaba estructura factorial a la matriz de correlaciones entre las variables Z_j , $j = 1, 2, \dots, p$ y los factores rotados, que en el caso ortogonal coincide con la matriz de cargas factoriales rotadas. Pues bien, se denomina estructura factorial de referencia a la matriz de correlaciones entre las variables

15.3. El análisis factorial en la práctica

261

Z_j y los factores de referencia. Si la rotación es ortogonal, coincide con la estructura factorial.

- *Matriz de transformación*: en el caso oblicuo se representa por Λ .
- *Estructura factorial oblicua*: \mathbf{V} , tal que $\mathbf{V} = \mathbf{A}\Lambda$; sus elementos son v_{jm} .
- *Cargas*: en el caso oblicuo el término “carga” se utiliza para denotar la correlación de la variable con el eje de referencia: $v_{jm} = r_{Z_j; \Lambda_m}$.

Mientras las rotaciones ortogonales intentan encontrar la estructura factorial más simple, las oblicuas hacen lo mismo pero con la estructura de referencia.

El método (directo) OBLIMAX maximiza la expresión $K = \frac{\sum_{j=1}^p \sum_{m=1}^k v_{jm}^4}{(\sum_{j=1}^p \sum_{m=1}^k v_{jm}^2)^2}$. Nótese que se trata del criterio QUARTIMAX ortogonal, pero incorporando el denominador, puesto que en la rotación oblicua ya no es constante.

El QUARTIMIN directo, también derivado del QUARTIMAX ortogonal, minimiza el criterio $N = \sum_{j=1}^p \sum_{m \leq q=1}^k v_{jm}^2 v_{jq}^2$, y recibe este nombre por minimizar términos de cuarto grado.

La generalización del criterio “minimizar $H = \sum_{j=1}^p \sum_{m < q=1}^k b_{jm}^2 b_{jq}^2$ ” para factores oblicuos se denomina OBLIMIN, y da lugar a métodos indirectos. Entre ellos, destaca el COVARIMIN, que se obtiene relajando la condición de ortogonalidad en el VARIMAX, minimizando las covarianzas de los cuadrados de los elementos de \mathbf{V} : $C^* = \sum_{m \leq q=1}^k \left(p \sum_{j=1}^p v_{jm}^2 v_{jq}^2 - \sum_{j=1}^p v_{jm}^2 \sum_{j=1}^p v_{jq}^2 \right)$. La versión COVARIMIN normalizada minimiza $C = \sum_{m \leq q=1}^k \left(p \sum_{j=1}^p \frac{v_{jm}^2}{h_j^2} \frac{v_{jq}^2}{h_j^2} - \sum_{j=1}^p \frac{v_{jm}^2}{h_j^2} \sum_{j=1}^p \frac{v_{jq}^2}{h_j^2} \right)$.

Se ha comprobado empíricamente que QUARTIMIN tiende a ser demasiado oblicuo y COVARIMIN demasiado ortogonal. Una solución intermedia es la rotación BIQUARTIMIN, que consiste en minimizar $B^* = H + \frac{C^*}{p}$, donde $\frac{C^*}{p}$ es la expresión completa del COVARIMIN. Una generalización de la rotación BIQUARTIMIN es $B^* = \alpha H + \beta \frac{C^*}{p}$. Sencillas operaciones aritméticas llevan a $B^* = \sum_{m < q=1}^k \left(p \sum_{j=1}^p v_{jm}^2 v_{jq}^2 - \gamma \sum_{j=1}^p v_{jm}^2 \sum_{j=1}^p v_{jq}^2 \right)$, con $\gamma = \frac{\beta}{\alpha + \beta}$. La rotación QUARTIMIN se obtiene con $\gamma = 0$, la BIQUARTIMIN con $\gamma = 0,5$ y la COVARIMIN con $\gamma = 1$. También se pueden obtener versiones normalizadas sin más que normalizar las cargas (dividirlas por h_{jm}^2).

El criterio BINORMALMIN (normalizado) es una alternativa al BIQUARTIMIN para corregir el sesgo de oblicuidad del criterio COVARIMIN. Minimiza $D = \sum_{m < q=1}^k \left(\frac{\sum_{j=1}^p \frac{v_{jm}^2}{h_j^2} \frac{v_{jq}^2}{h_j^2}}{\sum_{j=1}^p \frac{v_{jm}^2}{h_j^2} \sum_{j=1}^p \frac{v_{jq}^2}{h_j^2}} \right)$. BINORMALMIN suele ser mejor con datos muy simples o muy complejos; BIQUARTIMIN es más recomendable con datos moderadamente complejos.

El método de rotación OBLIMIN directo, en vez de proceder como B^* , que depende de los valores de la estructura, minimiza directamente una función de la matriz del patrón factorial primario: $F(\mathbf{A}) = \sum_{m < q=1}^k \left(\sum_{j=1}^p a_{jm}^2 a_{jq}^2 - \frac{\delta}{p} \sum_{j=1}^p a_{jm}^2 \sum_{j=1}^p a_{jq}^2 \right)$. Cuando $\delta = 0$, se tiene el QUARTIMIN directo.

Hay otros tipos de transformaciones oblicuas, pero únicamente se mencionarán (*i*) la ORTOBLICUA, que llega a la solución oblicua mediante una serie de transformaciones ortogonales intermedias; y (*ii*) el la PROMAX, muy popular, que actúa alterando los resultados de una rotación ortogonal (concretamente elevando las cargas de la rotación ortogonal a una potencia entre 2 y 4) hasta crear una solución con cargas factoriales lo más próximas a la estructura ideal. Cuanto mayor es esta potencia más oblicua es la solución obtenida.

15.3.2.2.3. ¿Rotaciones ortogonales u oblicuas?

La selección del método de rotación, ortogonal u oblicua, depende del objetivo perseguido. Si se pretende reducir el número de variables originales a un conjunto mucho menor de variables incorrelacionadas para su uso posterior en otra técnica, por ejemplo regresión, la rotación debe ser ortogonal. Si el objetivo es obtener unos factores teóricos significativos, puede resultar apropiada la aplicación de una rotación oblicua.

En R es muy sencillo implementar una rotación ortogonal u oblicua. Basta, por ejemplo, con utilizar la librería GPArotation (Bernaards and Jennrich, 2005) e indicarlo en el argumento `rotate` de la función `fa`. A modo de ejemplo, extrayendo los factores por el método de los factores principales y utilizando una rotación VARIMAX normalizada, sería:

```
library(GPArotation)
af_facprin2 <- fa(cor(TIC2021), nfactors=2, rotate="varimax", fm="pa", digits=3)
```

```
af_facprin2 # el objeto contiene información adicional no relevante en estos momentos
#> Factor Analysis using method = pa
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>          PA1 PA2   h2   u2 com
#> ebroad    0.38 0.59 0.50 0.505 1.7
#> esales    0.02 0.74 0.55 0.447 1.0
#> esocmedia 0.46 0.68 0.68 0.322 1.7
#> eweb      0.50 0.75 0.82 0.182 1.7
#> hbroad    0.91 0.20 0.87 0.131 1.1
#> hiacc     0.96 0.26 0.98 0.018 1.1
#> iuse      0.72 0.60 0.88 0.125 1.9
#>
#>          PA1  PA2
#> SS loadings  2.87 2.40
#> Proportion Var 0.41 0.34
#> Cumulative Var 0.41 0.75
```

Nótese que la salida por defecto es la normalizada. También se puede utilizar la librería `stats` indicando T o F en el argumento `normalize`, dependiendo de que se quiera o no, respectivamente, una rotación VARIMAX (u otra) normalizada .

```
library(stats)
varimax(loadings(af_facprin), normalize=T)
```

En el ejemplo del uso las TIC en los países de la UE-27, la rotación VARIMAX ha conseguido facilitar la interpretación de los factores comunes, ya que, tras la rotación, las variables relacionadas con el uso de las TIC a escala individual y de hogar cargan en el primer factor, mientras que las relacionadas con el uso de las TIC a nivel empresarial cargan en el segundo. Por tanto, ambos factores pueden considerarse indicadores de la dotación y uso de las TIC en los ámbitos familiar y empresarial, respectivamente. El lector puede probar (y comparar) con otras rotaciones sin más que incluirlas en el argumento `rotate`.

15.3.3. Post-análisis factorial

Realizado el AF, los siguientes procedimientos permiten comprobar la bondad del modelo obtenido:

Análisis de las correlaciones residuales

Se entiende por bondad de la solución factorial la medida del grado en que los factores del modelo explican las correlaciones entre las variables. Por ello, parece natural que tal medida se base en la comparación entre las correlaciones observadas y las que se derivan del modelo factorial (reproducidas) o, en términos matriciales, en la magnitud de las entradas de la matriz de correlaciones residuales $\mathbf{R}^{res} = \mathbf{R} - \mathbf{R}^{rep}$, donde $\mathbf{R} = \frac{1}{N}\mathbf{Z}\mathbf{Z}'$ y $\mathbf{R}^{rep} = \mathbf{A}\Phi\mathbf{A}' = \mathbf{\Gamma}\mathbf{A}'$ (relación fundamental entre el patrón y la estructura factorial; en caso de incorrelación entre los factores, $\Phi = \mathbf{I}$ y $\mathbf{R}^{rep} = \mathbf{A}\mathbf{A}'$). La matriz \mathbf{R}^{rep} se obtiene sin más que sustituir \mathbf{Z} por \mathbf{AF} en la expresión de \mathbf{R} .

Ahora bien, ¿cuál es el criterio apropiado para concluir si una solución factorial es aceptable o no? Para que sea aceptable, los elementos (los residuos) de \mathbf{R}^{res} deben ser cercanos a cero, y como todos los factores comunes han sido considerados, se supone que no existen más vínculos entre las variables y que la distribución de dichos residuos debe ser como la de correlación cero en una muestra del mismo tamaño. Por tanto, como $\sigma_{r=0} = \frac{1}{\sqrt{N-1}}$, entonces $S_{r_{res}} \leq \frac{1}{\sqrt{N-1}}$:¹³

- Si $S_{r_{res}} \gg \frac{1}{\sqrt{N-1}}$, es razonable pensar que existen relaciones adicionales significativas entre las variables y hay que modificar la solución factorial.
- Si $S_{r_{res}} \ll \frac{1}{\sqrt{N-1}}$, es razonable pensar que la solución factorial incluye relaciones que no están justificadas.
- Si $S_{r_{res}} \leq \text{pero no } \ll \frac{1}{\sqrt{N-1}}$, la solución es aceptable.

Otra posibilidad, también muy sencilla, propuesta por [Revelle \(2022\)](#), es utilizar $fit = 1 - \frac{\sum(\mathbf{R}-\mathbf{FF}')^2}{\sum(\mathbf{R})^2}$, que indica la reducción proporcional en la matriz de correlación debida al modelo factorial. Nótese que esta medida es sensible al tamaño de las correlaciones originales. Es decir, si los residuos son pequeños, pero las correlaciones son pequeñas, el ajuste es malo. Las medidas clásicas como el RMSE (raíz cuadrada del error cuadrático medio), o similares, también son susceptibles de uso.

En el ejemplo TIC seguido en este capítulo el ajuste realizado es muy bueno:

¹³Este criterio tiene como ventaja la simplicidad. Sin embargo, sería conveniente que tuviese en cuenta, al menos, el número de variables.

```

round(af_facprin2$residual, 3)
#>      ebroad esales esocmedia  eweb hbroad hiacc  iuse
#> ebroad    0.505 -0.068     0.008  0.068 -0.045  0.002  0.003
#> esales   -0.068  0.447     0.026  0.015  0.004 -0.012  0.021
#> esocmedia  0.008  0.026     0.322 -0.047  0.014 -0.005  0.017
#> eweb      0.068  0.015    -0.047  0.182  0.012  0.015 -0.042
#> hbroad   -0.045  0.004     0.014  0.012  0.131 -0.005  0.010
#> hiacc     0.002 -0.012    -0.005  0.015 -0.005  0.018  0.002
#> iuse      0.003  0.021     0.017 -0.043  0.010  0.002  0.125
af_facprin2$rms
#> [1] 0.02907475
af_facprin2$fit
#> [1] 0.9715865

```

NOTA IMPORTANTE

Como se avanzó en la introducción, el AF está enfocado al ajuste de las correlaciones entre las variables observadas mediante el patrón factorial correspondiente al modelo (15.2) (con los factores comunes y el factor único). Pues bien, si en el proceso reproductivo se utiliza el modelo sólo con los factores comunes, la matriz de correlaciones que se reproduce es \mathbf{R} , lo que implica el modelo ACP (modelo (15.1)). Si se incluye también el factor específico, la matriz de correlaciones que se reproduce es \mathbf{R}^* (modelo AF). Si en dicha reproducción se utilizasen los factores comunes y el término de error, se reproduciría \mathbf{R} con una diagonal principal cuyas entradas serían la unidad menos las estimaciones de las communalidades.

Test de bondad de ajuste

Se trata de un contraste de razón de verosimilitudes que se puede llevar a cabo cuando se extraigan los factores por el método de máxima verosimilitud. La hipótesis nula es la suficiencia de k factores comunes para explicación de las correlaciones entre las variables originales y de la varianza que comparten.

El estadístico del contraste es $-2\ln\lambda = np(\hat{a} - \ln\hat{g} - 1]$, donde \hat{a} y \hat{g} son las medias aritmética y geométrica, respectivamente, de los autovalores de la matriz $\hat{\Sigma}_{H_0}^{-1}\mathbf{S}$. Bajo H_0 , se distribuye asintóticamente como una χ_{df}^2 , con $df = \left(p + \frac{p(p+1)}{2}\right) - \left(p + pk + p - \frac{k(k-1)}{2}\right) = \frac{1}{2}(p-k)^2 - \frac{1}{2}(p+k)$.¹⁴

Este test se aplica de manera secuencial: se formula como hipótesis nula $k = 0$. Si no se rechaza, no hay factores comunes subyacentes. Si se rechaza, se sigue con $k = 1$. Si no se rechaza $k = 1$, se concluye que el modelo con un factor es una adecuada representación de la realidad; si se rechaza, se formula la hipótesis nula de que $k = 2$, y el proceso continúa hasta que no se rechace la hipótesis nula, siempre que el valor de k sea compatible con un número de grados de libertad positivo.

¹⁴ df indica la medida en que el modelo factorial ofrece una interpretación más simple que Σ .

15.3.4. Puntuaciones factoriales

Las puntuaciones factoriales son las estimaciones de los valores de los factores aleatorios no observados, es decir, de los elementos de $\mathbf{F}_{m \times m}$. Así, \hat{f}_{im} será la estimación del valor del m -ésimo factor para la i -ésima observación (elemento, individuo, objeto...). Cuando se extraen los factores por componentes principales las puntuaciones son exactas.

Estas estimaciones pueden ser usadas como inputs para posteriores análisis (regresión, cluster, etc.) en los que se trabaje con los mismos elementos o individuos, sustituyendo las variables originales por los nuevos factores obtenidos. La cuestión es: ¿cómo calcular estas puntuaciones?, porque tanto los factores como los errores no son observables sino aleatorios.

Los métodos más populares para obtener la estimación de las puntuaciones factoriales son:

- El de regresión por mínimos cuadrados ordinarios (MCO), donde $\hat{\mathbf{F}} = (\mathbf{A}'\mathbf{A})^{-1} \mathbf{A}'\mathbf{Z}$.
- El de Bartlett, basado en el método de estimación por mínimos cuadrados generalizados (MCG), con $\hat{\mathbf{F}} = (\mathbf{A}'\Psi^{-1}\mathbf{A})^{-1} \mathbf{A}'\Psi^{-1}\mathbf{Z}$. El mismo estimador se puede obtener por máxima verosimilitud asumiendo normalidad multivariante.
- El de Thompson (con un enfoque bayesiano), donde $\hat{\mathbf{F}} = (\mathbf{I} + \mathbf{A}'\Psi^{-1}\mathbf{A})^{-1} \mathbf{A}'\Psi^{-1}\mathbf{Z}$.
- El de Anderson-Rubin (que obtiene estimaciones MCG imponiendo la condición $\mathbf{F}'\mathbf{F} = \mathbf{I}$ ($\hat{\mathbf{F}} = (\mathbf{A}'\Psi^{-1}\mathbf{R}\Psi^{-1}\mathbf{A})^{-1} \mathbf{A}'\Psi^{-1}\mathbf{Z}$)).

Las ventajas y desventajas de cada uno de ellos pueden verse en [Mardia et al. \(1979a\)](#) y [De la Fuente \(2011\)](#).

En el ejemplo de las TIC, las puntuaciones de los dos factores extraídos con el método de los factores principales y rotados con VARIMAX (la rotación no afecta a las puntuaciones), calculadas por el método de regresión, para los países de la UE-27 (se muestran los de Bélgica, Bulgaria y la República Checa), se obtienen en **R** como sigue:

```
af_facprin3 <- fa(cor(TIC2021), nfactors=2, rotate="VARIMAX", fm="pa",
  scores="regression")
factor.scores(TIC2021, af_facprin3)$scores[1:3,]
#>          PA1          PA2
#> BE  0.6256359  1.01289866
#> BG -2.1820404 -0.03439974
#> CZ -0.2189723  1.08635525
```

15.4. Relaciones y diferencias entre el AF y el ACP

ACF y AF son aparentemente muy similares, pero en realidad son muy diferentes. Tanto ACP como AF son técnicas de reducción de la dimensionalidad que aparecen juntas en los paquetes estadísticos y persiguen objetivos muy similares, lo cual, en determinadas ocasiones, lleva al

lector a pensar que son intercambiables entre sí, cuando ello no es cierto. Por ello, este capítulo finaliza con un breve comentario sobre las diferencias más relevantes entre ambos enfoques.

La primera es que ACP es una mera transformación de los datos en la que no se hace ningún supuesto sobre la matriz de covarianzas o de correlaciones. Sin embargo, AF asume que los datos proceden de un modelo bien definido, el modelo (15.3), en el que los factores subyacentes satisfacen unos supuestos bien definidos.

En segundo lugar, en ACP el énfasis se pone en el paso desde las variables observadas a las componentes principales, mientras que en AF se pone en el paso desde los factores latentes a las variables observadas. Es cierto que en ACP se pueden retener k componentes y a partir de ellas aproximar (reproducir) las variables observadas; sin embargo, esta manera de proceder parece menos natural que la aproximación de las variables observadas en términos de los factores comunes y, además, al no tener en cuenta la unicidad de las variables, sobreestima las cargas factoriales y la dimensionalidad del conjunto de variables originales.

Una tercera diferencia es que, mientras que ACP obtiene componentes en función de las variables originales (los valores de las variables pueden ser estimados a posteriori en función de dichas componentes o factores), en AF las variables son, ellas mismas, combinaciones lineales de factores desconocidos. Es decir, mientras que en ACP la solución viene de la mano de la descomposición en valores singulares, en AF requiere procedimientos de estimación, normalmente iterativos.

La cuarta es que ACF es un procedimiento cerrado mientras que AF es abierto, en el sentido de que explica la varianza común y no toda la varianza.

Finalmente, como pudo verse en 15.3.2.1, cuando las varianzas de los factores únicos son prácticamente nulas, el método de los factores principales es equivalente a ACP, y cuando son pequeñas ambos dan resultados similares. Sin embargo, cuando son grandes, en ACP las componentes principales (tanto las retenidas como las que no se retienen) las absorben, mientras que el AF las considera y les da su lugar.

RESUMEN

El Análisis Factorial es una técnica de reducción de la dimensionalidad que trata de dar una explicación de la varianza compartida, o común, de las variables objeto de estudio (no de toda la varianza, como hace el análisis de componentes principales) mediante un número mucho menor de factores comunes latentes. Por consiguiente, solo tiene sentido implementarlo si dichas variables se encuentran fuertemente correlacionadas. Tras introducir al lector en los principales elementos teóricos del Análisis Factorial (el modelo básico y la solución factorial completa), se abordan las distintas etapas del procedimiento en su vertiente práctica: (i) el pre-análisis factorial, que responde a la pregunta de si procede o no llevarlo a cabo; (ii) el análisis factorial propiamente dicho, prestando especial atención a los métodos de extracción de los factores y a las rotaciones de los mismos para facilitar su interpretación; y (iii) el post-análisis factorial, que incluye una serie de procedimientos para determinar si la solución factorial obtenida es o no aceptable. Posteriormente, se aborda la cuestión de cómo estimar los valores de los factores obtenidos para cada elemento o individuo involucrado en el análisis, pues estas estimaciones pueden usarse como inputs en análisis posteriores (regresión, cluster, etc.) sustituyendo las variables originales por los factores obtenidos. El capítulo finaliza con algunos comentarios sobre las diferencias entre el análisis factorial y el de componentes principales, aparentemente muy similares, pero en realidad muy diferentes.

Capítulo 16

Escalamiento multidimensional

José Luis Alfaro Navarro^a y Manuel Vargas Vargas^a

^a Universidad de Castilla-La Mancha

16.1. Introducción

El escalado multidimensional (EMD) fue propuesto por primera vez a la Universidad de Princeton por Warren S. Torgerson a principios de la década de 1950 siendo un investigador importante en este campo Joseph Bernard Kruskal. El EMD engloba una variedad de técnicas multivariadas cuya finalidad es obtener la estructura (factores o dimensiones) de los individuos (o variables) subyacente a una matriz de datos empíricos, lo que se consigue al representar dicha estructura en una forma geométrica bi o tridimensional.

Por tanto, la idea del EMD es **representar los datos en baja dimensión** (usualmente 2 dimensiones) utilizando la información proporcionada por las distancias entre los datos. Esta técnica surge ya que cada vez con más frecuencia los datos particulares de los que se dispone y el objetivo del análisis hacen difícil su tratamiento con las medidas clásicas, por lo que se han ido diseñando nuevas medidas de distancia entre datos. Estas medidas se pueden utilizar para diferentes tareas: agrupamiento de casos, clasificación, detección de patrones o dimensiones subyacentes, recuperación de información, etc. Por lo tanto, EMD aborda algunas problemáticas que pueden ser analizadas con otras técnicas como, por ejemplo, análisis de componentes principales o factorial cuando el objetivo es representar muchas variables en pocas dimensiones mediante la identificación de la estructura interna de los datos, dimensiones o factores en base a la matriz de correlaciones como medida de proximidad entre las variables o el análisis cluster cuando el objetivo es analizar la proximidad entre los objetos, personas, productos, etc. estudiados.

El EMD analiza matrices de proximidad (similitud, disimilitud o distancia), por ello, es una alternativa más flexible que otros métodos multivariantes con los que comparte objetivos, ya que sólo requiere de una matriz con las proximidades entre los datos, que pueden representar

valoraciones personales, grado de acuerdo entre juicios, parecido entre objetos, frecuencias de aparición de rasgos, diferencias entre tratamientos, etc. La idea central es que las distancias que median entre los puntos se corresponden con las proximidades entre los objetos por medio de una función de ajuste resultante de un proceso iterativo de optimización, pudiéndose describir las relaciones entre los objetos sobre la base de las proximidades observadas ([López-González and Hidalgo-Sánchez, 2010](#))

En **R**, existen distintas funciones para desarrollar el EMD, desde las clásicas funciones `cmdscale()` del paquete **base** e `isoMDS()` del paquete **MASS** hasta el enfoque más actual, usado en este documento, recogido en el paquete **smacof** ([de Leeuw and Mair, 2009; Mair et al., 2022](#)) que proporciona al usuario una gran flexibilidad para especificar EMD. Utiliza siempre matrices de disimilitud y, desde la primera versión, se han implementado varios enfoques adicionales de EMD y despliegue, así como varias extensiones y funciones de utilidad.

A modo de ejemplo se va a usar la información relacionada con 7 variables de la sociedad de la información disponibles para 27 países europeos en la base de datos **TIC2021**, cuatro relacionadas con el uso de las TIC por parte de las empresas y tres de aspectos relacionados con el uso por parte de las personas y la equipación de los hogares. Dicha información, así como la descripción de las variables, puede consultarse en la base de datos **TIC2021** del paquete **CDR**.

16.2. Medición de distancias y similitudes

Tanto para el EMD como para muchas otras técnicas multivariantes, el concepto de distancia, entendida como medida de diferenciación entre objetos, constituye la base fundamental de la obtención y presentación de sus resultados. También son frecuentes los conceptos de “disimilitud”, muy parecido al de distancia, o de “similaridad”, dual en su sentido al de distancia. Se nombre como se nombre, la característica que hay que tener siempre presente es si la medida indica “alejamiento” entre los objetos (distancia o disimilitud) o “cercanía” (similaridad o proximidad).

Básicamente, se considera una medida de distancia a una función que asigna a cada par de objetos (o_i y o_j), que pueden contener mediciones de variables x e y, un número real, $d(o_i, o_j) = \delta_{ij}$, que debe cumplir las siguientes condiciones (para un análisis más detallado véase Sec. [12.3](#)):

- a) No negatividad $\delta_{ij} \geq 0$
- b) Simetría, $\delta_{ij} = \delta_{ji}$
- c) Identificación del objeto, $\delta_{ii} = 0$

Si además es semidefinida positiva y cumple la desigualdad triangular se dice que δ es una distancia métrica.

Aunque no se va a profundizar en ello, existen diferencias matemáticas en los requisitos que debe cumplir una medida para ser considerada una distancia o una distancia métrica, así como las condiciones para ser considerada una similaridad. Básicamente, se considera una medida de similaridad a una aplicación que asigna a cada par de objetos (o_i y o_j) un número real, s_{ij} ,

16.3. *Modelo de escalamiento multidimensional*

271

que cumple las mismas condiciones que la distancia salvo la condición c para la que tiene que cumplir que $s_{ij} \leq s_{ii}$.

Esta condición es más difícil de cumplir por lo que se emplean mucho más las medidas de distancia al ser más sencillo formular la propiedad c pues simplifica mucho el poder atribuir un valor de referencia cero para definir la distancia de un individuo a sí mismo. La similitud carece de este valor de referencia, siendo posible que la similitud de un individuo a sí mismo sea diferente de unos a otros. A pesar de esta dificultad, las medidas de similitud surgen de modo natural en muchos problemas relacionados con valoraciones subjetivas de similitud.

Para un conjunto finito de objetos, la **matriz de similaridad** es:

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{pmatrix} \quad (16.1)$$

El paso de una medida de similaridad (s_{ji}) a una distancia (δ_{ij}) se puede hacer de diversas formas. Las más usuales son:

- a) $\delta_{ij} = 1 - s_{ij}$
- b) $\delta_{ij} = \sqrt{1 - s_{ij}}$
- c) Si los valores de la diagonal de \mathbf{S} no son la unidad, $\delta_{ij} = \sqrt{s_{ii} + s_{jj} - 2s_{ij}}$

En general, cuando las características que se miden sobre los objetos son variables cuantitativas p -dimensionales, las distancias más usadas son la euclídea, la city-block, la de Minkowski o la de Mahalanobis (véase Sec. 12.3).

Cuando las variables son binarias (0 y 1), los coeficientes de similaridad más utilizados son el coeficiente de Jaccard o el de Sokal-Sneath; por último, en el caso más general en el que existan variables cuantitativas, binarias y/o cualitativas, se suele utilizar la distancia de Gower (véase Sec. 12.3).

Como se aprecia, las características de los datos que se quieren analizar influyen determinantemente en qué tipo de medida de proximidad utilizar. A su vez, la elección de una medida concreta puede modificar la configuración de los datos y, consecuentemente, los resultados de los análisis que se hagan a partir de ellos.

16.3. *Modelo de escalamiento multidimensional*

El EMD parte de una matriz de proximidades entre n objetos:

$$\Delta_{n \times n} = \begin{pmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & \cdots & \delta_{nn} \end{pmatrix} \quad (16.2)$$

y busca una representación de los n objetos en un espacio de menor dimensión, m , donde x_{ij} es la coordenada del objeto i en la dimensión j :

$$\mathbf{X}_{n \times m} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix} \quad (16.3)$$

de forma que se puede calcular la distancia euclídea entre cada par de objetos:

$$d_{ij} = \sqrt{\sum_{t=1}^m (x_{it} - x_{jt})^2} \quad (16.4)$$

y, construir una matriz de **distancias “reproducidas”**

$$\mathbf{D}_{n \times n} = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \quad (16.5)$$

que aproxime, en la medida de lo posible, a la matriz de proximidades, Δ .

El concepto básico del EMD es que las distancias entre los objetos en la configuración X , d_{ij} deben corresponder a las proximidades originales, δ_{ij} mediante una transformación, $d_{ij} = f(\delta_{ij})$, donde f es de algún tipo determinado.

En la práctica, no se suele encontrar un ajuste perfecto, por lo que existe un cierto grado de error. Por ello, se define el **Stress de Kruskal** como una medida de la bondad de ajuste del modelo:

$$Stress = \sqrt{\frac{\sum (f(\delta_{ij}) - d_{ij})^2}{\sum d_{ij}^2}} \quad (16.6)$$

En caso de un ajuste perfecto, el Stress sería 0, aumentando conforme más grandes sean los errores (diferencias entre las distancias “reproducidas” y las originales). Así, la solución proporcionada por el EMD será “mejor” cuanto más pequeña sea la medida del Stress. También es frecuente utilizar una variante, llamada **S-Stress**, definida como:

$$S - Stress = \sqrt{\frac{\sum (f(\delta_{ij})^2 - d_{ij}^2)^2}{\sum (d_{ij}^2)^2}} \quad (16.7)$$

Otra medida que se suele utilizar como grado de ajuste es el coeficiente de **correlación al cuadrado** (RSQ), que indica la proporción de variabilidad de los datos explicada por el modelo:

$$RSQ = \frac{[\sum (d_{ij} - d_{..})(f(\delta_{ij}) - f(\delta_{..}))]^2}{[\sum (d_{ij} - d_{..})^2][\sum (f(\delta_{ij}) - f(\delta_{..}))^2]} \quad (16.8)$$

Este valor oscila entre 0 y 1, y se interpreta de forma contraria a las medidas de Stress: mientras mayor sea el RSQ, mejor ajuste del modelo.

16.4. Tipos de escalamiento multidimensional

La elección de la función f que relaciona las proximidades originales y las distancias “reproducidas” produce dos tipos básicos de EMD, el EMD métrico (o clásico) y el EMD no métrico . En el primero, se considera que los datos están medidos en escala de intervalo o de razón y existe una relación funcional entre las distancias originales y las reproducidas; mientras que el segundo se suele aplicar cuando los datos están en escala ordinal o no se asume una relación funcional entre las distancias originales y las reproducidas, sino que sólo se conserva su ordenación.

16.4.1. Escalado multidimensional métrico

En el modelo de escalamiento métrico se asume que la relación entre las proximidades y las distancias es de tipo lineal, $d_{ij} = a + b\delta_{ij}$. De esta forma, se conserva la métrica de distancia original, entre puntos, lo mejor posible, siendo adecuado para el caso de variables cuantitativas. También se conoce como EMD clásico o como análisis de coordenadas principales.

En el ejemplo introductorio, se va a aplicar un EMD métrico con un doble objetivo: por un lado se usa la matriz de correlaciones entre la variables con el objetivo de analizar la similitud entre las mismas y, por otro lado, se determina la distancia entre observaciones con el objetivo de analizar la similitud existente entre observaciones, en este caso los países europeos.

En el primer caso, los “objetos” son las siete variables de la base de datos TIC2021 y se ha utilizado como medida de proximidad (similitud) el coeficiente de correlación entre las variables, por lo que se plantea un EMD métrico.

```
library('smacof')
library('CDR')
correlacion<- cor(TIC2021)
round(correlacion[1:3,],3)
#>           ebroad esales esocmedia eweb hbroad hiacc iuse
#> ebroad     1.000  0.377   0.587 0.704  0.422 0.521 0.632
```

```
#> esales    0.377  1.000    0.542 0.585  0.167 0.195 0.479
#> esocmedia 0.587  0.542    1.000 0.698  0.567 0.609 0.756
```

Los pasos a seguir son:

- Calcular la matriz de disimilaridades sobre la que actúa `smacof()`. En este ejemplo, se usa la matriz de correlaciones que se debe convertir en matriz de disimilaridades, mediante la función `sim2diss()`.

```
datos<-sim2diss(correlacion,method="corr",to.dist=TRUE)
```

La conversión de similitudes (correlaciones) en disimilaridades se ha hecho por el método “corr”, que utiliza la expresión general $\delta_{ij} = 1 - s_{ij}$. Existen otros métodos en la función `sim2diss()` para cuando la matriz de proximidades no sea de correlaciones. El argumento `to.dist=TRUE` permite convertir el resultado en un objeto de la clase `dist`.

- Una vez que disponemos de la matriz de disimilaridades, aplicamos el EMD métrico mediante la función `mds()`, versión equivalente a la función `smacofSym()`.

```
res<-mds(datos,ndim=2,type="ratio")
res
#>
#> Call:
#> mds(delta = datos, ndim = 2, type = "ratio")
#>
#> Model: Symmetric SMACOF
#> Number of objects: 7
#> Stress-1 value: 0.161
#> Number of iterations: 42
```

La Fig. 16.1 que representa los objetos según sus distancias “reproducidas” muestra la configuración final de los siete objetos:

```
plot(res)
```

La información numérica detallada se podría obtener con la información de la salida dada por: `res$conf` que mostraría las coordenadas de los objetos en las dos dimensiones; `res$confdist` que muestra la matriz de distancias reproducidas; `res$stress` para obtener la medida de stress de Kruskal y `res$spp` con la contribución porcentual de cada objeto al stress.

En este caso los resultados muestran que la medida de stress es razonablemente baja con un valor de 0.16, indicando una buena “reproducción” de las proximidades originales. Además, la “contribución” relativa al stress de cada uno de los objetos (delitos) es bastante homogénea, siendo la variable porcentaje de empresas con banda ancha (EBROAD) la que más contribuyen al stress y el nivel de acceso a internet de los hogares (HIACC), la que menos. Para ver gráficamente el grado de ajuste, se usa el gráfico de Shepard (Fig. 16.2) que compara las proximidades originales y las distancias obtenidas:

16.4. Tipos de escalamiento multidimensional

275

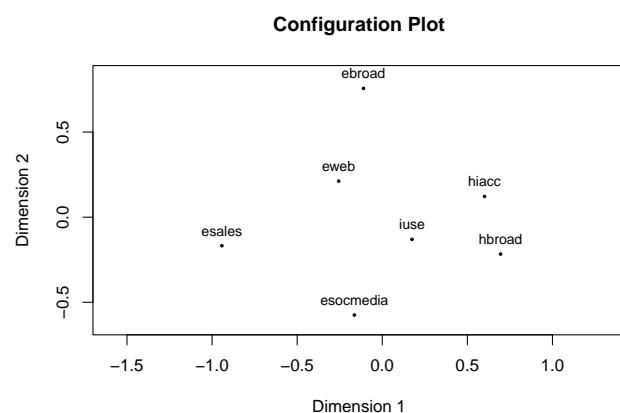


Figura 16.1: Gráfico de objetos en el plano de las distancias reproducidas.

```
plot(res, plot.type="Shepard")
```

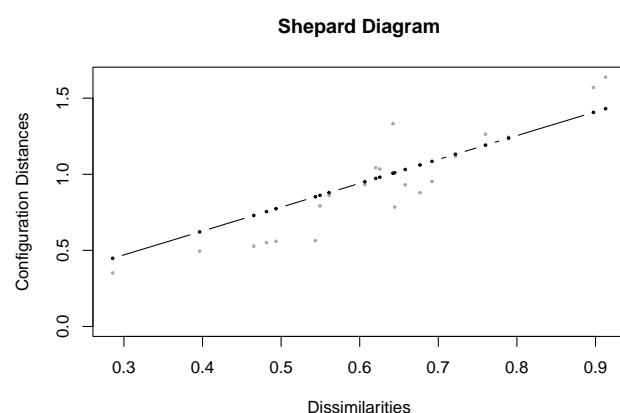


Figura 16.2: Gráfico de Shepard con método ‘ratio’.

El diagrama de Shepard incluye las proximidades originales entre pares de objetos (en gris claro) y las obtenidas por el EMD (en negro). También representa el método elegido; en este ejemplo, al usar el argumento `method="ratio"` estamos imponiendo una relación proporcional entre ambos tipos de similitudes, por lo que aparece una recta (que pasa por el origen). Dadas las diferencias que se ven en el gráfico, quizás la elección del método `ratio` (opción por defecto), no sea la más adecuada. Probando con el método `interval`, que impone una relación lineal entre ambos tipos de similitudes (recta que no tiene que pasar necesariamente por el origen), se obtendría la Fig. 16.3:

```
res2<-mds(datos,type="interval")
plot(res2,plot.type="Shepard")
```

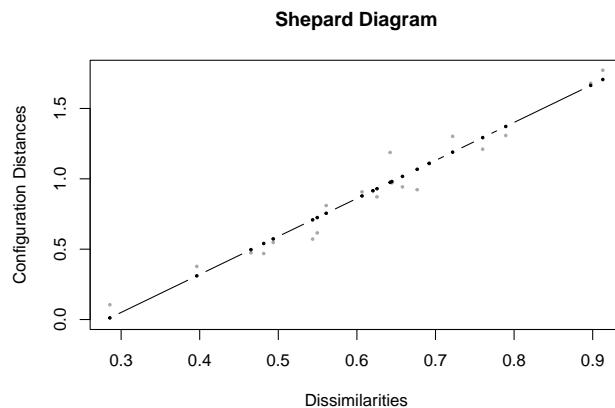


Figura 16.3: Gráfico de Shepard con método ‘interval’.

La medida de stress se ha reducido a la mitad, 0.087, indicando mejor ajuste, y el gráfico de Shepard muestra más concordancia entre las proximidades originales de los pares de objetos y las distancias reproducidas. Otra opción sería usar el método `mspline`, pero en este caso las diferencias son menores. Los tres métodos son métricos, puesto que se fija una forma funcional para relacionar las proximidades originales y las disimilitudes del modelo (un ratio, una función lineal o una función spline).

- Por último, para “interpretar” el sentido de las dimensiones en las que se representan los objetos, se recurre a ver cuáles están en los extremos. En la parte izquierda de la dimensión 1 están las variables relacionadas con el uso en las empresas mientras que en la parte derecha están las relacionadas con los hogares y las personas; se podría decir, entonces, que es una dimensión relacionada con el ámbito de uso de las TIC. En la dimensión 2, con menos distancias, y una interpretación menos clara, aparecen en la parte superior las variables relacionadas con el tipo de conexión y la existencia de web en las empresas y en la parte inferior las relacionadas con las redes sociales, las ventas o la frecuencia de uso de internet por parte de los individuos; se podría decir, **que es una dimensión asociada al uso dado a las TIC**.

Los pasos para desarrollar el mismo análisis agrupando países, observaciones en lugar de variables, serían similares pero usando la matriz de distancias en lugar de la matriz de correlaciones, por lo tanto:

```
library('factoextra')
d_euclidea <- get_dist(x = TIC2021, method = "euclidea")
res3<-mds(d_euclidea,ndim=2,type="ratio")
res3
#>
```

16.4. Tipos de escalamiento multidimensional

277

```
#> Call:
#> mds(delta = d_euclidea, ndim = 2, type = "ratio")
#>
#> Model: Symmetric SMACOF
#> Number of objects: 27
#> Stress-1 value: 0.112
#> Number of iterations: 77
```

```
plot(res3)
```

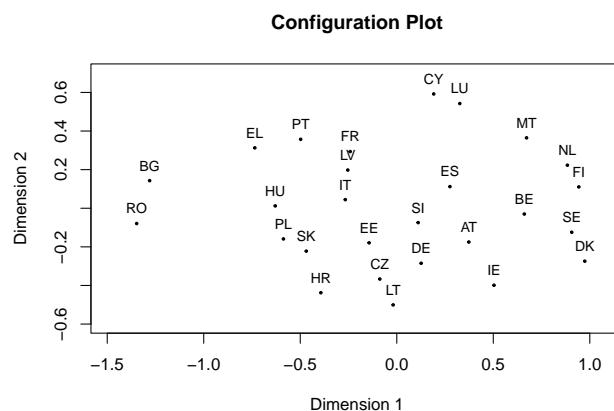


Figura 16.4: Gráfico de países sobre el plano de las distancias reproducidas.

En la Fig. 16.4, la interpretación de la dimensión 1 muestra la existencia de una diferencia clara entre los países del norte y los de última adhesión a la Unión Europea mientras que en la dimensión 2, con unas diferencias menores, aparecen en la parte superior los países de Chipre y Luxemburgo y en la parte inferior Lituania, Croacia, Irlanda y la República Checa.

16.4.2. Escalado multidimensional no métrico

En el modelo de escalamiento no métrico (también conocido como EMD ordinal) no se asume ninguna fórmula métrica que relacione las proximidades originales con las distancias reproducidas, sino que sólo describe un patrón creciente (o decreciente) entre ellas. No es significativo el valor de distancia, sino su relación con las distancias entre otros pares de objetos, por lo que construye distancias ajustadas que están en el mismo orden de rango que la proximidad original. Por ejemplo, si la distancia de los objetos separados A y B ocupa el tercer lugar en los datos de proximidades originales ordenadas, entonces también debería ocupar el tercer lugar en los datos de distancias reproducidas ordenadas. Así, el EMD no métrico busca conservar, no tanto los valores de las proximidades originales, sino la ordenación de los objetos en función de

dichas proximidades; es, por tanto, un modelo que se ajusta mejor a datos cualitativos que el métrico, aunque también se utiliza cuando se busca mayor flexibilidad en el ajuste.

Se va a aplicar un MDS no métrico a la matriz de correlaciones de las tasas anuales de siete delitos en 50 estados de EE.UU. (asesinatos, violaciones, robos, asaltos, allanamiento de morada, hurtos y sustracciones de vehículos). Los datos están disponibles en la librería `smacof` bajo el nombre de `crimes`:

```
data("crimes")
```

En este ejemplo, los “*objetos*” son los siete tipos de delito, y se ha utilizado como medida de proximidad (similitud) el coeficiente de correlación entre las tasas de delito (variables cuantitativas).

- El primer paso consiste en calcular la matriz de disimilaridades sobre la que actúa `smacof`, utilizando la función `sim2diss()`.

```
options(digits=3)
data<-sim2diss(crimes,method="corr",to.dist=FALSE)
data
#>           Murder Rape Robbery Assault Burglary Larceny Auto.Theft
#> Murder      0.000 0.693  0.812  0.436   0.849   0.970    0.943
#> Rape        0.693 0.000  0.671  0.548   0.566   0.632    0.748
#> Robbery     0.812 0.671  0.000  0.663   0.616   0.748    0.616
#> Assault     0.436 0.548  0.663  0.000   0.693   0.825    0.819
#> Burglary    0.849 0.566  0.616  0.693   0.000   0.447    0.548
#> Larceny     0.970 0.632  0.748  0.825   0.447   0.000    0.671
#> Auto.Theft  0.943 0.748  0.616  0.819   0.548   0.671    0.000
```

La conversión de similitudes (correlaciones) en disimilaridades se ha hecho por el método `corr`, que utiliza la expresión general $\delta_{ij} = \sqrt{1 - s_{ij}}$. Existen otros métodos en la función `sim2diss()` para cuando la matriz de proximidades no sea de correlaciones. El argumento `to.dist=TRUE` permite convertir el resultado en un objeto de la clase `dist` si fuese necesario.

Frente a los métodos métricos, que fijan una forma funcional para relacionar las proximidades originales y las disimilaridades del modelo (un ratio, una función lineal o una función spline), una alternativa más flexible es asumir una relación no métrica, donde sólo se conserve la ordenación de las proximidades originales. En este caso, se busca que las distancias reproducidas ordenen a los pares de objetos de forma idéntica a la original.

Para ello, se utiliza el método `ordinal` dentro de la función `mds()`:

```
res4<-mds(data,ndim=2,type="ordinal")
res4
#>
#> Call:
#> mds(delta = data, ndim = 2, type = "ordinal")
```

16.4. Tipos de escalamiento multidimensional

279

```
#>
#> Model: Symmetric SMACOF
#> Number of objects: 7
#> Stress-1 value: 0.002
#> Number of iterations: 15
```

Como se aprecia, la medida de stress es muy baja (0.002), indicando un muy buen ajuste, que resulta significativo como indica la Fig. 16.5 basada en la función `permtest()`:

```
ptestnm<-permtest(res4,nrep=50)
```

```
plot(ptestnm)
```

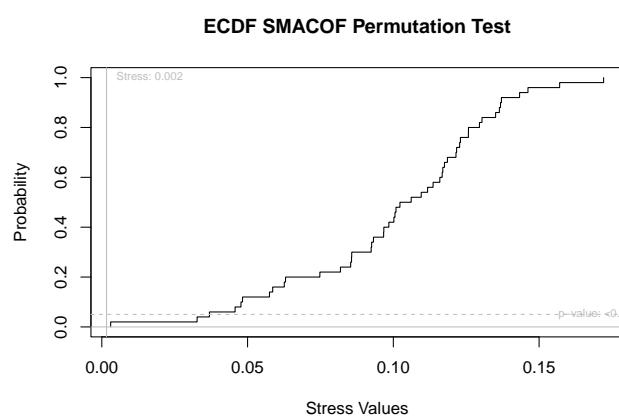


Figura 16.5: Test de permutaciones para evaluación de la significatividad de la medida de stress.

El gráfico de Shepard, representado en la Fig. 16.6, incluye las proximidades originales entre pares de objetos (en gris claro) y las obtenidas por el MDS (en negro), mostrando una alta concordancia entre las ordenaciones original y reproducida:

```
plot(res4,plot.type="Shepard")
```

La contribución porcentual de cada delito a la medida de stress y las coordenadas bidimensionales reproducidas serían:

```
res4$spp #Contribución porcentual de cada objeto al stress
#>      Murder      Rape     Robbery    Assault   Burglary    Larceny Auto.Theft
#>     8.064     6.443    39.270     0.104    10.618     9.023   26.478
res4$conf #Coordenadas de los objetos en dos dimensiones
```

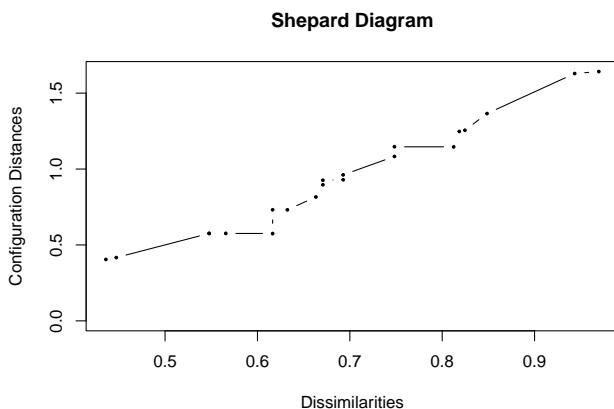


Figura 16.6: Gráfico de Shepard: concordancia entre las ordenaciones original y reproducida.

```
#>           D1      D2
#> Murder    -0.9673  0.01136
#> Rape      -0.1225 -0.37592
#> Robbery    0.0496  0.53424
#> Assault   -0.5630 -0.00483
#> Burglary   0.3925 -0.11326
#> Larceny    0.6015 -0.47400
#> Auto.Theft 0.6093  0.42240
```

El delito de robo (39.27 %) y el de sustracción de vehículos (26.48 %) son responsables del 65.75 % del stress, seguidos muy de lejos por los otros cinco tipos de delito.

La Fig. 16.7 muestra la representación bidimensional de la configuración final de los siete delitos según sus distancias “reproducidas”:

```
plot(res4)
```

Por último, para “interpretar” el sentido de las dimensiones en las que se representan los objetos, se recurre a ver cuáles están en los extremos. En la parte izquierda de la dimensión 1 están los delitos de asesinato y asalto, mientras que en la parte derecha están los de hurto, sustracción de vehículos y allanamiento; se podría decir, entonces, que es una dimensión relacionada con el grado de “personalización” del delito: los que afectan a personas directamente frente a los que no. La dimensión 2, con menos distancias (véase las escalas) y de más difícil interpretación, contrapone en la parte superior los delitos que más “beneficios” económicos producen (robos, sustracción de vehículos) frente a los que menos (violación o hurto); se podría decir, entonces, que es una dimensión asociada a la repercusión económica del delito.

Por último, destacar que puede ser interesante analizar la estabilidad de las soluciones del EMD (bien mediante jackknife, bootstrap, o elipses de pseudo-confianza). También puede interesar

16.4. Tipos de escalamiento multidimensional

281

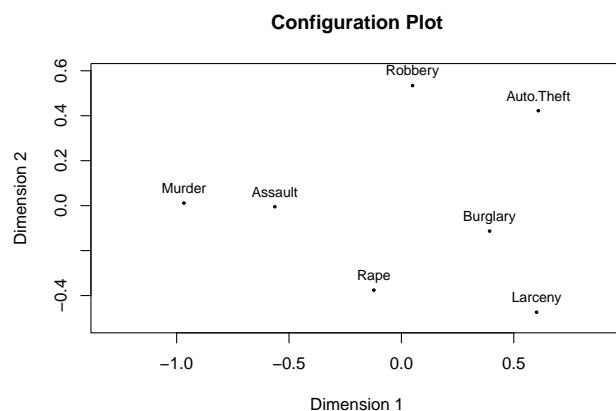


Figura 16.7: Representación bidimensional de las disimilitudes entre los siete tipos de delito.

plantear un modelo de EMD que permita valorar las diferencias individuales (abordable con la función `smacofIndDiff()`). Otra alternativa puede ser abordar un desplegamiento multidimensional, que representa conjuntamente objetos e individuos.

Resumen

La aplicación del análisis EMD implica tres pasos consecutivos:

- La determinación de las proximidades originales entre los objetos. Esta fase depende de las características de los objetos y del tipo de relación que se quiera/pueda establecer entre ellos. Actualmente, **R** dispone de paquetes que permiten estimar las matrices de proximidad a partir de los datos en bruto.
- La conversión de las proximidades en similaridades (si fuese necesario) y el ajuste entre las originales y las reproducidas por el EMD. Se debe elegir el tipo de EMD a utilizar, que depende de la función de ajuste: se puede optar por funciones de tipo `ratio`, `interval` o `mspline` (EMD métricos) o `ordinal` (EMD no métrico). La elección estará relacionada con el tipo de datos usados y el grado de ajuste (stress) de los modelos.
- La interpretación de los resultados a partir de la configuración obtenida, tanto del significado de las dimensiones como de la estructura de los objetos (cuáles se parecen, si existen grupos, etc.)

Capítulo 17

Análisis de correspondencias

Román Mínguez Salido^a y Manuel Vargas Vargas^a

^a Universidad de Castilla-La Mancha

17.1. Introducción

El **análisis de correspondencias** es un método gráfico descriptivo de reducción de la dimensión incluido entre los algoritmos de aprendizaje no supervisado. La idea principal es equivalente al método de componentes principales, pero aplicado a variables cualitativas. El objetivo es representar los valores (**niveles** en **R**) de variables cualitativas (**factores** en **R**) en ejes cuantitativos cuyas coordenadas representen la cercanía o lejanía entre los niveles de los factores. Es decir, es un método de reducción de la dimensionalidad para factores representables en pocas dimensiones.

Por sencillez, el punto de partida será una **tabla de contingencia RxC**, T , (véase Cap. 5) que recoge la frecuencia de cada par de niveles A_1, A_2, \dots, A_R del factor A y B_1, B_2, \dots, B_C del factor B :

Tabla 17.1: Ejemplo de tabla de contingencia RxC

	B_1	B_2	...	B_C	Total
A_1	n_{11}	n_{12}	...	n_{1C}	$n_{1\cdot}$
A_2	n_{21}	n_{22}	...	n_{2C}	$n_{2\cdot}$
...
A_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R\cdot}$
Total	$n_{\cdot 1}$	$n_{\cdot 2}$...	$n_{\cdot C}$	N

Cada fila representa el **perfil condicional** del nivel A_i , siendo la última el **perfil marginal**

del factor A . Igualmente, cada columna representa el **perfil condicional** del nivel B_j , siendo la última el **perfil marginal** del factor B .

Como se vió en el Cap. 5, si los factores fueran independientes, el **valor esperado** en cada casilla sería $E_{ij} = \frac{n_{i\cdot}n_{\cdot j}}{N}$, por lo que la diferencia tipificada, $r_{ij} = \frac{n_{ij} - E_{ij}}{\sqrt{E_{ij}}}$ es una medida de asociación entre las modalidades A_i y B_j . La matriz formada por estos “residuos estandarizados” (véase sección 23.5.4), $R = \{r_{ij}\}$ resume la asociación entre los atributos, y es el objetivo básico del análisis de correspondencias; básicamente, se realiza una proyección de las filas y columnas de la tabla de frecuencias relativas (transformadas) para obtener las coordenadas en ejes cuantitativos, representables en la forma habitual como diagramas de puntos.

Para un estudio en profundidad de esta técnica pueden consultarse [Greenacre \(2008\)](#) (en español) o [Beh and Lombardo \(2014\)](#). En el resto del capítulo se hará una breve exposición de la metodología y se exemplificará con el análisis de dos tablas de contingencia.

17.2. Metodología del análisis de correspondencias

Dada una tabla de contingencia T , a partir de las frecuencias observadas n_{ij} , se definen las **distancias** entre los perfiles:

- para los perfiles fila, $d_{ii'} = \sum_{k=1}^C \frac{1}{n_{i\cdot k}} \left(\frac{n_{ik}}{n_{i\cdot}} - \frac{n_{i'k}}{n_{i'\cdot}} \right)^2$
- para los perfiles columna, $d_{jj'} = \sum_{k=1}^R \frac{1}{n_{\cdot k}} \left(\frac{n_{kj}}{n_{\cdot j}} - \frac{n_{kj'}}{n_{\cdot j'}} \right)^2$

Estas distancias aumentan cuanto más se “*diferencien*” unos perfiles de otros. El análisis de correspondencias busca construir **dimensiones** (habitualmente, dos) y obtener las coordenadas de los niveles de ambos factores en dichas dimensiones

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}'_1 \\ \vdots \\ \mathbf{a}'_R \end{pmatrix}, \text{ con } \mathbf{a}_i = (a_{i1} \ a_{i2})', \text{ y } \mathbf{B} = \begin{pmatrix} \mathbf{b}'_1 \\ \vdots \\ \mathbf{b}'_C \end{pmatrix}, \text{ con } \mathbf{b}_j = (b_{j1} \ b_{j2})' \quad (17.1)$$

siendo \mathbf{a}_i las coordenadas del nivel fila A_i y \mathbf{b}_j las del nivel columna B_j en el plano, de forma que “*reproducen*” las distancias entre perfiles fila y columna y los residuos estandarizados (asociaciones):

$$\begin{aligned} d(\mathbf{a}_i, \mathbf{a}_{i'}) &= \sqrt{(a_{i1} - a_{i'1})^2 + (a_{i2} - a_{i'2})^2} \approx d_{ii'} \\ d(\mathbf{b}_j, \mathbf{b}_{j'}) &= \sqrt{(b_{j1} - b_{j'1})^2 + (b_{j2} - b_{j'2})^2} \approx d_{jj'} \\ \mathbf{a}'_i * \mathbf{b}_j &\approx r_{ij} \end{aligned} \quad (17.2)$$

Con las coordenadas contenidas en las matrices \mathbf{A} y \mathbf{B} , es posible “*visualizar*” la posición relativa de cada factor en las nuevas dimensiones. Esta estructura permite ver, tanto las “*distancias*” que hay entre los niveles de cada factor (mediante la distancia de representación en el plano),

como las “*asociaciones*” entre niveles de ambos factores (ya que mientras más asociación haya, más cerca se representarán en el plano).

Para resolver el problema de estimación de las matrices **A** y **B**, se busca una descomposición de la matriz de $\mathbf{R} = \{r_{ij}\}$ en valores singulares. Según la importancia que se da al ajuste de uno de los perfiles o a la matriz de residuos, se tienen diferentes métodos de selección, llamados **normalizaciones**, que pueden consultarse en [Greenacre \(2008\)](#).

17.2.1. Proyecciones fila, columna y simétrica

El punto de partida es la matriz de frecuencias relativas **P** cuyas entradas son n_{ij}/N , también llamada **matriz de correspondencias**. Definiendo el vector de unos **1**, con la dimensión adecuada, las masas, o frecuencias marginales, de filas y columnas, $r_i = \sum_{j=1}^C p_{ij}$ y $c_j = \sum_{i=1}^R p_{ij}$, respectivamente, se pueden expresar matricialmente como $\mathbf{r} = \mathbf{P} \mathbf{1}$ y $\mathbf{c} = \mathbf{P}' \mathbf{1}$ o, en forma de matrices diagonales,

$$\mathbf{D}_R = \text{diag}(r) \equiv \text{diag}(f_1, \dots, f_R) \quad \text{y} \quad \mathbf{D}_C = \text{diag}(c) \equiv \text{diag}(f_{,1}, \dots, f_{,C})$$

Se calcula la matriz de **residuos estandarizados** (véase Sec. @contaprox) como

$$\mathbf{S} = \mathbf{D}_R^{-\frac{1}{2}} (\mathbf{P} - \mathbf{rc}') \mathbf{D}_C^{-\frac{1}{2}} \quad (17.3)$$

La matriz **S** se descompone en valores singulares, calculando matrices las **U**, **D** y **V** tales que:

$$\begin{aligned} \mathbf{S} &= \mathbf{UDV}' \\ \mathbf{UU}' = \mathbf{V}'\mathbf{V} &= \mathbf{I}, \quad \mathbf{U}_{(R \times K)}, \quad \mathbf{V}_{(C \times K)}, \quad K = \min(R - 1, C - 1) \\ \mathbf{D} &= \text{diag}(\mu_1, \dots, \mu_K) \end{aligned} \quad (17.4)$$

donde los μ_i son los llamados **valores singulares**, estando ordenados de forma decreciente $\mu_1 \geq \mu_2 \geq \dots \geq \mu_K$.

A partir de esta descomposición se pueden obtener:

- las **coordenadas estándar de las filas**, $\Phi = \mathbf{D}_R^{-\frac{1}{2}} \mathbf{U}$, y sus **coordenadas principales**, $\mathbf{F} = \Phi \mathbf{D}$.
- las **coordenadas estándar de las columnas**, $\Gamma = \mathbf{D}_C^{-\frac{1}{2}} \mathbf{V}$, y sus **coordenadas principales**, $\mathbf{G} = \Gamma \mathbf{D}$.
- las **inercias principales**, $\lambda_i = \mu_i^2$.

Las coordenadas principales son las utilizadas para definir las **proyecciones fila** y **proyecciones columna**, que representan, en menor dimensión, los perfiles correspondientes, formando los llamados **mapas asimétricos**.

Por último, las matrices $\mathbf{A} = \mathbf{D}_R^{-\frac{1}{2}} \mathbf{UD}$ y $\mathbf{B} = \mathbf{D}_C^{-\frac{1}{2}} \mathbf{VD}$ representan las coordenadas de ambos perfiles en un espacio común, llamado **mapa simétrico**.

17.3. Procedimiento con R: la función ca()

Para realizar un análisis de correspondencias simple con **R** se puede utilizar el paquete **ca**, que contiene la función **ca()**. Esta función acepta como argumento de entrada o bien directamente una tabla de contingencia, o bien los datos originales como objeto matriz o data-frame. Incluso, el argumento puede ser una fórmula del tipo $\sim F1 + F2$ donde $F1$ y $F2$ son factores. Entre los argumentos adicionales se pueden incluir el número de dimensiones en el output así como filas o columnas suplementarias.

17.3.1. Caso práctico 1: tareas del hogar.

Como primer ejemplo, se van a utilizar los datos **housetasks**, contenidos en el paquete **factoextra**, que representan una tabla de contingencia con la frecuencia de ejecución de 13 tareas del hogar por los miembros de la pareja.

```
library('ca')
library('factoextra')
data('housetasks')
```

En primer lugar, la aplicación del test χ^2 de independencia (véase 5) permite contrastar si los factores son independientes o, por el contrario, están asociados:

```
chisq.test(housetasks)
#>
#> Pearson's Chi-squared test
#>
#> data: housetasks
#> X-squared = 1944.5, df = 36, p-value < 2.2e-16
```

Con un valor de $\chi^2 = 1944,5$ y un p-valor de $2.2e-16$, hay suficiente evidencia como para rechazar la hipótesis nula de independencia, indicando asociación entre ambos factores, por lo que tiene sentido analizar más en profundidad la estructura de dicha asociación.

La función **ca()** proporciona los valores singulares y, tanto para filas como para columnas, las masas (valores “*Mass*”); las distancias chi-cuadrado, que representan las distancias en esa métrica de cada fila respecto a la fila centroide (dada por la masa de las columnas, promedio de los vectores fila); las inercias explicadas, que representan la distancia cuadrática χ^2 respecto al perfil promedio (sin calcular raíces), ponderada por la masa (de la fila o columna) correspondiente; así como las coordenadas en el espacio proyectado:

```
options(digits = 2)
ca_house <- ca(housetasks, nd = 2)
ca_house
#>
#> Principal inertias (eigenvalues):
```

17.3. Procedimiento con **R**: la función **ca()**

287

```
#>      1       2       3
#> Value    0.542889 0.445003 0.127048
#> Percentage 48.69% 39.91% 11.4%
#>
#>
#> Rows:
#>   Laundry Main_meal Dinner Breakfast Tidying Dishes Shopping Official
#> Mass     0.10    0.088  0.062      0.080  0.070  0.065  0.069  0.055
#> ChiDist  1.15    1.017  0.786      0.716  0.594  0.550  0.466  0.984
#> Inertia  0.13    0.091  0.038      0.041  0.025  0.020  0.015  0.053
#> Dim. 1   -1.35   -1.188 -0.940     -0.690 -0.534 -0.256 -0.160  0.308
#> Dim. 2   -0.74   -0.735 -0.462     -0.679  0.651  0.663  0.605 -0.380
#>           Driving Finances Insurance Repairs Holidays
#> Mass      0.08    0.065    0.080   0.095   0.092
#> ChiDist   1.13    0.675    0.853   1.819   1.463
#> Inertia   0.10    0.030    0.058   0.313   0.196
#> Dim. 1   1.01    0.367    0.878   2.075   0.343
#> Dim. 2   -0.98   0.926    0.710  -1.296   2.151
#>
#>
#> Columns:
#>   Wife Alternating Husband Jointly
#> Mass    0.34    0.146    0.22    0.29
#> ChiDist 0.94    0.899    1.32    1.04
#> Inertia 0.30    0.118    0.38    0.31
#> Dim. 1 -1.14   -0.084   1.58    0.20
#> Dim. 2 -0.55   -0.437   -0.90   1.54
```

Las dos primeras dimensiones explican el 48.69 % y 39.91 % de la inercia respectivamente, por lo que la representación en un plano engloba al 88.6 % de la inercia global.

Las distancias chi-cuadrado indican lo cerca o lejos que está cada fila respecto al centroide de las mismas. En este ejemplo, la fila más distante del centroide de filas es “Repairs” (1.819), mientras que la columna más distante respecto del centroide de columnas es “Husband” (1.321).

Como las inercias miden la variabilidad de los perfiles, en este ejemplo, respecto a las filas, el nivel que mayor contribuye es “Repairs” (0.312874) mientras que por columnas es “Husband” (0.381373). Esto no es sorprendente ya que ambos niveles eran los más alejados del centro.

Con las coordenadas de las dimensiones se puede realizar un gráfico de las mismas utilizando la función **plot()**, pudiéndose optar por la proyección sólo de las filas (usando los argumentos **map = “rowprincipal”**, **what = c(“all”, “none”)**) o de las columnas (**map = “colprincipal”**, **what = c(“none”, “all”)**), tal como se muestra en la Fig. 17.1:

```
par(mfrow = c(1, 2))
plot(ca_house, map = "rowprincipal", what = c("all", "none"), xlab = "Perfiles fila")
plot(ca_house, map = "colprincipal", what = c("none", "all"), xlab = "Perfiles
→ columna")
```

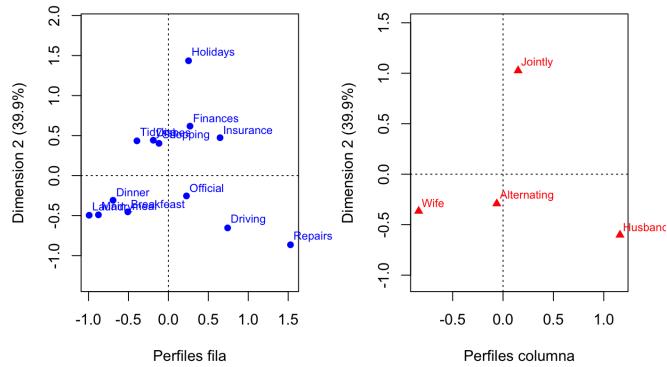


Figura 17.1: Proyecciones de los perfiles fila y columna

Respecto a las filas, se aprecian varios grupos: el compuesto por “*Breakfast*”, “*Dinner*”, “*Main_meal*” y “*Laundry*”; otro por “*Shopping*”, “*Dishes*” y “*Tidying*”; uno tercero por “*Insurance*” y “*Finance*”; y el compuesto por “*Driving*” y “*Official*”. Los niveles “*Holiday*” y “*Repairs*” están alejados del resto.

Las coordenadas simétricas permiten la representación de ambos factores a la vez (*map*= “*symmetric*”, *what*=*c*(“*all*”, “*all*”)), como se muestra en la Fig. ??.

```
plot(ca_house, map = "symmetric", what = c("all", "all"), xlab = "Proyección común de
↪ ambos factores")
```

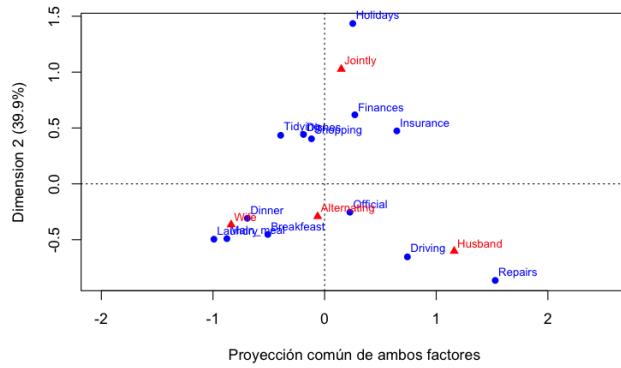


Figura 17.2: Proyección simétrica de ambos factores

El gráfico conjunto permite observar qué niveles de filas y columnas pueden estar más cercanos (aproximación a la asociación entre ellos). El grupo de “*Driving*” y “*Repairs*” está cercano a “*Husband*”; el grupo de “*Dinner*”, “*Breakfast*”, “*Laundry*” y “*Main_meal*” está cercano

a “*Wife*”; mientras que el nivel “*Jointly*” parece estar asociado a “*Holidays*”, “*Finance*”, e “*Insurance*”.

17.3.2. Caso práctico 2: accidentes 2020.

Como segundo ejemplo, se van a utilizar los datos `accidentes2020_data`, contenidos en el paquete CDR, en concreto, la información sobre “*tipo_accidente*” y “*estado_meteorológico*”. Para evitar pares de niveles con frecuencia nula, se eliminan los niveles “*Atropello a animal*”, “*Despeñamiento*” y “*Otro*” del factor “*tipo_accidente*” y los niveles “*Granizando*”, “*Nevando*”, “*NULL*” y “*Se desconoce*” del factor “*estado_meteorológico*”.

```
library('CDR')
library('dplyr')
data('accidentes2020_data')
datos <- data.frame(
  V1 = as.factor(accidentes2020_data$tipo_accidente),
  V2 = as.factor(accidentes2020_data$estado_meteorológico)
)
levelsV1 <- c("Alcance", "Choque contra obstáculo fijo", "Colisión frontal", "Colisión
  ↪ fronto-lateral", "Colisión lateral", "Colisión múltiple")
levelsV2 <- c("Despejado", "Lluvia débil", "Lluvia intensa", "Nublado")
datos_depu <- droplevels(filter(datos, (V1 %in% levelsV1) & (V2 %in% levelsV2)))
```

		V2			
		Despejado	Lluvia débil	Lluvia intensa	Nublado
#>	V1				
#>	Alcance	5525	403	84	449
#>	Choque contra obstáculo fijo	3258	308	43	224
#>	Colisión frontal	711	42	5	48
#>	Colisión fronto-lateral	6359	398	51	494
#>	Colisión lateral	3241	169	30	277
#>	Colisión múltiple	1619	173	29	111

Se comprueba que existe asociación y se obtienen los resultados del análisis de correspondencias:

```
tabla <- table(datos_depu)
chisq.test(tabla)
#>
#> Pearson's Chi-squared test
#>
#> data: tabla
#> X-squared = 104, df = 15, p-value = 2e-15
ca_tabla <- ca(tabla, k = 2)
ca_tabla
#>
#> Principal inertias (eigenvalues):
```

```

#>      1       2       3
#> Value  0.003804 0.000493 2.7e-05
#> Percentage 87.97% 11.4% 0.62%
#>
#>
#> Rows:
#>          Alcance Choque contra obstáculo fijo Colisión frontal
#> Mass      0.26864           0.1594        0.03351
#> ChiDist   0.03200           0.0817        0.06591
#> Inertia   0.00028           0.0011        0.00015
#> Dim. 1    -0.16016          -1.2818       0.72969
#> Dim. 2     1.36554          -0.9207      -1.84795
#>          Colisión fronto-lateral Colisión lateral Colisión múltiple
#> Mass      0.3036            0.15455       0.0803
#> ChiDist   0.0446            0.07682       0.1284
#> Inertia   0.0006            0.00091       0.0013
#> Dim. 1    0.6586            1.22996      -2.0813
#> Dim. 2    -0.8221           0.52856       0.1210
#>
#>
#> Columns:
#>          Despejado Lluvia débil LLuvia intensa Nublado
#> Mass      0.86121           0.0621        0.01006 0.06665
#> ChiDist   0.01344           0.2145        0.28954 0.08391
#> Inertia   0.00016           0.0029        0.00084 0.00047
#> Dim. 1    0.20552           -3.4619       -3.67116 1.12291
#> Dim. 2    -0.19007          -0.8381       8.05783 2.02007

```

Las dos primeras dimensiones explican el 87.97% y 11.4% respectivamente, por lo que la representación en un plano engloba al 99.37% de la inercia.

La representación gráfica de la proyección simétrica se muestra en la Fig. 17.3:

```

plot(ca_tabla, map = "symmetric", what = c("all", "all"), xlab = "Proyección común de
→ ambos factores")

```

Se observa que “*Lluvia intensa*” no está especialmente asociada a ningún tipo de accidente; “*Lluvia débil*” con “*Colisión múltiple*” y “*Choque contra obstáculo fijo*”; “*Despejado*” con “*Colisión fronto-lateral*”, “*Colisión frontal*” y “*Alcance*”; y “*Nublado*” con “*Colisión lateral*”.

17.3. Procedimiento con **R**: la función **ca()**

291

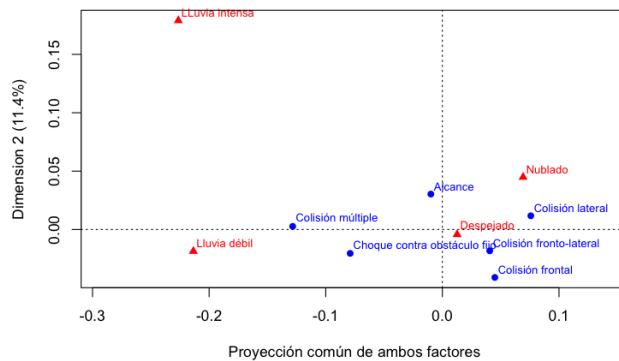


Figura 17.3: Proyección simétrica de ambos factores

Resumen

Dada una tabla de contingencia, el análisis de correspondencias reproduce: (i) las distancias entre niveles de cada factor en un espacio de menor dimensión, permitiendo la comparación gráfica entre ellos; (ii) la representación de los niveles de ambos factores en un espacio común.

En el primer caso, permite una visualización de la composición interna de cada factor, identificando los niveles que más se distancian del centroide. En el segundo, permite la representación de la asociación entre niveles de cada uno de los factores.

Parte III

Deep learning

Capítulo 18

Redes neuronales artificiales

Noelia Vállez Enano^a y José Luis Espinosa Aranda^a

^aUniversidad de Castilla-La Mancha

18.1. ¿Qué es el *deep learning*?

La inteligencia artificial es una disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico (de la Real Academia Española, 2023). Entre otros ejemplos se pueden encontrar en la actualidad tanto robots que son capaces de realizar tareas de manera similar a un humano en una fábrica, las denominadas como casas inteligentes o los vehículos autónomos.

Dentro de las técnicas utilizadas para la inteligencia artificial, se encuentran las técnicas clásicas de *machine learning*, ya explicadas en capítulos anteriores de este libro, las cuales tienen la habilidad de aprender sin haber sido explícitamente programadas para una tarea en particular, pudiendo ser utilizadas para varios fines y aplicaciones.

A su vez, dentro de estos algoritmos, se pueden enmarcar como un subconjunto de las mismas las técnicas de *deep learning*, las cuales intentan simular tanto la arquitectura como el comportamiento del sistema nervioso humano, en particular, de las redes de neuronas que componen el encéfalo y que se encargan de realizar tareas específicas (Fig. 18.1). Para ello, estas técnicas se basan en el concepto de redes neuronales, que intentan emular la forma de aprendizaje de los humanos (Goodfellow et al., 2016).

18.1.1. Diferencias entre las técnicas de *machine learning* tradicional y el *deep learning*

Como se vio en el Cap. ??, la metodologías de ciencia de datos tienen una etapa llamada preparación de datos que incluye la tarea de elección de variables, la cual ha sido tratada

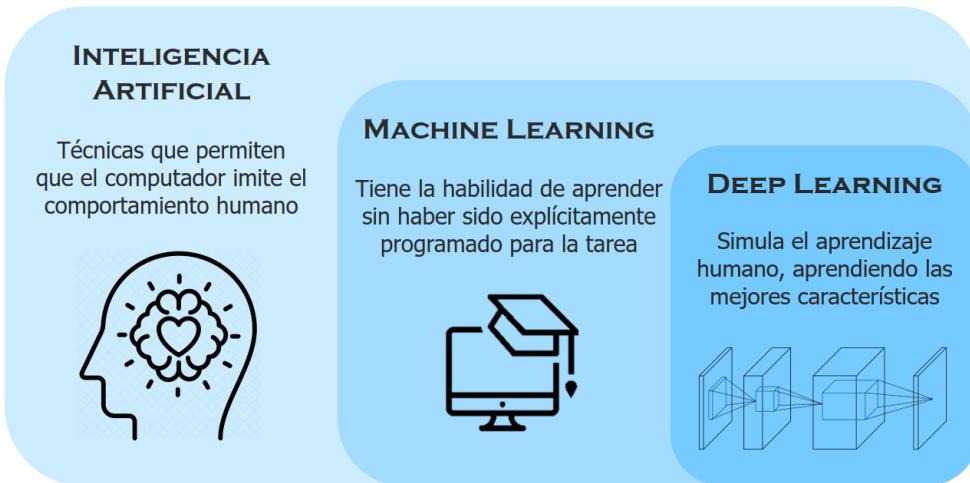


Figura 18.1: Inteligencia Artificial vs Machine learning vs Deep Learning

ampliamente en el Cap. ??, para realizar una selección de las mejores características que representen el problema a resolver, y que puedan ser comprendidas por el algoritmo de *machine learning* seleccionado de tal forma que sea capaz de solucionar el problema planteado.

Por ejemplo, en el caso de querer detectar una cara dentro de una imagen, sería necesario definir qué tipo de características servirían para detectar la misma, como podrían ser, a bajo nivel, determinados tipos de bordes de la imagen (Fig. 18.2). Estas características proporcionarían la base para detectar a nivel medio elementos de la cara como ojos, narices, orejas, etc. y, definitivamente, a alto nivel, reconocer donde hay una cara dentro de la imagen.



Figura 18.2: Detección de bordes de una imagen mediante el método de Scharr

Esta elección de características requiere en muchas ocasiones de la intervención humana, por lo que puede llevar mucho tiempo y diversos experimentos de prueba y error hasta poder encontrar una combinación de características que permita resolver el problema planteado.

Las técnicas de *deep learning*, a diferencia de las técnicas de *machine learning* tradicional, son capaces de aprender cuales son las mejores características que permitirán representar el

problema que se quiere resolver sin necesidad de la interacción humana a la misma vez que buscan la solución al mismo.

Continuando con el ejemplo anterior de la detección de caras, mientras que en las técnicas de *machine learning* sería necesario introducirle al algoritmo qué características base componen una cara para que sea capaz de reconocerlas, al utilizar *deep learning* únicamente sería necesario mostrarle suficientes imágenes de caras para conseguir que el algoritmo sea capaz de aprender a identificar una cara por sí mismo, identificando de forma automática las características más importantes de una cara.

Esta capacidad de aprender las mejores características necesarias por sí mismo hace que a nivel teórico las técnicas de *deep learning* puedan llegar a ser más potentes que el *machine learning* clásico, pero debido a la mayor complejidad del problema y, por consiguiente, al proceso de entrenamiento, también lleva a que que sean necesarios muchos más datos y una mayor potencia de cómputo para entrenarlas.

Este hecho explica que, aunque las bases de las técnicas de *deep learning* como el algoritmo del descenso del gradiente (Kiefer and Wolfowitz, 1952), el perceptrón (Rosenblatt, 1958), los algoritmos de retropropagación y el perceptrón multicapa (Rumelhart et al., 1986) y la primera red neuronal convolucional (LeCun et al., 1995), datan de varios años atrás, no sea hasta hace relativamente poco tiempo, cuando se ha podido empezar a utilizar estas técnicas. Esto se debe a diversos factores:

1. **La evolución en el hardware de procesamiento.** En particular, debido a la mejora de la capacidad de paralelismo masivo durante el cómputo que proporcionaron las nuevas tarjetas gráficas (GPU) al incorporar una gran cantidad de microprocesadores específicos, han podido ser utilizadas para las técnicas de *deep learning*. Originalmente su principal uso era representar modelos complejos 3D en los monitores, pero su utilización para técnicas de *deep learning* ha llevado recientemente al desarrollo de tarjetas específicas para este fin. Además, es posible disponer bajo demanda de estos recursos de computación como servicios a través de Internet. Esto es lo que se conoce como *cloud computing*.
2. **El Big data.** La gran cantidad de datos que se generan y almacenan en la actualidad, así como la mayor facilidad a la hora de trabajar con esos conjuntos de datos (gracias a las nuevas herramientas disponibles), han permitido cubrir la necesidad del gran volumen de datos iniciales necesarios.
3. **La evolución del software.** Recientemente ha habido un amplio interés tanto en buscar nuevos modelos para resolver todo tipo de problemas, como para mejorar las técnicas utilizadas para entrenar dichas redes neuronales. Esto ha llevado a la creación y mejora de diversos frameworks, librerías y aplicaciones relacionadas con el entrenamiento y despliegue de redes neuronales. Entre ellos, serían destacables Keras, Tensorflow, Pytorch, Caffe2, Matlab y OpenVINO.

18.2. Aplicaciones del *deep learning*

Las posibles aplicaciones de las técnicas de *deep learning* son muy diversas y, gracias a la continua investigación desarrollada en el área en la actualidad, no hacen más que aumentar. A continuación se comentan algunas de ellas:

1. **Clasificación de imágenes.** Aunque la clasificación de imágenes dentro del área de la visión por computador o artificial es una realidad hace muchos años, es con las técnicas de *deep learning* con las que se han logrado los mayores avances, en particular, utilizando las redes neuronales convolucionales. Estas redes permiten determinar a qué clase, perteneciente al conjunto de categorías utilizado para entrenar, se corresponde una determinada imagen.
2. **Detección de objetos.** Permite localizar los objetos contenidos en una imagen mediante un rectángulo, clasificándolo a su vez por su tipología. Por ejemplo, utilizando una cámara de seguridad instalada en una calle con este tipo de modelos sería posible localizar y diferenciar entre peatones y vehículos (Fig. 18.3).



Figura 18.3: Detección de peatones y vehículos utilizando una cámara térmica y técnicas de deep learning

3. **Segmentación semántica/de instancias.** De forma similar a la detección de objetos, la segmentación semántica permite localizar objetos contenidos en una imagen, además de su tipología, pero en este caso se marcan utilizando una máscara a nivel de píxel. La segmentación de instancias además es capaz de diferenciar entre diferentes instancias de una misma clase aun cuando se encuentren situadas de forma contigua.
4. **Reconocimiento del habla.** Permite a un computador procesar y comprender el habla humana. En la actualidad existen varios asistentes inteligentes basados en esta tecnolo-

gía que además son capaces de interpretar órdenes o instrucciones sencillas y actuar en consecuencia.

5. **Traducción automática.** Consiste en utilizar las técnicas de *deep learning* para traducir un texto automáticamente de una lengua a otra sin la necesidad de intervención humana. En la actualidad, no se limita únicamente a la traducción literal, palabra por palabra, del texto, si no que también tiene en cuenta el significado que tendría en el idioma original para adaptarlo al idioma destino (Fig. 18.4).

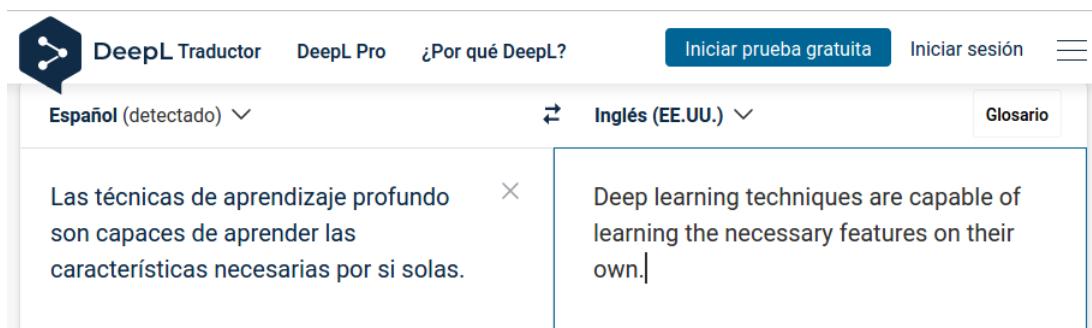


Figura 18.4: Traductor automático basado en Deep Learning

6. **Generación automática de imágenes/texto.** Permite obtener desde una imagen un texto descriptivo que indique el contenido de la imagen, o al contrario, a partir de un texto descriptivo generar una imagen basada en dicha descripción. Un ejemplo de este último caso sería Dall-E (Borji, 2022) (Fig. 18.5).

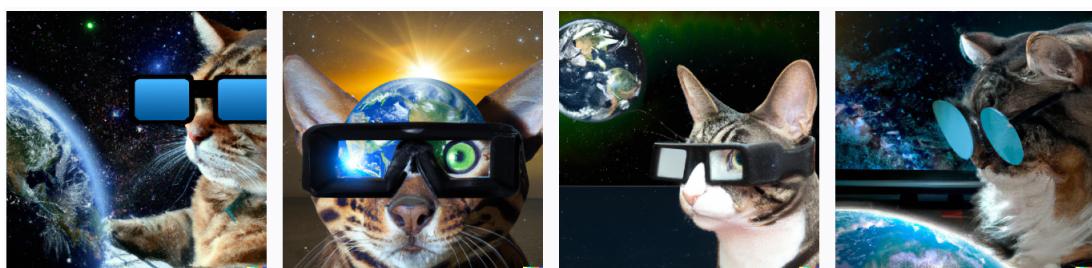


Figura 18.5: Algunas salidas posibles del generador de imágentes a partir de texto Dall-E, para el texto a “cat with glasses studying computer vision in the space with the Earth in the background”

7. **Automóvil autónomo.** Las técnicas de *deep learning* están siendo claves para el desarrollo del vehículo autónomo, capaz de circular sin la necesidad de la interacción de un conductor humano. Para lograr definitivamente un vehículo con estas características,

es necesario que sea capaz de ver, tomar decisiones y conducir al mismo tiempo. Esto se consigue en la actualidad integrando la información de gran cantidad de sensores que obtienen datos en tiempo real sobre el entorno, como serían cámaras, LIDAR, radares o ultrasónicos entre otros, y que son procesados por varias redes neuronales con el fin de que sea capaz de tomar una decisión en cuestión de milisegundos (Fig. 18.3).

8. **Chatbots con inteligencia artificial.** Son aplicaciones software que, utilizando la inteligencia artificial conversacional, son capaces de conversar mediante un chat escrito como si fueran un ser humano. Caben destacar los asistentes virtuales existentes en diversas páginas web y el reciente *ChatGPT* (OpenAI, 2022), el cual es capaz de mantener conversaciones con el usuario, resolver problemas sencillos, generar textos y resúmenes sobre cualquier tema o generar código en diversos lenguajes de programación a partir de una petición realizada mediante lenguaje natural.

18.3. Redes neuronales

Las redes neuronales artificiales (en inglés Artificial Neural Network (ANN)) tienen su origen en la definición de neurona artificial de (McCulloch and Pitts, 1943) y en el diseño del perceptrón por parte de Frank Rosenblatt (Rosenblatt, 1958). Cada ANN está formada por un conjunto de elementos conocidos como “neuronas” cuya organización está inspirada en la que siguen las redes neuronales de los seres vivos. Entre dos neuronas adyacentes existe una serie de conexiones a través de las cuales se envía la información como si de pulsos eléctricos se tratase. De forma aislada, cada neurona procesa la información recibida para producir un resultado que será utilizado por las siguientes neuronas con las que está conectada.

Cada ANN tiene como objetivo resolver una tarea concreta. Por ejemplo, una ANN podría estar diseñada para reconocer un dígito o una letra a partir de una imagen. Para conseguir resolver dicha tarea, la red sigue un proceso de aprendizaje automático. Este proceso se conoce como “entrenamiento” y requiere que se disponga de un conjunto de datos representativos de la tarea a resolver.

18.4. Perceptrón o neurona

El elemento básico de toda ANN es la neurona artificial, inspirada en las neuronas biológicas. Cada neurona tiene una serie de entradas y produce una única salida. Las entradas pueden ser variables extraídas de la tarea que se debe resolver o salidas de otras neuronas de la red.

Para calcular la salida, cada neurona realiza una suma ponderada de sus entradas utilizando una serie de pesos, \mathbf{w} donde $w_i \in \mathbb{R}$, y añade un término constante, $w_0 \in \mathbb{R}$. Por tanto, cada neurona actúa como un clasificador lineal que puede separar dos conjuntos diferentes dependiendo de si la salida es positiva o negativa (Figura 18.6).

Para cada vector de entrada, \mathbf{x} , la neurona aplicará los pesos, \mathbf{w} , como el producto escalar de ambos vectores:

18.4. Perceptrón o neurona

301

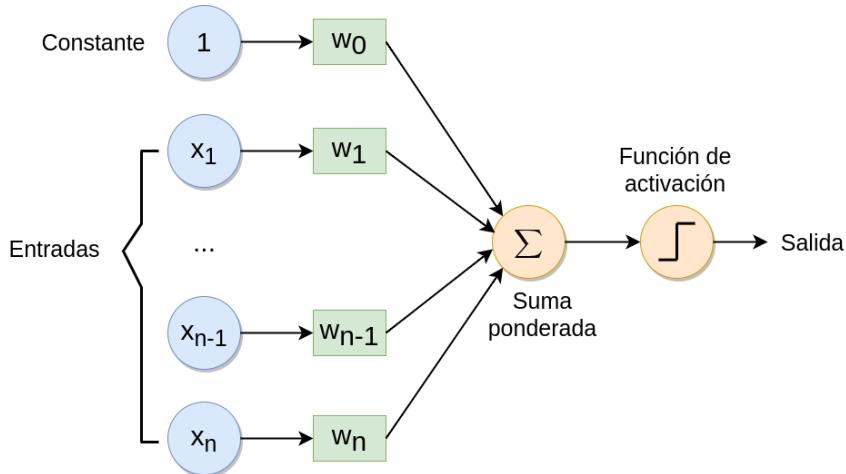


Figura 18.6: Estructura del perceptrón o neurona

$$\mathbf{w}'\mathbf{x} = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n. \quad (18.1)$$

Una vez obtenida la suma ponderada, típicamente se puede separar las entradas en dos conjuntos, obteniéndose como salida final un valor binario, siguiendo la fórmula:

$$f(\mathbf{w}'\mathbf{x}) = \begin{cases} 1 & \text{si } \mathbf{w}'\mathbf{x} > 0 \\ 0 & \text{en otro caso} \end{cases}. \quad (18.2)$$

18.4.1. Aprendizaje

Durante el proceso de aprendizaje, el perceptrón busca el ajuste automático de los valores de los pesos. Éstos deben seleccionarse de forma que minimicen el error de clasificación cometido sobre un conjunto de entrenamiento. El conjunto de entrenamiento estará compuesto por un conjunto de muestras del que se conoce su clase:

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}, \quad (18.3)$$

donde cada muestra, $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$, pertenece a una de las dos clases, $y_i = \{0, 1\}$.

El primer paso del aprendizaje o entrenamiento consiste en la inicialización de cada peso w_j a 0 o a algún otro valor aleatorio.

Tras ello, se calcula la clase estimada, \hat{y} , en un momento determinado, t , para cada muestra \mathbf{x}_i del conjunto de datos:

$$\hat{y}_i(t) = f(\mathbf{w}(t)^T \mathbf{x}_i) = f(w_0(t) + w_1(t) \cdot x_{i1} + \dots + w_n(t) \cdot x_{in}). \quad (18.4)$$

Tras obtener la salida para todas las muestras de entrenamiento, cada uno de los pesos, w_j , de la neurona se actualiza siguiendo la fórmula:

$$w_j(t+1) = w_j(t) + \lambda \cdot |y_i - \hat{y}_i(t)| \cdot x_{ij}. \quad (18.5)$$

donde $|y_i - \hat{y}_i(t)|$ será 0 cuando la clase predicha coincide con la clase real de la muestra y λ es la tasa de aprendizaje. La tasa de aprendizaje debe seleccionarse de antemano y controla la variación de los pesos entre iteraciones. En algunos casos el valor de λ es 0 o varía durante el proceso de entrenamiento.

Los dos pasos anteriores se repiten hasta que el error de clasificación es menor que un cierto umbral o el número de iteraciones alcanza un cierto valor fijado. Normalmente se suele utilizar el número de iteraciones como criterio de paro puesto que no siempre es posible alcanzar una tasa de error más baja que la deseada.

18.4.2. Convergencia

El teorema de la convergencia del perceptrón dice que, en los problemas en los que haya dos clases linealmente separables, es siempre posible encontrar unos pesos que realicen la separación en un número finito de iteraciones (Novikoff, 1962). Sin embargo, en la mayoría de los casos, no se está ante problemas linealmente separables, esto es, no es posible obtener un conjunto de variables que separen perfectamente las muestras de ambas clases. Por ello, es necesario el uso de ciertas estrategias que solucionen el problema de convergencia en estos casos. Algunas de las estrategias más utilizadas son:

- Algoritmo Pocket: Guarda la mejor solución obtenida hasta el final del entrenamiento.
- Algoritmo Maxover: Halla el margen de separación máximo permitiendo clasificaciones incorrectas.
- Algoritmo de Voto: Se utilizan múltiples perceptrones combinando sus salidas.

18.5. Perceptrón multiclasa

Una extensión lógica del uso del perceptrón es su empleo en la resolución de tareas de clasificación donde existan más de dos clases (Haykin, 1999). En ese caso se tendrá un conjunto de entrenamiento, D , de m muestras:

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}, \quad (18.6)$$

donde cada muestra $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$ pertenezca a una de las c clases posibles:

$$y_i = \{0, 1, \dots, c-1\}. \quad (18.7)$$

A diferencia del problema binario, en su versión multiclase lo que se definen son varios modelos, F , uno para cada una de las c clases:

$$F = \{f_0, f_1, \dots, f_{c-1}\} f_j : \mathbb{R}^n \rightarrow \mathbb{R}. \quad (18.8)$$

En este caso la salida no se selecciona en función de si el valor obtenido es positivo o negativo, sino que se asigna la clase del modelo que obtenga el valor más alto tras aplicar los pesos a la muestra. Esta estrategia recibe el nombre de “uno contra todos”:

$$\hat{y}_i = \operatorname{argmax}_j(f_j(\mathbf{x}_i)) j \in \{0, 1, \dots, c - 1\}. \quad (18.9)$$

En muchas ocasiones lo que se obtiene no es un único valor con la clase asignada como salida, sino que se obtiene un vector con las salidas binarias de cada uno de los modelos empleados. En ese caso, el vector contendrá un 1 en la posición de la clase asignada y un 0 en el resto de clases. Por ejemplo, el vector [0, 1, 0, 0, 0] representaría que una muestra ha sido asignada a la segunda clase en un problema de clasificación donde existen 5 clases posibles:

$$[(f_1(\mathbf{x}_i)), (f_2(\mathbf{x}_i)), \dots, (f_c(\mathbf{x}_i))]. \quad (18.10)$$

18.6. Funciones de activación

Además de los pesos, toda neurona tiene asociada una función de activación. Esta función se encarga de transformar la suma ponderada de las entradas en el resultado final. En las secciones anteriores se ha utilizado una función de activación con umbral 0, pero existen muchas otras. Algunas de las más utilizadas se enumeran a continuación.

Para algunas de ellas, se ha implementado una función, `plot_activation_function()`, que permite dibujarlas en **R**, y que se puede ver a continuación:

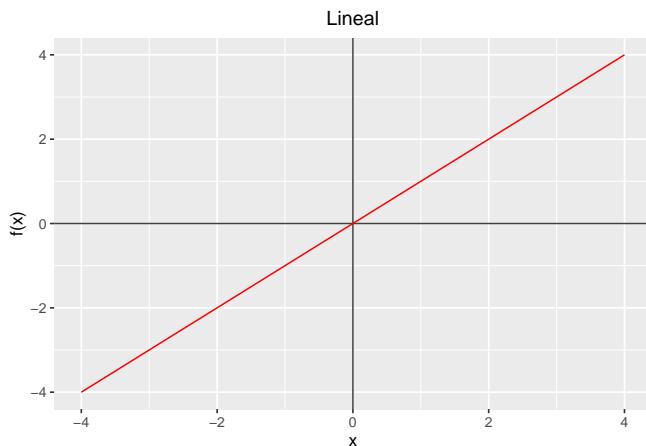
```
require(ggplot2)
plot_activation_function <- function(f, title, range){
  ggplot(data.frame(x=range), mapping=aes(x=x)) +
    geom_hline(yintercept=0, color='black', alpha=3/4) +
    geom_vline(xintercept=0, color='black', alpha=3/4) +
    stat_function(fun=f, colour = "red") +
    ggtitle(title) +
    scale_x_continuous(name='x') +
    scale_y_continuous(name='f(x)') +
    theme(plot.title = element_text(hjust = 0.5))
}
```

- **Función lineal.** Se trata de una función identidad donde la salida tiene el mismo valor que la entrada. Normalmente se aplica en problemas de regresión lineal. Por ejemplo, si se quiere predecir el número de días que lloverá en un mes determinado.

$$f(x) = x \quad (18.11)$$

Y se representa gráficamente de la siguiente forma:

```
f <- function(x){ x }
plot_activation_function(f, 'Lineal', c(-4,4))
```



- **Función umbral.** Esta función recibe también el nombre de función escalón. Si el valor de entrada es menor que el umbral la salida será 0. En caso contrario, la salida será 1. Si el umbral es 0, la función se reduce a mirar el signo del valor analizado.

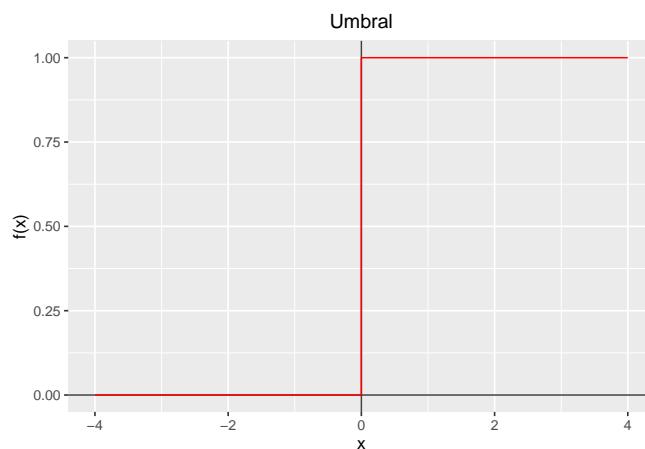
$$f(x) = \begin{cases} 0 & \text{si } x < u \\ 1 & \text{en otro caso} \end{cases} \quad (18.12)$$

Se representa gráficamente mediante el siguiente código, el cual se corresponde con una modificación de la función `plot_activation_function`, ya que la versión original no mostraría de forma correcta la gráfica al requerir representar dos valores en la posición 0, el valor 0 y el valor 1 del escalón:

```
df <- data.frame(x=c(-4, -3, -2, -1, 0, 1, 2, 3, 4), f=c(0,0,0,0,1,1,1,1,1))
ggplot(data=df, aes(x=x, y=f, group=1)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("Umbral") +
  scale_y_continuous(name='f(x)') +
  geom_hline(yintercept=0, color='black', alpha=3/4) +
  geom_vline(xintercept=0, color='black', alpha=3/4) +
  geom_step(color='red')
```

18.6. Funciones de activación

305

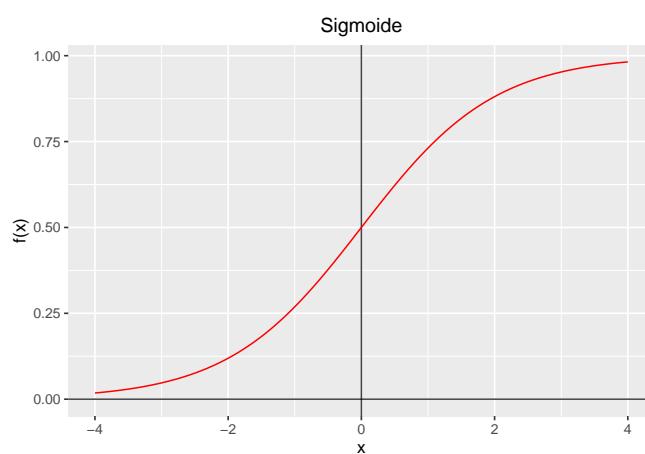


- **Función sigmoide.** También conocida como función logística, se trata de una de las funciones más utilizadas para asignar una clase. Si el punto de evaluación de la función es un valor negativo muy bajo, la función dará como resultado un valor muy cercano a 0, si se evalúa en 0, el resultado es 0,5 y si se evalúa en un valor positivo alto el resultado será aproximadamente 1.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (18.13)$$

Representándose gráficamente de la siguiente forma:

```
f <- function(x){1 / (1 + exp(-x))}
plot_activation_function(f, 'Sigmoid', c(-4,4))
```

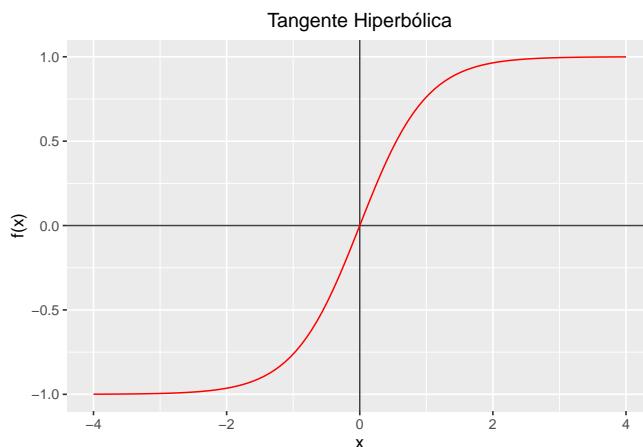


- **Función tangente hiperbólica.** El rango de valores de salida es [-1, 1], donde los valores altos tienden de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1 de forma similar a la sigmoid.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (18.14)$$

Siendo su representación gráfica de la siguiente forma:

```
tanh_func <- function(x){tanh(x)}
plot_activation_function(tanh_func, 'Tangente Hiperbólica', c(-4,4))
```

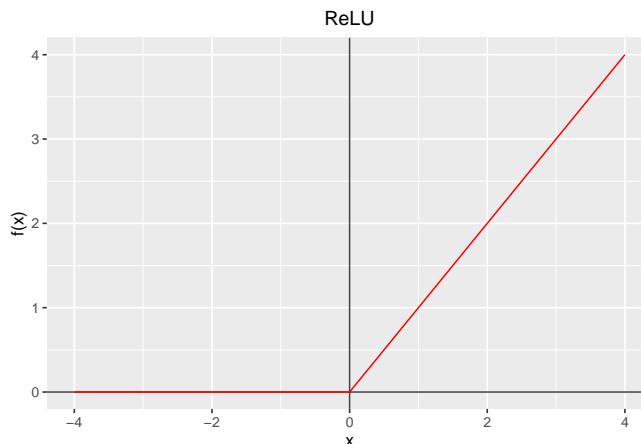


- **Función ReLU.** Se trata de la unidad lineal rectificada (del inglés Rectified Linear Unit). Es posiblemente la función de activación más utilizada actualmente en deep learning ([Nair and Hinton, 2010](#)).

$$f(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{en otro caso} \end{cases} \quad (18.15)$$

Y se representaría gráficamente de la siguiente manera:

```
rec_lu_func <- function(x){ ifelse(x < 0 , 0, x )}
plot_activation_function(rec_lu_func, 'ReLU', c(-4,4))
```



18.7. Perceptrón multicapa

Aunque el perceptrón puede aprender muchos tipos de lógica, no es posible que aprenda la operación XOR (OR exclusivo) que se diferencia del OR en que asigna un 1 a la salida cuando las dos entradas son distintas (Minsky and Papert, 1969). El perceptrón multicapa o, en inglés, Multilayer Perceptron (MLP) surge para dar una solución a este problema que es un paradigma de los problemas linealmente no separables, que realmente son la mayoría en el mundo real.

Un MLP está compuesto por varias capas con neuronas. La primera capa será la de entrada, que recibirá las variables que representan los elementos del problema a resolver. Por otro lado, la última capa representará las clases de salida (en las que hay que clasificar las entradas), esto es, la salida del MLP. Entre ambas capas existirán una o más capas “ocultas”. Las neuronas de una capa intermedia tienen como entrada la salida de la capa anterior y su salida es la entrada de las neuronas de la siguiente capa (Figura 18.7). Este tipo de capas también son llamadas *densas* o *totalmente conectadas*.

18.7.1. Aprendizaje

El MLP entra en la categoría de los algoritmos de propagación hacia adelante o *feedforward* ya que las entradas de las neuronas de una capa se combinan mediante la suma ponderada, pasan por una función de activación y el resultado es propagado a las neuronas de la capa siguiente. Este proceso se lleva a cabo desde la capa de entrada hasta la capa de salida.

Dado un conjunto de muestras de entrenamiento $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ donde cada $\mathbf{x}_i \in \mathbb{R}^d$ e $y_i \in \{0, 1\}$ (siendo d el número de características), la salida de la primera capa, \mathbf{z}_1 , para una entrada \mathbf{x} vendrá dada por la expresión:

$$\mathbf{z}_1 = \mathbf{W}'_{(1)} \mathbf{x} + \mathbf{b}_1, \quad (18.16)$$

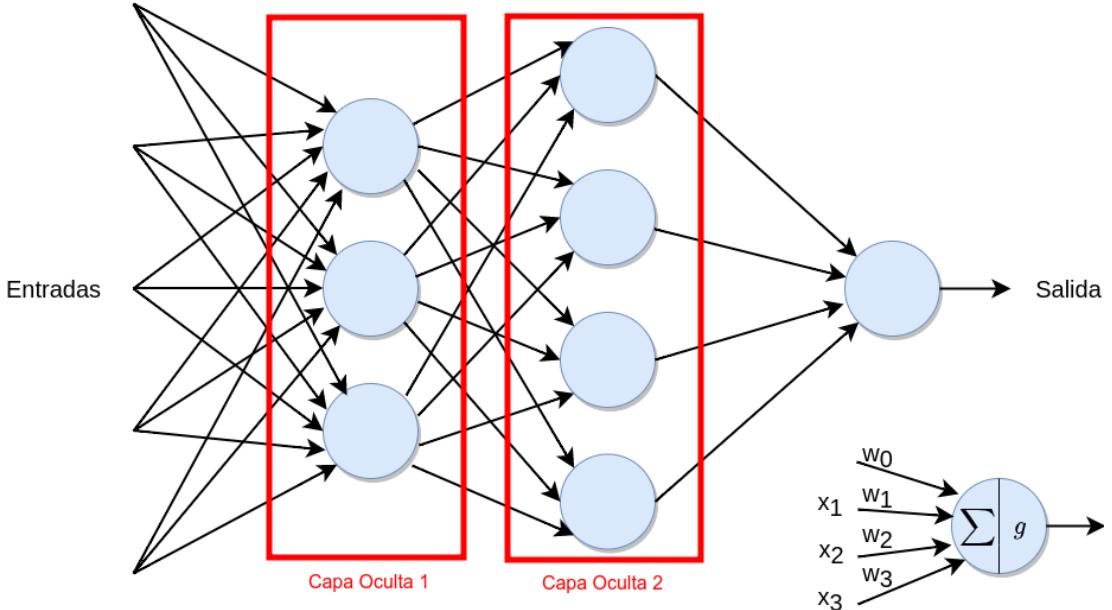


Figura 18.7: Estructura del perceptrón multicapa (MLP)

donde $\mathbf{b}_1 \in \mathbb{R}^h$ es un vector con las constantes de la primera capa, siendo h el número de variables de cada capa, y $\mathbf{W}_{(1)} \in \mathbb{R}^{h \times d}$ son los pesos de la capa. Tras aplicar la función de activación, $g(\cdot)$, al vector intermedio, $\mathbf{z} \in \mathbb{R}^h$, se obtiene:

$$\mathbf{h}_1 = g(\mathbf{z}_1). \quad (18.17)$$

La salida de una capa intermedia, $\mathbf{h}_i \in \mathbb{R}^h$, también está formada por variables intermedias que sirven de entrada a la siguiente capa. La función a calcular en la siguiente capa será por tanto:

$$\mathbf{h}_2 = g(\mathbf{W}'_{(2)} \mathbf{h}_1 + \mathbf{b}_2). \quad (18.18)$$

Siguiendo el mismo razonamiento, la salida de la última capa, \hat{y} , y por tanto de la red, vendrá dada por:

$$\hat{y} = g(\mathbf{W}'_{(n)} \mathbf{h}_{n-1} + \mathbf{b}_n). \quad (18.19)$$

Por ejemplo, si se tiene una red de tres capas la salida podrá calcularse como:

$$\hat{y} = g(\mathbf{W}'_{(3)} g(\mathbf{W}'_{(2)} g(\mathbf{W}'_{(1)} \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3). \quad (18.20)$$

Para entrenar y ajustar los pesos de este tipo de redes es necesario realizar el ajuste de la combinación de todos los pesos de la red. De forma similar a la búsqueda de los pesos de una

18.7. Perceptrón multicapa

309

sola neurona, será necesario encontrar la combinación de valores que clasifiquen bien todas las muestras del conjunto de entrenamiento o, en su defecto, que fallen en el menor número de muestras posible o minimicen alguna otra función de coste. En este punto es donde entra en juego la propagación hacia atrás o *backpropagation*.

La propagación hacia atrás es el mecanismo por el que el MLP ajusta de forma iterativa los pesos de la red con el objetivo de minimizar una función de coste que mide lo bueno o malo que es el resultado obtenido en un momento determinado (Rumelhart et al., 1986). Su único requisito de aplicación es que todas las operaciones de la red (incluidas las funciones de activación) sean diferenciables ya que se utiliza el algoritmo del descenso del gradiente para optimizar la función de coste.

El MLP utiliza diferentes funciones de coste o pérdida según el tipo de problema a resolver. Para los problemas de clasificación, la función de coste más utilizada es la Entropía Cruzada Media (en inglés Average Cross-Entropy). Para un problema binario esta función de coste se calcula como;

$$C(\hat{y}, y, \mathbf{W}) = -\frac{1}{n} \sum_{i=0}^n (y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)) + \frac{\alpha}{2n} \|\mathbf{W}\|_2^2, \quad (18.21)$$

donde $\alpha \|\mathbf{W}\|_2^2$ con $\alpha > 0$ es un término de regularización, L2, también conocido como penalización ya que penaliza los modelos complejos. α es un hiperparámetro cuyo valor se establece manualmente.

Para los problemas de regresión, la función de coste se basa en el Error Cuadrático Medio (*Mean Squared Error*):

$$C(\hat{y}, y, \mathbf{W}) = \frac{1}{2n} \sum_{i=0}^n \|\hat{y}_i - y_i\|_2^2 + \frac{\alpha}{2n} \|\mathbf{W}\|_2^2. \quad (18.22)$$

Cada iteración en el proceso de aprendizaje estará compuesta entonces por dos etapas, una de propagación hacia adelante y otra de propagación hacia atrás. En la primera etapa se introducen los valores de entrada a la red y se propagan las operaciones y los resultados hasta obtener la salida final de la red. En la segunda, el gradiente de la función de coste es propagado hacia atrás para actualizar los valores de los pesos de todas las capas y acercarse más a los valores que minimizan la función de coste.

En el algoritmo del descenso del gradiente, $\nabla C_{\mathbf{W}}$ se calcula y deduce de \mathbf{W} . Formalmente esto puede expresarse como:

$$\mathbf{W}^{t+1} = \mathbf{W}' - \lambda \nabla C_{\mathbf{W}}^t, \quad (18.23)$$

donde t es el estado de la red en una iteración determinada y λ es la tasa de aprendizaje cuyo valor debe ser superior a 0.

Al igual que en el caso del perceptrón único, el entrenamiento terminará cuando se alcance un número máximo de iteraciones o la mejora en la función de coste entre dos iteraciones consecutivas no supere cierto umbral.

Durante el proceso de aprendizaje, es necesario guardar en memoria los resultados de cada una de las muestras del conjunto de entrenamiento. Si el número de muestras o el tamaño de la red son grandes, es posible que no se disponga del suficiente espacio. Para resolver este problema, en una iteración no se utiliza todo el conjunto de entrenamiento, sino que se utiliza un subconjunto del mismo llamado *batch*. El conjunto de entrenamiento se divide en cada iteración, por tanto, en un número de *batches* disjuntos con un número de muestras por *batch*. Atendiendo a esta división, es posible definir una serie de hiperparámetros:

- Tamaño del *batch*. Número de muestras utilizadas en cada iteración para actualizar los pesos.
- Número de épocas. Número de pasadas completas sobre el conjunto de entrenamiento hasta terminar el proceso de aprendizaje.
- Número de iteraciones por época. Será el resultado de dividir el número total de muestras por el tamaño del *batch*.

Por ejemplo, si se tiene un conjunto de 55000 muestras y el tamaño del *batch* es de 100, cada época tendrá 550 iteraciones.

18.8. Instalación de librerías de *deep learning* en R: Tensorflow/Keras

El framework que se va a utilizar en este libro para trabajar con técnicas de *deep learning* será Tensorflow/Keras, debido a que es uno de los más completos en la actualidad, permitiendo realizar una configuración completa del proceso de entrenamiento y trabajar con diversos tipos de redes neuronales.

Para poder utilizar Tensorflow/Keras en **R**, es necesario realizar la instalación de la librería fuera de **R**. Por ello, si ya se dispone de una instalación del mismo sería posible utilizarla. No obstante, se recomienda seguir los pasos indicados a continuación para tener una instalación nativa de Tensorflow/Keras asociada directamente a R.

- **Paso 1** - Librería de Tensorflow en **R**

El primer paso será instalar el paquete de **tensorflow** en **R** [].

```
install.packages("tensorflow")
```

A continuación, será necesario tener una instalación de Conda en el sistema. Los usuarios tanto de Windows como de Linux/Mac podrán realizar directamente la instalación de una versión de Conda denominada Mini-Conda en el instalador del siguiente paso, la cual sería la opción recomendada para no tener que realizar una instalación externa de manera adicional.

NOTA

Otra manera disponible para los usuarios de Windows, pero no recomendada por los autores de este libro salvo que ya se disponga de Anaconda instalado, sería la de utilizar el programa y la librería directamente dentro de Anaconda, instalando una versión de R directamente en el sistema a través del siguiente link:

<https://docs.anaconda.com/anaconda/install/windows/>

- **Paso 2 - Instalación de tensorflow y keras**

Para continuar la instalación se activará la librería de Tensorflow y se ejecutará la función *install_tensorflow*

```
library(tensorflow)
install_tensorflow()
```

Al ejecutar esta función, los usuarios deberán marcar “Y” para aceptar la instalación de Mini-Conda, descartando aceptar la utilización de cualquier otro sistema Conda que pueda estar instalado previamente.

También se puede ejecutar la función *install_keras* del paquete **keras** para instalar Tensorflow.

```
install.packages("keras")
library(keras)
install_keras()
```

- **Paso 3 - Confirmar la instalación**

Para confirmar la instalación, se puede comprobar con los siguientes comandos (la salida puede variar según el equipo, pero la línea final tiene que ser similar a la indicada):

```
library(tensorflow)
tf$constant("Hello Tensorflow")  
  
tf.Tensor(b'Hello Tensorflow', shape=(), dtype=string)
```

18.9. Ejemplo de red para clasificación en R

En esta sección se entrena una red neuronal artificial para reconocer o clasificar los dígitos manuscritos del conjunto de datos MNIST (https://en.wikipedia.org/wiki/MNIST_database). Cada una de las imágenes de este conjunto de datos tiene un tamaño de 28×28 píxeles en escala de grises. En vez de extraer una serie de variables a partir de cada imagen, en este caso se utilizan cada uno de los $28 \times 28 = 784$ píxeles como variable de entrada (Figura 18.8).

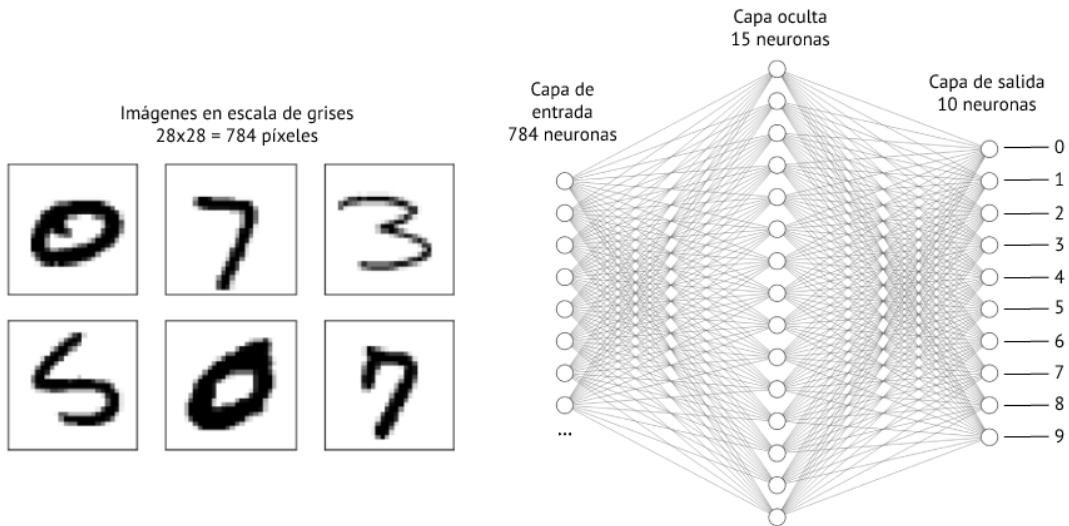


Figura 18.8: MLP para reconocimiento de dígitos manuscritos

18.9.1. Carga y visualización de los datos

El primer paso será cargar la librería **keras** que permite crear redes neuronales y conjunto de imágenes que se encuentra disponible públicamente:

```
library(keras)
mnist <- dataset_mnist()
```

A continuación, se puede ver el contenido de las variables generadas, donde cabe destacar que el conjunto de datos MNIST ya viene separado en dos subconjuntos, uno para entrenamiento y otro para test, compuestos por 60000 y 10000 imágenes respectivamente. En ambos casos, estos datos se almacenan en la variable de nombre *x*.

```
names(mnist)
#> [1] "train" "test"
dim(mnist$train$x)
#> [1] 60000    28    28
dim(mnist$train$y)
#> [1] 60000
dim(mnist$test$x)
#> [1] 10000    28    28
dim(mnist$test$y)
#> [1] 10000
```

Además, las imágenes de cada subconjunto vienen acompañadas de la clase a la que pertenecen

18.9. Ejemplo de red para clasificación en **R**

313

(dígito contenido en la imagen). En ambos casos, esta etiqueta se almacena en la variable con nombre *y*. A continuación se muestra un pequeño ejemplo que permitirá visualizar alguna de las imágenes contenidas en el conjunto de datos de entrenamiento junto con la etiqueta representando el dígito contenido:

```
par(mfcol=c(4, 4))
par(mar=c(0, 0, 3, 0), xaxs='i', yaxs='i')
for (j in 1:16) {
  im <- mnist$train$x[j, , ]
  im <- t(apply(im, 2, rev))
  image(x=1:28, y=1:28, z=im, col=gray((0:255)/255),
        xaxt='n', main=paste(mnist$train$y[j]))
}
```

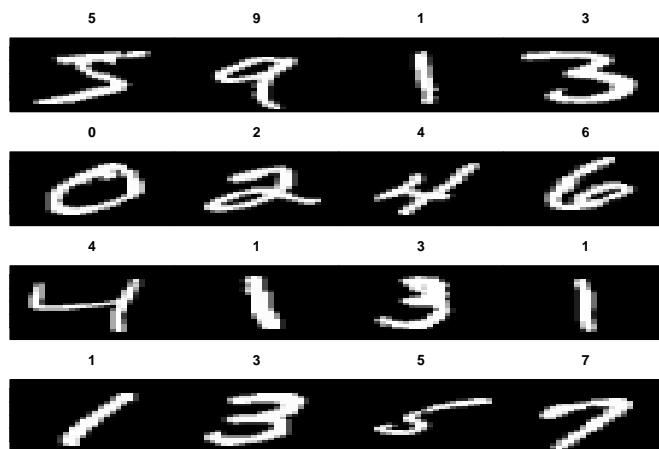


Figura 18.9: Algunas imágenes del conjunto de entrenamiento

18.9.2. Preprocesamiento

Una vez cargados los datos y comprobado su contenido, es posible realizar algún tipo de preprocesado. Dependiendo del tipo de problema se podrán realizar unas operaciones u otras. Por ejemplo, cuando se trabaja con imágenes es muy típico estandarizar los valores de color de las imágenes para mitigar las diferencias producidas por las diferentes condiciones de iluminación.

En este caso, solo se va a transformar los valores originales de la imagen (en rango de 0 a 255) a valores entre 0 y 1 dividiendo cada valor por el máximo, 255:

```
mnist$train$x <- mnist$train$x/255
mnist$test$x <- mnist$test$x/255
```

18.9.3. Generación de la red neuronal

El siguiente paso consiste en la generación de la red neuronal. Para ello, se define primero la estructura utilizando la interfaz *sequential* proporcionada por Tensorflow/Keras a través de la función *keras_model_sequential*:

```
model <- keras_model_sequential() |>
  layer_flatten(input_shape = c(28, 28)) |>
  layer_dense(units = 15, activation = "relu") |>
  layer_dense(10, activation = "softmax")
```

Como se puede observar, la red definida está compuesta por una capa de tipo *flatten* que se encarga de transformar los 28x28 valores a un vector de 784 elementos, para que a continuación una capa oculta *dense* de 15 neuronas con activación *relu* se encargue de realizar las primeras operaciones con esos datos. Al final, una última capa *dense* se encarga de obtener la probabilidad de que la imagen represente cada una de las posibles clases mediante una activación *softmax*¹:

```
summary(model, line_length=64)
```

```
#> Model: "sequential"
#>
#>   Layer (type)        Output Shape       Param #
#>   -----
#>   flatten (Flatten)    (None, 784)          0
#>   dense_1 (Dense)      (None, 15)           11775
#>   dense (Dense)        (None, 10)            160
#>   -----
#>   Total params: 11,935
#>   Trainable params: 11,935
#>   Non-trainable params: 0
#>   -----
```

Finalmente, es necesario compilar el modelo, indicando algunos de los parámetros de configuración necesarios para el proceso de entrenamiento, como la función de coste o pérdida, el optimizador a utilizar y las métricas a obtener:

```
model |>
  compile(
    loss = "sparse_categorical_crossentropy", # función utilizada para problemas de
    ↪ clasificación con varias clases
    optimizer = "sgd", # stochastic gradient descent
    metrics = "accuracy" # Precisión
  )
```

¹La activación *softmax* convierte un vector de números reales en una distribución de probabilidad de tal manera que la probabilidad de pertenecer a cada una de las categorías de salida siempre sume el 100 %.

18.9.4. Entrenamiento

Una vez generada la estructura de la red neuronal y definida la anterior configuración, es posible entrenarla mediante la función *fit()*. Para ello, se le debe indicar el conjunto de imágenes de entrenamiento, *x*, que debe utilizar y sus clases correspondientes, *y*. Además de otros parámetros, se podrá configurar el número de épocas, *epochs*, a entrenar (pasadas sobre el conjunto completo de entrenamiento), el tamaño del *batch* que se utilizará en cada iteración con *batch_size* (número de imágenes por iteración), qué porcentaje de elementos del conjunto de datos se utilizarán para validar el modelo con *validation_split* (imágenes utilizadas durante el entrenamiento pero solo para obtener una estimación real del error cometido) o la tasa de aprendizaje, *learning_rate*.

```
training_evolution <- model |>
  fit(
    x = mnist$train$x, y = mnist$train$y,
    epochs = 10, batch_size = 128,
    validation_split = 0.2,
    learning_rate = 0.1,
    verbose = 2
  )

#> Epoch 1/10
#> 375/375 - 2s - loss: 1.6313 - accuracy: 0.5266 - val_loss: 1.0455 - val_accuracy:
#>   0.7510 - 2s/epoch - 6ms/step
#> Epoch 2/10
#> 375/375 - 1s - loss: 0.8433 - accuracy: 0.7881 - val_loss: 0.6409 - val_accuracy:
#>   0.8434 - 1s/epoch - 3ms/step
#> Epoch 3/10
#> 375/375 - 1s - loss: 0.6022 - accuracy: 0.8427 - val_loss: 0.5031 - val_accuracy:
#>   0.8712 - 1s/epoch - 3ms/step
#> Epoch 4/10
#> 375/375 - 1s - loss: 0.5047 - accuracy: 0.8656 - val_loss: 0.4381 - val_accuracy:
#>   0.8830 - 1s/epoch - 3ms/step
#> Epoch 5/10
#> 375/375 - 1s - loss: 0.4526 - accuracy: 0.8767 - val_loss: 0.4019 - val_accuracy:
#>   0.8909 - 1s/epoch - 3ms/step
#> Epoch 6/10
#> 375/375 - 1s - loss: 0.4201 - accuracy: 0.8854 - val_loss: 0.3764 - val_accuracy:
#>   0.8959 - 1s/epoch - 3ms/step
#> Epoch 7/10
#> 375/375 - 1s - loss: 0.3976 - accuracy: 0.8896 - val_loss: 0.3593 - val_accuracy:
#>   0.8996 - 1s/epoch - 3ms/step
#> Epoch 8/10
#> 375/375 - 1s - loss: 0.3809 - accuracy: 0.8939 - val_loss: 0.3463 - val_accuracy:
#>   0.9022 - 1s/epoch - 3ms/step
#> Epoch 9/10
#> 375/375 - 1s - loss: 0.3678 - accuracy: 0.8975 - val_loss: 0.3359 - val_accuracy:
#>   0.9050 - 1s/epoch - 3ms/step
#> Epoch 10/10
```

```
#> 375/375 - 1s - loss: 0.3571 - accuracy: 0.8997 - val_loss: 0.3289 - val_accuracy:  
→ 0.9064 - 1s/epoch - 3ms/step
```

Tras el entrenamiento es posible ver su evolución mediante las gráficas de coste/pérdida y precisión:

```
plot(training_evolution)
```

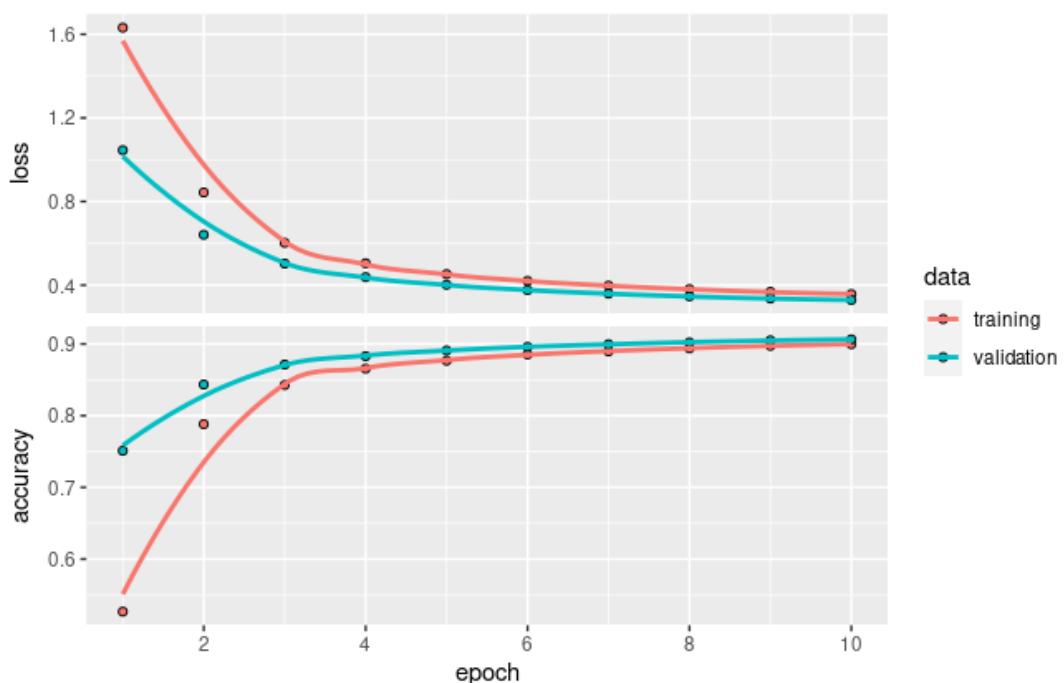


Figura 18.10: Evolución durante el entrenamiento de la función de precisión y de coste/pérdida de los conjuntos de entrenamiento y validación

Como se puede observar, la red entrenada tiene alrededor de un 90 % de precisión (porcentaje de aciertos al clasificar las imágenes) para las imágenes en los conjuntos de entrenamiento y validación. En el caso de la función de pérdida o coste, que mide el error cometido al realizar la clasificación, podemos ver como se reduce conforme la precisión del modelo aumenta.

18.9.5. Test

Una vez entrenado el modelo, es posible aplicarlo sobre el conjunto de test. Para ello, se puede realizar la predicción sobre cualquiera de las imágenes mediante la función *predict*, obteniendo la probabilidad de que pertenezca a una determinada clase:

18.9. Ejemplo de red para clasificación en **R**

317

```
predictions <- predict(model, mnist$test$x)
head(round(predictions, digits=3), 5)

#>      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]  [,10]
#> [1,] 0.000 0.000 0.000 0.003 0.000 0.000 0.000 0.995 0.000 0.002
#> [2,] 0.009 0.000 0.836 0.024 0.000 0.009 0.119 0.000 0.003 0.000
#> [3,] 0.000 0.962 0.013 0.006 0.001 0.001 0.003 0.002 0.010 0.002
#> [4,] 0.999 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
#> [5,] 0.001 0.000 0.007 0.000 0.836 0.004 0.011 0.012 0.017 0.111
```

También se puede utilizar la función *evaluate* para calcular tanto el coste o pérdida como la precisión de la red neuronal sobre el conjunto de test. Como se puede observar, se obtienen valores muy similares a los obtenidos durante el entrenamiento:

```
model |>
  evaluate(mnist$test$x, mnist$test$y, verbose = 0)

#>      loss  accuracy
#> 0.3310305 0.9045000
```

Con la función *predict* se puede también generar la matriz de confusión de la red para evaluar aciertos y fallos para cada clase:

```
prediction_matrix <- model |> predict(mnist$test$x) |> k_argmax()
confusion_matrix <- table(as.array(prediction_matrix), mnist$test$y)
confusion_matrix

#>
#>      0   1   2   3   4   5   6   7   8   9
#> 0 953  0 11  4  2 16 16  3  8  7
#> 1  0 1108 10  2  6  1  3 21 10  5
#> 2  4  3 901 27  5 11 14 27 13  6
#> 3  2  2 16 903  0 46  1  4 29 10
#> 4  1  0 16  0 899 16 12  9 11 43
#> 5  6  1  1 29  1 726  8  1 24 13
#> 6  9  4 19  3 10 21 902  0 10  0
#> 7  2  2 12 17  2 10  0 916 11 18
#> 8  3 15 35 20 10 38  2  3 839  9
#> 9  0  0 11  5 47  7  0 44 19 898
```

En la diagonal principal podemos observar el número de aciertos que obtiene el modelo entrenado para el conjunto de test, mientras que el resto de valores indican en cuantas ocasiones una clase es clasificada de manera incorrecta como otra diferente. A partir de esta matriz de confusión se puede calcular el valor de **accuracy** calculado mediante la función **evaluate** previa.

18.9.6. Guardado y reutilización del modelo

Finalmente, es posible almacenar el modelo entrenado mediante la función `save_model_tf`, que genera una carpeta con la red que se puede cargar y reutilizar mediante la función `load_model_tf`.

```
save_model_tf(object = model, filepath = "model")
reloaded_model <- load_model_tf("model")
round(predict(reloaded_model, mnist$test$x[1,1:28,1:28]), digits=4)

#>      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]  [,10]
#> [1,] 2e-04   0 1e-04 0.0028   0 1e-04   0 0.9948   0 0.002
```

18.10. Ejemplo de red para regresión en R

En esta sección se entrena una red neuronal artificial para predecir el precio de la vivienda según sus características en Madrid. Para ello se usará el dataset de `Madrid_Sale` disponibles en el paquete de *R* **Idealista18**, con datos inmobiliarios del año 2018 y que fue utilizado en el Cap. ???. Para ello, se tomarán las siguientes 7 variables que se usarán para realizar la estimación:

- *CONSTRUCTEDAREA*: metros cuadrados construidos.
- *ROOMNUMBER*: número de habitaciones.
- *BATHNUMBER*: número de baños.
- *HASLIFT*: si tiene ascensor.
- *DISTANCE_TO_CITY_CENTER*: distancia al centro de la ciudad.
- *DISTANCE_TO_METRO*: distancia a la parada de metro más cercana.
- *DISTANCE_TO_CASTELLANA*: distancia a la Castellana.

18.10.1. Carga y visualización de los datos

Considerando que ya se ha cargado previamente la librería `keras`, se carga el conjunto de datos indicando las variables a considerar:

```
library(idealista18)
data("Madrid_Sale")

variables <- c("CONSTRUCTEDAREA", "ROOMNUMBER", "BATHNUMBER",
              "HASLIFT", "DISTANCE_TO_CITY_CENTER", "DISTANCE_TO_METRO",
              "DISTANCE_TO_CASTELLANA")
x_madrid <- Madrid_Sale[variables]
x_madrid_mat <- unname(data.matrix(x_madrid))
y_madrid <- Madrid_Sale$PRICE
y_madrid_mat <- matrix(y_madrid, nrow = length(y_madrid), byrow = TRUE)
```

18.10. Ejemplo de red para regresión en **R**

319

El conjunto de datos contiene un total de 94815 elementos, que se dividirán en un 90 % para entrenamiento y un 10 % para test:

```
ind <- sample(c(TRUE, FALSE), length(y_madrid), replace=TRUE, prob=c(0.9, 0.1))
madrid_dat_train_x <- x_madrid_mat[ind, ]
madrid_dat_test_x <- x_madrid_mat[!ind, ]
madrid_dat_train_y <- y_madrid_mat[ind, ]
madrid_dat_test_y <- y_madrid_mat[!ind, ]
```

18.10.2. Preprocesamiento

Una vez cargados los datos y comprobado su contenido, es recomendable la normalización de las variables contenidas en el conjunto de datos debido a su heterogeneidad. Aunque sería posible para la red neuronal el adaptarse a esta situación, ciertamente puede complicar el proceso de entrenamiento haciéndola más imprecisa. Para ello, se utilizará la función `scale()` sobre las variables predictoras y se dividirá la variable del precio entre 100000 para reducir su escala:

```
madrid_dat_train_x <- scale(madrid_dat_train_x)
madrid_dat_test_x <- scale(madrid_dat_test_x)
madrid_dat_train_y <- madrid_dat_train_y/100000
madrid_dat_test_y <- madrid_dat_test_y/100000
```

18.10.3. Generación de la red neuronal

El siguiente paso consiste en la generación de la red neuronal. Para ello, al igual que en la sección 18.9.3, se define primero la estructura utilizando la interfaz *sequential* proporcionada por Tensorflow/Keras a través de la función `keras_model_sequential()`:

```
model <- keras_model_sequential() |>
  layer_dense(units=128, activation="relu", input_shape=7) |>
  layer_dense(units=64, activation="relu") |>
  layer_dense(units=16, activation="relu") |>
  layer_dense(units=1)
```

Como se puede observar, la red está compuesta por varias capas ocultas tipo *dense*, en las que las tres primeras tienen una activación *relu*. Al final, una última capa *dense* se encarga de obtener el valor de la estimación y, al contrario que en el ejemplo previo, no incluye ningún tipo de función de activación debido a que el valor de la misma ya es comprensible tanto para el modelo como para su interpretación. Esto sería equivalente a utilizar la función de activación lineal.

```
summary(model, line_length=64)
```

```
#> Model: "sequential_1"
#>
#>   Layer (type)        Output Shape       Param #
#>   =====
#>   dense_5 (Dense)     (None, 128)        1024
#>   dense_4 (Dense)     (None, 64)         8256
#>   dense_3 (Dense)     (None, 16)         1040
#>   dense_2 (Dense)     (None, 1)          17
#>   =====
#> Total params: 10,337
#> Trainable params: 10,337
#> Non-trainable params: 0
#> -----
```

Finalmente, se compila el modelo indicando los parámetros de configuración necesarios para el proceso de entrenamiento. En este caso la función de coste o pérdida se corresponderá con el Error Medio Cuadrático y la métrica con el error medio absoluto:

```
model |>
  compile(
    loss = "mse", # mean squared error
    optimizer = "sgd", # stochastic gradient descent
    metrics = "mae" # mean average error
  )
```

18.10.4. Entrenamiento

Una vez generada la estructura de la red neuronal y definida la anterior configuración, se entrena la misma utilizando la función `fit()`, configurando el resto de parámetros de forma similar a como se vio en la sección 18.9.4:

```
training_evolution <- model |>
  fit(
    x = madrid_dat_train_x, y = madrid_dat_train_y,
    epochs = 50, batch_size = 512,
    validation_split = 0.2,
    learning_rate = 0.1,
    verbose = 2
  )
```

Tras el entrenamiento es posible ver su evolución mediante las gráficas de coste/pérdida y error:

```
plot(training_evolution)
```

18.10. Ejemplo de red para regresión en **R**

321

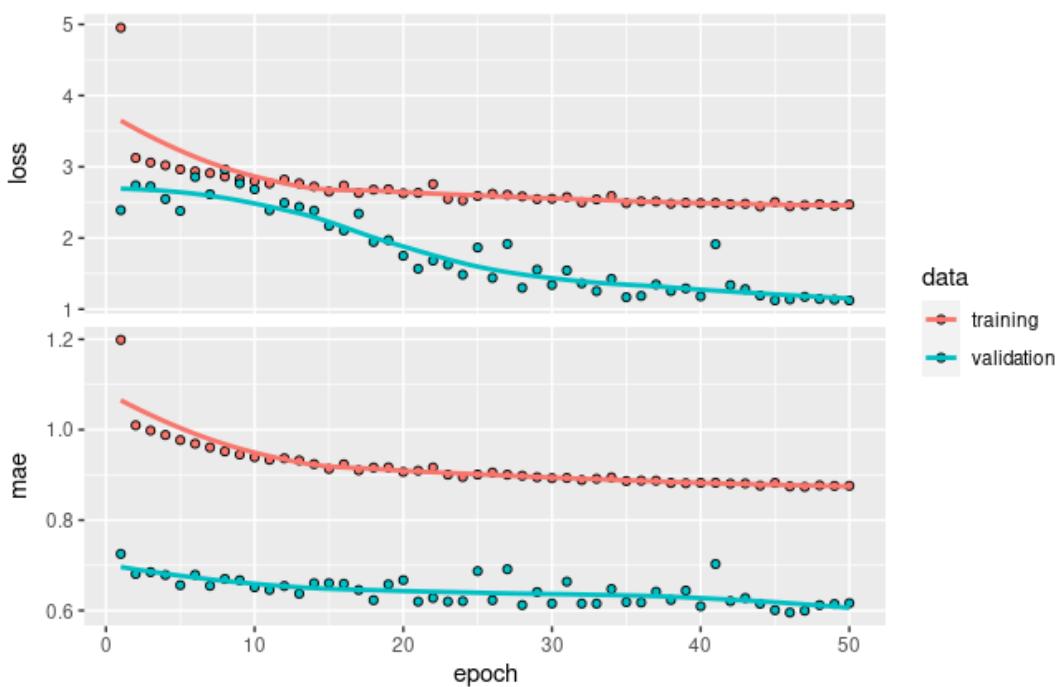


Figura 18.11: Evolución durante el entrenamiento de la precisión y la pérdida de los conjuntos de entrenamiento y validación

Como se puede observar, en este caso el modelo tiene aún posibilidad de mejora, ya que la pérdida sigue siendo alta y no se ha estancado, por lo que incrementando el número de épocas y el tiempo de entrenamiento se podría obtener un mejor resultado.

18.10.5. Test

Una vez entrenado el modelo, es posible aplicarlo sobre el conjunto de test mediante la función `predict()`, obteniendo la estimación para cada una de las viviendas:

```
predictions <- predict(model, madrid_dat_test_x)
head(predictions, 5)
#> [1]
#> [1,] 6.669374
#> [2,] 5.895504
#> [3,] 3.887646
#> [4,] 6.390513
#> [5,] 5.721725
```

Y mediante la función `evaluate()` se calcula tanto el coste o pérdida como el error de la red neuronal sobre el conjunto de test, el cual tendremos que multiplicar por 100000 para obtener el resultado en la escala original del conjunto de datos:

```
model |>
  evaluate(madrid_dat_test_x, madrid_dat_test_y, verbose = 0)
#> loss mae
#> 2.4195166 0.9227165
```

Resumen

- En este capítulo se ha explicado en detalle el concepto de redes neuronales artificiales, incluyendo los elementos que la componen, desde el perceptrón o neurona básica hasta el perceptrón multicapa, pasando el perceptron multiclase, junto al proceso de aprendizaje de los mismos.
- Además, se han definido las funciones de activación clásicas utilizadas en las redes neuronales artificiales, las cuales se encargan de transformar la suma ponderada de las entradas en el resultado final de la capa.
- Finalmente, se han explicado los pasos necesarios para poder entrenar una red neuronal artificial utilizando la librería Tensorflow/Keras en **R**, resolviendo el problema de clasificación de dígitos manuscritos representado en el conjunto de datos MNIST y un problema de regresión para estimar el precio de viviendas según sus características representado en el conjunto de datos de Idealista18.

Capítulo 19

Redes neuronales convolucionales

Noelia Vállez Enano^a y José Luis Espinosa Aranda^a

^aUniversidad de Castilla-La Mancha

19.1. Introducción

Las redes neuronales convolucionales (en inglés *Convolutional Neural Network*, CNN) son una extensión de las redes neuronales artificiales en las que se incluyen capas convolucionales para aprender a extraer, de forma automática, las características de los datos de entrenamiento al inicio de la arquitectura (Fig. 19.1).

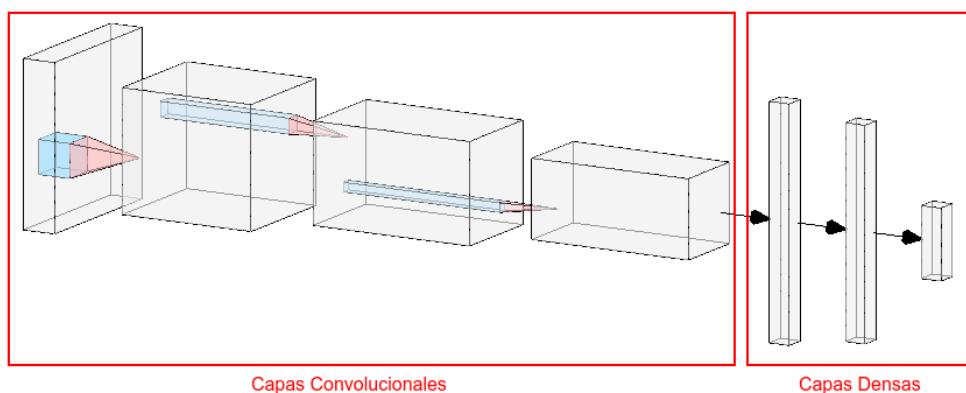


Figura 19.1: Estructura general de una CNN

Las primeras capas convolucionales de la red aprenden a extraer características generales de los datos de entrada mientras que las últimas capas convolucionales extraen características

mucho más específicas. Cuanto más larga es la red (o más *profunda*) mayor cantidad de detalles podrá aprender a distinguir. Esto es lo que ha propiciado la aparición del término “aprendizaje profundo” ([Goodfellow et al., 2016](#)).

Tras las capas convolucionales suelen encontrarse las capas *densas* o *totalmente conectadas* de la misma tipología de las vistas en el Cap. 18. Esta parte de la red será la encargada de realizar la clasificación de las muestras según los valores de las características extraídas en la parte convolucional. Por tanto, se dice que este tipo de redes tiene dos partes: una parte de extracción de características (realizada por la red convolucional) y una parte de clasificación o regresión (como las vistas en el Cap. 18).

19.2. Convolución

Aunque las redes neuronales artificiales (ANN) pueden utilizarse con los valores de color de una imagen como variables para reconocer qué hay en ella (ver Cap. 18), no es posible extraer información espacial de esta forma. Para lidiar con este problema, las CNN incorporan capas convolucionales para extraer características de las muestras de entrada, incluyendo información de la estructura espacial ([LeCun et al., 1995](#)).

Las convoluciones realizan una tarea similar al sistema visual humano, de hecho, se inspiran en cómo el ser humano percibe y procesa las características de los objetos. Aunque se diseñaron principalmente para ayudar a resolver tareas de visión por computador donde la entrada de la red es una imagen, es posible utilizarlas también con entradas vectoriales o series temporales.

Una convolución aplica un filtro sobre la entrada siguiendo un proceso de ventana deslizante. El filtro (o *kernel*) no es otra cosa que una matriz con unos pesos que se centrará en cada uno de los valores de la entrada para realizar una media ponderada de los valores de la entrada por los valores del filtro ([Garcia et al., 2015](#)). El tamaño de los filtros suele ser, por tanto, impar.

La Figura 19.2 muestra el resultado de aplicar la operación de convolución con un filtro de tamaño 3x3 sobre una matriz de entrada de tamaño 5x5. La salida será una matriz, M , de tamaño 3x3, donde cada elemento será la suma ponderada de multiplicar los elementos del filtro centrado en esa posición de la entrada por los valores de la entrada.

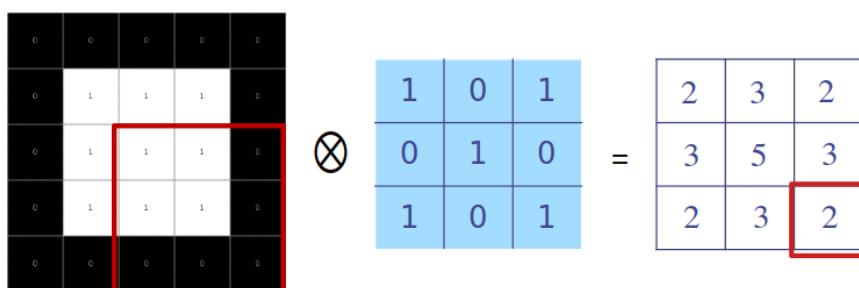


Figura 19.2: Ejemplo de convolución. De izquierda a derecha: entrada (negro = 0, blanco = 1), filtro y salida.

En general, una convolución en dos dimensiones se define como:

$$\mathbf{M}[x, y] = \sum_{s=-a}^a \sum_{t=-b}^b \mathbf{F}[s, t] \mathbf{I}[x - s, y - t], \quad (19.1)$$

donde \mathbf{F} es el filtro a aplicar, \mathbf{I} es la matriz de entrada, \mathbf{M} es la matriz de resultado que recibe también el nombre de “mapa de características” y a y b son los desplazamientos desde el centro del filtro a cualquier otro valor.

Por tanto, cada valor del ejemplo de la Figura 19.2 se obtiene como:

$$\begin{aligned} M_{1,1} &= 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 = 2 \\ M_{1,2} &= 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 3 \\ &\dots \\ M_{2,2} &= 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 5 \\ &\dots \\ M_{3,3} &= 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 = 2 \end{aligned} \quad (19.2)$$

La elección de los valores del filtro obtendrá matrices de salida que realcen o suavicen ciertas partes de la entrada. Por ejemplo, si la entrada es una imagen, es posible definir filtros que realcen los bordes, que los suavicen o incluso que detecten dichos bordes y cómo de marcados están. La Figura 19.3 muestra el resultado de aplicar distintos filtros a una imagen de entrada.

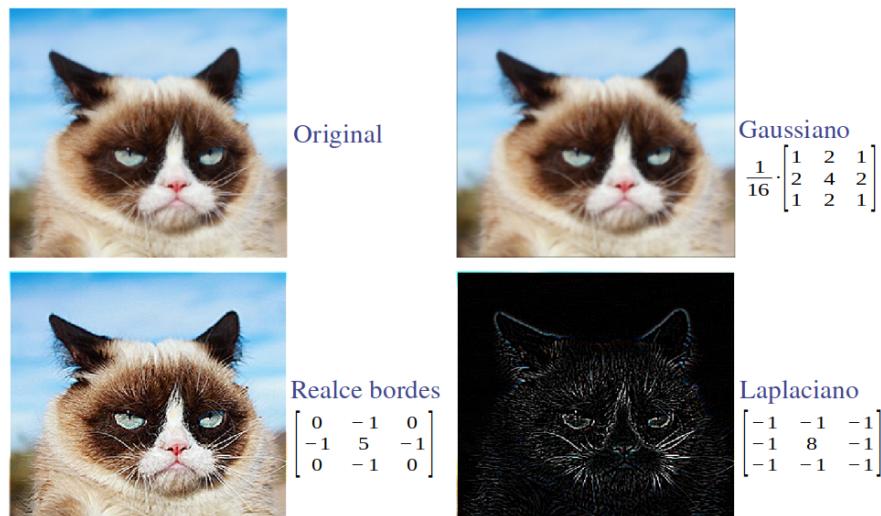


Figura 19.3: Resultado de aplicar diferentes filtros de convolución sobre una imagen dada.

Los valores (o pesos) de los filtros, que se ajustaban tradicionalmente, en un inicio, de forma manual según el problema a resolver. En los frameworks actuales estos filtros se ajustan durante

el proceso de entrenamiento de la CNN junto al resto de pesos de la red. Esto permite encontrar valores que maximicen la precisión final de la red.

19.3. Neuronas convolucionales

Las capas convolucionales de la CNN no estarán compuestas por perceptrones, sino por neuronas convolucionales que realizan las operaciones comentadas. Estas neuronas cuentan con matrices de pesos y no con vectores de pesos como lo hace el perceptrón. En este caso, tanto la entrada como la salida de la neurona son matrices. Para una neurona j , la salida \mathbf{Y}_j se calcula como la combinación lineal de las salidas de las neuronas de la capa anterior \mathbf{Y}_i operando cada una de ellas con el filtro \mathbf{F}_{ij} correspondiente a esa conexión de forma que:

$$\mathbf{Y}_j = g(\mathbf{B}_j + \sum \mathbf{F}_{ij} \otimes +\mathbf{Y}_i), \quad (19.3)$$

donde \mathbf{B}_j y g representan el *bias* y la función de activación respectivamente. La mayoría de las CNN utilizan la ReLU como activación o alguna variante de ésta. Esta activación funciona muy bien con el método del descenso del gradiente utilizado para encontrar los pesos.

Cada neurona dará lugar a un *mapa de activaciones*. La salida de una capa convolucional será entonces un conjunto de estos mapas (Fig. 19.4)

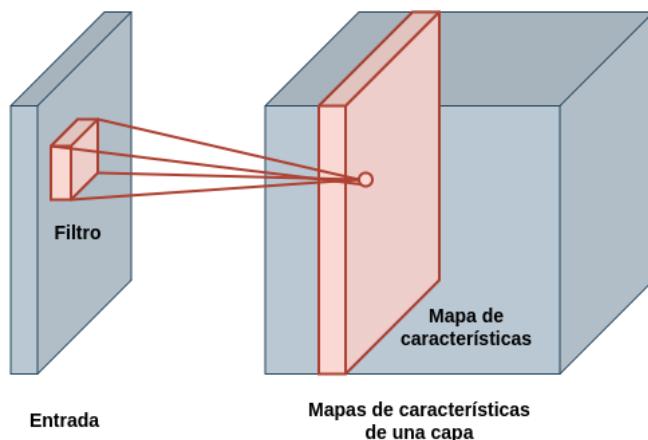


Figura 19.4: Conjunto de mapas de activaciones de una determinada capa. Cada filtro de la capa da lugar a un mapa diferente.

En el caso de que la entrada no sea una matriz 2D sino que sea una matriz 3D como, por ejemplo, una imagen, los filtros contarán con una tercera dimensión. La Figura 19.5 muestra algunos de los rellenos más empleados.

19.4. Relleno del borde

327

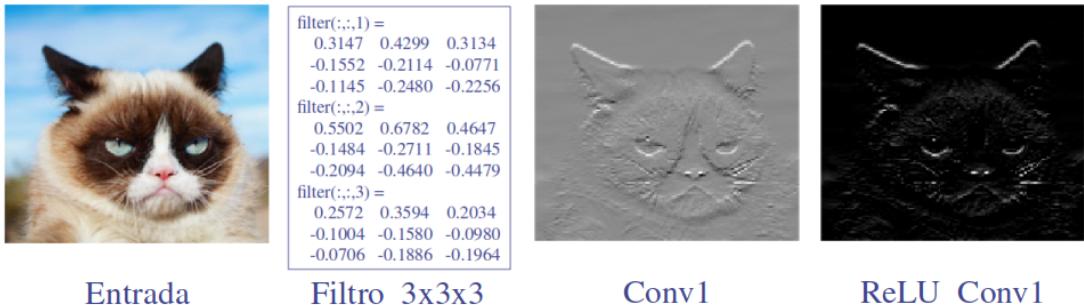


Figura 19.5: Resultado de aplicar un filtro 3D a una imagen antes y después de pasar por el filtro de activación

19.4. Relleno del borde

Si se aplica el filtro convolucional a una entrada, la salida será algo más pequeña al no poder centrar el filtro en los bordes de la matriz. Para poder hacerlo, se suele incrementar la entrada de la capa con un relleno (en inglés *padding*). El relleno se puede realizar con ceros, con algún valor, con el valor más cercano del borde, etc. La Figura 19.6 muestra algunos de los rellenos más empleados.

	Valor Constante	Valores del borde	Espejo

Figura 19.6: Distintos tipos de relleno del borde

19.4.1. Desplazamiento

El desplazamiento (en inglés *stride*) básico con el que se aplica un filtro convolucional es de 1. Sin embargo, la aplicación de muchos filtros repartidos en capas a lo largo de la red hace que sea especialmente difícil mantener todos los datos generados en un momento determinado del entrenamiento. Para reducir este volumen de datos, se suelen aplicar las convoluciones con un desplazamiento mayor que 1. Esto reduce el tamaño del mapa de activaciones obtenido por una determinada capa (Fig. 19.7).

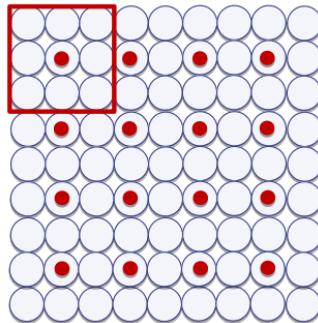


Figura 19.7: Desplazamiento 2x2 del filtro. El punto es el centro de la zona en la que se aplica el filtro en cada momento.

19.5. Capas de agrupación

La ejecución en secuencia de varias capas convolucionales es muy efectiva a la hora de decidir si ciertas características están o no presentes en la entrada. Sin embargo, una de sus ventajas y a la vez limitaciones es que mantiene la localización espacial de las características. Aunque es necesario cierta información espacial como, por ejemplo, el que hubiera unos bigotes cerca de una boca sería característico de una imagen que contuviese un gato, pequeños movimientos del contenido de la imagen producirían mapas de características diferentes.

Una forma de mitigar este problema es usar capas de agrupación (en inglés *pooling*). Estas capas agrupan un número de valores adyacentes de los mapas de características obteniendo un nuevo conjunto de mapas más pequeños. Es posible emplear distintos tipos de operaciones con las que realizar la agrupación. Los más empleados suelen ser el *max pooling* y el *average pooling* (Goodfellow et al., 2016) que seleccionan el máximo de los valores u obtienen su media respectivamente (Fig. 19.8). El tamaño más típico es 2x2.

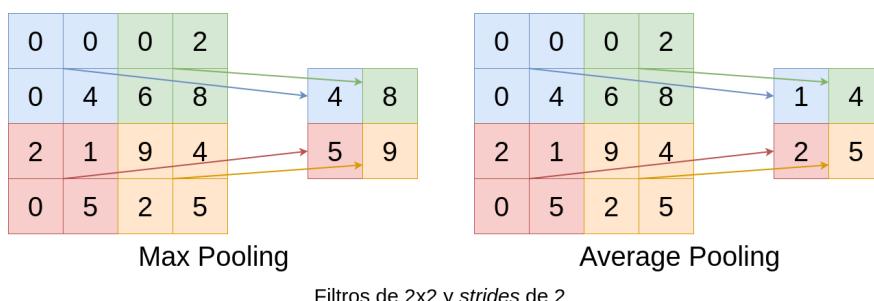


Figura 19.8: Resultado de emplear dos métodos de agrupación diferentes para reducir la dimensión de los datos

19.6. Desvanecimiento del gradiente

La primera red convolucional fue propuesta en 1982 ([Fukushima and Miyake, 1982](#)). Esta arquitectura recibió el nombre de Neocognitron y ya constaba de capas convolucionales y capas de *pooling*. Siguiendo la misma idea, en 1998 se diseñó otra CNN para resolver el problema de reconocimiento de dígitos manuscritos, MNIST ([LeCun et al., 1998](#)). A esta arquitectura de CNN se la conoce con el nombre de LeNet y es una de las arquitecturas más pequeñas que podemos definir para resolver un problema de clasificación (Fig. 19.9). El extracto de características, consta de dos capas convolucionales alternadas con 2 capas de *pooling* que obtienen un total de 400 variables. La parte final con el clasificador está compuesta por 3 capas densas de 120, 84 y 10 neuronas.

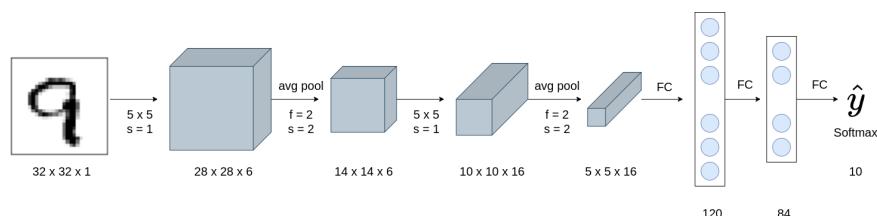


Figura 19.9: Arquitectura LeNet

A pesar de los buenos resultados obtenidos por la arquitectura, el uso de estos métodos para resolver problemas reales estaba aún lejos debido a la carga computacional requerida para su entrenamiento. No fue hasta el año 2012, cuando los ganadores del concurso ImageNet Challenge presentaron una nueva arquitectura llamada AlexNet, que las CNN volvieron a estar en el punto de mira de los investigadores ([Deng et al., 2012](#)). A partir de ese momento, y teniendo en cuenta los grandes avances computacionales de las tarjetas gráficas (GPU) que permitían ejecutar operaciones matriciales de forma eficiente, se empezaron a desarrollar cada vez más arquitecturas diferentes.

Durante los primeros años, las arquitecturas desarrolladas tenían cada vez más capas y más filtros en cada una de ellas para extraer la mayor cantidad de información posible de la entrada. Sin embargo, las arquitecturas más profundas se encontraron con un problema: el desvanecimiento del gradiente.

Ciertas funciones de activación como, por ejemplo, la sigmoide, comprimen el espacio de entrada entre 0 y 1. Esto hace que grandes cambios en la entrada produzcan cambios muy pequeños en la salida, haciendo que la derivada sea pequeña. Como los gradientes de la red se calculan durante la propagación hacia atrás capa a capa siguiendo la regla de la cadena, si los valores son muy cercanos a 0, la multiplicación de muchos de estos valores hará que el gradiente de la red caiga rápidamente. Un gradiente muy pequeño hará que los pesos de las capas iniciales apenas se modifiquen con cada iteración y no lleguen a obtener valores adecuados durante el entrenamiento.

Algunas soluciones a este problema son:

- El uso de activaciones tipo ReLU que no obtienen valores muy pequeños en su derivada.

- Capas de normalización. Si se normalizan los datos de entrada ya no habrá grandes cambios entre ellos y los valores estarán lejos de los extremos de la sigmoide.
- Uso de bloques con conexiones residuales que suman el valor de la entrada del bloque a su salida.

19.7. Sobreajuste

Cuanto mayor es el número de parámetros de la red, mayor probabilidad hay de que “memorice” los datos de entrenamiento. Esto se debe a la cantidad de características que la red es capaz de extraer y medir. Si la red es muy profunda, aprenderá cosas muy concretas del conjunto de entrenamiento, lo que dará lugar a modelos que no generalizan bien con nuevos datos (Tetko et al., 1995).

Además de esto, la no linealidad que añaden las funciones de activación puede hacer que se encuentren fronteras de decisión que modelen datos que no son linealmente separables, pero también facilita que se produzca el sobreajuste (Fig. 19.10).

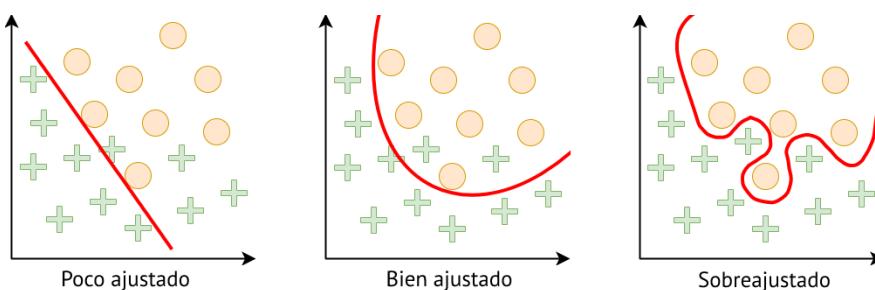


Figura 19.10: Tipos de ajuste del modelo a los datos

Para evitar que se produzca el sobreajuste se suelen emplear técnicas de regularización. Se trata de técnicas que impiden que los modelos sean demasiado complejos mejorando su capacidad de generalización. Algunas de estas técnicas son:

- *Dropout*. Durante el entrenamiento, algunas activaciones se ponen a 0 de forma aleatoria (entre el 10 % y el 50 %). Esto hace que una capa de la red no dependa siempre de los mismos nodos anteriores.
- *Early Stopping*. Se trata de parar el entrenamiento antes de que se produzca el sobreajuste y seleccionar ese modelo como final. Para ello se utilizan dos conjuntos: uno de entrenamiento y otro de validación. Cuando las curvas de pérdida de ambos conjuntos comienzan a diverger, se para el entrenamiento y se selecciona el modelo resultante del momento anterior al comienzo de la divergencia (Fig. 19.11).
- *Regularización L1*. Penaliza los pesos grandes por lo que fuerzan a los pesos a tener valores cercanos a 0 (sin ser 0). Añade un término de penalización a la función de coste sumando

19.8. Generación de datos de entrenamiento artificiales

331

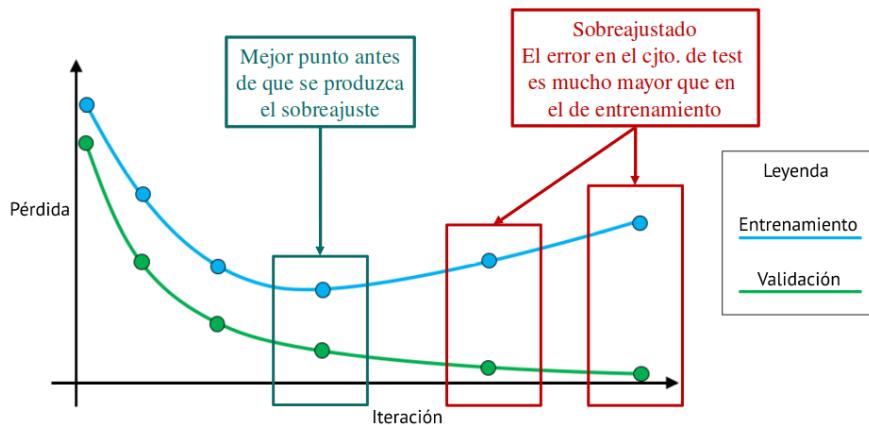


Figura 19.11: Selección del modelo antes del sobreajuste

todos los pesos de la matriz y multiplicado por un valor α que es otro hiperparámetro que debe ser seleccionado manualmente:

$$\alpha \|\mathbf{W}\|_1 = \alpha \sum_i \sum_j |w_{ij}|. \quad (19.4)$$

- **Regularización L2 o weight decay.** Parecida a la regularización L2 pero con una expresión algo diferente:

$$\frac{\alpha}{2} \|\mathbf{W}\|_2^2 = \frac{\alpha}{2} \sum_i \sum_j w_{ij}^2. \quad (19.5)$$

19.8. Generación de datos de entrenamiento artificiales

Como se ha comentado anteriormente, las técnicas de *deep learning* suelen requerir de gran cantidad de datos para su correcto funcionamiento. En muchas situaciones, se dispone de un conjunto limitado para poder entrenar los modelos de forma correcta, por lo que para tratar de suplir la falta de datos se recurre a la generación de datos artificiales, técnica conocida con el nombre de, con la expresión en inglés, *data augmentation* (Shorten and Khoshgoftaar, 2019).

Esta técnica realiza pequeñas variaciones en los datos del conjunto de entrenamiento del que se dispone para obtener nuevos, manteniendo el significado semántico de los mismos. Esto también permite mejorar la generalización de los modelos. Por ejemplo, si se tienen imágenes donde una de ellas contiene un elemento de la clase *gato*, las modificaciones que se realicen deben permitir poder reconocer esa misma clase a partir de las imágenes modificadas.

Algunos ejemplos de técnicas de *data augmentation* en imagen pueden ser: la realización de rotaciones, modificación del contraste o cambios en la iluminación, reescalados, adición/eliminación de ruido o cambio en las proyecciones de las mismas.

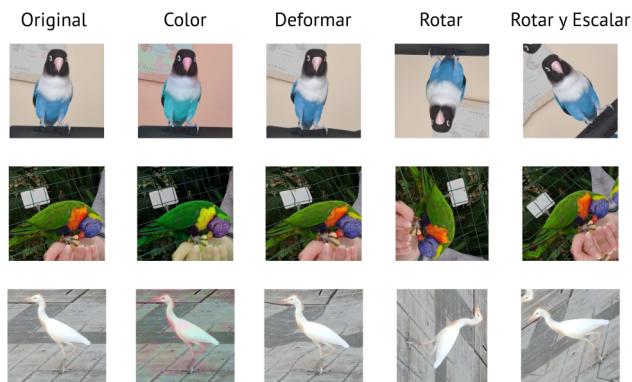


Figura 19.12: Ejemplos de técnicas de generación de datos artificiales

Para agregar diferentes tipologías de esta técnica de *data augmentation* en **R**, se pueden incluir capas de preprocessado en el modelo secuencial, que serán ejecutadas de manera aleatoria únicamente durante el entrenamiento. En el siguiente ejemplo se realizarán rotaciones aleatorias, volteados horizontales y acercamientos a la imagen:

```
data_augmentation <-
  keras_model_sequential() |>
  layer_random_rotation(0.1) |>
  layer_random_flip("horizontal") |>
  layer_random_zoom(0.1)
```

A continuación se muestran los diferentes tipos de preprocessado disponibles para imagen y redes neuronales convolucionales:

```
layer_random_crop()
layer_random_flip()
layer_random_flip()
layer_random_translation()
layer_random_rotation()
layer_random_zoom()
layer_random_height()
layer_random_width()
layer_random_contrast()
```

NOTA

Otros tipos de *data augmentation* disponibles en **keras** y **R** para otro tipo de datos pueden consultarse en
https://tensorflow.rstudio.com/guides/keras/preprocessing_layers

19.9. Ejemplo en R para el conjunto de datos CIFAR10

En esta sección se verá cómo entrenar una red neuronal convolucional para ser capaces de clasificar las clases contenidas en el conjunto de datos **CIFAR10**. La descarga debe hacerse a través del siguiente enlace https://drive.google.com/file/d/1-pFGg-bkooss1fNp5UNYR0-hLUMDP_XO/view?usp=sharing y el conjunto tiene que guardarse en una carpeta **data** dentro del proyecto de trabajo para asegurar la reproducibilidad del capítulo. Cada una de las imágenes contenidas en el mismo contiene un único elemento que puede ser clasificado como avión, coche, pájaro, gato, ciervo, perro, rana, caballo, barco o camión.

Nota

Existe otra versión del conjunto de datos denominada como **CIFAR100**, en la cual se definen un total de 100 posibles categorías en las que las imágenes contenidas pueden ser clasificadas. El ejemplo a continuación puede ser replicado con este mismo conjunto de datos.

https://www.rdocumentation.org/packages/keras/versions/2.9.0/topics/dataset_cifar100

Cada una de las imágenes contenidas en el conjunto tiene un tamaño de 32x32 píxeles en color, representándose mediante los 3 canales RGB, siendo diferente al ejemplo del capítulo 18, en el cual se trabaja con imágenes en escala de grises y, por tanto, un único canal.

A continuación, se verán los pasos seguidos, siendo de forma general muy similares a los ya descritos en el Cap. 18, pero adaptando la red al tipo de dato utilizado.

19.9.1. Carga y visualización de los datos

El primer paso será cargar la librería **keras**, para así poder crear las redes neuronales necesarias y también para cargar el conjunto de imágenes CIFAR10 que se encuentra disponible públicamente:

```
library(keras)
load("data/cifar10.RData")
```

A continuación, se puede ver el contenido de las variables generadas, donde se puede observar como el conjunto de datos CIFAR10 ya viene separado en dos subconjuntos que pueden ser utilizados para entrenamiento y para test. Además se puede ver que el conjunto de entrenamiento

está compuesto por 50000 imágenes, mientras que el conjunto de test por 10000. En ambos casos, estas imágenes se almacenan en la variable x .

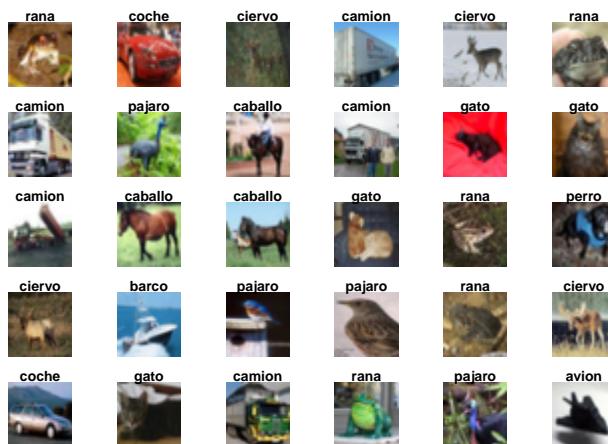
```
names(cifar)
#> [1] "train" "test"
dim(cifar$train$x)
#> [1] 50000   32    32     3
dim(cifar$train$y)
#> [1] 50000      1
dim(cifar$test$x)
#> [1] 10000   32    32     3
dim(cifar$test$y)
#> [1] 10000      1
```

Además, las imágenes de cada subconjunto tienen definida la clase a la que pertenecen, en este caso, cualquiera de las 10 clases indicadas anteriormente. En ambos casos, esta etiqueta se almacena en la variable y . A continuación, se muestra un pequeño ejemplo que permitirá mostrar alguna de las imágenes contenidas en el conjunto de datos de entrenamiento junto con su etiqueta:

```
class_names <- c('avion', 'coche', 'pajaro', 'gato', 'ciervo',
                 'perro', 'rana', 'caballo', 'barco', 'camion')

index <- 1:30

par(mfcol = c(5,6), mar = rep(1, 4), oma = rep(0.2, 4))
cifar$train$x[index,,,] |>
  purrr::array_tree(1) |>
  purrr::set_names(class_names[cifar$train$y[index] + 1]) |>
  purrr::map(as.raster, max = 255) |>
  purrr::iwalk(~{plot(.x); title(.y)})
```



19.9.2. Preprocesamiento

Una vez cargados los datos y comprobado su contenido, al igual que se explicó en el Cap. 18, es posible realizar algún tipo de preprocesado. Al estar trabajando con imágenes, es muy típico estandarizar los valores de color de las imágenes para mitigar las diferencias producidas por las diferentes condiciones de iluminación.

En este caso, al igual que en el Cap. 18, se va a transformar los valores originales de la imagen (en rango de 0 a 255) a valores entre 0 y 1 dividiendo cada valor por el máximo, 255:

```
cifar$train$x <- cifar$train$x/255
cifar$test$x <- cifar$test$x/255
```

19.9.3. Generación de la red neuronal

En esta ocasión se creará la red neuronal convolucional en dos pasos, para además mostrar cómo se pueden utilizar las funciones proporcionadas por la librería **keras** para definir una CNN en varias partes, combinándolas poco a poco.

En el primero, se definirá, utilizando la interfaz *sequential* proporcionada por Tensorflow/Keras a través de la función *keras_model_sequential*, la base convolucional de la red combinando varias capas *Conv2d* con *MaxPooling2D*. Esta es la parte de la red que se encargará de aprender las características necesarias que permitirán representar el contenido de la imagen. Otra de las diferencias principales que se puede observar en esta red es que, al aceptar imágenes de 3 canales, RGB, en vez de imágenes en escala de grises, el tamaño de la entrada de la primera capa tiene que reflejar esto *input_shape = c(32,32,3)*.

```
model <- keras_model_sequential() |>
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = "relu",
                 input_shape = c(32,32,3)) |>
  layer_max_pooling_2d(pool_size = c(2,2)) |>
  layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = "relu") |>
  layer_max_pooling_2d(pool_size = c(2,2)) |>
  layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = "relu")
```

Como se puede observar, en esta parte de la red se reduce la dimensión de la información de manera paulatina en cada capa, obteniendo las características representativas del objeto contenido en cada imagen hasta llegar a un tamaño de $4 \times 4 \times 64$.

```
summary(model, line_length=74)
```

```
#> Model: "sequential_2"
#> -----
#> Layer (type)          Output Shape         Param #
#> ======
```

```
#> conv2d_2 (Conv2D)           (None, 30, 30, 32)      896
#> max_pooling2d_1 (MaxPooling2D) (None, 15, 15, 32)      0
#> conv2d_1 (Conv2D)           (None, 13, 13, 64)     18496
#> max_pooling2d (MaxPooling2D) (None, 6, 6, 64)        0
#> conv2d (Conv2D)             (None, 4, 4, 64)      36928
#> =====
#> Total params: 56,320
#> Trainable params: 56,320
#> Non-trainable params: 0
#> -----
```

Ahora, será necesario añadir capas que permitan transformar los resultados de la parte convolucional de la red implementada a un valor de probabilidad de que la imagen represente cada una de las posibles clases de la imagen.

Para ello, primero se inserta una capa de tipo *flatten* que se encarga de transformar la salida de la última capa convolucional $4 \times 4 \times 64$ a un vector de 1024 elementos. A continuación, una capa oculta *dense* de 64 neuronas con activación *relu* se encarga de realizar las primeras operaciones con esos datos y de reducir la dimensionalidad. Finalmente, una última capa *dense* con activación *softmax* se encarga de obtener la probabilidad de que la imagen represente cada una de las 10 posibles clases:

```
model |>
  layer_flatten() |>
  layer_dense(units = 64, activation = "relu") |>
  layer_dense(units = 10, activation = "softmax")
```

A continuación, se puede observar como quedaría la estructura final del modelo implementado:

```
summary(model, line_length=74)
```

```
#> Model: "sequential_2"
#> -----
#> Layer (type)          Output Shape       Param #
#> =====
#> conv2d_2 (Conv2D)      (None, 30, 30, 32)      896
#> max_pooling2d_1 (MaxPooling2D) (None, 15, 15, 32)      0
#> conv2d_1 (Conv2D)      (None, 13, 13, 64)     18496
#> max_pooling2d (MaxPooling2D) (None, 6, 6, 64)        0
#> conv2d (Conv2D)        (None, 4, 4, 64)      36928
#> flatten_1 (Flatten)   (None, 1024)            0
#> dense_7 (Dense)       (None, 64)              65600
#> dense_6 (Dense)       (None, 10)              650
#> -----
#> Total params: 122,570
#> Trainable params: 122,570
```

19.9. Ejemplo en **R** para el conjunto de datos CIFAR10

337

```
#> Non-trainable params: 0
#> -----
```

NOTA

Un detalle a tener en cuenta con respecto al ejemplo del Cap. 18 es el parámetro *Total params*. Este valor indica el número de parámetros que contiene nuestra red neuronal y, en cierta manera, el tamaño de la misma. Se puede observar que en este caso tiene un mayor tamaño al contar con un total de 122570 parámetros con respecto a los 11935 del ejemplo previo.

Finalmente, es necesario compilar el modelo, indicando algunos de los parámetros de configuración necesarios para el proceso de entrenamiento, como serían el optimizador a utilizar, la función de coste y las métricas a calcular para poder evaluar la red entrenada:

```
model |> compile(
  optimizer = "sgd", # stochastic gradient descent
  loss = "sparse_categorical_crossentropy", # función utilizada para problemas de
  # clasificación con varias clases
  metrics = "accuracy" # Precisión
)
```

19.9.4. Entrenamiento

Una vez generada la estructura de la red neuronal convolucional, es posible entrenarla para resolver el problema de clasificación mediante la función *fit*. Para ello, se le debe indicar el conjunto de imágenes de entrenamiento, *x*, que debe utilizar y sus etiquetas correspondientes, *y*. Además de otros parámetros, se podrá configurar el número de *epochs* a entrenar (pasadas sobre el conjunto completo de entrenamiento), el tamaño del batch que se utilizará en cada iteración con *batch_size* (número de imágenes por iteración), qué porcentaje de elementos del conjunto de datos se utilizarán para validar el modelo con *validation_split* (imágenes utilizadas durante el entrenamiento pero solo para obtener una estimación real del error cometido) o la tasa de aprendizaje, *learning_rate*, entre otros.

```
training_evolution <- model |>
  fit(
    x = cifar$train$x, y = cifar$train$y,
    epochs = 10, batch_size = 32,
    validation_split = 0.2,
    learning_rate = 0.1,
    verbose = 2
  )
```

NOTA

Como se puede observar, el *batch_size* configurado es menor que el del Cap. 18 (32 vs 128). Esto es debido a que, el número máximo de imágenes que un mismo equipo utilizado para entrenar podrá procesar en una iteración vendrá determinado por el tamaño de la red neuronal, es decir, por la variable *Total params* indicada en la Nota anterior. Cuanto mayor sea el tamaño de la red, menor será el número máximo de imágenes que podrá tener el batch.

```
#> Epoch 1/10
#> 1250/1250 - 12s - loss: 2.1097 - accuracy: 0.2316 - val_loss: 1.9339 - val_accuracy:
→ 0.2958 - 12s/epoch - 9ms/step
#> Epoch 2/10
#> 1250/1250 - 8s - loss: 1.7478 - accuracy: 0.3667 - val_loss: 1.6987 - val_accuracy:
→ 0.3965 - 8s/epoch - 6ms/step
#> Epoch 3/10
#> 1250/1250 - 8s - loss: 1.5464 - accuracy: 0.4399 - val_loss: 1.4731 - val_accuracy:
→ 0.4707 - 8s/epoch - 7ms/step
#> Epoch 4/10
#> 1250/1250 - 9s - loss: 1.4304 - accuracy: 0.4866 - val_loss: 1.3653 - val_accuracy:
→ 0.5149 - 9s/epoch - 7ms/step
#> Epoch 5/10
#> 1250/1250 - 8s - loss: 1.3477 - accuracy: 0.5199 - val_loss: 1.3407 - val_accuracy:
→ 0.5257 - 8s/epoch - 6ms/step
#> Epoch 6/10
#> 1250/1250 - 7s - loss: 1.2784 - accuracy: 0.5437 - val_loss: 1.2563 - val_accuracy:
→ 0.5564 - 7s/epoch - 6ms/step
#> Epoch 7/10
#> 1250/1250 - 7s - loss: 1.2118 - accuracy: 0.5705 - val_loss: 1.2331 - val_accuracy:
→ 0.5720 - 7s/epoch - 6ms/step
#> Epoch 8/10
#> 1250/1250 - 8s - loss: 1.1539 - accuracy: 0.5954 - val_loss: 1.1807 - val_accuracy:
→ 0.5882 - 8s/epoch - 6ms/step
#> Epoch 9/10
#> 1250/1250 - 7s - loss: 1.1015 - accuracy: 0.6135 - val_loss: 1.1516 - val_accuracy:
→ 0.5935 - 7s/epoch - 6ms/step
#> Epoch 10/10
#> 1250/1250 - 7s - loss: 1.0526 - accuracy: 0.6286 - val_loss: 1.1014 - val_accuracy:
→ 0.6128 - 7s/epoch - 6ms/step
```

Tras el entrenamiento, se puede observar la evolución del mismo mediante las gráficas de coste/perdida y precisión.

```
plot(training_evolution)
```

Como se puede observar, la red entrenada es capaz de alcanzar un 60 % de precisión tanto en los conjuntos de entrenamiento como los de validación

19.9. Ejemplo en **R** para el conjunto de datos CIFAR10

339

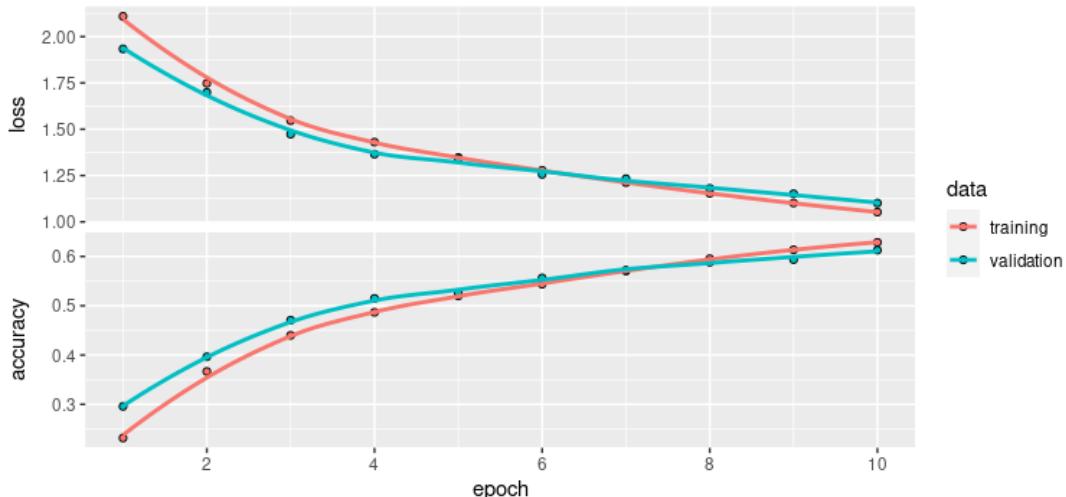


Figura 19.13: Evolución durante el entrenamiento de la precisión y la pérdida de los conjuntos de entrenamiento y validación

19.9.5. Test

Una vez entrenado el modelo, es posible aplicarlo sobre el conjunto de test proporcionado. Para ello, se puede realizar la predicción sobre cualquiera de las imágenes mediante la función `predict()`, obteniendo la probabilidad de que pertenezca a una determinada clase:

```
predictions <- predict(model, cifar$test$x)
head(round(predictions, digits=2), 5)
```

```
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,] 0.03 0.00 0.18 0.47 0.01 0.14 0.16    0 0.00 0.00
#> [2,] 0.04 0.07 0.00 0.00 0.00 0.00 0.00    0 0.89 0.00
#> [3,] 0.08 0.28 0.00 0.00 0.00 0.00 0.00    0 0.55 0.07
#> [4,] 0.82 0.01 0.04 0.00 0.00 0.00 0.00    0 0.11 0.00
#> [5,] 0.00 0.00 0.05 0.11 0.11 0.03 0.69    0 0.00 0.00
```

También se puede utilizar la función `evaluate()` para calcular tanto el coste o pérdida como la precisión de la red neuronal sobre el conjunto de test. Como se puede observar, se obtienen valores muy similares a los obtenidos durante el entrenamiento:

```
evaluate(model, cifar$test$x, cifar$test$y, verbose = 0)
```

```
#>      loss accuracy
#> 1.094381 0.611100
```

Con la función *predict* se puede también generar la matriz de confusión de la red para evaluar qué pares de clases está cometiendo un mayor número de errores:

```
prediction_matrix <- model |> predict(cifar$test$x) |> k_argmax()
confusion_matrix <- table(as.array(prediction_matrix), cifar$test$y)
confusion_matrix
```

```
#>
#>      0   1   2   3   4   5   6   7   8   9
#> 0 650  25  57  12  39  12  3  21  82  35
#> 1  43 790  15  18  13  11  16  12  43 179
#> 2 119  20 676 180 325 170 112  94  33  26
#> 3  23  15  57 427  76 184  54  48  18  18
#> 4   1   0  13  20 236  12   5  21   3   2
#> 5   6   7  52 145  41 488  24  82   5  10
#> 6  14   5  71 110 119  42 755  13   4  14
#> 7   7   6  30  46 126  60  16 676   9  18
#> 8 114  58  15  19  20  13   6   5 777  62
#> 9  23  74  14  23   5   8   9  28  26 636
```

19.9.6. Otros ejemplos de interés

- Transfer learning and fine tuning. Explicación de estas técnicas para clasificar imágenes que contienen perros y gatos. https://tensorflow.rstudio.com/guides/keras/transfer_learning
- <https://tensorflow.rstudio.com/guides/>
- <https://tensorflow.rstudio.com/examples/>

RESUMEN

- Este capítulo presenta las características de las redes neuronales convolucionales y sus diferencias con el perceptrón multicapa.
- Además, se exponen los principales problemas a la hora de diseñar este tipo de redes profundas y sus posibles soluciones.
- Finalmente, se han explicado los pasos necesarios para poder entrenar una red neuronal convolucional utilizando la librería Tensorflow/Keras en **R**, resolviendo el problema de clasificación de las 10 clases del conjunto de datos CIFAR10.

Parte IV

Ciencia de datos de texto y redes

Capítulo 20

Minería de textos

Víctor Casero-Alonso^a, Ángela Celis^a y María Lozano Zahonero^b

^aUniversidad de Castilla-La Mancha y ^b Università degli Studi di Roma Tor Vergata.

20.1. Introducción

En la actualidad, entre el 80 % y el 90 % de los datos que se generan diariamente son datos no estructurados (vistos en el Cap. ??). Un ejemplo típico de datos no estructurados son los textos, desde los comentarios o mensajes de las redes sociales, reseñas, blogs y microblogs, chats o whatsapp hasta las noticias periodísticas, los discursos políticos o las obras literarias. En consecuencia, aprender a procesar y analizar datos exige aprender a procesar y analizar textos.

Los textos precisan, sin embargo, un tratamiento especial. A diferencia de la mayoría de los datos que se tratan en este libro, que son datos estructurados, los datos textuales requieren que se les otorgue un orden y estructura para su manejo y análisis con el software R. Además, al utilizar un lenguaje natural –es decir, un idioma como, por ejemplo, el español, el chino o el inglés–, los textos no pueden ser procesados directamente por un ordenador. Es preciso “traducirlos” antes a un lenguaje formal que los ordenadores puedan entender.

La **minería de textos** (en inglés, *text mining*), también conocida como **análisis de textos** (en inglés, *text analysis*), puede definirse como el proceso para detectar, extraer, clasificar, analizar y visualizar la información no explícita que contienen los textos, transformando los datos textuales en datos estructurados y el lenguaje natural en lenguaje formal a fin de determinar, después, de manera automática, patrones recurrentes y desviaciones de los mismos. La minería de textos utiliza muchas técnicas y métodos diferentes, la mayor parte de los cuales proceden del **procesamiento del lenguaje natural** (PLN), un ámbito de la inteligencia artificial que se ocupa de la comunicación entre los seres humanos y las máquinas mediante el tratamiento computacional del lenguaje humano.

Este capítulo constituye una primera aproximación a la minería de textos con **R**. Su objetivo es proporcionar un marco teórico y aplicado básico de este ámbito. Para ello, en la Sec. 20.2, se presentan los conceptos y fases fundamentales de la minería de textos. La Sec. 20.3 está dedicada al análisis de sentimientos, que constituye uno de los campos de la minería de textos de mayor desarrollo en la actualidad. La Sec. 20.4 indica paquetes de **R** que permiten realizar análisis textuales de distintos tipos. Cierra el capítulo un ejemplo, en el que se aplica y se amplía lo estudiado anteriormente. Dos referencias útiles sobre el tema son [Fraudejas Rueda \(2022\)](#) y [Jockers \(2014\)](#).

20.2. Conceptos y tareas fundamentales

Lo primero que se necesita para hacer un análisis de textos son los textos. Esta afirmación podría parecer banal, pero no lo es. El volumen de textos en circulación es ingente, pero, en la mayor parte de los casos, es necesario realizar una serie de operaciones complejas para poder extraer y recopilar los datos textuales que se quiere analizar. Es también difícil muchas veces acceder después a estos datos, ya que los textos pueden presentar formatos muy heterogéneos, no siempre interpretables o fáciles de convertir en un formato interpretable. Baste pensar, por ejemplo, en una nota escrita a mano. Dado que este capítulo es una primera aproximación a la minería de textos, se parte del supuesto de que el texto o los textos están disponibles ya en un fichero, denominado **corpus**, legible por **R**. En este contexto, *corpus* es la colección de textos con el mismo origen, por ejemplo, el *corpus* de las obras de un autor, que para poder manejarse requiere metadatos con detalles adicionales.

20.2.1. Preparación de los datos

Una vez constituido el *corpus*, la primera fase es la **preparación de los datos**. Los textos suelen contener un cierto grado de “suciedad”, es decir, elementos que alteran o impiden el análisis. De una buena “limpieza” inicial, dependerá en gran parte la validez de los resultados que se obtengan. Entre las operaciones de “limpieza” generales figuran una serie de transformaciones cuya finalidad es evitar el recuento incorrecto de palabras, como el cambio de mayúsculas por minúsculas y la eliminación de los signos de puntuación, los números y los espacios en blanco en exceso.

La siguiente operación de preparación, que tiene un importante peso en el análisis, es la eliminación de las **palabras vacías** (en inglés, **stopwords**). En la lengua no todas las palabras tienen el mismo tipo de significado. Las palabras con significado léxico, como *mesa* o *corpus*, son palabras a las que corresponde un concepto que se puede definir o explicar. Otras palabras, sin embargo, son palabras funcionales, cuyo contenido es puramente gramatical. Son palabras como el artículo *el*, la preposición *de* o la conjunción *o*: se puede explicar cómo se usan, pero no definirlas asociándolas a un concepto porque carecen de contenido léxico-semántico.

Las palabras vacías son, con gran diferencia respecto de las palabras léxicas, las más frecuentes de la lengua, pero, dado su escaso o nulo significado léxico, en los análisis de tipo semántico, como el análisis de sentimientos o el modelado de temas, carecen de valor informativo, por lo que es conveniente eliminarlas. No es aconsejable eliminarlas, sin embargo, en otros tipos de

análisis, como los análisis estilométricos, donde tienen un importante valor informativo como se verá en la Sec. 20.2.4. Las palabras vacías pertenecen a clases cerradas, es decir, a clases de palabras con un número de elementos limitado, finito. Es posible confeccionar, por tanto, listas de palabras vacías para facilitar su eliminación. En el ejemplo de aplicación que se verá en la Sec. 20.5, se aprenderá a usar estas listas y se podrá apreciar con detalle la diferente información que proporciona una tabla de frecuencias con y sin palabras vacías.

20.2.2. Segmentación del texto: tokenización

La segunda fase de la minería de textos consiste en la **segmentación del texto**, denominada también **tokenización**. El texto se divide en ***tokens***, secuencias de texto con valor informativo. De esta manera, se pasa del lenguaje natural a un lenguaje formal comprensible por el software, dándole formato de ‘vector’ o ‘tabla’. Así se pueden aplicar algunas de las herramientas que se utilizan con datos numéricos para manejar el texto y obtener resúmenes y visualizaciones que muestren la información no explícita contenida en él en forma de patrones recurrentes.

Generalmente, los *tokens* son **palabras**, es decir, secuencias de caracteres entre dos espacios en blanco y/o signos de puntuación, pero pueden ser también **oraciones**, **líneas**, **párrafos** o **n-gramas**. Como se verá en el ejemplo de aplicación, un primer análisis del significado consiste en eliminar las palabras vacías y obtener las frecuencias¹ de las palabras con valor informativo para responder a la pregunta “¿Qué se dice?” (Silge and Robinson, 2017).

Nota

También puede ser útil obtener la **tasa de riqueza léxica** (TTR, del inglés *type-token ratio*). Esta mide la relación entre el número de palabras diferentes que contiene un texto (*types*) dividido entre las palabras totales de ese texto (*tokens*)^a.

$$TTR = \frac{\text{Types}}{\text{Tokens}}$$

^aVéase <https://www.fundeu.es/consideraciones-teoricas/>

20.2.2.1. N-gramas

El análisis puede proseguir estudiando la frecuencia de los **n-gramas**, secuencias de *n* palabras consecutivas en el mismo orden. Se tienen así bigramas o 2-gramas (secuencias de dos palabras), trigramas o 3-gramas (secuencias de tres palabras), etc. El estudio de los *n-gramas* responde al principio de Firth: “*You shall know a word by the company it keeps*” (Firth, 1957, 11). Este principio es el fundamento del llamado **análisis de colocaciones**: para conocer el significado de una palabra es preciso conocer las palabras con las que aparece, el contexto relevante. En un sentido amplio, el análisis de colocaciones consiste en examinar los contextos izquierdo y/o derecho de una palabra. La segmentación en *n-gramas* permite tener en cuenta este contexto relevante que indicará, por ejemplo, que *banco* es, con toda probabilidad, un asiento en las

¹Frecuencias relativas si se comparan distintos textos.

secuencias *banco de madera* o *banco en la terraza*, pero no lo es en secuencias como *banco de peces*, *banco de arena*, *banco de inversiones*, *banco de datos* o *banco de pruebas*. La división en *n-gramas* permitirá también considerar en el análisis, al menos hasta cierto punto, el peso de la ambigüedad, la negación o el distinto significado que pueden tener las palabras según el ámbito temático. Por ejemplo, la forma *larga* no tiene el mismo significado en los bigramas *falda larga*, *mano larga* y *cara larga*, ni tiene tampoco el mismo valor informativo en *es larga / no es larga* o en *de larga experiencia* (valor positivo) y en *se me hizo larga* (valor negativo). En el ejemplo de aplicación (Sec. 20.5), se verá la segmentación en *n-gramas* en la práctica, y cómo la visualización de redes contribuye a complementar el análisis.

20.2.3. *Stemming* y lematización

La tokenización se puede refinar mediante el ***stemming***, o reducción de las palabras “flexionadas” a su raíz, y la **lematización**, o extracción del lema de cada palabra. Un ejemplo de *stemming* sería reducir las palabras *texto*, *textos*, *textual* y *textuales*, que R cuenta como cuatro palabras diferentes, a la raíz “text”. El *stemming* puede proporcionar un recuento más preciso en algunos casos, pero en otros, al eliminar los sufijos de las palabras, puede crear confusión. Además, como en el ejemplo anterior, las raíces pueden no coincidir con palabras existentes, lo que hace que sean difíciles de interpretar y resulten extrañas si se visualizan en nubes de palabras. Con la lematización se reducen las formas flexionadas de una misma palabra al lema, que es la forma que encabeza la entrada de la palabra en el diccionario. Por ejemplo, si se quiere buscar el significado de la palabra *niñas* no se encontrará como tal sino bajo el lema *niño* y si se quiere buscar *iremos* se tendrá que buscar el lema *ir*. En el caso anterior, la lematización reduciría las formas *texto*, *textos*, *textual* y *textuales* a dos lemas: *texto* y *textual*. La lematización evita la dispersión de significado en varias formas, pero a veces es compleja y puede conducir a la pérdida de información pertinente.

20.2.4. Campos de aplicación de la minería de textos

La minería de textos tiene varios campos de aplicación. Entre ellos destacan tres:

1. El **análisis de sentimientos** se tratará con detalle en la Sec. 20.3 y en el ejemplo de aplicación (Sec. 20.5.4).
2. El **modelado de temas o tópicos** (en inglés, *topic modelling*), como su propio nombre indica, tiene por objeto identificar los temas principales sobre los que versa el texto haciendo uso de técnicas de clasificación no supervisada del campo del aprendizaje automático, como por ejemplo LDA (*Latent Dirichlet Allocation*). Se ilustrará en el Cap. 41.
3. La **estilometría** o **análisis estilométrico** es una aplicación de la minería de textos cuya finalidad consiste en determinar las relaciones existentes entre el estilo de los textos y los metadatos incluidos en ellos. Se utiliza principalmente en la atribución de autoría. El concepto base es el de **huella lingüística**, constituida por el conjunto de rasgos lingüísticos que caracterizan el estilo de un autor como un estilo individual y único y permiten identificarlo. Un punto clave es que, contrariamente a lo que podría pensarse, los rasgos

que conforman en mayor medida la huella lingüística son los que tienen un mayor índice de frecuencia. La mayor parte de los enfoques utilizan el vector de las “palabras más frecuentes” (MFW, por sus siglas en inglés), que son, como se ha visto antes, las palabras vacías y no las palabras con significado léxico, para determinar el estilo de un autor. Esto es debido fundamentalmente a que las palabras vacías se usan de manera involuntaria e inconsciente, configurando de esta manera, sin ningún tipo de filtros racionales, una clave estilística idiosincrásica ([Lozano Zahonero, 2020](#)). De lo anterior se deduce fácilmente que en este tipo de análisis no deben eliminarse las palabras vacías.

En la actualidad, el análisis estilométrico se usa en ámbitos muy dispares: desde la criminología o los servicios de inteligencia para identificar a los autores de mensajes o notas en casos de asesinatos, terrorismo, secuestro o acoso, por ejemplo, hasta el derecho civil o la literatura en cuestiones de derechos de autor o detección de plagio, entre muchas otras cuestiones.

20.3. Análisis de sentimientos

El **análisis de sentimientos** (en inglés, *sentiment analysis*) es una aplicación de la minería de textos que tiene como finalidad la detección, extracción, clasificación, análisis y visualización de la dimensión subjetiva asociada a los temas o tópicos presentes en los textos. La dimensión subjetiva comprende no solo los sentimientos, sino también las **emociones**, sensaciones y estados afectivos y anímicos, así como las opiniones, creencias, percepciones, puntos de vista, actitudes, juicios y valoraciones. De ahí que reciba también el nombre de **minería de opinión** (en inglés, *opinion mining*) ([Lozano Zahonero, 2020](#)).

El análisis de sentimientos asigna a esta dimensión subjetiva una polaridad, que puede ser positiva o negativa ([Pang and Lee, 2008](#)). Algunas técnicas añaden además una polaridad neutra. En algunos casos, el análisis de sentimientos se refina hasta llegar a las emociones básicas: este subcampo del análisis de sentimientos se conoce como **detección de emociones**.

La primera aplicación del análisis de sentimientos fue la investigación de mercados. A partir del año 2000, se registra un crecimiento exponencial de textos como reseñas, chats, foros, blogs, microblogs o comentarios y mensajes de las redes sociales, en los que predomina la expresión de emociones y opiniones personales. Mediante el análisis de sentimientos se extrae de ellos información que permite conocer los gustos del consumidor y diseñar productos a su medida. Esta idea se extenderá después a otros ámbitos, en especial a aquellos en los que predomina la comunicación persuasiva como las campañas publicitarias o políticas. Recientemente, ha empezado a utilizarse también con fines predictivos y preventivos en muchas esferas: desde cuáles son los políticos, las empresas, las películas, canciones u obras literarias que obtendrán un mayor rendimiento, mejores resultados o más votos o ventas hasta cómo detectar y prevenir, por ejemplo, conductas suicidas mediante el análisis de mensajes en las redes sociales.

En el análisis de sentimientos y la detección de emociones existen dos enfoques principales: el enfoque basado en el aprendizaje automático (*machine learning*), en el que se usan algoritmos de aprendizaje supervisado, y el enfoque semántico basado en diccionarios o **lexicones**. Este último enfoque es el que se verá en detalle en el ejemplo de aplicación.

En **R** están implementados varios lexicones para el análisis de sentimientos. Dos de los más utilizados son **bing**, de Bing Liu y colaboradores ([Liu, 2015](#)), y **NRC**, de Saif Mohammad y Peter Turney, ambos incluidos tanto en el paquete **tidytext** como en **syuzhet** ([Jockers, 2017](#)). Estos lexicones tienen en común que están basados en unigramas, es decir, en palabras sueltas, y que tienen como idioma original el inglés, si bien hay versiones traducidas automáticamente a distintas lenguas. La diferencia principal entre los dos lexicones es que **bing** clasifica las palabras de forma binaria en polaridad positiva/negativa, mientras que **NRC** además de la polaridad positiva/negativa permite detectar también ocho emociones básicas (*ira, miedo, anticipación, confianza, sorpresa, tristeza, alegría, asco*). En el ejemplo de aplicación se compararán ambos diccionarios. Como se verá, los resultados del análisis dependerán en buena medida del lexicón elegido, así como del idioma del texto y de si el lexicón se elaboró originalmente en ese idioma o es una versión traducida automáticamente de otra lengua.

20.4. Minería de textos en R

En **R** existen diversos paquetes y funciones que facilitan la minería de textos, entre los que destacan:

- **tidytext**: con la filosofía del **tidyverse**, puede combinarse con los conocidos paquetes **dplyr**, **broom**, **ggplot2**, etc. Se puede destacar la función **unnest_tokens()**, que automatiza el proceso de *tokenización* y el almacenamiento en formato *tidy* en un único paso.
- **tm**: destaca por tener soporte *back-end* de base de datos integrada, gestión avanzada de metadatos y soporte nativo para leer en varios formatos de archivo.
- **tokenizers**: incluye *tokenizadores* de palabras, oraciones, párrafos, *n*-gramas, *tweets*, expresiones regulares, así como funciones para contar caracteres, palabras y oraciones, y para dividir textos más largos en documentos separados, cada uno con el mismo número de palabras.
- **wordcloud**: permite visualizar **nubes de palabras**. Las palabras más frecuentes aparecen en mayor tamaño permitiendo de un vistazo obtener las palabras clave del texto.
- **quanteda**: maneja **matrices de documentos-términos** y destaca en tareas cuantitativas como recuento de palabras o sílabas.
- **syuzhet**: incluye distintas funciones que facilitan el análisis de textos, en particular el *análisis de sentimientos* de textos literarios.
- **gutenbergr**: almacena las obras del proyecto Gutenberg²; muy útil si se quieren analizar textos literarios.

²Proyecto desarrollado por Michael Hart en 1971 para crear una biblioteca de libros electrónicos gratuitos, y accesibles en internet, a partir de libros en soporte físico, generalmente de dominio público. Cuenta con más de 50 000 libros.

20.5. Ejemplo de aplicación

20.5.1. Declaración institucional del Estado de Alarma 2020

La “Declaración institucional del presidente del Gobierno anunciando el Estado de Alarma en la crisis del coronavirus” (en adelante, “la Declaración”), dada en La Moncloa el 13 de marzo de 2020 es el objeto de análisis. Esta se puede encontrar en el paquete CDR que acompaña este libro. Se le van a aplicar las operaciones y técnicas mencionadas en la Sec. 20.2.

```
library("CDR")
data("declaracion")
```

20.5.2. Segmentación en palabras y oraciones

Las primeras tareas del análisis son la preparación, limpieza y segmentación o tokenización de los textos, como se vió en las Sec. 20.2.1 y 20.2.2. A continuación, se verá una segmentación en palabras individuales. La función `tokenize_words()` del paquete `tokenizers` prepara el texto convirtiéndolo a minúsculas, elimina todos los signos de puntuación y finalmente segmenta el texto en palabras.

```
library("tokenizers")
palabras <- tokenize_words(declaracion)
tokenizers::count_words(declaracion)
#> [1] 922
```

Con la última sentencia se obtiene la longitud de la Declaración, el número de palabras utilizadas: 922.

La frecuencia de cada palabra se puede obtener y presentar con el código de abajo. La primera sentencia crea la tabla de frecuencias, la tercera la transforma en el tipo `tibble`, creando la columna recuento, y ordena la tabla de forma descendente, de mayor a menor frecuencia.

```
library("tidyverse")
tabla <- table(palabras[[1]])
( tabla <- tibble(palabra = names(tabla),
                  recuento = as.numeric(tabla)) |>
    arrange(desc(recuento)) )
#> # A tibble: 390 x 2
#>   palabra      recuento
#>   <chr>        <dbl>
#> 1 de            43
#> 2 y             41
#> 3 la            35
#> 4 a             31
#> 5 los           26
```

```
#> 6 en      22
#> 7 que     20
#> 8 el      17
#> 9 al      14
#> 10 para   14
#> # ... with 380 more rows
```

En la primera fila de la salida se indican las dimensiones de la `tibble`, por lo que se puede ver que en esta Declaración hay 390 “palabras” distintas (considera los números como palabras).

El resultado son las palabras más utilizadas en el texto, que, como puede apreciarse, son palabras vacías. Esto no debería sorprender porque, como ya se ha visto, estas palabras son las más frecuentes. En la siguiente Sección, se verá cómo eliminarlas para obtener datos con valor informativo.

Para otras formas de segmentar el texto (oraciones, párrafos, *tweets*, etc.): véase `?tokenize_words`. Por ejemplo, para segmentar en oraciones:

```
oraciones <- tokenize_sentences(declaracion)
count_sentences(declaracion)
#> [1] 44
```

Las tres primeras oraciones y la última se obtienen con el siguiente código.

```
oraciones[[1]][1:3] # primeras 3 oraciones
#> [1] "Buenas tardes."
←
#> [2] "Estimados compatriotas."
←
#> [3] "En el día de hoy, acabo de comunicar al Jefe del Estado la celebración, mañana,
← de un Consejo de Ministros extraordinario, para decretar el Estado de Alarma en
← todo nuestro país, en toda España, durante los próximos 15 días."
oraciones[[1]][count_sentences(declaracion)] # última oración
#> [1] "Buenas tardes."
```

También podría medirse la longitud de cada oración, en número de palabras, normalmente para comparaciones con otros textos. Para ello hay que separar cada oración en palabras y obtener la longitud de cada oración, con la función `sapply()`, que puede verse en la Fig. 20.1.

```
palabras_oracion <- tokenize_words(oraciones[[1]])
longitud_o <- sapply(palabras_oracion, length)
head(longitud_o)
#> [1] 2 2 39 33 33 32
```

20.5. Ejemplo de aplicación

351

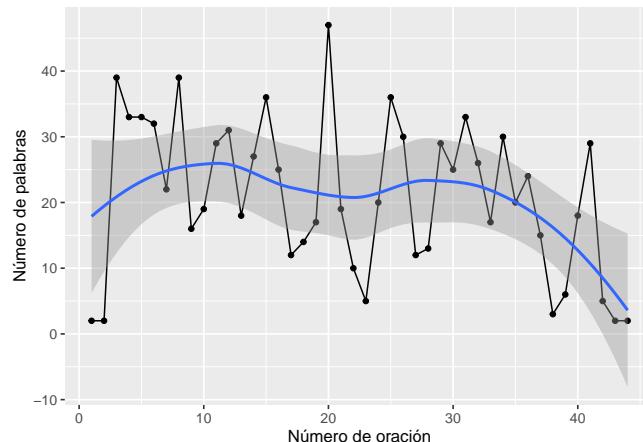


Figura 20.1: Número de palabras en cada oración de la Declaración

20.5.3. Análisis exploratorio

20.5.3.1. Eliminación de palabras vacías

Se llevará a cabo con el paquete `stopwords`, que contiene listas de *palabras vacías* en diferentes idiomas. Para el ejemplo, se define una tabla con la misma estructura que la tabla de la Declaración con las 308 palabras vacías españolas que tiene el paquete:

```
library("stopwords")
tabla_stopwords <- tibble(palabra = stopwords("es"))
```

La siguiente sentencia ‘limpia’ la tabla de la Declaración quitando las palabras vacías españolas. Además, se hace uso de la función `kable()` para una visualización más sofisticada de la tabla (con la longitud que se desee):

```
tabla <- tabla |> anti_join(tabla_stopwords)
knitr::kable(tabla[1:10],  
            caption = "Palabras más frecuentes (sin palabras vacías)")
```

El resultado, Tabla 20.1, se puede considerar el primer análisis léxico con valor informativo: la palabra más frecuente es *virus*, seguida de *recursos* y *social*. Se podría ver que en total hay 319 palabras no vacías distintas.

El método de eliminar palabras con el paquete `stopwords` no es perfecto. Por ejemplo, *va* y *cada* (posiciones 9 y 10 de la tabla) no son muy informativas. En estos casos, como se ha visto antes, se pueden utilizar listas de palabras vacías de otros paquetes como, por ejemplo `tidytext` o `tokenizers` o el listado en español propuesto por [Fradejas Rueda \(2022\)](#), o pueden confeccionarse listas *ad hoc*.

Tabla 20.1: Palabras más frecuentes (sin palabras vacías)

palabra	recuento
virus	9
recursos	7
social	5
alarma	4
conjunto	4
emergencia	4
españa	4
semanas	4
va	4
cada	3

20.5.3.2. Nubes de palabras

Una manera habitual de mostrar la información de forma visual es con las denominadas **nubes de palabras**, acudiendo a la función `wordcloud()` del paquete con el mismo nombre. Al tener dicha función un componente aleatorio, se fija con `set.seed()` (para la reproducibilidad del gráfico por parte del lector).

```
set.seed(12)
library("wordcloud")
wordcloud(tabla$palabra, tabla$recuento,
          max.words = 50, colors = rainbow(3))
```



Figura 20.2: Nube de palabras más frecuentes de la Declaración

El resultado se muestra en la Fig. 20.2. Como se puede observar, el tamaño de letra de la

palabra, y en este caso también el color, están relacionados con su frecuencia.

20.5.4. Análisis de sentimientos y detección de emociones

20.5.4.1. Lexicón bing

El diccionario `bing`, como se ha visto en la Sec. 20.3, es uno de los repertorios léxicos para el *análisis de sentimientos* que se pueden encontrar en R. Es un diccionario de polaridad (positiva/negativa) cuyo idioma original es el inglés. Se puede obtener con la función `get_sentiments()` del paquete `tidytext`. Contiene 2005 palabras positivas y 4781 palabras negativas, por lo que hay un marcado sesgo hacia la polaridad negativa.

Para ilustrar el uso de `bing`, se ha traducido al inglés (automáticamente) la Declaración. A continuación se carga el texto y se genera el objeto `tabla`, replicando el procedimiento descrito arriba de preparación, limpieza, segmentación en palabras, eliminación de palabras vacías (obviamente, en idioma inglés).

```
data("EN_declaracion")
tabla <- table(tokenize_words(EN_declaracion)[[1]])
tabla <- tibble(word = names(tabla),
                recuento = as.numeric(tabla))
tabla <- tabla |> anti_join(tibble(word=stopwords("en"))) |>
  arrange(desc(recuento))
```

Los sentimientos positivos de la Declaración se pueden obtener con:

```
library("tidytext")
pos <- get_sentiments("bing") |>
  dplyr::filter(sentiment=="positive")
pos_EN <- tabla |> semi_join(pos)
knitr::kable(pos_EN)
```

Análogamente, se pueden obtener los sentimientos negativos. Las siete palabras más frecuentes de cada tipo que aparecen en la Declaración se presentan conjuntamente en la Tabla 20.2.

20.5.4.2. Lexicón NRC

Para poder observar las similitudes y diferencias en el análisis según el diccionario elegido, se aplica también NRC a la Declaración (véase la Tabla 20.2).

Con el diccionario NRC pueden detectarse emociones. La misma palabra puede tener asociada distintas emociones/sentimientos. En la Fig. 20.3 se puede observar la dispar frecuencia de palabras de cada tipo:

Tabla 20.2: Palabras más frecuentes de la Declaración utilizando ‘bing’ y ‘NRC’

positivas bing	fr	negativas bing	fr	positivas NRC	fr	negativas NRC	fr
extraordinary	6	virus	9	resources	7	virus	9
protect	4	alarm	4	extraordinary	6	alarm	4
work	4	emergency	4	protect	4	emergency	4
like	3	vulnerable	3	maximum	3	government	3
decisive	2	difficult	2	public	3	discipline	2
good	2	hard	2	council	2	avoid	1
adequate	1	unfortunately	2	good	2	combat	1

```
emo <- get_sentiments("nrc")
emo |> ggplot(aes(sentiment)) +
  geom_bar(aes(fill=sentiment), show.legend = FALSE)
```

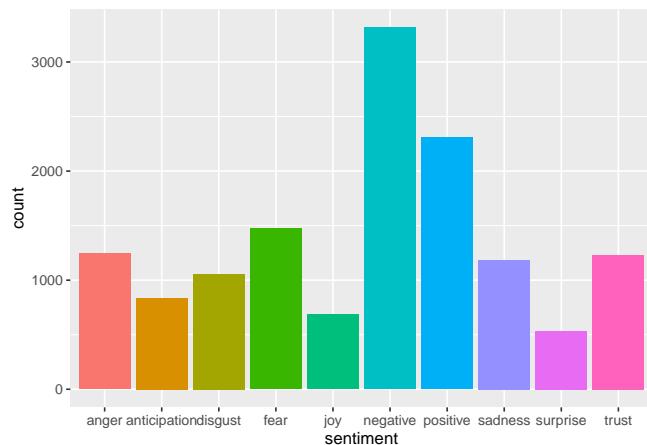


Figura 20.3: Gráfico de barras con la frecuencia de las emociones del lexicón NRC

El análisis de sentimientos y la detección de emociones de la Declaración mediante NRC se puede realizar con el siguiente código, mediante el cual se obtiene la tabla de frecuencias por emociones y sentimientos:

```
emo_tab <- tabla |> inner_join(emo)
head(emo_tab, n=7)
#> # A tibble: 7 x 3
#>   word          recuento sentiment
#>   <chr>        <dbl> <chr>
#> 1 virus         9 negative
#> 2 resources     7 joy
#> 3 resources     7 positive
```

20.5. Ejemplo de aplicación

355

```
#> 4 resources      7 trust
#> 5 extraordinary 6 positive
#> 6 alarm          4 fear
#> 7 alarm          4 negative
```

Como se ha mencionado antes, algunas palabras tienen asociados distintos sentimientos, por ejemplo, *resources*. La información de la tabla se puede visualizar bien con un gráfico de barras (Fig. 20.4) bien con nubes de palabras (Fig. 20.5).

```
emo_tab |>
  dplyr::count(sentiment) |>
  ggplot(aes(x=sentiment, y=n)) +
  geom_bar(stat = "identity", aes(fill=sentiment), show.legend = FALSE) +
  geom_text(aes(label = n), vjust=-0.25)
```

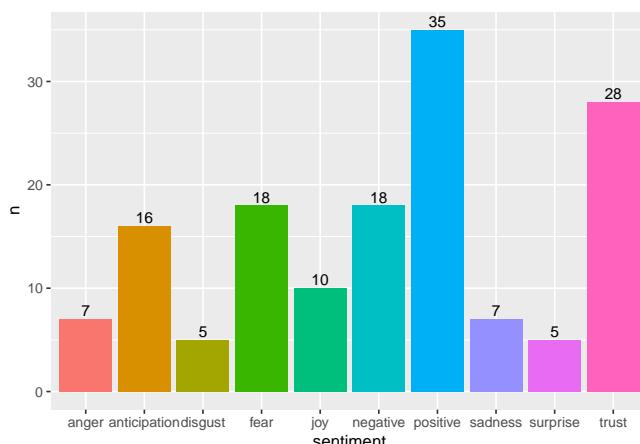


Figura 20.4: Frecuencia de emociones de la Declaración utilizando NRC

Entre las distintas opciones para dibujar nubes de palabras para el análisis de sentimientos es interesante la que se obtiene con el paquete *syuzhet* dado que permite visualizar las palabras agrupadas por emociones. Su obtención requiere distintos pasos en los que primero las palabras se agrupan por emoción y después se organizan en una **matriz de documentos** con la función *TermDocumentMatrix()* del paquete *tm*. Finalmente la función *comparison.cloud()* permite visualizar el gráfico (tiene distintos argumentos opcionales que admiten distintas posibilidades). En el ejemplo que figura a continuación solo se han escogido tres emociones³:

```
library("syuzhet")
palabras_EN2 <- get_tokens(EN_declaracion)
emo_tab2 <- get_nrc_sentiment(palabras_EN2, lang = "english" )
```

³Se deja al lector el análisis de la Declaración con más emociones, en castellano, etc.

```

emo_vec <- c(
  paste(palabras_EN2[emo_tab2$anger > 0], collapse = " "),
  paste(palabras_EN2[emo_tab2$anticipation > 0], collapse = " "),
  paste(palabras_EN2[emo_tab2$disgust > 0], collapse = " "))
library("tm")
corpus <- Corpus(VectorSource(emo_vec))
TDM <- as.matrix(TermDocumentMatrix(corpus))
colnames(TDM) <- c('anger', 'anticipation', 'disgust')
set.seed(1)
comparison.cloud(TDM, random.order = FALSE,
                 colors = c("firebrick", "forestgreen", "orange3"),
                 title.size = 1.5, scale = c(3.5, 1), rot.per = 0)

```



Figura 20.5: Nube de palabras de tres emociones NRC seleccionadas

20.5.5. *N-gramas*

El siguiente código muestra la obtención de *n-gramas* con `tokenizers`.

```

bigramas <- tokenize_ngrams(declaracion, n = 2,
                             stopwords = tabla_stopwords$palabra)
head(bigramas[[1]], n = 3)
#> [1] "buenas tardes"          "tardes estimados"      "estimados compatriotas"
trigramas <- tokenize_ngrams(declaracion, n = 3,

```

20.5. Ejemplo de aplicación

357

```
stopwords = tabla_stopwords$palabra)
head(trigramas[[1]], n = 3)
#> [1] "buenas tardes estimados"      "tardes estimados compatriotas"
#> [3] "estimados compatriotas día"
```

Se ha procedido a eliminar de los bigramas y trigramas aquellas combinaciones con al menos una palabra vacía (*stopword*).

Se procede ahora a obtener los bigramas con `tidytext`. Para el resto de *n-gramas* el procedimiento es análogo, haciendo las modificaciones oportunas. En el último paso se ordenan por frecuencia (de mayor a menor):

```
declara2 <- tibble(texto = declaracion)
bigramas <- declara2 |>
  unnest_tokens(bigram, texto, token = "ngrams", n = 2) |>
  dplyr::count(bigram, sort = TRUE)
bigramas[1:5, ]
#> # A tibble: 5 x 2
#>   bigram          n
#>   <chr>        <int>
#> 1 todos los      6
#> 2 de la         5
#> 3 de los         5
#> 4 del estado     5
#> 5 estado de       5
```

Una forma de eliminar las palabras vacías es:

```
bigramas_limpios <- bigramas |>
  tidyrr::separate(bigram, c("word1", "word2"), sep = " ") |>
  dplyr::filter(!word1 %in% tabla_stopwords$palabra) |>
  dplyr::filter(!word2 %in% tabla_stopwords$palabra) |>
  tidyrr::unite(bigram, word1, word2, sep = " ")
bigramas_limpios[1:5, ]
#> # A tibble: 5 x 2
#>   bigram          n
#>   <chr>        <int>
#> 1 autoridades sanitarias      2
#> 2 buenas tardes            2
#> 3 disciplina social        2
#> 4 haga falta              2
#> 5 ministros extraordinario 2
```

20.5.5.1. Significado y contexto

Como se ha visto en la Sec. 20.2.2, con los *n-gramas* se puede hacer un análisis de colocaciones para extraer los distintos significados y valores informativos a partir del contexto. En este caso,

se puede ver cómo la palabra *atender* cambia de sentido cuando va precedida de *no* o *sin*. A continuación, se filtran los bigramas cuya primera palabra es *no*:

```
bigramas_no <- bigramas |>
  tidyverse::separate(bigram, c("word1", "word2"), sep = " ") |>
  dplyr::filter(word1 == "no") |>
  dplyr::count(word1, word2, sort = TRUE)
bigramas_no
#> # A tibble: 3 x 3
#>   word1 word2     n
#>   <chr> <chr> <int>
#> 1 no    atiende     1
#> 2 no    cabe        1
#> 3 no    es          1
```

Estos resultados se pueden utilizar para el análisis de sentimientos y la detección de emociones.

20.5.6. Análisis de redes

En esta Sección se proporcionan las instrucciones para realizar un **análisis básico de redes** (ver Cap. 21), utilizando los paquetes *igraph* y *ggraph*. Dada la corta extensión de la Declaración no es posible obtener conclusiones. En la Fig. 20.6 se pueden ver los gráficos de redes de bigramas, tanto sin palabras vacías como con ellas.

```
library("igraph")
library("ggraph")
set.seed(1)
graf_bigramas_1 <- bigramas_limpios |>
  tidyverse::separate(bigram, c("first", "second"), sep = " ") |>
  dplyr::filter(n > 1) |>
  graph_from_data_frame()
g1 <- ggraph(graf_bigramas_1, layout = "fr") +
  geom_edge_link(arrows = arrow(length = unit(4, 'mm'))) +
  geom_node_point(size=0) +
  geom_node_text(aes(label = name))
graf_bigramas <- bigramas |>
  tidyverse::separate(bigram, c("first", "second"), sep = " ") |>
  dplyr::filter(n > 2) |>
  graph_from_data_frame()
g2 <- ggraph(graf_bigramas, layout = 'fr') +
  geom_edge_link0() +
  geom_node_point(size=0) +
  geom_node_label(aes(label = name))
library("patchwork")
g1+g2
```

20.5. Ejemplo de aplicación

359

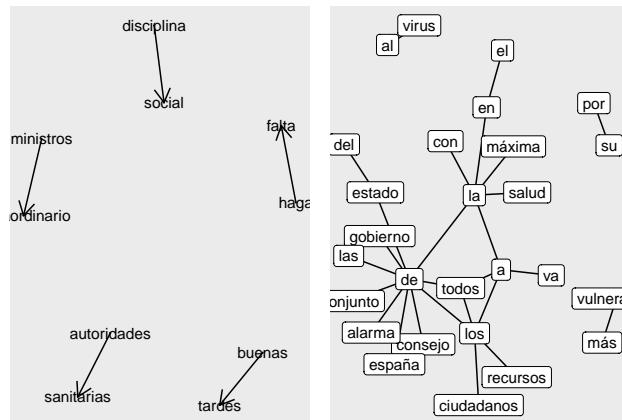


Figura 20.6: Redes de bigramas sin palabras vacías y con ellas

Resumen

En este capítulo se introduce al lector en la minería de textos, en particular:

- Se presentan los conceptos y tareas fundamentales de este ámbito, así como sus principales campos de aplicación. Se pone de relieve la importancia de la preparación de los datos y su segmentación (a distintos niveles) para obtener buenos resultados, acordes con el objetivo de la investigación.
- Se muestra el uso de **R** para el análisis de textos y de sentimientos.
- Se presenta un ejemplo de aplicación para ilustrar las técnicas de minería de textos.
- Se mencionan otros análisis plausibles de minería de textos, como la estilometría o el modelado de temas (véase el Cap. 41).

Capítulo 21

Análisis de grafos y redes sociales

José J. Galán

Universidad Complutense de Madrid

21.1. Introducción

El origen de la teoría de grafos se debe al problema de los siete puentes de Königsberg (Paul Euler, 1736), que es considerado el primer artículo sobre teoría de grafos. El problema se centra en la ciudad Königsberg en Prusia, ahora Kaliningrado (Rusia), donde existen varios puentes y el problema plantea trazar una ruta que cruce todos los puentes una única vez (ver Fig. 21.1). Euler mediante el uso de grafos demostró que no era posible.

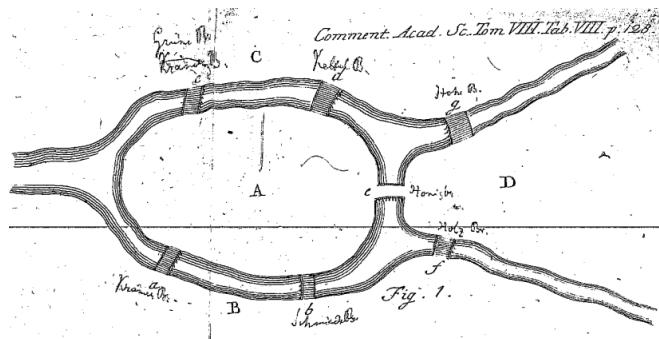


Figura 21.1: Siete puentes de Königsberg, Euler (1736).

Pero, ¿qué relación tiene un concepto creado en 1736, el de grafo, con algo tan reciente como las redes sociales?. Informalmente se puede hablar de las redes sociales (RRSS) como las relaciones existentes entre personas, un hilo invisible que une a las personas en relación con algo que tienen

en común. En algunos casos es muy evidente porque se crean grupos específicos de personas que comparten una afición y en otros casos es menos evidente porque, por ejemplo, pueden compartir un amigo en común sin saberlo. Estos hilos “invisibles” se unen y forman una red que se puede representar como un grafo, el mismo concepto de grafo que describió Euler, y que permite establecer diferentes caminos para unir a las personas que forman la red.

21.2. Teoría de grafos

Informalmente se puede decir que un **grafo** es un conjunto de **nodos** (vértices) que pueden estar unidos por **aristas** (enlaces).

Si se piensa en cada nodo como una persona y en cada arista como la relación que los une, entonces se podría representar mediante grafos una **red social** (ver Fig. ??).

Antes de ver el primer ejemplo se muestran las librerías necesarias en este capítulo.

```
library("igraph")
library("CDR")
```

En este primer ejemplo en **R** se puede observar cómo se obtienen los datos de una red social. El conjunto **datos_facebook** está incluido en el paquete **CDR**. A continuación, se representan las relaciones de los miembros que la componen mediante un grafo.

```
grafo_facebook <- graph.data.frame(datos_facebook, directed = F)
plot(grafo_facebook, vertex.label = NA, vertex.size = 8)
```

Más formalmente una **red social** puede modelizarse con una estructura de red invisible (relación familiar, amistad, trabajo ...) que une mediante relaciones a distintos actores a través de sus intereses o valores comunes, estableciendo una relación personal entre individuos o grupos de individuos conectados.

Existen distintos grafos dependiendo de las características de la red social representada, algunos ejemplos de estos grafos son:

- El grafo de amistad (ver Fig. 21.2), donde cada nodo representa una persona y la arista conecta dos personas si dentro de la red social son amigos.

```
amistades <- data.frame(
  persona = c("A", "B", "C", "D", "E"),
  amigo = c("B", "C", "A", "E", "A"))
grafo_amistades <- graph_from_data_frame(amistades)
```

- El grafo de comunidades (ver Fig. 21.2), donde también cada nodo representa una persona y la arista les conecta si dentro de la red social pertenecen a la misma comunidad, entendiendo por comunidad un grupo de individuos que comparten intereses o características en común.

```
comunidades <- data.frame(
  persona = c("A", "B", "C", "D", "E"), comunidad = c("1", "2", "1", "2", "1") )
grafo_comunidades <- graph_from_data_frame(comunidades)
```

```
par(mfrow=c(1,2))
plot(grafo_amistades, vertex.label = V(grafo_amistades)$name, main="Grafo amistades")
plot(grafo_comunidades, vertex.label = V(grafo_comunidades)$name, main="Grafo
↪ comunidades")
```

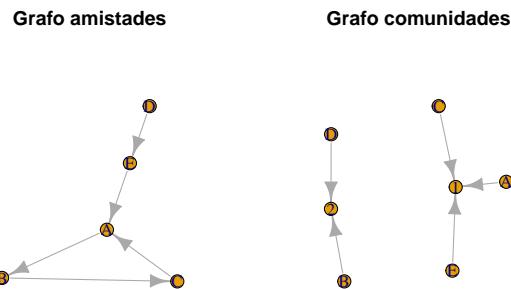


Figura 21.2: Grafo de amistades y comunidades.

21.3. Elementos de un grafo

El análisis de RRSS mediante la teoría de grafos requiere conocer previamente una serie de conceptos básicos ([Perez Sola, 2021](#)) que se enumeran a continuación.

- Los **vértices** representan nodos que se unen mediante aristas. En una red social cada vértice representa una de las personas de dicha red, unidas en ocasiones por intereses comunes a otras.
- Las **aristas** son las relaciones que unen los nodos. Son **dirigidas** (Fig.21.3) si tienen un sentido definido y **no dirigidas** (Fig. 21.3) en caso contrario.

El siguiente código computa un grafo dirigido:

```
grafodirigido <- graph.data.frame(datos_grafos, directed = T)
```

Para un grafo no dirigido, basta con especificar `directed = F` en la función `graph.data.frame()`:

```
grafonodirigido <- graph.data.frame(datos_grafos, directed = F)
```

En otras RRSS, como **LinkedIn**, las aristas podrían representar la relación que une las personas. Las personas forman parte de un grupo con intereses comunes, formando un grafo no dirigido. Pero también se pueden seguir a alguien sin necesariamente ser seguido; en ese caso las relaciones se pueden representar como un grafo dirigido.

- Un **grafo** es un conjunto de vértices y aristas que se puede representar mediante $G = (V, E)$, donde V es el conjunto de nodos o vértices del grafo y E es un conjunto de pares de vértices llamado arista, arco o edge.

Se presenta un grafo “sencillo” en la Fig. 21.3 (sólo se indican las aristas y **R** es capaz de interpretar los vértices), sobre el cual se explicará la matriz de adyacencia, grado y camino:

```
grafo <- graph(edges = c(1, 2, 1, 3, 1, 4, 2, 4, 3, 5, 4, 5))
```

```
par(mfrow=c(1,3))
plot(grafodirigido, vertex.label = V(grafodirigido)$name, main="Grafo dirigido")
plot(grafonodirigido, vertex.label = V(grafonodirigido)$name, main="Grafo no dirigido")
plot(grafo, main="Grafo \"sencillo\"")
```

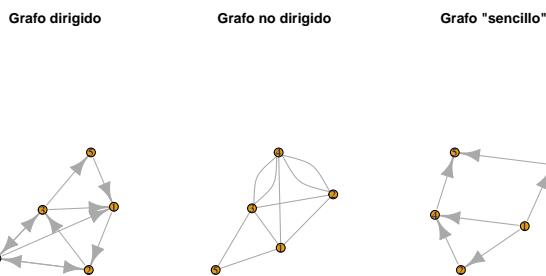


Figura 21.3: Grafo dirigido, no dirigido y sencillo.

21.3. Elementos de un grafo

365

- La información recogida en un grafo también se puede expresar mediante números, organizados en una matriz denominada **matriz de adyacencia**, $A_{n \times n}$, lo que facilita los cálculos computacionales en grandes redes. Cada entrada de la matriz, a_{ij} , indica el número de aristas que comparten los vértices (o nodos) i-ésimo y j-ésimo (si no existe relación entre ellos, entonces $a_{ij} = 0$); cada fila de la matriz indica el número de aristas que comparte el vértice i-ésimo con cada uno de los otros vértices. La suma de todas las entradas a_{ij} de una fila es el grado del vértice correspondiente. En los grafos no dirigidos $A_{n \times n}$ es simétrica, ya que si el vértice o nodo 1 conecta con el 2, entonces el 2 también conecta con el 1. En los grafos dirigidos, donde cada arista tiene una orientación, esto no tiene por qué ocurrir: el vértice o nodo 1 puede conectar con el 1 pero no al revés. En este tipo de grafos la matriz de adyacencia no es simétrica.

- El **grado** o valencia de un nodo x es el numero de aristas que concurren en dicho nodo, y se representa mediante $grado(x)$, $g(x)$ o $gr(x)$, lo cual en R se calcula con la función `degree`, siendo un vértice de grado 0 un vértice aislado. En un grafo G hay un grado máximo $\Delta(G)$ y un grado mínimo $\delta(G)$; el grado del grafo, $g(G)$, es la suma de los grados de todos sus vértices. En una red social representa el número de relaciones que existen; en una red social como Facebook podría significar conocer cuántos amigos tiene cada persona.

A continuación se muestra la matriz de adyacencia del grafo visto previamente:

```
matriz_adyacencia <- get.adjacency(grafo, sparse = FALSE)
matriz_adyacencia
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]     0    1    1    1    0
#> [2,]     0    0    0    1    0
#> [3,]     0    0    0    0    1
#> [4,]     0    0    0    0    1
#> [5,]     0    0    0    0    0
```

Ahora muestra el grado del mismo grafo:

```
degree(grafo)
#> [1] 3 2 2 3 2
```

- Un **camino** es un conjunto de aristas no recursivas. Entre dos vértices puede existir más de un camino, además puede haber varios y se puede incluir el mismo vértice en el camino más de una vez. Evidentemente, siempre habrá un **camino más corto**: que será aquel que menos aristas ha recorrido. Si entre todos los pares de vértices existe un camino, entonces el grafo se denomina *conexo*.

El siguiente código se utiliza para mostrar el camino más corto entre los vértices 2 y 5 de grafo utilizado de ejemplo.

```
# Camino más corto entre el nodo 2 y el 5
caminos <- get.shortest.paths(grafo, from = "2", to = "5")
```

```
V(grafo)[caminos$vpath[[1]]]
#> + 3/5 vertices, from 9547b44:
#> [1] 2 4 5
```

21.4. Procedimiento con R: el paquete igraph

Existen diversos paquetes en **R** para representar grafos, pero el más utilizado y popularizado, por sencillez y eficacia, es **igraph** ([Csardi and Nepusz, 2006](#)). Se trata de un paquete que permite crear y manipular grafos para analizar redes en **R** de forma muy sencilla (Fig. 21.4).

A continuación se muestra un sencillo ejemplo sobre cómo crear un grafo dirigido con la librería **igraph**

```
nodes <- data.frame("nodos" = c("A", "B", "C", "D", "E"))
edges <- data.frame(
  "from" = c("A", "C", "B", "A", "A", "A"),
  "to" = c("B", "D", "C", "D", "E"))
red <- graph_from_data_frame(edges, directed = TRUE, vertices = nodes)
plot(red, vertex.size = 50)
```

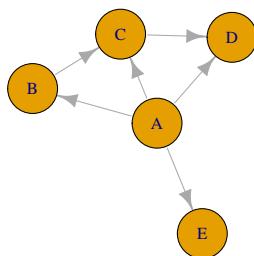


Figura 21.4: Ejemplo de grafo con ‘igraph’

Para crear un grafo, Fig. @ref{fig:grafobasico}, a partir de un **dataframe** se ha usado la función **graph_from_data_frame()** con los siguientes argumentos:

`graph_from_data_frame(edges, directed = TRUE, vertices = nodes)` donde:

- **edges** es un data frame donde las dos primeras columnas representan una lista de aristas.

21.4. Procedimiento con **R**: el paquete *igraph*

367

- **directed** es un valor lógico que indica si es un grafo dirigido o no dirigido.
- **vertices** es un data frame con los valores de los vértices o NULL.

El siguiente código muestra las relaciones entre los actores de dos películas. **nodes** contiene el nombre de cada actor y su descripción, es imprescindible que los nombres que más adelante se introducen en **edges** existan en **nodes**. Al mismo tiempo no es obligatorio declarar los nodos ya que pueden ser extraídos de las relaciones. **edges** contiene la relaciones, **from** y **to**, además de la película donde coinciden. Siendo este último dato descriptivo y no necesario.

```
nodes <- data.frame("actores" = c(
  "Jim Carrey", "Arnold Swarzenegger", "George Clooney",
  "Cameron Diaz"
), "descripcion" = c("actor", "actor", "actor", "actriz"))
edges <- data.frame(
  "from" = c("Jim Carrey", "Jim Carrey", "George Clooney", "Jim Carrey"),
  "to" = c(
    "Arnold Swarzenegger", "George Clooney", "Arnold Swarzenegger",
    "Cameron Diaz"
  ), "pelicula" = c(
    "Batman y Robin", "Batman y Robin",
    "Batman y Robin", "La mascara"
)
)
red <- graph_from_data_frame(edges, directed = F, vertices = nodes)
plot(red, vertex.size = 50)
```

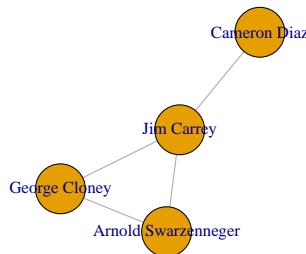


Figura 21.5: Grafo representativo de la relación de actores respecto a películas

En la Fig. 21.5 se puede observar como el actor Jim Carrey tuvo relación con todos los actores de la red propuesta, mientras que la actriz Cameron Diaz solo participó con uno de ellos (el propio Jim Carrey).

21.5. Análisis de influencia en un grafo aplicado a RRSS

Existen paquetes para obtener información de distintas RRSS; por ejemplo, en **R** se puede utilizar el paquete **Rfacebook** para conectarse a **Facebook** y obtener información de los contactos existentes. Para ello será necesario activar la API, Interfaz de Programación de Aplicaciones, desde <https://developers.facebook.com>. La información necesaria se puede encontrar en su página web <https://developers.facebook.com/docs>. Para ilustrar un ejemplo didáctico, sin que el lector necesite conocimientos de desarrollo para descargar los datos, se ha generado un fichero Excel que simula la relación entre amigos de una red social como podría ser Facebook generando un grafo dirigido, tipo de grafo habitual en este tipo de redes.

Se incorporan los datos y se muestra su cabecera.

En primer lugar, se utiliza el siguiente código para recoger los datos de un fichero CSV, con dos columnas, separadas por un espacio. Cada columna mediante un número identificador representa una persona, la unión de estas dos personas es el resultado de una relación, estas relaciones pueden visualizarse con el siguiente código.

```
datos_facebook <- graph.data.frame(datos_facebook, directed = F)
#datos_facebook # descomentar para ver las relaciones
```

En esta ocasión se utiliza la función `graph.data.frame()` del paquete **igraph** para crear un objeto de tipo grafo, dirigido en este caso, a partir de un data frame en **R**. Seguidamente, mediante `plot()` se muestra el grafo al mismo tiempo que se establecen sus propiedades. Ver Fig. 21.6. Nótese que la estructura de los datos de entrada para construir el grafo es diferente a la función `graph_from_data_frame()` vista en el apartado anterior, pero ambas cumple el objetivo de construir un grafo y por ello se presentan ambas opciones.

```
grafo_facebook <- graph.data.frame(datos_facebook, directed = T)
plot.igraph(grafo_facebook,
           layout = layout.fruchterman.reingold,
           vertex.label = NA, vertex.label.cex = 1, vertex.size = 3, edge.curved = TRUE
)
```

Siguiendo con el ejemplo, se ven las relaciones de la red social, se puede observar que el número de aristas que concurren en cada vértice indica el número de personas con las que se relaciona el individuo, representado por dicho vértice. Indica, por tanto, el número de relaciones que mantiene dicho individuo.

```
table(degree(grafo_facebook))
#>
#>   9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
#>   1  2  4  4  7  4  8 14 19 15 14 19 22 15 11 13  9  5  3  3  4  1  4
```

Ahora sobre el mismo ejemplo se personalizan los datos. Las RRSS son enormes y, por tanto, es útil centrarse en una subred para estudios concretos. A modo de ejemplo, para focalizar este caso

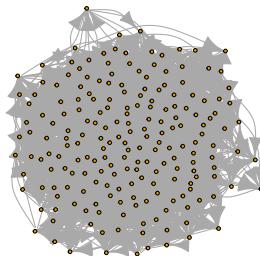


Figura 21.6: Aplicación de ‘igraph’ a RRSS.

de estudio el ejemplo se centrará en aquéllos cuyo grado sea igual o superior a 26, asignándoles su nombre.

```
bad_network <- V(grafo_facebook)[degree(grafo_facebook) <= 26]
grafo_facebook <- delete.vertices(grafo_facebook, bad_network)
V(grafo_facebook)$name <- c(
  "Gema", "Patricia", "Ramon", "José", "Maria", "Ángeles",
  "Gabriel", "Javier", "Victor", "Leonor", "Ana", "Isabel", "Cristóbal", "Rosa",
  ↪ "Aurora"
)
plot(grafo_facebook, vertex.size = 20)
```

En la Fig. 21.7 se pueden ver las relaciones entre las personas incluidas en el grafo (a quién siguen y por quiénes son seguidas). Por ejemplo, a Gema no la sigue nadie. Leonor, otro caso extremo igual que Gema, es seguida por Aurora, Ángeles y Gabriel, pero ella no sigue a nadie.

21.5.1. Centralidad

Un grafo no tiene una centralidad real porque no tiene coordenadas (Easley, 2010), pero existen distintas medidas de centralidad que, en una red/grafo social, permitirán identificar el networking social de cada individuo, es decir, su influencia.

En teoría de grafos y análisis de RRSS, el concepto de **centralidad** refiere a la importancia o prominencia de los vértices o nodos en un determinado grafo o red social. En el caso de una red social de amigos, como, por ejemplo, Facebook, la centralidad de un nodo (persona) representa el número de amigos que tiene.

Son innumerables las medidas de **centralidad** (generalmente normalizadas o estandarizadas) que pueden encontrarse en la literatura sobre la cuestión para determinar y comparar, de forma

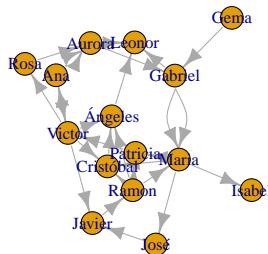


Figura 21.7: Aplicación de ‘Igraph’ a RRSS.

cuantitativa, la importancia relativa de un nodo en el conjunto de la red. La **centralidad** no es un atributo intrínseco de los nodos, sino un atributo estructural: un valor que depende de las relaciones de dicho nodo con los demás de la red. Generalmente, el nodo central suele tener la mayor centralidad, mientras que la mínima suele corresponder a los nodos periféricos.

En los grafos dirigidos, cuantas más aristas reciba un nodo (persona) más personas están intentando interactuar con ella y más prestigio tendrá en la red. Pero si la interacción hacia esta persona no es directa y se realiza a través de un camino más largo pasando por más nodos quiere decir que su influencia es elevada, ya que más personas han recibido esa influencia.

Existen diversas técnicas de obtener la centralidad (Wasserman, 1995), entre las que destacamos la centralidad por intermediación y la de vector propio por ser dos conceptos diferentes en el análisis de redes. Ahora se verá más en detalle como la primera mide la influencia de un nodo en la transmisión de información, mientras que la segunda es una medida más amplia de la centralidad global de un nodo en un grafo.

- La técnica de **centralidad de intermediación (betweenness)** se basa en el número de caminos mínimos (camino más corto entre dos vértices en un grafo ponderado) en los que un nodo está involucrado. Por lo tanto, en una red social una persona tendrá mayor influencia cuanto mayor betweenness tenga, porque comunicará mucha información a través de los nodos de la red. Si puede llegar a un grupo grande, aunque sea a través de un nodo a quien nadie sigue, como Gema, puede alcanzar un gran nivel de viralización porque su información llegara hasta muchas personas.

El siguiente código muestra la centralidad mediante intermediación correspondiente a los nodos del ejemplo actual

21.5. Análisis de influencia en un grafo aplicado a RRSS

371

```
betweenness_centrality <- betweenness(grafo_facebook)
betweenness_centrality
#>      Gema Patricia Ramon José María Ángeles Gabriel Javier
#> 0.000000 33.500000 30.200000 10.500000 53.300000 7.000000 37.800000 27.200000
#>   Victor Leonor Ana Isabel Cristóbal Rosa Aurora
#> 42.700000 0.000000 9.333333 0.000000 6.000000 4.666667 23.800000
```

Maria es el nodo con mayor porcentaje, ello quiere decir que es quien tiene un mayor número de enlaces o conexiones con otros nodos, siendo el nodo de mayor importancia en términos de conectividad.

- La **Centralidad de vector propio (eigenvector)**. Se basa en la centralidad de los nodos con los que se relaciona. En concreto la centralidad de valor propio es proporcional a la suma de las centralidades de sus nodos vecinos. Se representa mediante $c_i = \lambda \sum_j a_{ij} c_j$, donde λ es la constante de proporcionalidad y a_{ij} es el valor de la fila i y la columna j de la matriz de adyacencia \mathbf{A} de la red social.

El siguiente código muestra la centralidad mediante vector propio correspondiente a los nodos del ejemplo actual

```
eigencentrality <- eigen_centrality(grafo_facebook)$vector
eigencentrality
#>      Gema Patricia Ramon José María Ángeles Gabriel Javier
#> 0.1618269 0.6090993 0.7768737 0.3148866 0.9257523 0.6787138 0.7180398 0.4714266
#>   Victor Leonor Ana Isabel Cristóbal Rosa Aurora
#> 1.0000000 0.4725741 0.7663131 0.2086397 0.7620632 0.3831565 0.7000984
```

Observamos que según esta centralidad, basada en el número de enlaces y conexiones con otros nodos importantes, el mayor porcentaje es obtenido por el nodo Victor. Este nodo es por tanto central en términos de conexión con otros nodos importantes de la red siendo identificado como el nodo líder o de mayor influencia de la red.

21.5.2. Detección de comunidades

En el análisis de una red social es importante detectar las distintas comunidades que la componen, entendiendo por comunidad un grupo de personas afines, gracias a las comunidades podremos estudiar los distintos grupos que lo forman en función de sus relaciones y afinidad (Missaouri, 2015).

Existen diversos algoritmos en **R** para detectar comunidades (Borgatti, 2022), pero ninguno ha demostrado que pueda actuar a la perfección con todos los grafos debido a la gran tipología que existe. Así, según el ejemplo anterior podemos detectar las siguientes comunidades:

- La **detección de comunidades mediante betweenness** tiene implementado en el paquete **igraph** un algoritmo para detectar comunidades por lo que a continuación se expone.

```
communities <- cluster_edge_betweenness(grafo_facebook)
# head(communities, 10) # descomentar para ver las comunidades
```

- La **detección de comunidades mediante walktrap** es también muy utilizada en RRSS, se basa en el concepto de que las caminatas aleatorias cortas permanecen a la misma comunidad y no esta basado en una medida de centralidad.

A continuación, se presenta el código del algoritmo walktrap, mediante el cual se detectan dos comunidades:

```
communities <- walktrap.community(grafo_facebook)
```

21.5.3. Representación con grafos

Ahora que se ha observado la detección de comunidades se realiza su representación final mediante grafos.

- El siguiente código muestra la red mediante un **grafo de tipo Eigenvector** donde se han detectado mediante este algoritmo tres comunidades. Según este método, no es tan importante que tengas muchos amigos, lo importante es que tus amigos sean muy influyentes (ver Fig. 21.8)

```
cl_g_network <- leading.eigenvector.community(grafo_facebook)
plot(cl_g_network, grafo_facebook,
     edge.arrow.size = 0.25, main =
     "Leading Eigenvector Community", vertex.size = 50
)
```

- A continuación el código que muestra la red mediante el **grafo de tipo betweenness** donde en la Fig.21.9 se observan siete comunidades.

```
betweenness_grafo <- edge.betweenness.community(grafo_facebook)
```

- Ahora se puede observar el código para obtener el **grafo de tipo Walktrap**. Se pueden observar dos comunidades detectadas con **Walktrap**, ver Fig.21.9

```
Walktrap_grafo <- walktrap.community(grafo_facebook, steps = 5, modularity = TRUE)
#Debe indicarse el largo de la caminata aleatoria, se recomienda usar 5 caminatas.
```

```
par(mfrow=c(1,2))
plot(betweenness_grafo, grafo_facebook, edge.arrow.size = 0.25, vertex.size = 50,
     main="Grafo Betweenness")
plot(Walktrap_grafo, grafo_facebook, edge.arrow.size = 0.25,
     vertex.label = (grafo_facebook)$name, vertex.size = 50, main="Grafo Walktrap")
```

21.5. Análisis de influencia en un grafo aplicado a RRSS

373

Leading Eigenvector Community

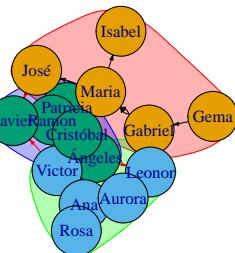
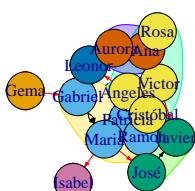


Figura 21.8: Aplicación de ‘Igraph’ a RRSS

Grafo Betweenness



Grafo Walktrap

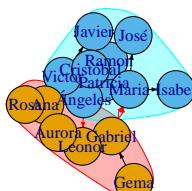


Figura 21.9: Grafo Betweenness y Walktrap.

21.6. Entorno social en el universo cinematográfico Marvel

Se va a analizar el Universo Cinematográfico de Marvel como una red social donde cada héroe tiene un grado de relación con otro. Se realiza este estudio utilizando los datos `marvel.edges` del paquete CDR que contiene dos columnas formateadas para representar la red social del Universo Marvel, donde la primera columna es el nombre de un personaje del Universo cinematográfico Marvel y la segunda el nombre de otro con el que coincide en alguna película, representado cada relación una arista entre dos nodos. Con estos datos se forma el grafo correspondiente a su red social, donde las coincidencias en la misma película entre héroes representan relaciones y cada héroe un nodo. En el siguiente código cargamos el fichero y formamos el `grafo_marvel`.

```
grafo_marvel <- graph.data.frame(marvel_edges, directed = F)
plot(grafo_marvel)
```

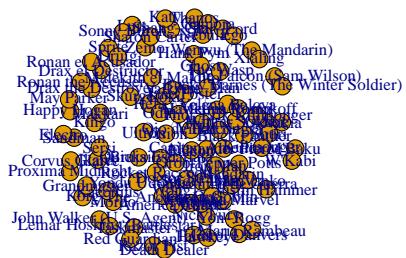


Figura 21.10: Grafo original sobre el Universo cinematografico Marvel

Son muchas las relaciones mostradas en la 21.10, por lo que a continuación se muestran aquellos héroes que tienen dos o más relaciones. Esto crea un grafo más visible, incluido un subgrafo de dos nodos (véase 21.11).

```
nodos_poca_realacion <- which(degree(grafo_marvel) < 2)
grafo <- delete.vertices(grafo_marvel, nodos_poca_realacion)
```

A continuación, se presenta el número de relaciones que tiene cada nodo, cada héroe, mediante la centralidad de grado en R usando la función `degree()` del paquete `igraph`. Se puede apreciar como Iron Man y Capitán América son quienes mayor número de relaciones tiene y por lo tanto quienes más gozan de popularidad e influencia.

21.6. Entorno social en el universo cinematográfico Marvel

375

```
grado_nodos <- degree(grafo)
#sort(grado_nodos)# descomentar para ver la centralidad de grado de los héroes.
```

Ahora, se analizan las comunidades que tiene, identificando cada una con un color distinto. En esta ocasión se utiliza el algoritmo Louvain, se considera el algoritmo más popular por su fácil interpretación, flexibilidad, alta calidad de las comunidades y eficiencia en tiempo de ejecución. El resultado podemos verlos en la 21.11.

```
comunidades <- cluster_louvain(grafo)
```

```
par(mfrow=c(1,2))
plot(grafo, vertex.label = V(grafo)$name, main="Relaciones de héroes")
plot(grafo, vertex.color = comunidades$membership, vertex.label = V(grafo)$name,
     main="Comunidades de héroes")
```

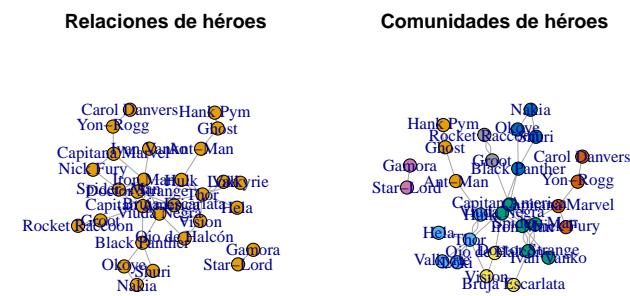


Figura 21.11: Grafo de relaciones y comunidades de héroes.

Por otra parte, se observan las comunidades, es decir, grupos de personas que tienen algo en común, en este caso héroes que comparten escenas en películas. De esta manera recorremos el grafo anterior para mostrar por separado cada una de las comunidades encontradas, rápidamente se observa que la Comunidad 3 es la comunidad con más miembros por lo tanto es la comunidad más popular y la que posee un mayor interés compartido.

```
comunidades <- cluster_louvain(grafo)
num_comunidades <- length(unique(comunidades$membership))

for (i in 1:num_comunidades) {
  nodos_comunidad <- which(comunidades$membership == i)
  subgrafo <- induced_subgraph(grafo, nodos_comunidad)
}
```

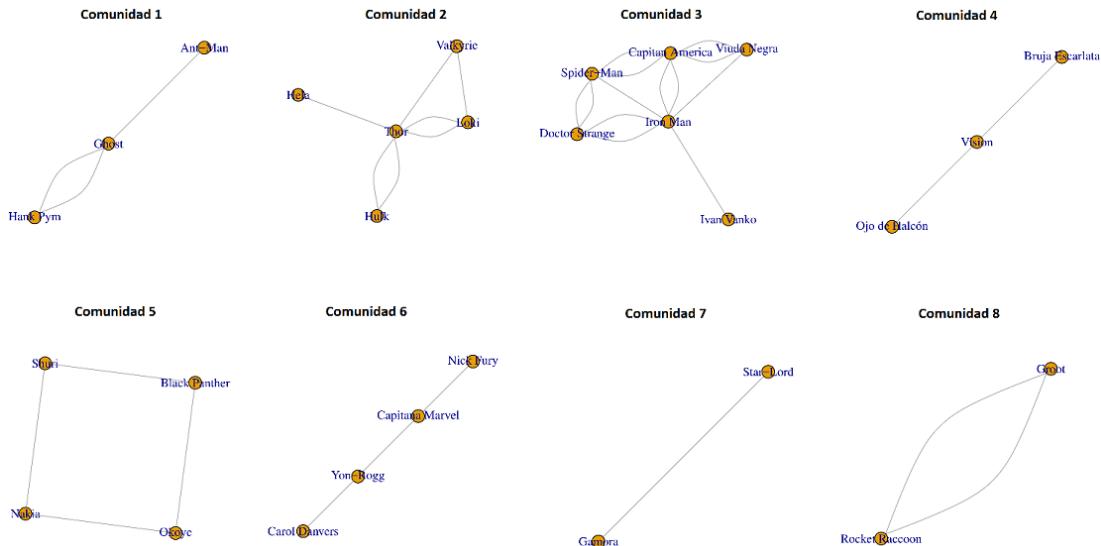


Figura 21.12: Distintas comunidades

A continuación, Fig. 21.13, el análisis se centra en la comunidad con más relaciones. Para ello, se identifica la comunidad más grande y se genera un subgrafo, mediante la función `induced_subgraph()` del paquete `igraph`, para su visualización.

```
comunidades <- cluster_louvain(grafo)
tamanos_comunidades <- table(comunidades$membership)
indice_comunidad_max <- which.max(tamanos_comunidades)
nodos_comunidad_max <- which(comunidades$membership == indice_comunidad_max)

subgrafo <- induced_subgraph(grafo, nodos_comunidad_max)
```

La Fig. 21.13 muestra el tamaño de los héroes según el número de relaciones y con un color distinto por cada héroe, lo que hace que el gráfico final sea más llamativo y fácil de interpretar.

```
# Calcular el grado de cada nodo
grados <- degree(subgrafo)

# Ajustar el tamaño de los nodos proporcionalmente a su grado
tamaños <- 80 * grados / max(grados)

# Generar un vector de colores aleatorios
colores <- sample(colors(), vcount(subgrafo), replace = TRUE)
```

21.6. Entorno social en el universo cinematográfico Marvel

377

```
par(mfrow=c(1,2))
plot(subgrafo, main="Grafo con más relaciones")
plot(subgrafo, vertex.color = colores, vertex.size = tamaños, main="Grafo adaptado por
  ↵ color y tamaño")
```

Grafo con más relaciones Grafo adaptado por color y tamaño

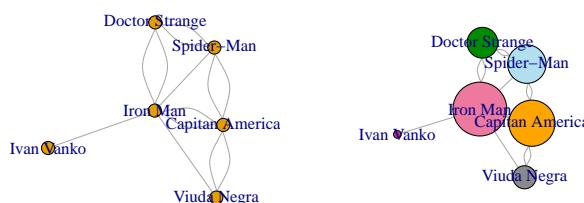


Figura 21.13: Grafo con más relaciones y grafo adaptado con color y tamaño

Las comunidades pueden ser representadas por otros algoritmos. A continuación se representan los algoritmos ya comentados, Betweenness, el cual es útil cuando los nodos que conectan distintas comunidades son los más importantes y se quiere asegurar que se incorporen en una comunidad y el algoritmo Walktrap, el cual detecta eficazmente comunidades de tamaños similares.

```
# edge_betweenness
comunidades <- cluster_edge_betweenness(grafo)
# walktrap
comunidades <- cluster_walktrap(grafo)
```

::: {.infobox_resume data-latex=““} ### Resumen {-} Este capítulo ha introducido la teoría de grafos y su relación con las RRSS, incluyendo los siguientes apartados:

- Teoría de grafos, se presenta como realizar un primer grafo mediante un data frame.
- Elementos de un grafo, se introducen los conceptos básicos: vértice, arista, grafico dirigido y no dirigido, grado y camino entre otros.
- Procedimiento con R: el paquete **igraph**, se centra en la estructura necesaria para componer un grafo.
- Análisis de influencia en un grafo aplicado a RRSS, se utiliza la teoría de grafos en el entorno de RRSS, introduciendo los conceptos de centralidad y comunidad.

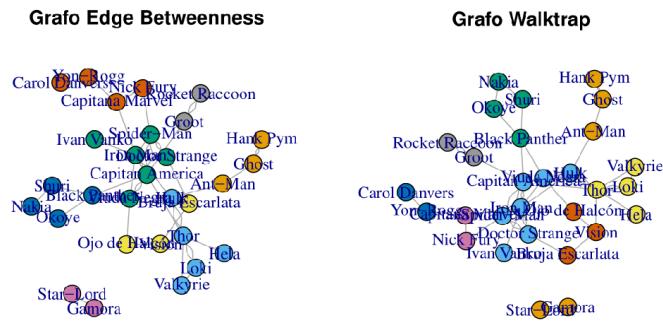


Figura 21.14: Comunidades según el algoritmo utilizado

- Entorno social en el Universo Cinematográfico Marvel, análisis de la red social que representan los héroes de la citada saga, aplicando los conocimientos de los apartados ante

Parte V

Ciencia de datos espaciales

Capítulo 22

Trabajando con datos espaciales

Gema Fernández-Avilés¹

Universidad de Castilla-La Mancha

22.1. Introducción

Los **datos espaciales**, también conocidos como **datos geográficos** o **datos georeferenciados**, son aquellos datos relacionados o que contienen información de una localización o área geográfica de la superficie de la Tierra. El primer análisis de datos geoespaciales fue hecho por el médico John Snow en 1854. Éste produjo un famoso mapa que muestra las muertes causadas por un brote de cólera (que mató a 127 personas en 3 días) en Soho, Londres así como la ubicación de las bombas de agua en el área (Fig. 22.1). Snow descubrió que había un agrupamiento significativo de muertes alrededor de una determinada bomba, y al quitar la manija de la bomba se detuvo el brote. Los datos con los que trabajó Snow y aquellos que contienen coordenadas son considerados datos espaciales.

El análisis espacial de Snow es considerado el antecedente más antiguo conocido de la ciencia de datos ([Baumer et al. \(2021\)](#)): (i) la información clave se obtuvo mediante la combinación de tres fuentes de datos (las muertes por cólera, las ubicaciones de las bombas de agua y el mapa de calles de Londres); (ii) se puede crear un modelo espacial directamente a partir de los datos y (iii) el problema solo se resolvió cuando la evidencia basada en datos se combinó con un modelo plausible que explicaba el fenómeno físico. Es decir, Snow era médico y su conocimiento sobre la transmisión de enfermedades fue suficiente para convencer a sus colegas de que el cólera no se transmitía por el aire.

¹Quisiera agradecer a Diego Hernández la ayuda prestada en la elaboración de este capítulo.

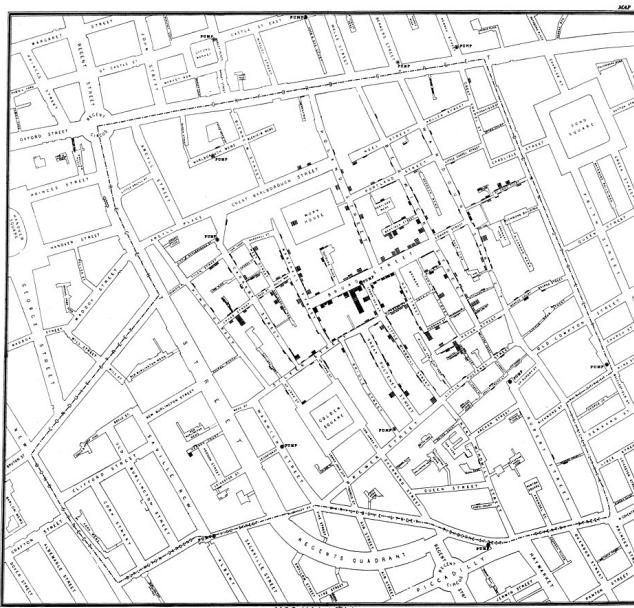


Figura 22.1: Mapa de cólera en Londres según Snow. Fuente: Wikipedia

22.1.1. Estadística para datos espaciales

El área que se encarga de estudiar y analizar los datos espaciales es la **estadística espacial** o la estadística para datos espaciales ([Cressie \(1993\)](#), ?).

Debido a que los datos espaciales surgen en una gran variedad de campos y aplicaciones, también hay una gran variedad de tipos de datos espaciales, estructuras y escenarios ([Schabenberger and Gotway, 2005](#), p. 6). La Fig. 22.2 representa la clasificación de datos espaciales proporcionada por [Cressie \(1993\)](#) basada en la naturaleza del dominio espacial en estudio. Cressie distingue tres tipos de datos espaciales:

- (I) datos geoestadísticos,
- (II) datos de patrones de puntos y
- (III) datos lattice o reticulares.

El estudio de los datos geoestadísticos se aborda en el Cap.[23](#), el análisis de los datos *lattice* se lleva a cabo en el Cap. [24](#) dedicado a la Econometría espacial y los datos de patrones de puntos se analizan en el Cap. [25](#).

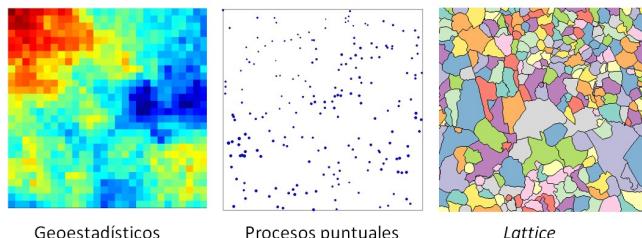


Figura 22.2: Clasificación de datos espaciales propuesta por Cressie (1993)

22.2. Conceptos clave

Visto el contexto original de los datos espaciales y antes de entrar en detalle en su análisis, se debe tener en cuenta una serie de conceptos clave. La Fig. 22.3, representa la localización de los accidentes de tráfico registrados en la ciudad de Madrid durante el año 2020. Sin embargo, tal representación no aporta información útil para su análisis. Por ejemplo, sería interesante añadir un mapa de carreteras junto con la localización de los accidentes.

```
library("CDR")
library("tidyverse")
ggplot(data = accidentes2020_data,
       aes(x = coordenada_x_utm, y = coordenada_y_utm)) +
  geom_point(col="blue", size = 0.1, alpha = 0.3) +
  coord_fixed()
```

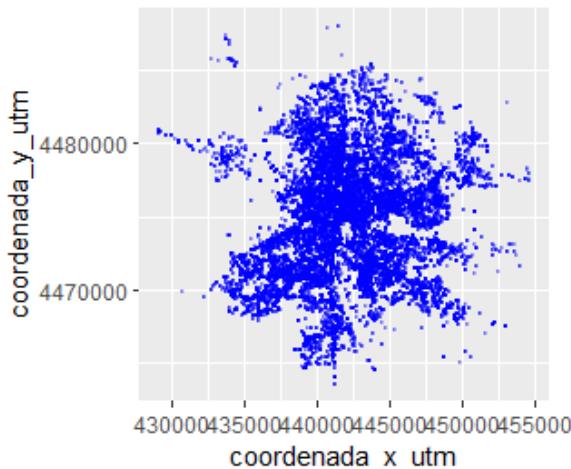


Figura 22.3: Accidentes de Tráfico en Madrid (2020)

Ademas de las coordenadas, en la representación de geodatos es importante el marco o contexto

espacial, así como el conocimiento del (i) **Sistema de referencia de coordenadas** o Coordinate reference system (**CRS**) en el que están georeferenciadas o proyectadas las coordenadas y (ii) el tipo de datos con el que se está trabajando: vectores o ráster.

```
library("sf")
accidentes2020_sf <- st_as_sf(accidentes2020_data,
  coords = c("coordenada_x_utm", "coordenada_y_utm"),
  crs = 25830 # proyección ETRS89/ UTM zone 30N. Área de uso: Europa
)

library("mapSpain")
madrid <- esp_get_munic(munic = "^Madrid$") |>
  st_transform(25830)

# descara imagen de un de mapa estático de las carreteras de Madrid
tile <- esp_getTiles(madrid, "IDErioja", zoommin = 2)

ggplot() +
  tidyterra::geom_spatraster_rgb(data = tile) +
  geom_sf(data = accidentes2020_sf,
    col = "blue", size = 0.1, alpha = 0.3) +
  coord_sf(expand = FALSE)
```

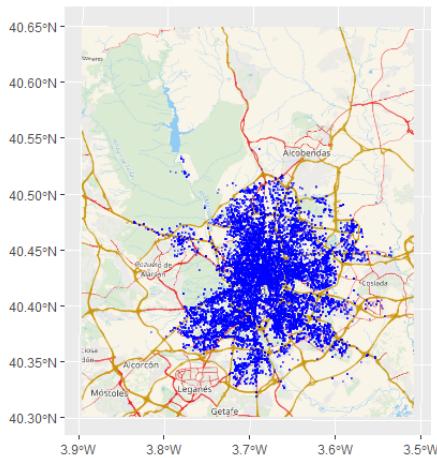


Figura 22.4: Accidentes de tráfico en Madrid proyectados y con mapa de carreteras (2020)

La Fig. 22.4 permite observar ciertos patrones en la ocurrencia de accidentes. Por ejemplo, apenas se producen accidentes en la Casa de Campo o en el Monte del Pardo, y parece observarse cierta concentración en la ciudad y en las autopistas de salida de la ciudad.

22.2.1. Sistema de referencia de coordenadas

Los CRS permiten identificar con exactitud la posición de los datos sobre el globo terráqueo. Cuando se trabaja con datos espaciales provenientes de distintas fuentes de información es necesario comprobar que dichos datos se encuentran definidos en el mismo CRS. Ésto se consigue transformando (o proyectando) los datos a un CRS común. Una buena referencia para profundizar este tema es el Cap. 2 de [Pebesma and Bivand \(2022\)](#).

En la Fig. 22.5 se muestran los puertos en un mapa mundial. Todos los vienen representados por el punto rojo. ¿A qué se debe? A que los datos están en distintos CRS.

```
library("giscoR")

paises <- gisco_get_countries()
puertos <- gisco_get_ports()
paises_robin <- st_transform(paises, st_crs("ESRI:54030")) #Proyección Robinson

plot(st_geometry(paises_robin), main = " ")
plot(st_geometry(puertos), add = TRUE, col="2", pch=20, lwd=2.5)
```

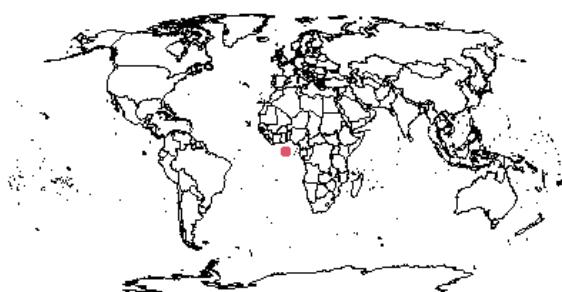


Figura 22.5: Localización de los puertos en el mapamundi (distinto CRS en los puertos y el mapa)

Los dos tipos de CRS que existen se describen a continuación:

- (i) **Geográficos**: aquellos en los que los parámetros empleados para localizar una posición

espacial son la latitud (Norte-Sur [-90°,90°]) y la longitud (Este-Oeste [-180°,180°]). Están basados en la geometría esférica. En este caso las distancias entre dos puntos son **distancias angulares**.

- (ii) **Proyectados:** permiten reducir la superficie de la esfera terrestre (3D) a un sistema cartesiano (2D). Para ello, es necesario transformar las coordenadas longitud y latitud en coordenadas cartesianas X e Y . La unidad de distancia, habitualmente, es el **metro**.

Tras proyectar los puertos al mismo CRS que el mapamundi utilizando la proyección de Robinson (la proyección cartográfica más convencional para mapamundis), la Fig. 22.6 muestra adecuadamente el mapa de la Fig. 22.5.

```
st_crs(puertos) == st_crs(paises_robin) # Comprueba CRS
#> [1] FALSE
puertos_robin <- st_transform(puertos, st_crs(paises_robin))
plot(st_geometry(paises_robin), main = " ")
plot(st_geometry(puertos_robin), add = TRUE, col=4, pch=20)
```

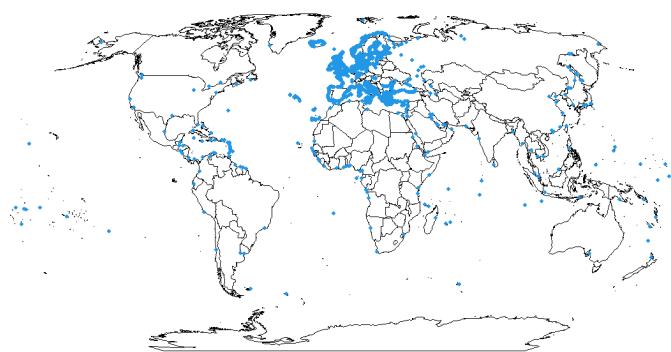


Figura 22.6: Localización de los puertos en el mapa mundi (mismo CRS puertos y mapa)

¿Qué proyección uso? El CRS adecuado para cada análisis depende de la localización y el rango espacial de los datos. El paquete `crssuggest` (Walker, 2022) facilita la labor, sugiriendo la escala de estudio o el tipo de análisis más adecuado para cada zona.

22.2.2. Formatos de datos espaciales

En el ámbito del análisis espacial, el formato de datos espaciales se puede clasificar en función del modelo de datos. Se pueden distinguir dos tipos de modelos de datos (Lovelace et al., 2019): vectoriales y ráster².

22.2.2.1. Datos de vectores

Este modelo está basado en puntos georeferenciados. Los **puntos** pueden representar localizaciones específicas, como la localización de los Hospitales y Centros de Salud de la ciudad de Toledo (Fig. 22.7).

```
ggplot() +
  geom_sf(data = hosp_toledo,
          aes(fill = "Hospitales y Centros Sanitarios",
              color = "blue") +
  labs(title = NULL, fill = NULL) +
  theme_minimal() +
  theme(legend.position = "right")
```

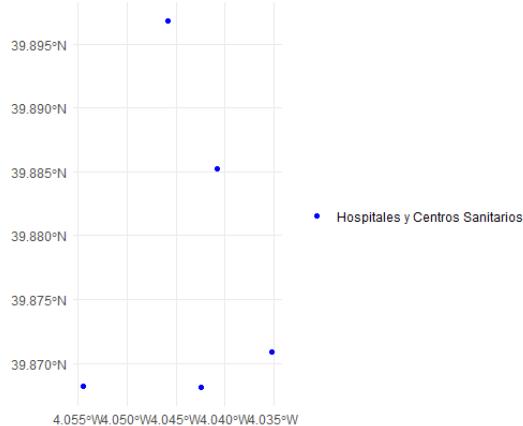


Figura 22.7: Hospitales y Centros de Salud en Toledo

Los puntos también pueden estar conectados entre sí, de manera que formen geometrías más complejas, como **líneas** y **polígonos**.

En la Fig. 22.8, el río Tajo está representado como una línea (`tajo`, sucesión de puntos unidos entre sí) y la ciudad de Toledo como un polígono (`toledo`, línea de puntos cerrada formando un continuo).

²Un análisis detallado puede verse en Hernangómez and Fernández-Avilés (2022)

```
ggplot(toledo) +
  geom_sf(fill = "cornsilk2") +
  geom_sf(data = tajo, col = "lightblue2", lwd = 2, alpha = 0.7) +
  geom_sf(data = hosp_toledo, col = "blue") +
  coord_sf(xlim = c(-4.2, -3.8), ylim = c(39.8, 39.95)) +
  theme_minimal()
```

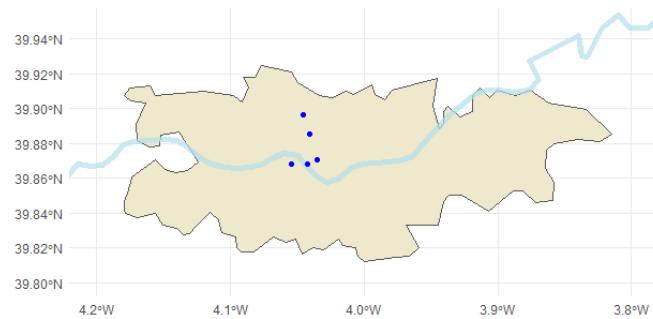


Figura 22.8: Datos vector: Puntos, líneas y polígonos

Las extensiones más habituales de los archivos que contienen datos de vectores se muestran a continuación:

Tabla 22.1: Ficheros con datos vector

Tipo	Extensión
Shapefile	.shp, .shx, .dbf
GeoPackage vector	.gPKG
GeoJson	.geojson
GPX	.gpx
Geography Markup Language	.gml
Keyhole Markup Language	.kml
Otros	.csv, .txt, xls

ESRI Shapefile surgió como uno de los primeros formatos de intercambio de datos geográficos y en la actualidad es, quizás, el formato más empleado. Sin embargo, tiene una serie de limitaciones: es un formato multiarchivo y el CRS es opcional.

22.2.2.2. Datos ráster

Los datos raster son datos proporcionados en una rejilla de píxeles (regulares o no) denominada **matriz**. El caso más popular de un ráster es una fotografía, donde la imagen se representa como una serie de celdas, determinadas por la resolución de la imagen, es decir, el tamaño del píxel (por ejemplo, 5 x 5 unidades, si es regular, 5 x 10 unidades, si es irregular) y el valor del pixel (RGB, por ejemplo) que determina el color que presenta cada uno de estos píxeles. En el ámbito de los datos espaciales, un archivo ráster está formado por una malla de píxeles georreferenciada, tal y como muestra la Fig. 22.9. Aquí se visualiza el conjunto de datos `elev` dentro del paquete CDR que representan los datos de la altitud de la provincia de Toledo en metros.

```
library("terra")
elev <- rast(system.file("external/Toledo_DEM.asc", package = "CDR"))
plot(elev, main = " ")
polys <- as.polygons(elev, dissolve=FALSE)
plot(polys, add = TRUE, border = "grey90")
plot(st_geometry(Tol_prov), add = TRUE)
```

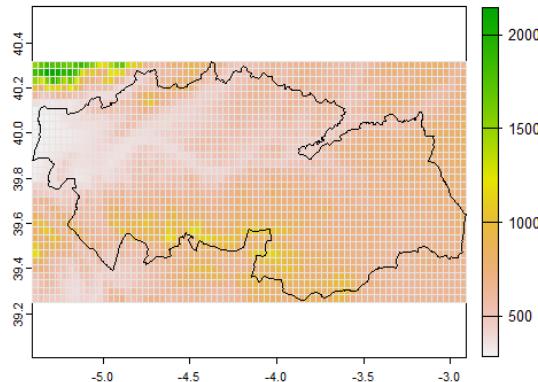


Figura 22.9: Datos ráster. Altitud de la provincia de Toledo

En la Fig. 22.9, el objeto ráster `elev` tiene únicamente una capa. Eso implica que cada píxel tiene asociado un único valor, en este caso, la altitud media del terreno observado. Las extensiones más habituales de los archivos que contienen datos ráster se muestran a continuación:

Tabla 22.2: Ficheros con datos raster

Tipo	Extensión
ASCII Grid	.asc

Tipo	Extensión
GeoTIFF	.tif, .tiff
Enhanced Compression Wavelet	.ecw

22.3. Mi primer mapa

Definidos los elementos clave de los datos espaciales se llevará a cabo la representación en un mapa de la distribución municipal de la renta neta per cápita (`renta_municipio_data`) por municipio (`municipios`) en el periodo 2019 en España³. Los datos están incluidos en el paquete CDR.

Ambos conjuntos deben tener, al menos, un campo en común, `codigo_ine` en este caso, para su unión.

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

ggplot(munis_renta) +
  geom_sf(aes(fill = `2019`), color = NA) +
  scale_fill_continuous(
    labels = scales::label_number(
      big.mark = ".", decimal.mark = ",", suffix = " €" )) +
  theme_minimal()
```

La Fig. 22.10 presenta un mapa temático o de coropletas, es decir, una visualización sencilla de cómo varía la distribución de una variable (en este caso la renta neta media por persona) en un área geográfica (España). Adicionalmente, una serie de elementos gráficos característicos de los objetos espaciales puede verse en la información contenida en el objeto `munis_renta`: los datos son de tipo vector, el tipo de geometría es MULTIPOLYGON, el CRS es ETRS89 y una leyenda explica el significado de la variable.

```
head(munis_renta)[1:3, ]
#> Simple feature collection with 3 features and 4 fields
#> Geometry type: MULTIPOLYGON
#> Dimension: XY
#> Bounding box: xmin: -3.140179 ymin: 36.73817 xmax: -2.741701 ymax: 37.24562
#> Geodetic CRS: ETRS89
#> name cpro cmun 2019 geom
```

³Un análisis detallado puede verse en ([Hernangómez and Fernández-Avilés, 2022](#)).

22.4. ¿Cómo (no) mentir con la visualización?

391

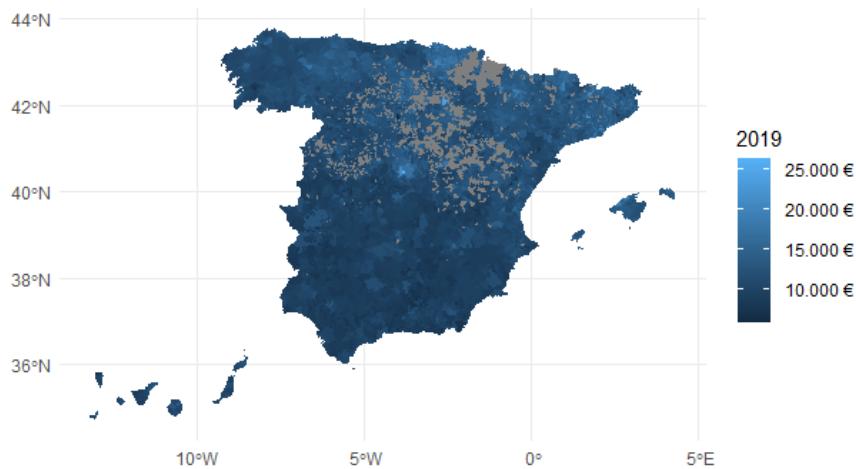


Figura 22.10: Distribución de la renta neta media por persona (€) en 2019

```
#> 1 Abla 04 001 10192 MULTIPOLYGON (((-2.775594 3...
```

```
#> Abrucena 04 002 10021 MULTIPOLYGON (((-2.787566 3...
```

```
#> Adra 04 003 8192 MULTIPOLYGON (((-3.051988 3...
```

22.4. ¿Cómo (no) mentir con la visualización?

Si se realiza un mapa de coropletas como el de la Fig. 22.10, puede que la información aparezca distorsionada. Algunas consideraciones básicas en visualización son:

- La escala de color.
- La distribución de los datos.
- La definición de intervalos.

¿Cómo es la distribución de la variable renta? La variable no sigue una distribución Normal (la renta sigue una distribución Gamma⁴), y si el objetivo es mostrar patrones espaciales de la variable, para una mejor representación será necesario dividir los datos en clases con el paquete `classInt` (Bivand, 2020). De entre las distintas posibilidades que ofrece la función

⁴Las características de la distribución Gamma pueden verse en el Cap. ??.

`classIntervals()`, se utiliza el método de Fisher-Jenks, que consiste en un mapa de cortes naturales que utiliza un algoritmo no lineal para agrupar observaciones de modo que se maximice la homogeneidad dentro del grupo. Este algoritmo está desarrollado específicamente para la clasificación de datos espaciales y su visualización en mapas. Además, se eliminan los municipios sin datos (sombreados en color gris) y se elige una escala de color adecuada. El mapa de la Fig. 22.11 proporciona ahora una visualización adecuada de la variable renta.

```
munis_renta_clean <- munis_renta |>
  filter(!is.na(`2019`))

# crea Fisher-Jenks clases
library(classInt)
fisher <- classIntervals(munis_renta_clean$`2019`,
  style = "fisher", n = 10
)

ggplot(munis_renta_clean) +
  geom_sf(aes(fill = cut(`2019`, fisher$brks)), color = NA) +
  scale_fill_viridis_d(option= "A",
    labels= scales::label_number(suffix= "€")) +
  guides(fill = guide_colorsteps()) +
  labs(fill= "Fisher-Jenks") +
  theme_minimal()
```

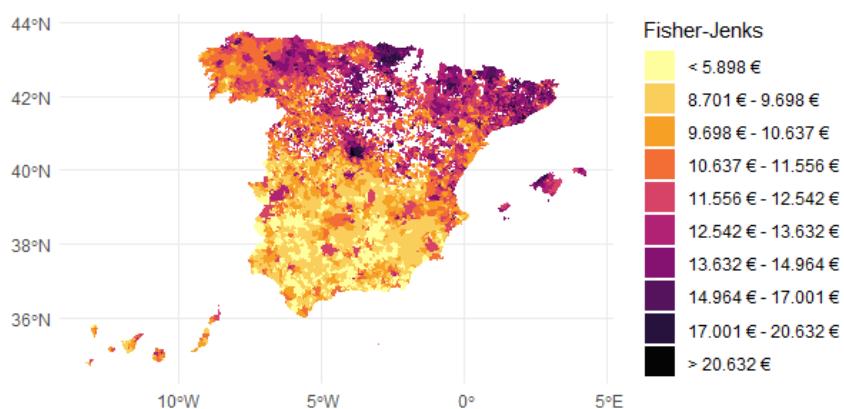


Figura 22.11: Renta neta per cápita (€) por tramos según Fisher-Jenks

22.5. Mapas espacio-temporales

La dimensión temporal es cada vez más importante en el ámbito espacial, por ello, es importante representar en el tiempo los procesos espaciales. La Fig. 22.13 representa la temperatura mínima registrada en España del 6 al 10 de Enero de 2021, CDR::tempmin_data, durante la [Borrasca Filomena](#).

```
tmin_sf <- st_as_sf(tempmin_data,
  coords = c("longitud", "latitud"),
  crs = 4326 # coordenadas geográficas longitud/latitud WGS84
)

esp <- esp_get_ccaa() |> # sf objeto, contorno de España
  filter(ine.ccaa.name != "Canarias") # excluye Canarias
```

La primera pregunta se debe formular es: ¿tengo el CRS de las estaciones de monitoreo en la misma proyección que el contorno de España?

```
st_crs(tmin_sf) == st_crs(esp)
#> [1] FALSE
esp2 <- st_transform(esp, st_crs(tmin_sf))
st_crs(tmin_sf) == st_crs(esp2)
#> [1] TRUE
```

Comprobado el CRS, es habitual representar las coordenadas con las que se trabaja. La Fig. 22.12 muestra la localización de las estaciones de monitoreo en España que registran la temperatura.

```
ggplot(esp2) +
  geom_sf() +
  geom_sf(data = tmin_sf) +
  theme_light()
```

Por último, se representa el mapa espacio-temporal con la función `ggplot()` indicando en el argumento `facet_wrap()` la dimensión temporal.

Nota

Los paquetes `tmap` ([Tennekes, 2018](#)) y `mapsdf` ([Giraud, 2022](#)) son referentes para mapas temáticos y pueden utilizarse como alternativa.

```
# definición de intervalos
cortes <- c(-Inf, seq(-20, 20, 2.5), Inf)
colores <- hcl.colors(15, "PuOr", rev = TRUE)
```

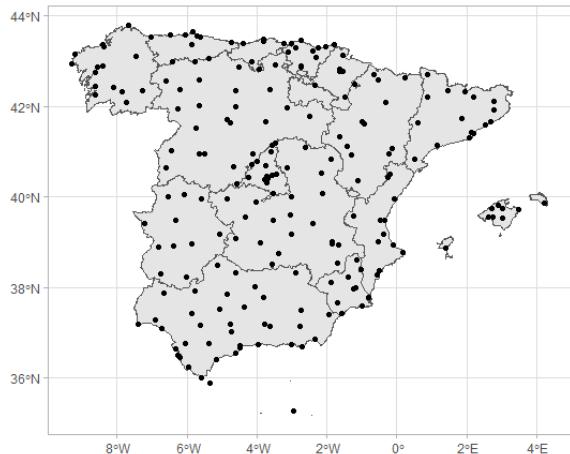


Figura 22.12: Estaciones de AEMET en la Península Ibérica

```
tmin_sf_sptem <- tmin_sf |>
  arrange(fecha, desc(tmin))

ggplot() +
  geom_sf(data = esp2, fill = "grey95") +
  geom_sf(data = tmin_sf, aes(color = tmin), size=3, alpha= .7) +
  facet_wrap(vars(fecha), ncol = 3) +
  labs(color = "Temp. mín") +
  scale_color_gradientn(
    colours = colores,
    breaks = cortes,
    labels = ~str_c(. , " °"),
    guide = "legend")
```

22.6. Mapas interactivos

El desarrollo de la informática ha propiciado también el desarrollo de la geocomputación, que está relacionada con los desarrollos webs, y permite, entre otras cosas, la representación de mapas interactivos.

A modo de ejemplo, el mapa de la Fig. 22.14 representa el mapa Fig. 22.1 de forma interactiva con la librería `leaflet`. Estos mapas dinámicos, ampliables y desplazables, son más informativos que los mapas estáticos y, además, son una alternativa que pueden proporcionar una experiencia diferente y una mayor interacción al usuario.

22.6. Mapas interactivos

395

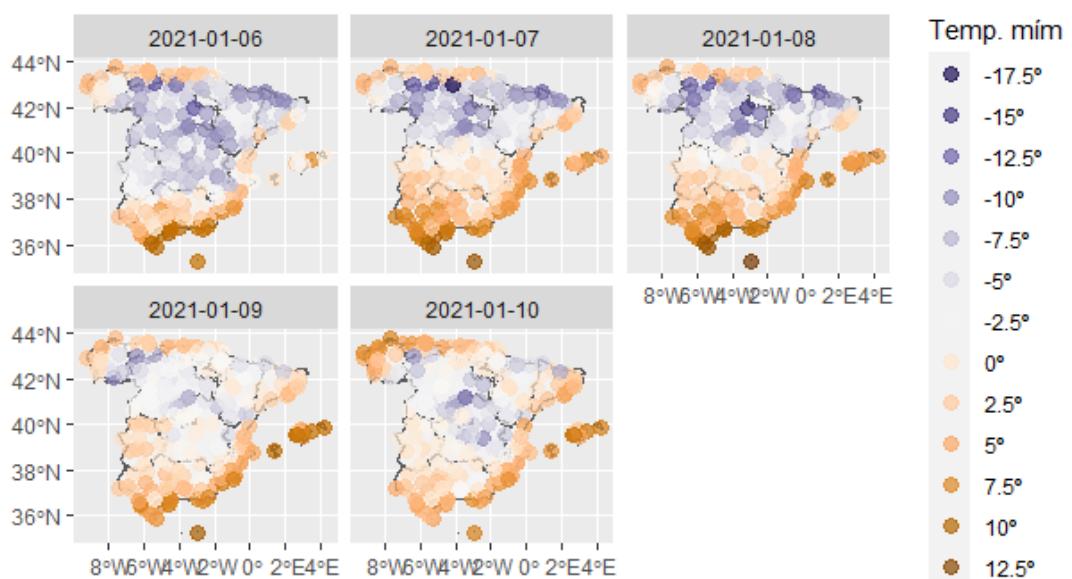


Figura 22.13: Temperatura mínima en España (6-10 enero 2021)

```
library("leaflet")
library("isdas")
data("snow_deaths")
data("snow_pumps")

## crea mapa interactivo
snow_map <- leaflet() |>
  setView(lng = -0.136, lat = 51.513, zoom = 16) |>
  addTiles() |>
  addMarkers( data = snow_deaths, ~long, ~lat,
    clusterOptions = markerClusterOptions(),
    group = "Deaths" ) |>
  addMarkers(data = snow_pumps, ~long, ~lat,
    group = "Pumps" )
snow_map
```

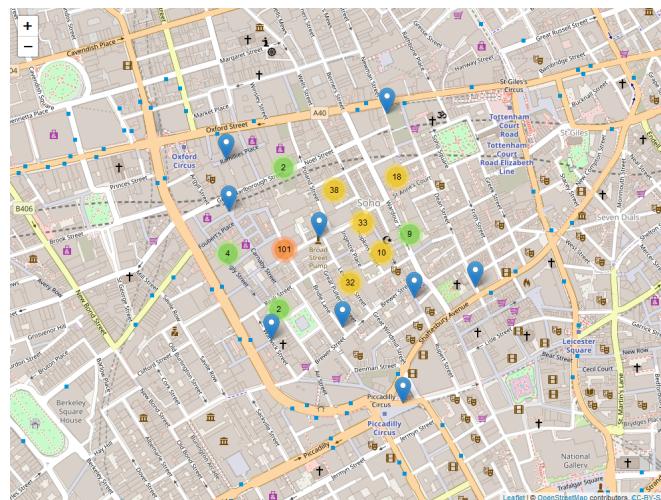


Figura 22.14: Mapa interactivo de las muertes por cólera en Londres según Snow en 1854

Reumen

Los datos espaciales son aquellos que contienen información de una zona geográfica de la tierra. Vienen definidos por coordenadas y por un sistema de referencia de coordenadas que debe tenerse en cuenta para su representación.

Existen dos tipos de formatos de datos: vector y ráster.

Los datos espaciales pueden clasificarse en: geoestadísticos, reticulares y puntuales.

Capítulo 23

Geoestadística

Gema Fernández-Avilés^a y José-María Montero^a

^aUniversidad de Castilla-La Mancha

23.1. Introducción

El término “geoestadística” apareció por primera vez en Matheron (1962), y en él “geo” enfatiza la referencia a las Ciencias de la Tierra, extendiendo el ámbito de la estadística tradicional, cuyo objetivo es el uso de métodos probabilísticos-inferenciales¹, con la incorporación del componente geográfico.

La geoestadística estudia los fenómenos regionalizados, que son aquellos que:

- Se extienden en el espacio, siendo el dominio espacial, D , continuo (se puede observar en cualquiera de sus puntos) y fijo (las ubicaciones observadas no son estocásticas; se seleccionan, por el procedimiento que sea, a juicio del investigador)².
- Presentan una organización o estructura debida a la dependencia espacial existente.

El objetivo fundamental de la geoestadística es sacar provecho de la dependencia espacial existente para llevar a cabo predicciones (interpolaciones) óptimas en ubicaciones o áreas de interés (en este sentido se habla de predicciones puntuales o por bloques, respectivamente), o la realización de mappings sobre todo el dominio o parte de él. Al ser D continuo, no se puede

¹Véanse Caps. ?? y ??.

²Los datos geoestadísticos son tan solo una parte de los datos espaciales: otra parte de ellos, son los datos “laticce”, poligonales o regionales, donde D es discreto (códigos postales, provincias, regiones, países...) y las ubicaciones observadas no son estocásticas. De su estudio se suele encargar la econometría espacial (véase Cap. 24). También hay otro tipo de datos espaciales que surgen en dominios que pueden ser continuos o discretos, pero donde la selección de las ubicaciones observadas no depende del investigador (en este sentido D es aleatorio). Se trata de los denominados procesos de puntos (véase Cap.25).

hacer una representación exhaustiva del fenómeno, pero sí se puede reconstruir a partir de las observaciones disponibles.

Las consecuencias de utilizar la estadística clásica, que no considera la dependencia espacial, cuando la hay, son muy graves y pueden verse en [Montero et al. \(2015\)](#).

El ámbito de aplicación de la geoestadística es enorme: minería, industria petrolífera, geología, meteorología, control de la calidad del aire, ecología, epidemiología, salud pública, criminología, economía, etc. Así, por ejemplo, en el ámbito del control de la calidad del aire en las grandes urbes, la concentración de ozono en aire se mide en una serie de estaciones de seguimiento, y a partir de dichas mediciones se reproduce el comportamiento del proceso sobre toda la urbe.

En conclusión, las dos partes del análisis geoestadístico son: el análisis estructural de la dependencia espacial y la predicción (que se suele acompañar del calificativo “*krigeada*”). Pero antes de estudiarlas, detengámonos en algunos preliminares.

23.2. Preliminares

Dado que los procedimientos geoestadísticos no pueden ser aplicados directamente sobre los fenómenos regionalizados como tales, porque son realidades físicas, se necesita una descripción matemática de los mismos a la que puedan ser aplicados: la *variable regionalizada* (*v.r.*) o *regionalización*, definida en un espacio geográfico, y que se supone que mide y representa correctamente dicho fenómeno.

Formalmente, cuando \mathbf{s} recorre D , el conjunto $z(\mathbf{s}), \mathbf{s} \in D$, se denomina *v.r.*, siendo $z(\mathbf{s}_i), i = 1, 2, 3, \dots$ una colección de valores regionalizados.

Desde la perspectiva probabilística, cada uno de los valores que toma *v.r.* puede interpretarse como el resultado de un mecanismo aleatorio, la variable aleatoria, *v.a.* ([véase 15.3.](#)). Si se toman valores regionalizados en todos los puntos del dominio, D , es decir, si se considerase *v.r.*, ésta podría ser vista como un conjunto infinitamente grande de *v.a.*, una en cada punto de D , que se conoce como *función aleatoria* (*f.a.*), *proceso estocástico* o *campo aleatorio* espacial, $Z(\mathbf{s}), \mathbf{s} \in D$, donde Z representa el fenómeno de interés. Pues bien, *v.r.* se interpreta como una realización de una *f.a.* espacial, y esta es una decisión metodológica clave en geoestadística.

Es importante tener en cuenta que, (*i*) frecuentemente, *v.r.* es muy irregular a escala local, lo que impide su representación mediante una función determinista; y (*ii*) muestra cierta organización o estructura espacial. La interpretación de *v.r.* como una realización de una *f.a.* espacial permite considerar estos dos aspectos:

- En cada localización \mathbf{s} , $Z(\mathbf{s})$ es una *v.a.* (de ahí el aspecto errático).
- Para un conjunto de puntos dado, $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k$, las *v.a.* $Z(\mathbf{s}_1), Z(\mathbf{s}_2), \dots, Z(\mathbf{s}_k)$ están ligadas por una red de correlaciones espaciales que son las responsables de la similitud en los valores que toman (de ahí el aspecto estructurado).

Las *f.a.* $Z(\mathbf{s})$ pueden ser estacionarias (en sentido estricto o de segundo orden), intrínsecamente estacionarias o no estacionarias, y el hecho de que tengan uno u otro tipo de estacionariedad determina el análisis geoestadístico.

23.3. Análisis estructural de la dependencia espacial

399

Una *f.a.* espacial es *estrictamente estacionaria* si las familias de *v.a.* $Z(\mathbf{s}_1), Z(\mathbf{s}_2), \dots, Z(\mathbf{s}_k)$, tienen la misma distribución de probabilidad conjunta que $Z(\mathbf{s}_1 + \mathbf{h}), Z(\mathbf{s}_2 + \mathbf{h}), \dots, Z(\mathbf{s}_k + \mathbf{h})$, $\forall k$, $\forall \mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k$ y $\forall \mathbf{h} \in \mathbb{R}^d$ (donde \mathbf{h} es un vector de traslación), siempre que $\mathbf{s}_1 + \mathbf{h}, \mathbf{s}_2 + \mathbf{h}, \dots, \mathbf{s}_k + \mathbf{h} \in D$. Es decir, la distribución de probabilidad conjunta de $Z(\mathbf{s}_1 + \mathbf{h}), Z(\mathbf{s}_2 + \mathbf{h}), \dots, Z(\mathbf{s}_k + \mathbf{h})$ no se ve afectada por una traslación \mathbf{h} , y por tanto, ni ella, ni las funciones de densidad de dimensión inferior a k , dependen de las localizaciones consideradas.

La estacionariedad estricta es una condición muy restrictiva. Por ello, en la práctica lo que se suele asumir es la *estacionariedad de segundo orden*, que limita la estacionariedad a los dos primeros momentos de la *f.a.*³

Si una *f.a.* es estrictamente estacionaria, también es estacionaria de segundo orden. Sin embargo, la relación inversa no tiene por qué ser cierta.

La estacionariedad de segundo orden implica la existencia de la varianza de la *f.a.*, y deja fuera los fenómenos con infinita capacidad de variación. En este caso, si las diferencias $Z(\mathbf{s} + \mathbf{h}) - Z(\mathbf{s})$ son estacionarias de segundo orden, se dice que la *f.a.* es *intrínsecamente estacionaria*.

Aquellas *f.a.* cuya esperanza y/o varianza dependan de la localización (no son invariantes a las traslaciones) se denominan *no estacionarias*.

Salvo indicación de lo contrario, se asumirá la estacionariedad de segundo orden.

Finalmente, unos breves comentarios sobre la importancia de la estacionariedad. Es imposible inferir la ley de probabilidad que gobierna la *f.a.* espacial a partir de una sola realización de la misma (una sola regionalización), pues sería como tener una muestra de tamaño 1. Pero en la práctica ese será el caso. Bueno, ni siquiera eso. Solo se dispondrá de una parte de la regionalización: la correspondiente a las localizaciones observadas. La solución a tan importante limitación es adoptar la hipótesis de estacionariedad u homogeneidad espacial. Es decir, sustituir la repetición de realizaciones de la *f.a.* espacial por repeticiones en el espacio; dicho de otra forma, suponer que los valores observados en distintas localizaciones de D tienen las mismas características estadísticas y pueden ser considerados, en términos estadísticos, como realizaciones de la misma *f.a.*⁴ Por tanto, la hipótesis de estacionariedad significa que la ley espacial que gobierna *f.a.*, o parte de ella, es invariante a traslaciones; no depende de las localizaciones específicas observadas sino solo de \mathbf{h} .

La hipótesis de estacionariedad permitirá actuar como si todas las *v.a.* que conforman la *f.a.* tuviesen la misma distribución de probabilidad (o los mismos momentos), haciendo posible el proceso inferencial. Por eso se le da tanta importancia a que la *f.a.* sea estacionaria, del tipo que sea.

23.3. Análisis estructural de la dependencia espacial

³En geoestadística lineal el interés se centra en los dos primeros momentos de la *f.a.*, por lo cual tan sólo es necesaria la estacionariedad de segundo orden. Es decir, la esperanza y la varianza existen, son constantes y no dependen de la localización \mathbf{s} . La covarianza existe para cada par de *v.a.* $Z(\mathbf{s})$ y $Z(\mathbf{s} + \mathbf{h})$ y sólo depende de \mathbf{h} .

⁴Estas realizaciones no son independientes, y se suele asumir también la hipótesis de ergodicidad (véase 21.1).

23.3.1. Semivariograma

La estadística espacial se basa en la suposición de que las unidades georeferenciadas cercanas están relacionadas (son dependientes) de alguna manera (Getis, 1999), y tanto más cuanto más cercanas estén (Tobler, 1970a).

Los procesos con dependencia espacial se reconocen, visualmente, porque muestran un patrón en el espacio; en los que no la tienen, el patrón es el de la aleatoriedad. La Fig. 23.1 muestra una simulación de una *f.a.* con dependencia espacial (panel izquierdo) frente a unos datos totalmente aleatorios (panel derecho).

```
library(geoR)
library(fields)
par(mfrow = c(1, 2))
set.seed(728)

sim_dep <- grf(401, grid = "reg", cov.pars = c(1, 0.8), messages = FALSE)
points.geodata(sim_dep,
  main = "Dependencia espacial",
  col = tim.colors(), cex.max = 2
)

sim_indep <- grf(401, grid = "reg", cov.pars = c(0.01, 0), messages = FALSE)
points.geodata(sim_indep,
  main = "Aleatoriedad",
  col = tim.colors(), cex.max = 2
)
```

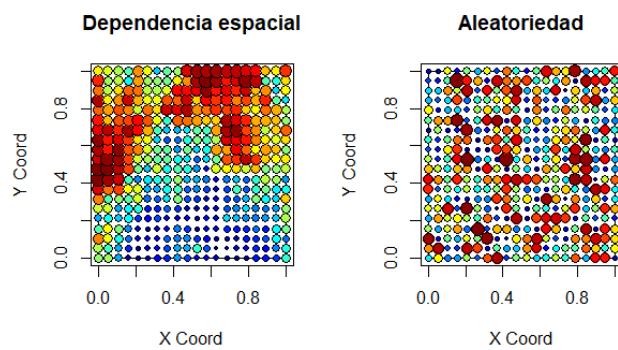


Figura 23.1: Dependencia espacial frente a aleatoriedad

Pasando del terreno de las simulaciones a la realidad, la Fig. 23.2 muestra la temperatura

23.3. Análisis estructural de la dependencia espacial

401

máxima en España el 6 de agosto de 2022⁵, en plena ola de calor (este es el ejemplo real que se utilizará a lo largo del capítulo). En ella puede observarse claramente una estructura de dependencia espacial, con máximas cercanas a 40 grados en la meseta central y Extremadura, de 30 grados o menos en la cordillera cantábrica y las costas atlántica, cantábrica y andaluza, y entre 30 y 35 grados en el resto del país (básicamente Murcia, Comunidad Valenciana y Cataluña).

```
library(CDR)
#summary(CDR::tempmax_data)

# renombra objetos por simplicidad en el análisis
ESP <- tempmax_data$ESP
ESP_utm <- tempmax_data$ESP_utm
grd_sf <- tempmax_data$grd_sf
grd_sp <- tempmax_data$grd_sp
temp_max_utm_sf <- tempmax_data$temp_max_utm_sf
temp_max_utm_sp <- tempmax_data$temp_max_utm_sp

library(ggplot2)
br_paper <- c(-Inf, seq(17.5, 45, 2.5), Inf)
pal_paper <- hcl.colors(15, "YlOrRd", rev = TRUE)

ggplot(ESP_utm) +
  geom_sf() +
  geom_sf(data = temp_max_utm_sf, aes(col = tmax), size = 4) + # temp_max_utm
  theme_light() +
  scale_color_gradientn(colours = pal_paper)
```

Ahora bien, para poder llevar a cabo predicciones geoestadísticas es necesario representar, previamente, los patrones de dependencia espacial observados mediante funciones que indiquen cuál es la estructura de dicha dependencia espacial. Dichas funciones son los semivariogramas. Dado que la identificación de la estructura de la dependencia espacial existente en el fenómeno de interés es la clave del éxito del proceso predictivo, al semivariograma se le considera la piedra angular de la predicción geoestadística (Montero et al., 2015).

Un semivariograma se define como la semivarianza de los incrementos de la *f.a.*:

$$\gamma(\mathbf{s}_i - \mathbf{s}_j) = \frac{1}{2}V[Z(\mathbf{s}_i) - Z(\mathbf{s}_j)], \forall \mathbf{s}_i, \mathbf{s}_j \in D. \quad (23.1)$$

que, en el caso habitual de *f.a.* estacionarias de segundo orden o intrínsecamente estacionarias (sin deriva), se transforma en:

$$\gamma(\mathbf{h}) = \frac{1}{2}V(Z(\mathbf{s} + \mathbf{h}) - Z(\mathbf{s})) = \frac{1}{2}E\left((Z(\mathbf{s} + \mathbf{h}) - Z(\mathbf{s}))^2\right), \quad (23.2)$$

Nótese que:

⁵Los datos ya procesados para el análisis se encuentran en `CDR::tempmax_data`. Una descripción puede verse con `summary(CDR::tempmax_data)`. Estos datos han sido descargados con la librería `climaemet` y `mapSpain`. Un desarrollo completo de manipulación de datos espaciales puede verse en Pizarro et al. (2021).

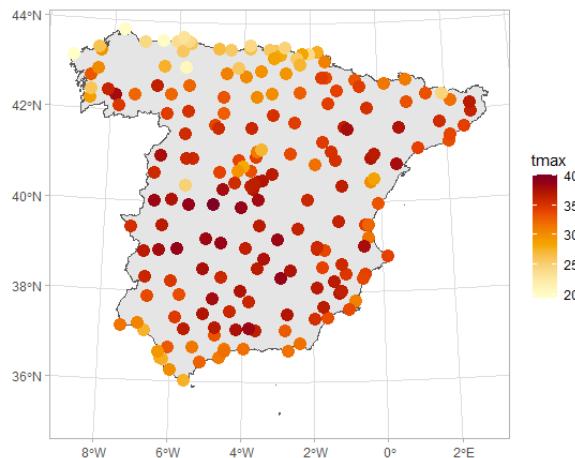


Figura 23.2: Temperatura máxima en España peninsular, 6 de agosto de 2022

- Si hay dependencia espacial (positiva⁶, es lo normal), la diferencia entre los valores de la *f.a.* en los puntos separados por una pequeña distancia será poca y más o menos la misma, es decir, dichas diferencias serán poco variables, y el valor del semivariograma, a pequeñas distancias, será pequeño.
- Si aumenta la distancia, la dependencia espacial se reduce y la diferencia entre los valores de la *f.a.* en los puntos separados por distancias intermedias y grandes no será tan parecida como en el caso anterior, sino mayor; y variará más: Es decir, el valor del semivariograma aumenta con la distancia.
- Si la distancia aumenta lo suficiente como para que ya no haya dependencia espacial, las diferencias entre los valores de la *f.a.* separados por tal distancia alcanzarán la variabilidad de la *f.a.* en estudio, y si ésta es estacionaria de segundo orden, el semivariograma se estabilizará en torno a ella.

En el caso estacionario, las funciones de covarianza, $C(\mathbf{h})$, también pueden ser utilizadas para representar la estructura de la dependencia espacial, si bien se prefiere el semivariograma porque no requiere el conocimiento de la media de la *f.a.* en estudio. Además, el semivariograma cubre un espectro más amplio de fenómenos regionalizados que la función de covarianza, ya que ésta no puede definirse en el caso de estacionariedad intrínseca. Los detalles pueden verse en Montero et al. (2015) y Montero and Larraz (2008). \index{dependencia espacial}

Cuando el semivariograma depende tanto de la dirección como de la longitud del vector \mathbf{h} que une las localizaciones \mathbf{s} y $\mathbf{s} + \mathbf{h}$, se denomina *anisotrópico*. Cuando solo depende de la distancia ($|\mathbf{h}| = \|\mathbf{h}\|$, porque en el espacio euclídeo el módulo y la norma coinciden) se denomina *isotrópico*.

⁶Valores cercanos similares. Si es negativa, los valores vecinos son diferentes.

23.3. Análisis estructural de la dependencia espacial

403

Un semivariograma no puede ser cualquier función. Tiene que ser nulo en el origen, no negativo, verificar que $\gamma(\mathbf{h}) = \gamma(-\mathbf{h})$, debe ser una función condicionalmente definida negativa y tener un ritmo de crecimiento inferior a $|\mathbf{h}|^2$, es decir, $\lim_{|\mathbf{h}| \rightarrow \infty} \frac{\gamma(\mathbf{h})}{|\mathbf{h}|^2} = 0$ cuando el proceso es no estacionario (sin deriva), siendo finito en caso de procesos estacionarios de segundo orden.

El análisis del comportamiento de un semivariograma a pequeñas, medias y grandes distancias es de sumo interés, como se verá a continuación.

En general, a distancias medias y grandes, los semivariogramas asociados a *f.a.* estacionarias de segundo orden crecen, monótonamente, desde el origen con la distancia, hasta alcanzar un valor límite, la *varianza a priori* de la *f.a.* (o covarianza para $\mathbf{h} = 0$, $C(\mathbf{0})$), bien de forma exacta o asintóticamente. Dicho valor límite se denomina *meseta*, m , y la distancia a la cual se alcanza se conoce como *alcance* o *rango*, a . Por tanto, el rango es la distancia a partir de la cual ya no hay dependencia espacial. Cuando m se alcanza asintóticamente, el alcance no queda perfectamente definido y se toma como alcance, a efectos prácticos, a' , la distancia a la cual el semivariograma toma el valor 0,95m.

En el caso no estacionario (por ejemplo, si hay deriva) o intrínsecamente estacionario el semivariograma no tiene meseta.

El comportamiento a pequeñas distancias, sobre todo cerca del origen, que es donde más dependencia espacial hay, está muy relacionado con el grado de continuidad y regularidad de *f.a.* Cuanto más continua y regular sea, más suaves y estructuradas serán las realizaciones que produzca, y más regular será el comportamiento del semivariograma cerca del origen (Fig. 23.3, panel izquierdo; representación bidimensional).

Los semivariogramas con un comportamiento lineal cerca del origen son típicos de *v.r.* continuas, al menos por partes, pero no diferenciables. Su representación gráfica tridimensional está llena de picos. La amplitud de las fluctuaciones aumenta con la distancia entre localizaciones y es proporcional a la pendiente de la tangente en el origen (Fig. 23.3, panel derecho; representación bidimensional).

```
library(fields)
library(geoR)
par(mfrow = c(1, 2))
set.seed(123)

fa_gauss <- grf(1225, grid = "reg", cov.pars = c(1, .25), cov.model = "gaussian")
image(fa_gauss, col = tim.colors())

fa_sph <- grf(1225, grid = "reg", cov.pars = c(1, .25), cov.model = "spherical")
image(fa_sph, col = tim.colors())
```

Las *v.r.* regulares (aquellas cuya gráfica tridimensional no tiene picos) se identifican con un comportamiento semivariográfico parabólico en el origen. Si dicho comportamiento persiste a largas distancias, puede que exista una fuerte deriva.

Las discontinuidades en el origen (que teóricamente no pueden darse) son frecuentes en la práctica. Su amplitud se denomina “efecto pepita” (nugget effect) y son típicas de variables

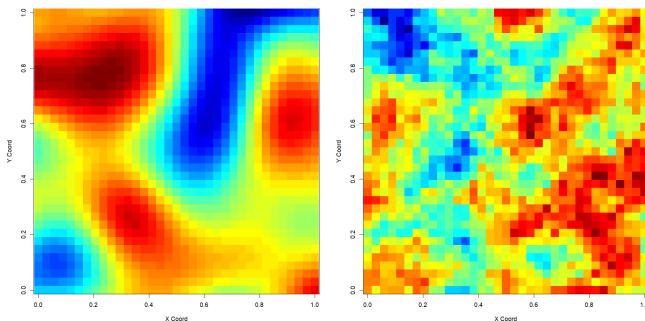


Figura 23.3: Representación bidimensional de dos *f.a.*: con semivariograma parabólico en el origen (izquierda); con semivariograma lineal en el origen (derecha)

regionalizadas muy irregulares y, quizás, discontinuas. Las causas más frecuentes del “efecto pepita” son la existencia de una estructura con alcance inferior a la distancia más corta entre localizaciones y los errores de posicionamiento o de medida (véase Chilès and Delfiner (1999) para más detalle).

El caso límite del efecto pepita es el “efecto pepita puro”. En ese caso, el semivariograma es constante cualquiera que sea la distancia, indicando ausencia de dependencia espacial.

La Fig. 23.4 muestra gráficamente los principales elementos de un semivariograma.

```
library(geoR)
semivar <- function(x, ...) {
  1 - cov.spatial(x, ...)
}
curve(semivar(x, cov.pars = c(0.8, 0.4), cov.model = "sph"), 0.0, 1,
      xlab = "Distancia",
      ylab = expression(bold(gamma("|h|"))), lwd = 4, lty = 1, col = "4", xlim = c(0.03,
      ↪ 1), ylim = c(0, 1)
)
abline(v = 0.4, col = 2, lty = 2, lwd = 2) # alcance
abline(h = 1, col = 3, lty = 2, lwd = 2) # meseta
legend(-0.05, 0.15, "Efecto pepita")
legend(0, 0.95, "Meseta")
legend(0.25, 0.5, "Alcance")
legend(0.5, 0.75, "Ausencia de dependencia")
```

```
knitr::include_graphics("img/semivar-parts.png")
```

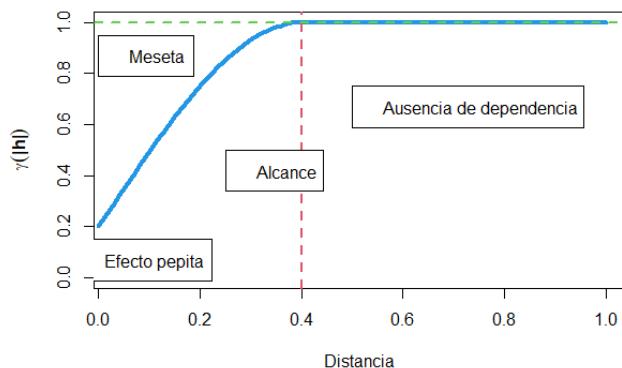


Figura 23.4: Elementos del semivariograma (meseta unitaria)

23.3.2. Modelos de semivariogramas válidos

Las funciones que verifican las condiciones que debe cumplir un semivariograma (véase 23.3.1) se conocen como semivariogramas válidos. El incumplimiento de alguna de ellas tiene perversas consecuencias en el proceso predictivo (por ejemplo, varianzas de los errores de predicción negativas). Su tipología, siguiendo el enfoque sugerido en Journel and Huijbregts (1978) y Montero et al. (2015), es la siguiente:⁷

23.3.2.1. Semivariogramas con meseta

Están asociados a *f.a.* estacionarias de segundo orden. Los más utilizados son:

- **Esférico.** Válido en \mathbb{R}^1 , \mathbb{R}^2 y \mathbb{R}^3 , y viene dado por:

$$\gamma(|\mathbf{h}|) = \begin{cases} m \left(1,5 \frac{|\mathbf{h}|}{a} - 0,5 \left(\frac{|\mathbf{h}|}{a} \right)^3 \right) & \text{si } |\mathbf{h}| \leq a \\ m & \text{si } |\mathbf{h}| > a. \end{cases} \quad (23.3)$$

Tiene un comportamiento lineal cerca del origen, indicando continuidad y cierto grado de irregularidad en la *f.a.* A grandes distancias alcanza la meseta cuando $|\mathbf{h}| = a$. Estas dos características son propias de muchas regionalizaciones observadas en la realidad; de ahí su popularidad.

⁷Aquí se presentan los semivariogramas isotrópicos más utilizados en la práctica. Los semivariogramas anisotrópicos pueden ser representados por semivariogramas isotrópicos sin más que llevar a cabo una transformación lineal de las coordenadas, y porque las anisotropías anisotropías) pueden representarse separadamente mediante semivariogramas isotrópicos. Para un análisis detallado, consultese, por ejemplo, Montero et al. (2015).

- **Exponencial.** Válido en $\mathbb{R}^d, d \geq 1$ y viene dado por:

$$\gamma(|\mathbf{h}|) = m \left(1 - \exp \left(-\frac{|\mathbf{h}|}{a} \right) \right). \quad (23.4)$$

Igual que el esférico, cerca del origen exhibe un comportamiento lineal, siendo menor la pendiente. A diferencia de él, solo alcanza la meseta asintóticamente. A efectos prácticos, se toma como alcance la distancia para la cual el semivariograma alcanza el valor del 95 % de la meseta, $a' \approx 3a$.

- **Gausiano.** Válido en $\mathbb{R}^d, d \geq 1$. Está definido por:

$$\gamma(|\mathbf{h}|) = m \left(1 - \exp \left(-\frac{|\mathbf{h}|^2}{a^2} \right) \right). \quad (23.5)$$

A diferencia del esférico y el exponencial, tiene un comportamiento parabólico cerca del origen. Por consiguiente, está asociado con *f.a.* estacionarias de segundo orden infinitamente diferenciables y, en consecuencia, muy regulares. Igual que el modelo exponencial, alcanza la meseta sólo asintóticamente, con $a' \approx a\sqrt{3}$.

- **Efecto pepita puro.** Refleja la ausencia de dependencia espacial:

$$\gamma(|\mathbf{h}|) = \begin{cases} m & \text{si } |\mathbf{h}| = 0 \\ 0 & \text{si } |\mathbf{h}| > 0 \end{cases}, \quad m > 0. \quad (23.6)$$

- **K-Bessel.** Válido in $\mathbb{R}^d, d \geq 1$. Su expresión es:

$$\gamma(|\mathbf{h}|) = m \left(1 - \frac{1}{2^{\alpha-1}\Gamma(\alpha)} \left(\frac{|\mathbf{h}|}{a} \right)^\alpha K_\alpha \left(\frac{|\mathbf{h}|}{a} \right) \right), \quad \alpha > 0, \quad (23.7)$$

donde K_α es la función de segunda especie de orden α . Este modelo puede representar cualquier tipo de comportamiento cerca del origen. Por ejemplo, para $\alpha = 0,5$ se obtiene el modelo exponencial.

23.3.2.2. Semivariogramas con meseta y efecto hoyo

- **J-Bessel.**

Un semivariograma no tiene por qué ser necesariamente una función monótona no decreciente, sino que puede tener “ondas” (efecto hoyo). Tal es el caso del modelo J-Bessel, que puede ser utilizado en presencia de dependencia espacial negativa o, específicamente, en caso de alternancia entre dependencia positiva y negativa. Válido en $\mathbb{R}^d, d \leq 2(\alpha + 1)$, su expresión viene dada por:

$$\gamma(|\mathbf{h}|) = m \left(1 - \left(\frac{2a}{|\mathbf{h}|} \right)^\alpha \Gamma(\alpha + 1) J_\alpha \left(\frac{|\mathbf{h}|}{a} \right) \right), \quad (23.8)$$

23.3. Análisis estructural de la dependencia espacial

407

donde α es un parámetro de forma, a es un parámetro de escala, Γ es la función de Euler que interpola el factorial y J_α es la función J-Bessel de primera especie de orden α .

La Fig. 23.5 muestra una representación gráfica de los anteriores semivariogramas.

```
library(gstat)
show.vgms(models = c("Sph", "Exp", "Gau", "Nug", "Bes", "Wav"))
```

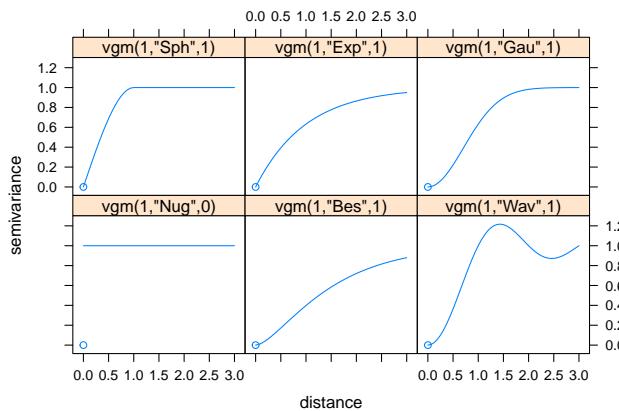


Figura 23.5: Representación de semivariogramas con meseta válidos (meseta y alcance efectivo unitarios; a excepción del efecto pepita puro)

23.3.2.3. Semivariogramas sin meseta

Estos modelos van más allá de la hipótesis estacionaria de segundo orden y corresponden a *f.a.* con una capacidad ilimitada de dispersión espacial, es decir, a *f.a.* intrínsecamente estacionarias, pero no estacionarias de segundo orden.

- **Potencial.** Válido en $\mathbb{R}^d, d \geq 1$ y definido por:

$$\gamma(|\mathbf{h}|) = (|\mathbf{h}|)^\alpha, \quad \text{con } 0 < \alpha < 2, \quad (23.9)$$

- **Logarítmico.** Válido en $\mathbb{R}^d, d \geq 1$ y con expresión:

$$\gamma(|\mathbf{h}|) = b \log |\mathbf{h}| \quad \text{si } |\mathbf{h}| \geq 0, \quad (23.10)$$

donde b es una constante.

Una representación gráfica de ambos semivariogramas puede verse en la Fig. 23.6.

```
library(gstat)
show.vgms(models = c("Pow", "Log"), sill = 1, range = c(2, 1), nugget = 0)
```

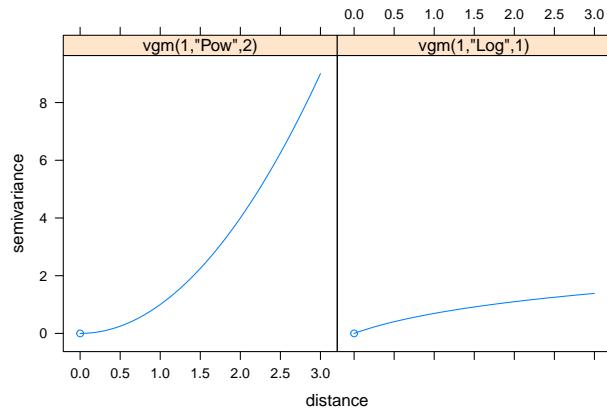


Figura 23.6: Representación de semivariogramas sin meseta válidos

23.3.3. Semivariograma empírico

Dado que la única información de la que se dispone es una realización observada de la *f.a* objeto de estudio, en la práctica la estructura de la dependencia espacial se estima mediante el semivariograma empírico.

En el marco de la estacionariedad intrínseca (que incluye la estacionariedad estricta y de segundo orden), y en \mathbb{R}^d , $d \geq 1$, se estiman (insegadamente) los valores semivariográficos para un número determinado de distancias, por el método de los momentos:

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2\#N(\mathbf{h})} \sum_{N(\mathbf{h})} (Z(\mathbf{s}_i + \mathbf{h}) - Z(\mathbf{s}_i))^2, \quad (23.11)$$

donde $\#N(\mathbf{h})$ es el número de parejas de localizaciones separadas por el vector \mathbf{h} .

La función que mejor ajusta las estimaciones de los valores semivariográficos anteriormente referidos se denomina *semivariograma empírico*, y también se suele denotar por $\hat{\gamma}(\mathbf{h})$.

Los valores semivariográficos se suelen computar para distancias inferiores a la mitad del diámetro de D , porque, para distancias superiores, el número de parejas de localizaciones suele ser pequeño para proporcionar estimaciones fiables. En la práctica, como en muchas de las direcciones no hay un número de parejas suficiente para calcular el semivariograma con cierta fiabilidad, lo habitual es construir un *semivariograma empírico omnidireccional*, es decir, que depende solo de la distancia (longitud del vector \mathbf{h}) y no de la dirección. Para ello se crean regiones de tolerancia, que no se solapen, basadas en intervalos de distancia (normalmente de la

23.3. Análisis estructural de la dependencia espacial

409

misma longitud) y un angulo de tolerancia. En concreto, la tolerancia se especifica en el módulo de \mathbf{h} ($\pm\Delta|\mathbf{h}|$) y su dirección ($\pm\Delta\theta$). Para más detalles y ejemplos, véase [Montero et al. \(2015\)](#).

La Fig. 23.7 muestra la ubicación de los puntos semivariográficos, indicando el número de parejas a cada distancia, en el caso de las temperaturas máximas en España (06/08/2022).

```
vgm_tmax <- variogram(tmax ~ 1, temp_max_utm_sf,
  cutoff = 250000 # 250 km
)
plot(vgm_tmax, plot.numbers = TRUE, pch = "+", lwd = 2, cex = 2)
```

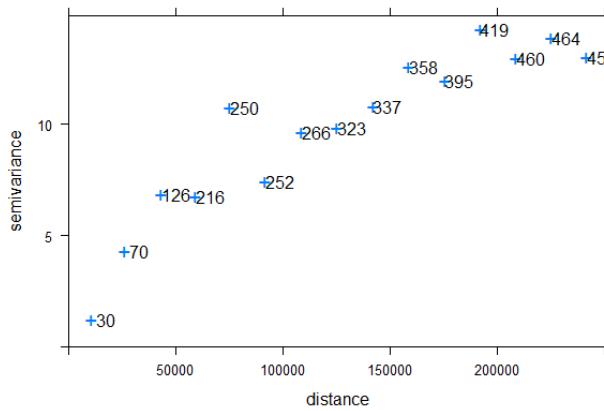


Figura 23.7: Valores semivariográficos. Temperaturas máximas (06/08/2022)

En el ejemplo ilustrado, las distancia mínima entre dos estaciones meteorológicas es 1.125m y la máxima 1.027.597m. Sin embargo, dada la geometría del mapa de España (aunque de Huelva a Gerona hay 987 km en linea recta, más de dos terceras partes de las ciudades españolas no están separadas más de 500 km.), se consideró 250.000m (1/4 de la distancia máxima) como distancia máxima a la hora de calcular los valores semivariográficos, ya que a partir de dicha distancia el número de parejas no es lo suficientemente grande como para obtener resultados fiables. Por convenio, `gstat` divide la distancia en 15 intervalos (`geoR` se divide en 13 porque los autores lo hicieron un viernes 13).

23.3.4. Ajuste semivariográfico

Cualquier función que dependa de una distancia y una dirección no es necesariamente un semivariograma, pues para ello tienen que cumplir los requisitos especificados en 23.3.1. Esta es la razón por la que el semivariograma empírico no puede utilizarse directamente para realizar predicciones geoestadísticas. Por ello, a los valores semivariográficos estimados se les ajusta

una función que represente un semivariograma válido. Sin embargo, esta tarea, clave para el éxito del posterior proceso predictivo, no es sencilla ni existe consenso en torno a ella. El ajuste puede ser *manual*, utilizando métodos visuales y gráficos, o *automático*, que usa procedimientos estadísticos. Una combinación de ambos es muy recomendable.

El ajuste manual pudiera parecer “no muy científico” pero, dado que lo más importante a la hora del ajuste no es tanto la bondad del ajuste para todos los puntos semivariográficos sino lo bien que un semivariograma válido representa las principales características del fenómeno, especialmente el tipo de estacionariedad (comportamiento a largas distancias) y, sobre todo, el tipo de continuidad (comportamiento cerca del origen), resulta ser un procedimiento muy práctico si se guía por las anteriores consideraciones. En este sentido, cualquier conocimiento sobre el fenómeno en estudio es bienvenido.

El ajuste automatizado mediante procedimientos estadísticos incluye los *métodos de mínimos cuadrados* (tanto ordinarios como generalizados y ponderados), que son los más populares en la práctica, y los *métodos basados en máxima verosimilitud*, que incluyen, entre otros, el tradicional método máximo verosímil, la máxima verosimilitud restringida y el método de la verosimilitud compuesta.

La Fig. 23.8 muestra el semivariograma empírico correspondiente a los puntos semivariográficos de la Fig. 23.7. De todos los modelos con meseta, el semivariograma ajustado ha sido un exponencial con alcance 76.404,64 metros y meseta 13,74.

```
fit_vgm_tmax <- fit.variogram(vgm_tmax,
  model = vgm(model = c("Sph", "Exp", "Gau", "Nug", "Bes",
  → "Wav")), fit.sills = TRUE, fit.ranges = TRUE,
  → fit.kappa = TRUE, fit.method = 7)
fit_vgm_tmax
#>   model    psill     range
#> 1  Exp 13.74102 76404.64
attr(fit_vgm_tmax, "SSER")
#> [1] 6.200657e-07
plot(vgm_tmax, fit_vgm_tmax, lwd = 2, col = "2", pch = "*", cex = 3)
```

El método de ajuste ha sido el que figura por defecto en la función `vgm`: mínimos cuadrados ponderados con ponderaciones $\frac{N_{|h|}}{|h|^2}$, que funciona bien en la práctica y selecciona el semivariograma que mejor ajuste cuando el número de parejas es elevado y la distancia pequeña, que es la parte del semivariograma que hay que ajustar bien porque a pequeñas distancias es cuando más dependencia espacial hay. Respecto a los parámetros iniciales, aunque el investigador puede especificar los que considere convenientes, se recomienda utilizar los que tiene la función por defecto: (i) alcance igual a 1/3 de la distancia máxima en la muestra; (ii) como efecto pepita se toma la media de los tres primeros valores semivariográficos; y (iii) como meseta parcial (meseta menos efecto pepita), la media de los cinco últimos valores semivariográficos.

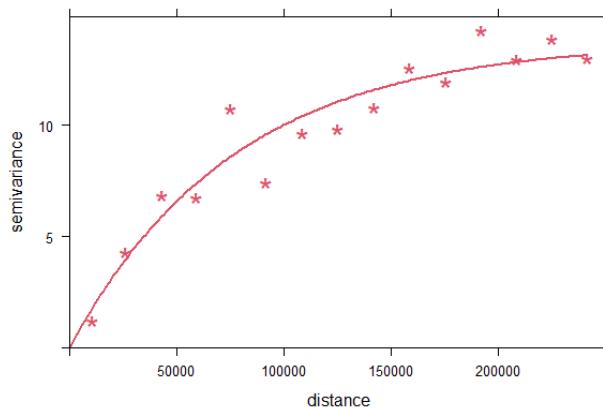


Figura 23.8: Semivariograma empírico. Temperaturas máximas (06/08/2022)

23.4. Kriging

Seleccionado el semivariograma válido que mejor se ajusta a los puntos semivariográficos, se aborda el proceso predictivo. El método predictivo que usa la geoestadística es conocido como *kriging* en honor al ingeniero de minas D.G. Krige.

El kriging tiene como objetivo predecir el valor de una *f.a.*, $Z(\mathbf{s})$, en uno o más puntos (o bloques) no observados, a partir de la regionalización observada (pueden ser puntos o bloques) en un dominio D , y proporciona el mejor predictor lineal insesgado de la *v.r.* de interés en tales puntos o bloques no observados⁸. La limitación a la clase de predictores lineales obedece a que, bajo estacionariedad de segundo orden, solo se requiere el conocimiento de los momentos de segundo orden de la *f.a.* Con más información estructural, pueden definirse predictores no lineales.

Las principales ventajas del kriging sobre los métodos de interpolación espacial deterministas (método de la distancia inversa, splines, regresión polinomial, etc.), es que (*i*) considera la estructura de la dependencia espacial (dando lugar a mejores predicciones), (*ii*) proporciona, junto con la predicción, la varianza del error de predicción y (*iii*) es un interpolador exacto.

Dependiendo del tipo de estacionariedad que se considere en la *f.a.* el kriging puede ser: universal (no estacionariedad en media) u ordinario (estacionariedad de segundo orden o intrínseca). Nos centraremos en el kriging ordinario (*KO*). La generalización al caso universal (hay derivas: la media depende de las localizaciones en vez de ser constante) puede verse en [Montero et al. \(2015\)](#).

En términos formales, *KO* se plantea como sigue: Sea $Z = \{Z(\mathbf{s}), \mathbf{s} \in D\}$ una *f.a.* con estacionariedad de segundo orden o intrínseca y con media desconocida (cuando se conoce, *KO*

⁸En lo que sigue, la exposición se centrará en la predicción puntual a partir de datos puntuales (es decir, en el soporte puntual). La generalización a bloques puede verse en [Montero et al. \(2015\)](#).

se denomina kriging simple). Sea el predictor lineal krigeado $Z^*(\mathbf{s}_0) = \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i)$, donde las ponderaciones $\lambda_i, i = 1, 2, \dots, n$, se obtienen imponiendo al error de predicción las condiciones de esperanza nula y mínima varianza.

El sistema de ecuaciones que proporciona dichas ponderaciones óptimas es:

$$\begin{cases} \sum_{j=1}^n \lambda_j \gamma(\mathbf{s}_i - \mathbf{s}_j) + \alpha = \gamma(\mathbf{s}_i - \mathbf{s}_0), & i = 1, \dots, n \\ \sum_{i=1}^n \lambda_i = 1, \end{cases} \quad (23.12)$$

siendo la varianza del error de predicción: $\sigma_{OK}^2(\mathbf{s}_0) = \sum_{i=1}^n \lambda_i \gamma(\mathbf{s}_i - \mathbf{s}_0) + \alpha$, donde α es el multiplicador de Lagrange involucrado en el proceso de optimización.

Retomando el ejemplo de las temperaturas máximas en la España peninsular el 6 de agosto de 2022, a continuación se muestra el código necesario para la creación de un *mapping* de predicción de dichas temperaturas.

```
kriged_tmax <- krig(tmax ~ 1,
  temp_max_utm_sp,
  grd_sp,
  model = fit_vgm_tmax
)
#> [using ordinary kriging]

kriged_df <- as.data.frame(kriged_tmax, xy = T, na.rm = T)

ggplot() +
  geom_tile(data = kriged_df,
    aes(x = coords.x1, y = coords.x2, fill = var1.pred)
  ) +
  geom_sf(data = ESP_utm, col = "black", fill = NA) +
  scale_fill_gradientn(colours = pal_paper,
    breaks = br_paper,
    labels = function(x) {
      paste0(x, "%")
    },
    guide = guide_legend(reverse = TRUE, title = "Temp. max.")
  ) +
  theme_light() +
  theme(panel.background = element_blank(),
    panel.border = element_blank(),
    axis.title = element_blank(),
  )

```

El *mapping* de la Fig. 23.9 tiene poco valor si no se acompaña de otro que muestre la desviación típica de los errores de predicción.

23.4. Kriging

413

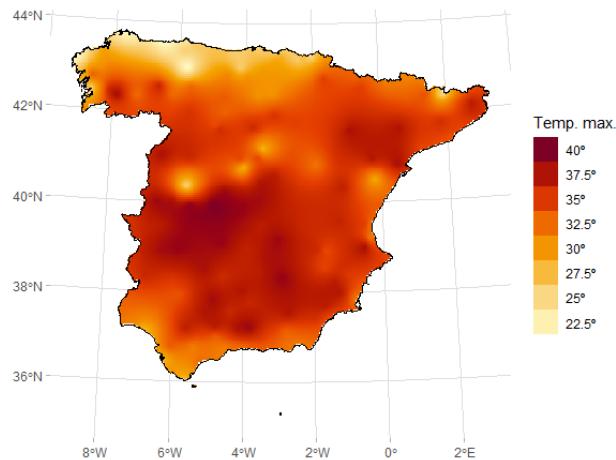


Figura 23.9: Mapping de temperaturas máximas (06/08/2022).

```
ggplot(kriged_df) +
  geom_contour_filled(aes(coords.x1, coords.x2, z = sqrt(var1.var)),
    breaks = c(0, 2, 2.5, 3, 3.5, 4, max(sqrt(kriged_df$var1.var))))
  ) +
  geom_sf(data = ESP_utm, col = "black", fill = NA) +
  geom_sf(data = temp_max_utm_sf, col = "blue", shape = 4) +
  scale_fill_manual( # paleta colores
    values = c("springgreen", hcl.colors(8, "PuRd", rev = TRUE)),
    guide = guide_legend(title = "Desv. típica\n error predicción")
  ) +
  theme_light() +
  theme(panel.background = element_blank(),
    panel.border = element_blank(),
    axis.title = element_blank(),
  )
```

Como se aprecia en la Fig. 23.10, cuanto mayor es el número de localizaciones observadas alrededor del punto de predicción, menor es la desviación típica del error de predicción.

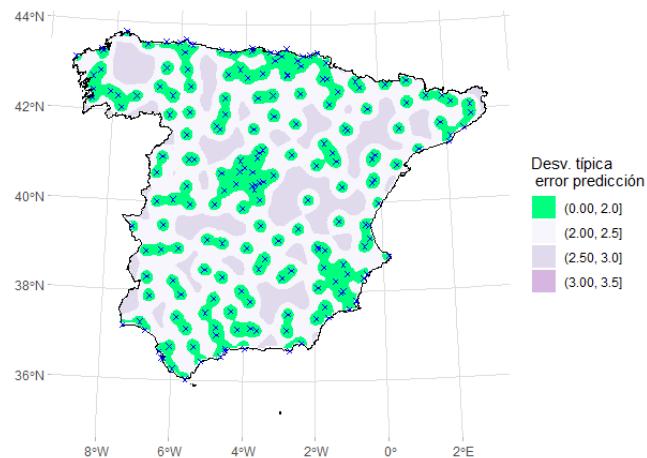


Figura 23.10: Desviaciones típicas del error de predicción

RESUMEN. La geoestadística estudia de fenómenos regionalizados, que son aquellos que se extienden en el espacio y presentan una organización o estructura debida a la dependencia espacial existente. Su objetivo es sacar provecho de dicha dependencia espacial para llevar a cabo predicciones (interpolaciones) óptimas en ubicaciones o áreas de interés, o la realización de *mappings* sobre todo el dominio o parte de él. Las dos partes del análisis geoestadístico son: (*i*) el análisis estructural de la dependencia espacial y (*ii*) la predicción “krigeada”. La estructura de dependencia espacial se representa mediante un semivariograma. La elección del semivariograma entre el elenco de funciones semivariográficas válidas es la clave del éxito de la predicción geoestadística, y por ello al semivariograma se le considera la piedra angular de la geoestadística. La técnica que utiliza la geoestadística para predecir se denomina *kriging*, y presenta un buen número de ventajas sobre los tradicionales métodos de interpolación espacial deterministas al considerar la estructura espacial de las observaciones.

Capítulo 24

Modelos econométricos espaciales

Andrés Vallone^a y Coro Chasco^{b,c}

^a Escuela de Ciencias Empresariales-Instituto de Políticas Públicas Universidad Católica del Norte ^b Departamento de Economía Aplicada, Universidad Autónoma de Madrid ^c Universidad de Neubria

24.1. La dependencia espacial

En muchas ocasiones, los fenómenos de estudio no son independientes del espacio geográfico en el cual se producen. Esto se refleja en la **primera ley de la Geografía** enunciada por [Tobler \(1970b\)](#) “Todas las cosas están relacionadas entre sí, pero las cosas más próximas en el espacio tienen una relación mayor que las distantes” ([Tobler, 1970b](#), p 236). Esta situación, produce una violación del supuesto básico de independencia de las variables aleatorias requerido por el método de estimación de mínimos cuadrados ordinarios (MCO).

En este contexto, los MCO ya no son óptimos y, por tanto, los estadísticos de contraste t y F pueden llevar a conclusiones erróneas([Anselin, 1988](#)). Por ello es necesario encontrar la manera de incorporar el espacio geográfico en los procesos de modelación. En este capítulo se abordará esta cuestión, mostrando primero los métodos de exploración de datos espaciales, para luego presentar las formas de modelización del espacio y los métodos de estimación.

Los modelos de econometría espacial se centran en manejar las situaciones de **dependencia espacial**. Existe dependencia espacial cuando lo que sucede en una locación i está influenciado por lo que sucede en una locación j y viceversa ([Anselin, 2013](#)). La dependencia espacial se traduce en que los valores de la variable en las locaciones i y j con $i \neq j$ están correlacionados entre sí, hecho que se conoce como **autocorrelación espacial** ([Anselin, 2013](#)). La autocorrelación espacial puede ser positiva, cuando las locaciones con valores similares tienden a estar juntas (altos con altos, bajos con bajos) o negativa, cuando las unidades espaciales tienden a estar rodeadas de vecinos con valores diferentes. Los patrones espaciales formados por la existencia

de autocorrelación se muestran en la Figura 24.1. La ausencia de algún tipo de autocorrelación es lo que se entiende como aleatoriedad espacial (Anselin, 2013).

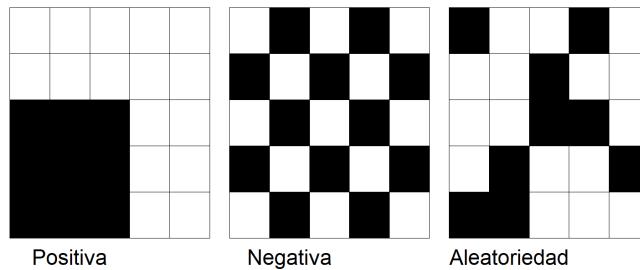


Figura 24.1: Patrones de autocorrelación espacial

24.1.1. Modelización del espacio

El espacio puede jugar un rol importante en la determinación de los procesos a modelizar. Por ello, resulta relevante encontrar una forma de incorporar el espacio en los procesos de estimación. Para modelizar la interacción de una variable consigo misma es natural pensar en el concepto de autocorrelación. No obstante, a diferencia de la autocorrelación temporal, que es unidireccional (sólo el pasado puede afectar el presente), en el caso del espacio la influencia es multidireccional en el entorno o vecindario de la localidad de análisis y, por tanto, es crucial definir el **vecindario**.

La matriz de vecindad o contigüidad $\mathbf{W}_{n \times n}$ muestra la relación entre las n localidades analizadas, y por tanto la interacción existente entre ellas. Es una matriz simétrica y binaria, de forma que $w_{ij} = 1$ si las localidades i y j son vecinas y cero si no lo son. Por tanto, $w_{ii} = 0$ puesto que una localidad no puede ser vecina de sí misma. Existen distintos criterios de definición de vecindad dependiendo del proceso que se desee modelizar y las características de los datos. Si se cuenta con un mapa de polígonos, entonces podemos utilizar alguno de los criterios que se presentan en la Figura 24.2 para configurar la matriz \mathbf{W} .

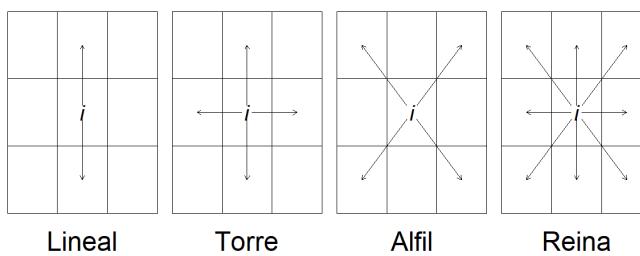


Figura 24.2: Criterios de vecindad

Las matrices \mathbf{W} generadas bajo el **criterio lineal** consideran como vecinas a la localidad i todas aquellas localidades que comparten un borde situadas en la misma dirección cardinal, norte sur

o este oeste, de esta localidad. El resto de los criterios de contigüidad siguen los movimientos de las piezas del ajedrez para definir la vecindad de la localidad i . La construcción de una matriz de vecindad bajo el **criterio de la torre** implica considerar como vecinos de la localidad i aquellas localidades situadas al norte, sur, este u oeste y que comparten un borde en común con dicha localidad. El uso del **criterio de alfil** considera como vecindad de la localidad i aquellas localidades situadas al noreste, noroeste, sureste o suroeste de la localidad i y que tengan, al menos, un punto en común. El **criterio de la reina** considera como vecindario de la unidad espacial i a las localidades en todas las direcciones cardinales y que tengan al menos un punto en común con ella (Martori et al., 2008).

Dependiendo del fenómeno que se analice, la matriz de contigüidad puede ser construida considerando un vecindario más amplio; por ejemplo, considerando como vecinos de la localidad i a los vecinos de los vecinos de dicha localidad, en este caso se dice que la matriz de vecindad es de orden 2 (los vecinos y los vecinos de los vecinos). Las matrices \mathbf{W} , se utilizan para capturar el efecto del vecindario a partir de medias ponderadas basadas en la cercanía de las unidades espaciales. Es por ello que las matrices de vecindad se estandarizan por filas. Los elementos de la matriz estandarizada se obtienen de la siguiente manera:

$$w_{ij}^e = \frac{w_{ij}}{\sum_{j=1}^n w_{ij}} \quad (24.1)$$

En palabras simples, se divide cada elemento de una fila de la matriz \mathbf{W} por la suma de dicha fila. Esto asegura que cada elemento de la matriz \mathbf{W} estandarizada se encuentre entre 0 y 1, y que la suma de cada una de sus filas sea siempre 1. Las matrices de vecindad estandarizadas llevan el nombre de matrices de pesos espaciales. A partir de ahora, cuando se haga referencia a la matriz \mathbf{W} se estará haciendo referencia a una matriz de pesos espaciales.

Para el cálculo de las matrices de vecindad se utilizará el paquete `spdep` (Bivand, 2022). La función `poly2nb()` construye la relación de vecindad a partir de los polígonos de un objeto espacial según el criterio y el orden que se indique; la función `nb2listw()` transforma la relación de vecindad en una lista de pesos espaciales. Para el ejemplificar la construcción de la matriz \mathbf{W} , se utilizará el conjunto de datos del estudio de Guerry (1833) utilizados en (Anselin, 2017). Esta base de datos contiene información respecto a estadísticas morales, criminales y sociales de las distintas provincias de Francia en 1830¹.

```
library("spdep")
library("CDR")

reina<-poly2nb(guerry, queen=TRUE)
w_reina<-nb2listw(reina, style="W", zero.policy=TRUE)
w_reina
#> Characteristics of weights list object:
#> Neighbour list object:
#> Number of regions: 85
#> Number of nonzero links: 420
```

¹Una descripción completa de la base de datos está disponible en <https://geodacenter.github.io/data-and-lab/Guerry/>

```
#> Percentage nonzero weights: 5.813149
#> Average number of links: 4.941176
#>
#> Weights style: W
#> Weights constants summary:
#>   n  nn  S0      S1      S2
#> W 85 7225 85 37.2761 347.6683
```

Para construir una matriz de contigüidad de la torre de orden 1 se utilizan las mismas funciones, cambiando el parámetro `queen` de la función `poly2nb()`

```
torre<-poly2nb(guerry, queen=FALSE)
w_torre<-nb2listw(torre, style="W", zero.policy=TRUE)
```

Cuando se trabaja con datos a nivel puntual, o cuando existen situaciones geográficas de no contigüidad como, por ejemplo, una isla, la construcción de la matriz de contigüidad no es tan evidente. En estos casos, resulta más oportuno definir la matriz de vecindad a partir de **criterios de distancia**. Las matrices de \mathbf{W} basadas en distancias pueden tener configuraciones continuas de la matriz respecto a la distancia d entre las localidades i y j de tal manera que $w_{ij} = 1/d_{ij}$ o $w_{ij} = 1/d_{ij}^2$ y $w_{ii} = 0$. Otras configuraciones implican considerar un numero k de vecinos más cercanos a cada localidad de tal manera que:

$$\begin{cases} w_{ij}(k) = 0 & i = j, \forall k \\ w_{ij}(k) = 1 & d_{ij} \leq d_i(k) \\ w_{ij}(k) = 0 & d_{ij} > d_i(k) \end{cases} \quad (24.2)$$

donde $d_i(k)$ es la k -esima menor distancia entre las localidades i y j . Utilizando funciones de [Bivand \(2022\)](#) y [Pebesma \(2022\)](#) y los datos de [Vallone and Chasco \(2020\)](#) se calculará la matriz de 5 vecinos más próximos de las áreas urbanas chilenas con más de 2000 habitantes.

```
library("sf")
#Se extraen las coordenadas de las ciudades
coord <- st_coordinates(cities)
# Calcula la vecindad de 5 vecinos más cercanos
w5knn <- knearneigh(coord, k=5, longlat= T) |> knn2nb()
```

El uso de matrices de k vecinos puede forzar la vecindad entre localidades, considerando vecinas a localidades que estén muy distantes entre ellas. Para evitar este problema se puede usar una configuración de vecindad basada en una distancia límite d_{max} , de tal manera que:

$$\begin{cases} w_{ij} = 1 & d_{ij} \leq d_{max} \\ w_{ij} = 0 & d_{ij} > d_{max} \end{cases} \quad (24.3)$$

El problema de este criterio de vecindad es la posibilidad de generar unidades espaciales aisladas cuando d_{max} se fija en un valor muy bajo. Este problema se evita fijando la distancia máxima (d_{max}) de tal manera que se asegure que todas las unidades espaciales tengan al menos un vecino.

```
#Calcula la k=1 matriz W
knn1 <- knearneigh(coord) |> knn2nb()
# Obtiene la distancia critica
distancia_critica <- max(unlist(nbdists(knn1,coord)))
#genera la matriz de vecindad de distancia w_ij < d_max
k1 <- dnearneigh(coord, 0, distancia_critica)
w_dist <- nb2listw(k1)
```

Debe considerarse que la configuración basada en k vecinos y en la distancia censurada dan lugar a matrices binarias, mientras que las matrices \mathbf{W} basadas en distancias, no. Para realizar el cálculo de la matriz \mathbf{W} basado en la distancia inversa ($1/d_{ij}$) se utilizará en mismo conjunto de datos que en las matrices \mathbf{W} basadas en distancias. Dos elementos deben considerarse: utilizar una función decreciente respecto distancia (en este caso una hipérbola) para satisfacer la ley de Tobler (Tobler, 1970b) y segundo, dado que la incidencia que puede tener una localidad j que se encuentre muy lejana a la localidad i tiende a cero, (Tobler, 1970b), la matriz suele censurarse a una distancia máxima d_{max} a partir de la cual la incidencia entre unidades espaciales es nula, el criterio para la fijación de d_{max} es el mismo que el utilizado para evitar la existencia de unidades espaciales aisladas.

```
knn1 <- knearneigh(coord) |> knn2nb()
distancia_critica <- max(unlist(nbdists(knn1,coord)))
k1 <- dnearneigh(coord, 0, distancia_critica)
#Calcula la distancia entre los vecinos
dist_list<- nbdists(k1, st_coordinates(cities))
#Calcula la distancia inversa
i_dist_list <- lapply(dist_list, function(x) 1/x)
#Crea la matriz W
w_dist_i <- nb2listw(k1, glist=i.dist_list, style="W")
```

Se ha indicado anteriormente que la matriz \mathbf{W} se utiliza para capturar los efectos del espacio a partir de medias ponderadas, es decir mediante la matriz \mathbf{W} se puede de construir el *retardo* espacial. El retardo espacial $\mathbf{W}\mathbf{y}$ de la variable \mathbf{y} se obtiene al multiplicar dicha variable por la matriz \mathbf{W} ; por tanto, cada elemento del retardo espacial puede ser interpretado como la media ponderada de las observaciones de la variable \mathbf{y} en el vecindario de cada localidad i .

24.2. Medidas de autocorrelación espacial

Una buena herramienta para entender y comprender las medidas de autocorrelación espacial es el **diagrama de Moran** (Anselin, 1996). El diagrama de Moran relaciona una variable con lo que sucede en su entorno mediante su retardo espacial en una gráfico de puntos. La Fig. 24.3

presenta un diagrama de Moran para la base *clergy* del conjunto de datos Guerry, esta variable muestra el ratio de sacerdotes católicos sobre la población de cada provincia francesa. La linea horizontal discontinua en la Figura 24.3 muestra el promedio del retardo espacial, mientras que la linea vertical discontinua indica el promedio de la variable *clergy*. El dividir el diagrama a partir de dichos promedios permite generar cuatro zonas: el área “HH” que contiene a las localidades cuyo valor de la variable *clergy* es superior al promedio y su vecindario también. Las localidades que se sitúen en el área “LL” presentan valores de la variable *clergy* inferiores al promedio y su entorno también. El área “LH” contiene a las localidades cuyo valor de la variable *clergy* es inferior al promedio, pero su vecindario supera el valor promedio, lo contrario sucede en el área “HL”.

A partir del diagrama es posible observar la situación de una variable respecto a su entorno. Si las localidades se sitúan mayoritariamente en las zonas “HH” y “LL”, las localidades con altos valores (superiores al promedio) de la variable de interés están rodeadas por localidades con altos valores de dicha variable, y las localidades con valores bajos (inferiores al promedio) están rodeadas de localidades con valores bajos, lo cual es una señal de existencia de autocorrelación espacial positiva. Si la concentración tiene lugar en las áreas “HL” y “LH” la autocorrelación espacial negativa.

```
library("spdep")
library("CDR")
w_reina_francia<-poly2nb(guerry, queen=TRUE) |>
  nb2listw()
## Diagrama de Moran
moran.plot(guerry$clergy,w_reina_francia, xlab="Clergy",
            ylab="Retardo espacial de Clergy")
```

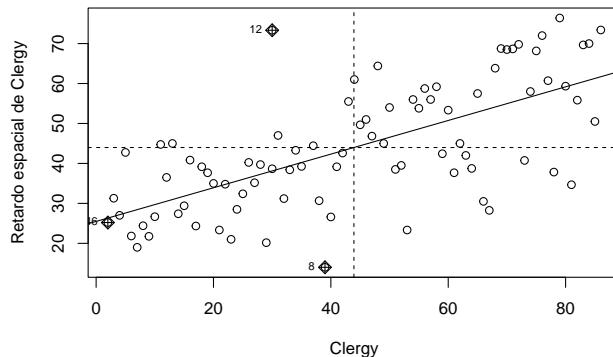


Figura 24.3: Diagrama de Moran de la variable Clergy

La Fig. 24.3 da indicios de la existencia de autocorrelación positiva, es decir que las localidades francesas tienden a estar rodeadas de localidades con numero similares de clérigos. El diagra-

ma de Moran es una herramienta gráfica, para comprobar estadísticamente la existencia de autocorrelación espacial se utilizará el indicador I de Moran.

24.2.1. El indicador I de Moran

Se define $\mathbf{z} = \mathbf{y} - \bar{\mathbf{y}}$, el indicador de Moran se calcula como:

$$I = \frac{n}{\sum_{i=1}^n \sum_{j=1}^n w_{ij}} \frac{\sum_{i=1}^n \sum_{j=1}^n z_i w_{ij} z_j}{\sum_{i=1}^n z_i^2} \quad (24.4)$$

Su campo de variación es $[-1, 1]$ y el signo coincide con el tipo de autocorrelación: valores positivos son indicativos de autocorrelación positiva y valores negativos son indicadores de la existencia de autocorrelación negativa. Nótese que I no es sino el cociente entre la covarianza de la variable \mathbf{y} y su retardo espacial $\mathbf{W}\mathbf{y}$, y la varianza de la variable \mathbf{y} . Por tanto, el coincide con el coeficiente de una regresión lineal de $\mathbf{W}\mathbf{y}$ sobre \mathbf{y} . La linea con pendiente positiva presente en 24.3 es precisamente el resultado de la regresión lineal de $\mathbf{W}\mathbf{y}$ e \mathbf{y} y por tanto su pendiente es el indicador I de Moran. En este sentido, cuanto mayor sea la pendiente de esta recta mayor será el grado de autocorrelación espacial existente. La significatividad estadística del indicador I se establece bajo la hipótesis nula de **aleatoriedad espacial**. La aleatoriedad espacial implica la inexistencia de autocorrelación en la variable analiza; es decir, considera que la variable que se analiza está distribuida en forma aleatoria entre las localidades. En este contexto, p -valores bajos permiten rechazar la hipótesis de aleatoriedad espacial, indicando la existencia de autocorrelación espacial en la variable estudiada (Anselin, 2013).

```
moran.test((guerry$clergy),w_reina_francia,randomisation=TRUE
            ,alternative="two.sided")
#>
#> Moran I test under randomisation
#>
#> data: (guerry$clergy)
#> weights: w_reina_francia
#>
#> Moran I statistic standard deviate = 6.1632, p-value = 7.13e-10
#> alternative hypothesis: two.sided
#> sample estimates:
#> Moran I statistic      Expectation      Variance
#> 0.421118648     -0.011904762     0.004936422
```

El p -valor permite rechazar la hipótesis nula de aleatoriedad espacial a favor de la existencia de autocorrelación positiva.

24.3. Modelos econométricos espaciales de corte transversal

Los modelos espaciales deben ser identificados, antes de proceder a su estimación y contraste. Para ello, es importante disponer de una estrategia de identificación propia, que permita al investigador estimar los parámetros poblacionales a partir de la observación de una muestra de datos.

Tradicionalmente, la econometría espacial ha resuelto este problema asumiendo que la especificación de los modelos es algo que se conoce a priori, bien a partir de la teoría económica existente o bien aplicando ciertas estrategias consistentes en la comparación de varios modelos competitivos. Dentro de esta última opción, se pueden destacar dos estrategias de modelización ampliamente utilizadas: la que va de lo particular (modelo básico sin efectos de autocorrelación espacial) a lo general (modelo con variables explicativas espacialmente retardadas), y la que parte de un modelo general para terminar en un modelo de autocorrelación espacial más sencillo. A partir estos los dos enfoques previos, es posible plantear la estrategia híbrida de [Elhorst \(2010\)](#).

Según se presenta en la Fig 24.4, la estrategia híbrida comienza con la estimación de un **modelo básico sin efectos espaciales**:

$$\mathbf{y} = \alpha \boldsymbol{\iota}_n + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (24.5)$$

siendo \mathbf{y} el vector de la variable dependiente, de orden $(n \times 1)$; \mathbf{X} la matriz de variables explicativas, de orden $(n \times k)$; $\boldsymbol{\iota}_n$ un vector formado por unos, de orden $(n \times 1)$; α , $\boldsymbol{\beta}$ son el conjunto de $(p+1)$ parámetros a estimar; y $\boldsymbol{\epsilon}$ es el vector de perturbaciones aleatorias, de orden $(n \times 1)$, que se distribuye como $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma_\epsilon \mathbf{I}_n)$, siendo \mathbf{I}_n la matriz identidad de orden n .

Este modelo se estima por MCO y luego se llevan a cabo dos contrastes LM del Multiplicador de Lagrange sobre los errores de la regresión para contrastar si son ruido blanco desde el punto de vista espacial. Se trata de dos tests que contrastan una única hipótesis alternativa: el LMLAG, para la hipótesis de variable dependiente espacialmente retardada, y el LMERR, para la hipótesis de dependencia residual. La hipótesis básica se rechaza en cuanto que alguno de los estadísticos de contraste, que se distribuyen como una Chi cuadrado con 1 grado de libertad (χ^2_1) bajo la hipótesis nula, resulte estadísticamente significativo.

En primer lugar, si alguno de los tests LM resulta significativo, se recomienda seleccionar el **modelo Durbin espacial** (SDM), que es un modelo general ([Anselin, 1988](#)):

$$\mathbf{y} = \rho \mathbf{W}\mathbf{y} + \alpha \boldsymbol{\iota}_n + \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon} \quad (24.6)$$

siendo ρ un parámetro y $\boldsymbol{\theta}$ un vector de p parámetros autorregresivos espaciales.

Este modelo general incluye dos tipos de interacción espacial: los efectos endógenos ($\mathbf{W}\mathbf{y}$) y exógenos ($\mathbf{W}\mathbf{X}$). La variable endógena espacialmente retardada ($\mathbf{W}\mathbf{y}$) referida al mismo momento del tiempo que la variable dependiente (\mathbf{y}) produce en los estimadores MCO una situación

24.3. Modelos econométricos espaciales de corte transversal

423

de simultaneidad y, por tanto, sesgo, ineficiencia e inconsistencia. Por eso, se recomienda su estimación por el método de máxima verosimilitud, o “maximum likelihood” en inglés (ML), que supone normalidad en la distribución de los errores (ver [Anselin \(1988\)](#), Cap. 6).

La estimación ML de este modelo permite utilizar la ratio de verosimilitud o “likelihood ratio” (LR), cuya distribución sigue una Chi al cuadrado con k grados de libertad, como estadístico para contrastar las hipótesis nulas $H_0(\boldsymbol{\theta} = 0)$ y $H_0(\rho = 0)$, siendo las hipótesis alternativas las opciones contrarias. En este punto, se pueden dar tres casos:

- 1) Si no se rechaza la primera hipótesis, pero sí la segunda, siempre y cuando los valores de los tests LMLAG > LMERR, el SDM debería simplificarse a un modelo del retardo espacial o modelo autorregresivo espacial de orden 1 (SAR):

$$\mathbf{y} = \rho \mathbf{W}\mathbf{y} + \alpha \boldsymbol{\iota}_n + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (24.7)$$

Este modelo se estima por ML si los errores de la estimación por MCO se distribuyen como una normal.

- 2) Si no se rechaza la segunda hipótesis, pero sí la primera, y los valores de los tests LMERR > LMLAG, debería seleccionarse el **modelo del error espacial** (SEM):

$$\begin{cases} \mathbf{y} = \alpha \boldsymbol{\iota}_n + \mathbf{X}\boldsymbol{\beta} + \mathbf{u} \\ u = \lambda \mathbf{W}\boldsymbol{\epsilon} + \boldsymbol{\epsilon} \end{cases} \quad (24.8)$$

siendo λ un parámetro autorregresivo espacial a estimar. La estimación MCO produciría estimadores insesgados, consistentes, pero ineficientes. Por eso, se considera aceptable estimar el modelo SEM por MCO realizando una inferencia robusta de la matriz de varianzas y covarianzas de los estimadores por el método KP-HET propuesto por [Kelejian and Prucha \(2010\)](#), que tiene en cuenta la existencia conjunta de heteroscedasticidad y autocorrelación espacial en los errores de la regresión.

- 3) Si se rechazan ambas hipótesis nulas o no hubiera acuerdo entre los resultados del test LR y los tests LM, entonces el SDM sería el modelo que mejor describiría los datos.

En segundo lugar, si tras la estimación MCO del modelo básico ninguno de los tests LM fuera estadísticamente significativo, entonces dicho modelo tendría que ser reestimado como un **modelo espacial regresivo transversal** (SLX):

$$\mathbf{y} = \alpha \boldsymbol{\iota}_n + \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon} \quad (24.9)$$

Este modelo puede estimarse por MCO ya que, si las variables explicativas son exógenas, también lo serán sus correspondientes retardos espaciales. Este modelo puede considerar todas las

variables exógenas espacialmente retardadas o un subconjunto de ellas, para contrastar la hipótesis nula $H_0(\boldsymbol{\theta} = 0)$. Si esta hipótesis fuese rechazada debería elegirse el modelo básico como el que mejor describe los datos, es decir, no existiría evidencia alguna de la necesidad de efectos de autocorrelación espacial para explicar la variable dependiente. Pero si, por el contrario, la hipótesis $H_0(\boldsymbol{\theta} = 0)$ fuese rechazada, habría que estimar el modelo SDM con las variables \mathbf{WX} estadísticamente significativas, para contrastar, de nuevo, la hipótesis nula $H_0(\rho = 0)$. Si se rechaza esta hipótesis, el modelo seleccionado sería el SDM, pero en caso contrario, sería el modelo SLX el que mejor describiría los datos.

Todos estos modelos pueden también estimarse con metodología bayesiana utilizando el enfoque Markov Chains Monte Carlo (MCMC), tal y como se explica en [LeSage and Pace \(2009\)](#) Cap. 5.

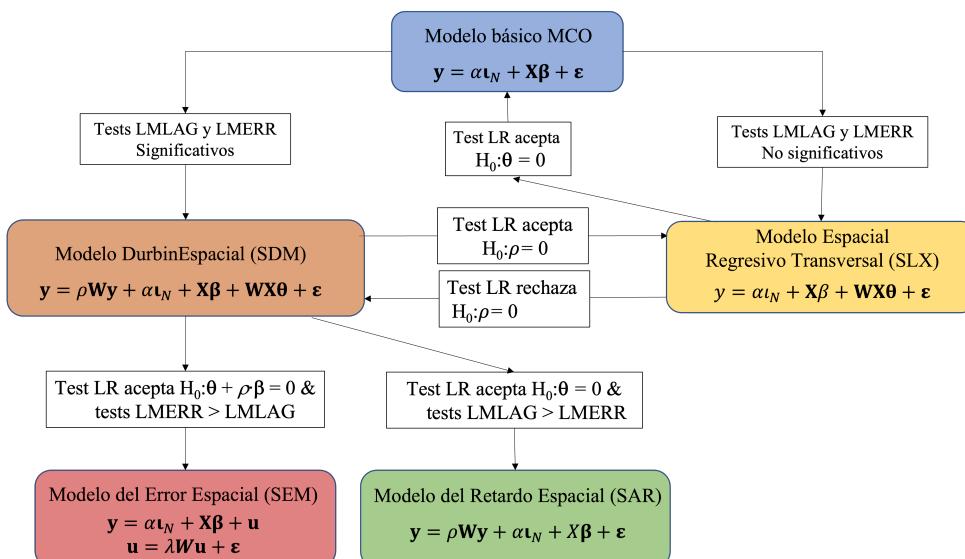


Figura 24.4: Estrategia de especificación híbrida ([Elhorst, 2010](#))

El siguiente conjunto de secuencias de código muestran cómo estimar los modelos que intervienen en la estrategia de modelización de Elhorst. Para ello, se utiliza un conjunto de datos de los 120 municipios grandes de España (capitales de provincia y ciudades con población superior a 50.000 habitantes) que forman parte de las áreas urbanas de España ([Mella and Chasco \(2006\)](#)). Con estos datos, se formula un modelo de crecimiento económico urbano en España, en el que la tasa media de variación del PIB per cápita, en logaritmos, durante el período 1985-2003 (LPGH), se explica en función del PIB per capita en logaritmos de 1985 (LGH85), la tasa de variación del número de entidades bancarias en el período 1985-2003 (BANK), el porcentaje de personas con educación secundaria y universitaria sobre la población de 16 y más años en el año 2001 (UNI01) y la tasa del número de patentes por habitante en el año 2000 (PAT00).

```
library(spatialreg)
library(tseries)
```

24.3. Modelos econométricos espaciales de corte transversal

425

```

# Matriz de pesos espaciales
coord <- coordinates(gdpmap)
k6 <- knearneigh(coord, k=6)
dmins <- knn2nb(k6) |> nb2listw(style="W")
# Estimación modelo básico por MCO
gdp_ols <- lm(LPGH~LGH85+BANK+UNI01+PAT00, data=gdpmap)
summary(gdp_ols)
jarque.bera.test(gdp_ols$res) # Test normalidad de residuos
lm.LMtests(gdp_ols, dmins, test="all") # Grupo de tests LM

# Estimación modelo SDM por ML
gdp_sdm <- lagsarlm(LPGH~LGH85+BANK+UNI01+PAT00, data=gdpmap, listw=dmins,
                     type="mixed")
summary(gdp_sdm)

# Estimación modeelo SAR por ML
gdp_sar <- lagsarlm(LPGH~LGH85+BANK+UNI01+PAT00, data=gdpmap, listw=dmins)
summary(gdp_sar)
LR.Sarlm(gdp_sdm, gdp_sar) # Test LR: SDM vs. SAR

# Estimación del modelo SEM por ML
gdp_err <- errorsarlm(LPGH~LGH85+BANK+UNI01+PAT00, data=gdpmap, listw=dmins,
                       tol.solve=1e-16)
summary(gdp_err)
LR.Sarlm(gdp_sdm, gdp_err) # Test LR: SDM vs. SEM

# Estimación del modelo SLX por MCO

# Cálculo retardos espaciales
gdpmap$WLGH85 <- lag(dmins, gdpmap$LGH85)
gdpmap$WBANK <- lag(dmins, gdpmap$BANK)
gdpmap$WUNI01 <- lag(dmins, gdpmap$UNI01)
gdpmap$WPAT00 <- lag(dmins, gdpmap$PAT00)

gdp_slx <- lm(LPGH~LGH85+BANK+UNI01+PAT00+WLGH85+WBANK+WUNI01+WPAT00, data=gdpmap)
summary(gdp_slx) # Modelo SLX completo

gdp_slx2 <- lm(LPGH~LGH85+BANK+UNI01+PAT00+WLGH85+WPAT00, data=gdpmap)
summary(gdp_slx2) # Modelo SLX parsimonioso
LR.Sarlm(gdp_sdm, gdp_slx2) # Test LR: SDM vs. SLX

```

24.3.1. Estimación SAR

A continuación se muestra la salida de la estimación del modelo SAR, pudiéndose observar que todos los parámetros estimados, incluido ρ , son estadísticamente significativos.

#>

```

#> Call:lagsarlm(formula = LPGH ~ LGH85 + BANK + UNI01 + PAT00, data = gdpmmap,
#>     listw = dmins)
#>
#> Residuals:
#>      Min       1Q   Median      3Q      Max
#> -0.0184657 -0.0034523  0.0012278  0.0032544  0.0194746
#>
#> Type: lag
#> Coefficients: (asymptotic standard errors)
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 2.5745e-01 3.0703e-02 8.3851 < 2.2e-16
#> LGH85       -3.2962e-02 4.4472e-03 -7.4119 1.246e-13
#> BANK        6.6493e-05 1.3086e-05 5.0811 3.753e-07
#> UNI01        4.6884e-04 8.4080e-05 5.5762 2.459e-08
#> PAT00        4.7256e-02 1.8113e-02 2.6090  0.009082
#>
#> Rho: 0.36545, LR test value: 16.353, p-value: 5.2578e-05
#> Asymptotic standard error: 0.078929
#>      z-value: 4.6302, p-value: 3.6538e-06
#> Wald statistic: 21.438, p-value: 3.6538e-06
#>
#> Log likelihood: 447.309 for lag model
#> ML residual variance (sigma squared): 3.7602e-05, (sigma: 0.006132)
#> Number of observations: 122
#> Number of parameters estimated: 7
#> AIC: -880.62, (AIC for lm: -866.27)
#> LM test for residual autocorrelation
#> test value: 6.4996, p-value: 0.01079

```

24.3.2. Comparando SAR con SDM

A continuación se muestra el resultado del test comparando el modelo SDM con el SAR. A la luz de los valores de la LR se rechaza que el valor de los parámetros restringidos sea cero y, por tanto, el modelo SDM es más adecuado para explicar esta variable que el modelo SAR.

```
#>
#> Likelihood ratio for spatial linear models
#>
#> data:
#> Likelihood ratio = 11.559, df = 4, p-value = 0.02095
#> sample estimates:
#> Log likelihood of gdpsdm Log likelihood of gdpsar
#> 453.0885 447.3090
```

24.3.3. Interpretación de los estimadores de los modelos de autocorrelación espacial

Sólo en los modelos de autocorrelación espacial en los que el efecto endógeno ($\mathbf{W}\mathbf{y}$) no está presente en la parte derecha del modelo, los coeficientes estimados ($\hat{\beta}$) pueden interpretarse de forma directa, como en el modelo básico sin efectos espaciales. Es decir, el efecto marginal de un cambio del valor de una variable explicativa continua en la variable explicada coincide con la estimación del coeficiente correspondiente a dicha variable, para todas y cada una de las localizaciones.

$$\frac{\partial y_i}{\partial x_{i,k}} = \beta_k \quad (24.10)$$

En los modelos SAR y SDM, la correcta interpretación de los estimadores implica antes pasar de su forma estructural a su forma reducida. Así, por ejemplo, en el modelo SAR de la expresión (24.7) la forma reducida sería (bajo ciertas condiciones de invertibilidad):

$$\mathbf{y} = (\mathbf{I} - \rho\mathbf{W})^{-1}(\alpha\boldsymbol{\iota}_n + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) \quad (24.11)$$

El término $(\mathbf{I} - \rho\mathbf{W})^{-1}$ se denomina multiplicador espacial y, utilizando la expansión potencial, puede expresarse también del modo siguiente:

$$(\mathbf{I} - \rho\mathbf{W})^{-1} = \mathbf{I} + \rho\mathbf{W} + \rho^2\mathbf{W}^2 + \dots \quad (24.12)$$

Si se utiliza esta nueva expresión en la ecuación (24.11) se observa más claramente que el valor de \mathbf{y} en una determinada localización i es función no sólo del valor de las variables explicativas en esa localización, sino también del valor de las explicativas en las localizaciones vecinas (a través de término $\rho\mathbf{W}\mathbf{X}\boldsymbol{\beta}$), del valor de las explicativas en las localizaciones vecinas a las vecinas (a través del término $\rho^2\mathbf{W}^2\mathbf{X}\boldsymbol{\beta}$), etc., hasta llegar a los límites del sistema espacial en estudio.

$$E(y|\mathbf{X}) = \mathbf{X}\boldsymbol{\beta} + \rho\mathbf{W}\mathbf{X}\boldsymbol{\beta} + \rho^2\mathbf{W}^2\mathbf{X}\boldsymbol{\beta} + \rho^3\mathbf{W}^3\mathbf{X}\boldsymbol{\beta} + \dots \quad (24.13)$$

[LeSage and Pace \(2009\)](#) presentan el cambio (también llamado efecto o impacto) experimentado por \mathbf{y} en una localización i , sea cual sea i , debido a un cambio en el valor \mathbf{x}_k sobre otra localización, j , sea cual sea j . Dicho conjunto de impactos o efectos se presenta en una matriz completa, $\mathbf{S}_k(\mathbf{W})_{ij}$, de orden $n \times n$. Así, cada variable explicativa \mathbf{x}_k del modelo tendrá una matriz completa propia de impactos sobre la variable dependiente.

$$\mathbf{S}_k(\mathbf{W})_{ij} = \frac{\partial y_i}{\partial x'_k} = (\mathbf{I}_n - \rho\mathbf{W})^{-1}\beta_k \quad (24.14)$$

En los modelos SAR y SDM, también podemos distinguir efectos directos e indirectos. El efecto directo sería el efecto causado por cambios en el valor de \mathbf{x}_k , en una localización i sobre el valor de \mathbf{y} en esa misma localización. Estos efectos son los valores de la diagonal principal de la matriz,

$\mathbf{S}_k(\mathbf{W})_{ii}$. El efecto indirecto viene dado por el resto de los valores de la matriz $\mathbf{S}_k(\mathbf{W})_{ij}$, que serían los “bucles de retroalimentación” en los que el valor de \mathbf{x}_k , en una localización j afecta al valor de \mathbf{y} en la localización i , y viceversa, pudiéndose dar recorridos más largos en los que el efecto en una localización podría llegar a la última localización observada n y luego volver de nuevo al punto de partida.

Por ejemplo, $\mathbf{S}_k(\mathbf{W})_{11}$ es el efecto directo de un cambio unitario en el valor de la variable \mathbf{x}_k en la localización 1 (\mathbf{x}_{k1}) sobre el valor de la variable \mathbf{y} en esa misma localización (y_1), mientras que el valor de $\mathbf{S}_k(\mathbf{W})_{12}$ sería el efecto indirecto de un cambio unitario en el valor de la variable \mathbf{x}_k , en la primera localización, sobre el valor de la variable \mathbf{y} en la segunda (y_2). En las filas, la matriz $\mathbf{S}_k(\mathbf{W})_{ij}$ tiene los efectos de un cambio unitario en \mathbf{x}_k en la variable \mathbf{y} desde cada localización i “hacia” todas y cada una de las localizaciones j , mientras que las columnas representan el efecto de un cambio unitario en \mathbf{x}_k en la variable \mathbf{y} , provocado “desde” todas y cada una de las localizaciones i sobre la localización j .

Dado que no es posible contrastar si todos los impactos directos e indirectos contenidos en la matriz $\mathbf{S}_k(\mathbf{W})_{ij}$ son significativamente distintos de cero, o construir intervalos de confianza para ellos, LeSage y Pace proponen llevar a cabo el proceso inferencial sobre el valor medio de los efectos directos y totales, extrayendo los efectos indirectos por diferencia:

$$\bar{M}(k)_{directo} = \text{tr}(\mathbf{S}_k(\mathbf{W}))/n \quad (24.15)$$

$$\bar{M}(k)_{total} = \boldsymbol{\iota}'_{\mathbf{n}} \mathbf{S}_k(\mathbf{W}) \boldsymbol{\iota}_{\mathbf{n}}/n \quad (24.16)$$

$$\bar{M}(k)_{indirecto} = \bar{M}(k)_{total} - \bar{M}(k)_{directo} \quad (24.17)$$

donde \bar{M} indica que se trata de un efecto promedio.

El siguiente conjunto de secuencias presentan el cálculo de las matrices de efectos directos, indirectos y totales, y la inferencia para los modelos SAR y SDM correspondientes al ejemplo del modelo estimado para los municipios urbanos de España.

```
library(coda)
# Cálculo de los efectos para el modelo SAR (LeSage y Pace)
Wsp <- as(as_dgRMatrix_listw(dmins), "CsparseMatrix")
trMat <- trW(Wsp, type="mult")
set.seed(1234) # Simulaciones para el proceso inferencial
gdpsar_impacts <- impacts(gdpsar, tr=trMat, R=1000)
summary(gdpsar_impacts, zstats=TRUE, short=TRUE)
HPDinterval(gdpsar_impacts, choice="direct")
HPDinterval(gdpsar_impacts, choice="indirect")
HPDinterval(gdpsar_impacts, choice="total")
plot(gdpsar_impacts, choice="direct")
plot(gdpsar_impacts, choice="indirect")
plot(gdpsar_impacts, choice="total")
plot(gdpsar_impacts, trace=TRUE, density=FALSE, choice="total")

# Cálculo de la matriz de impactos para la variable LGH85
clear.pr <- rep(NA, dim(gdpmap)[1])
names(clear.pr) <- gdpmap$MUNICIPIO
```

24.3. Modelos econométricos espaciales de corte transversal

429

```

svec <- rep(0,dim(gdpmap)[1])
eye <- matrix(0,nrow=dim(gdpmap)[1],ncol=dim(gdpmap)[1])
diag(eye) <- 1
for(i in 1:length(clear.pr)){
  cvec <- svec
  cvec[i] <- 1
  res <- solve(eye - gdpsar[["rho"]]*Wsp) %*% cvec*gdpsar[["coefficients"]][["LGH85"]]
  clear.pr[i] <- res[i]
}
mult <- solve(eye - gdpsar[["rho"]]*Wsp)
deriv_LGH85 <- solve(eye - gdpsar[["rho"]]*Wsp)*gdpsar[["coefficients"]][["LGH85"]]

# Cálculo de los efectos para el modelo SDM (LeSage y Pace)
set.seed(1234) # Simulaciones para el proceso inferencial
gdpsdm_impacts <- impacts(gdpsdm, tr=trMat, R=1000)
summary(gdpsdm_impacts, zstats=TRUE, short=TRUE)
HPDinterval(gdpsdm_impacts, choice="direct")
HPDinterval(gdpsdm_impacts, choice="indirect")
HPDinterval(gdpsdm_impacts, choice="total")
plot(gdpsdm_impacts, choice="direct")
plot(gdpsdm_impacts, choice="indirect")
plot(gdpsdm_impacts, choice="total")
plot(gdpsdm_impacts, trace=TRUE, density=FALSE, choice="total")

```

24.3.4. Impacto del SDM

A continuación se presenta la salida del cálculo de los efectos para el modelo SDM (LeSage and Pace, 2009) pudiéndose observar que todos son estadísticamente significativos y, salvo en el caso de la variable LGH85, todos ellos son positivos. El signo negativo del coeficiente de LGH85 demuestra la existencia de convergencia en renta en el grupo de grandes ciudades españolas. El impacto total de un crecimiento del 10% del PIB per cápita en una ciudad en el período inicial (1985) supuso una caída de la tasa media de variación del PIB per cápita en el período 1985-2003 del -0,63% en dicha ciudad. Este impacto es la suma del efecto directo causado por el crecimiento del PIB per cápita en la propia ciudad (-0,43), que es el efecto directo, y el efecto indirecto proveniente del crecimiento del PIB per cápita en el resto de ciudades (-0,20). Por su parte, el efecto total del crecimiento de 10 patentes por habitante supuso un crecimiento del PIB per cápita en el período del 4,38%, del cual un 0,5% procedía del crecimiento de las patentes per cápita en la propia ciudad efecto directo y el 3,88% restante fue causado indirectamente por el crecimiento de las patentes en el resto de ciudades.

```

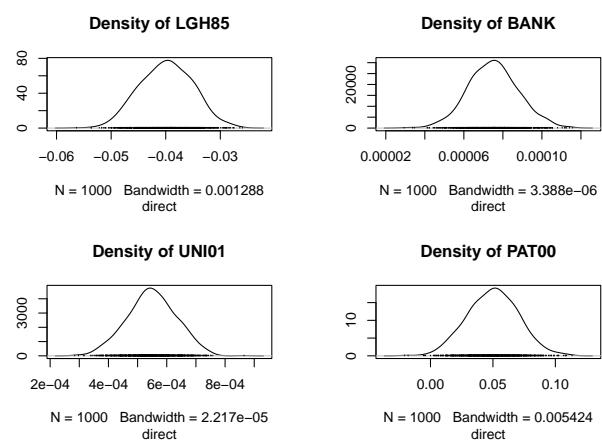
#> Impact measures (mixed, trace):
#>          Direct      Indirect       Total
#> LGH85 -3.967704e-02 -1.656718e-02 -5.624422e-02
#> BANK   7.484574e-05 -3.878932e-05  3.605642e-05
#> UNI01  5.459202e-04  3.900067e-04  9.359269e-04
#> PAT00  5.029859e-02  1.621204e-01  2.124189e-01

```

```

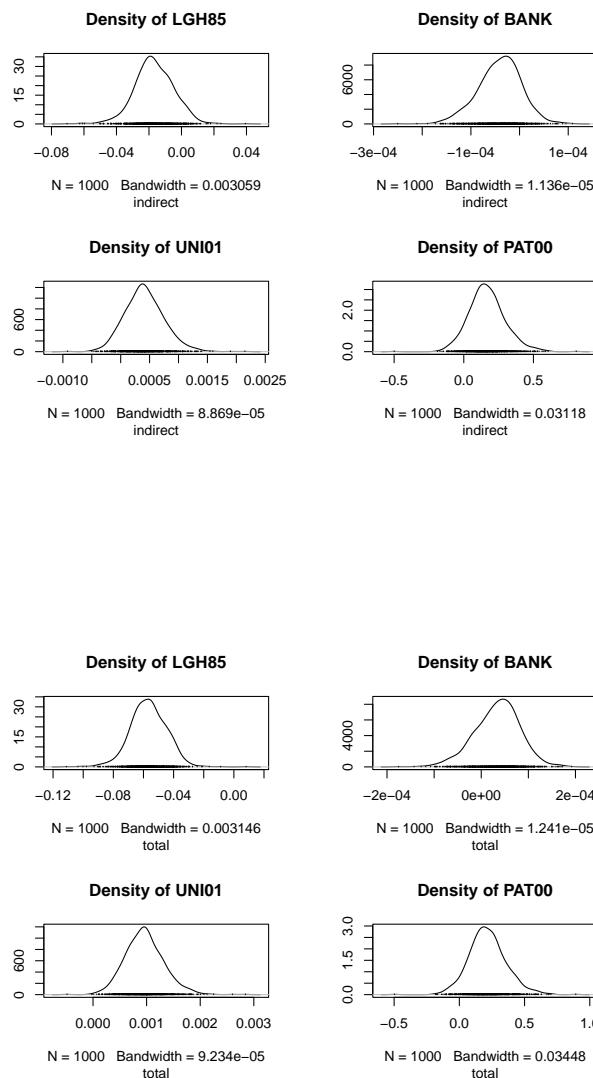
#> =====
#> Simulation results ( variance matrix):
#> =====
#> Simulated standard errors
#>      Direct   Indirect     Total
#> LGH85 4.835913e-03 1.205378e-02 1.208788e-02
#> BANK   1.289395e-05 4.475846e-05 4.830235e-05
#> UNI01  8.434442e-05 3.397464e-04 3.621312e-04
#> PAT00  2.037202e-02 1.317136e-01 1.441993e-01
#>
#> Simulated z-values:
#>      Direct   Indirect     Total
#> LGH85 -8.246556 -1.3907437 -4.6859617
#> BANK    5.826543 -0.9252065  0.6980269
#> UNI01   6.504595  1.2126777  2.6527099
#> PAT00   2.454730  1.2664303  1.5035704
#>
#> Simulated p-values:
#>      Direct   Indirect     Total
#> LGH85 2.2204e-16 0.16430  2.7865e-06
#> BANK   5.6587e-09 0.35486  0.4851604
#> UNI01  7.7903e-11 0.22525  0.0079848
#> PAT00  0.014099  0.20536  0.1326920

```



24.3. Modelos econométricos espaciales de corte transversal

431



Como puede observarse en los gráficos anteriores, para cada variable explicativa se estiman tres estimadores, de forma que el efecto total causado por el cambio unitario en el valor de dicha variable sobre el valor de la variable explicada, en una ciudad determinada, es la suma de dos efectos, uno directo, ocasionado por el cambio acaecido en la propia ciudad, y otro indirecto, proveniente del cambio acaecido en el resto de ciudades de España, existiendo tantos efectos como ciudades.

Resumen

En este capítulo se introduce a la componente espacial en la estimación econométrica y, en particular, el efecto de dependencia espacial inherente en alguna de las variables involucradas en el proceso de modelización. Primero, se observa la heterogeneidad espacial de los datos a partir de los mapas temáticos y se presenta el indicador de autocorrelación espacial de Moran. Posteriormente, se construye la matriz de pesos espaciales bajo distintas especificaciones. Por último, se muestra la taxonomía de los modelos econométricos espaciales, presentando la estrategia de especificación híbrida y la interpretación de los coeficientes estimados.

Capítulo 25

Procesos de puntos

Jorge Mateu^a y Mehdi Moradi^b

^aUniversidad Jaume I ^bUmeå Universitet

25.1. Introducción

La **estadística espacial** es una rama de la estadística que se ha desarrollado rápidamente durante los últimos treinta años, tanto en el plano teórico como en el práctico. A ello ha contribuido, de manera significativa, la creciente disponibilidad de potencia computacional y variedad en software, que han estimulado la capacidad de resolver problemas cada vez más complejos. Lo cierto es que estos problemas tienen como elemento común la estructura espacial. En general, se observa un desarrollo científico que ha ocurrido en el campo de la estadística espacial: problemas bien definidos con un carácter común saltaron a la agenda del investigador, y la disponibilidad de datos motivaron nuevos desarrollos teóricos.

La estadística espacial reconoce y explota las ubicaciones espaciales de los datos al diseñar, recopilar, administrar, analizar y mostrar dichos datos. Los datos espaciales suelen ser dependientes, y se necesitan clases de modelos espaciales que permitan la predicción de procesos y la estimación de parámetros. **Patrones espaciales** ocurren en una variedad sorprendentemente amplia de disciplinas científicas: los ecólogos estudian las interacciones entre plantas y animales, los silvicultores y agricultores deben investigar la capacidad de las plantas y tener en cuenta las variaciones del suelo en sus experimentos. Así pues, cualquier disciplina que trabaje con datos recopilados de diferentes ubicaciones espaciales, necesita desarrollar modelos que indican cuándo hay dependencia entre mediciones en diferentes lugares. Referencias modernas sobre estadística espacial incluyen los libros de [Diggle \(2013\)](#); [Cressie and Wikle \(2015\)](#); [Montero et al. \(2015\)](#); [Wikle et al. \(2019\)](#); [Diggle and Giorgi \(2019\)](#) entre otros.

Este capítulo se centra en **patrones espaciales de puntos**. Datos en forma de un conjunto de puntos, distribuidos irregularmente dentro de una región del espacio, surgen en muchos contextos diferentes; por ejemplo localizaciones de incendios forestales (Fig. 25.1), delitos (Fig.

[25.2](#)), árboles en un bosque, nidos en una colonia de cría de pájaros, ubicación de núcleos en una sección microscópica de tejido, depósitos de oro mapeados en un estudio geológico, estrellas en un cúmulo estelar, accidentes de tráfico, terremotos, llamadas de teléfonos móviles, avistamientos de animales o casos de una enfermedad rara.

Se llama **patrón espacial de puntos**, cualquier conjunto de datos de este tipo. La disposición espacial de los puntos es el principal foco de investigación. Son muchos los campos de la ciencia donde este tipo de estructuras son de interés; por ejemplo, en ecología, epidemiología, geociencia, astronomía, econometría e investigación criminal. El análisis estadístico de la disposición espacial de los puntos puede revelar características importantes, como una tendencia a que los yacimientos de oro se encuentren cerca de una gran falla geológica, o a que los casos de una enfermedad sean más frecuentes cerca de una fuente de contaminación.

El análisis de los datos de patrones de puntos ha proporcionado evidencia fundamental para importantes investigaciones, desde la transmisión del cólera hasta el comportamiento de los asesinos en serie y la estructura a gran escala del universo. Los puntos en un patrón de puntos pueden tener todo tipo de atributos. Un estudio forestal podría registrar cada ubicación, especie y diámetro del árbol; un catálogo de estrellas puede dar sus posiciones en el cielo, masas, formas y colores; las ubicaciones de los casos de enfermedades pueden estar vinculadas a registros clínicos detallados. Esta información auxiliar adjunta a cada punto en el patrón de puntos se llama **marca** y en ese caso se habla de un patrón de puntos marcado. La colección de localizaciones de un patrón puntual puede venir definida en una **región plana** (Sec. [25.2](#)) o bien en una **red lineal** (Sec. [25.3](#)), haciendo que las distancias dejen de ser euclidianas para pasar a ser del camino más corto. Esto introduce ciertos cambios metodológicos en cuanto a las construcciones de ciertas características, que en el caso de intensidades de primer orden trataremos en este capítulo.

25.2. Patrones puntuales espaciales en \mathbb{R}^2

La teoría de procesos puntuales espaciales constituye la base para el análisis de eventos observados geográficamente a través de sus coordenadas (longitud, latitud) en un espacio bidimensional. Esta rama de los procesos puntuales pertenece al campo de la estadística espacial en conjunción con la de procesos estocásticos. De hecho, un proceso puntual espacial es un proceso estocástico cuyas realizaciones consisten en un conjunto numerable de puntos en el plano (patrón puntual). Heurísticamente, se trata de un conjunto de datos que se encuentran en una región concreta (o área de estudio).

Sea $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, $0 \leq n < \infty$, una realización (patrón puntual) observada de un proceso puntual simple (i.e. sin múltiples eventos por localización) y finito X en \mathbb{R}^2 en la región $W \subset \mathbb{R}^2$ y con la métrica (distancia) asociada $d(u, v)$. En general, las realizaciones consisten en un conjunto numerable de puntos (llamados en muchas ocasiones eventos). Consultar las Figs. [25.1](#) y [25.2](#) para ver algunos ejemplos de patrones puntuales. Para cualquier conjunto arbitrario $A \subset \mathbb{R}^2$, el cardinal de X viene dado por la función de conteo

$$N(X \cap A) = \sum_{x \in X} \mathbf{1}\{x \in A\} < \infty.$$

Además, y gracias a la fórmula de Campbell ([Baddeley et al., 2015](#)), para cualquier función medible $f : \mathbb{R}^2 \rightarrow [0, \infty)$ se cumple que

$$\mathbb{E} \left[\sum_{x \in X} f(x) \right] = \int_{\mathbb{R}^2} f(u) \lambda(u) du,$$

donde $\lambda(\cdot)$ determina la **función de intensidad** de X , y gobierna su distribución espacial. De hecho, $\lambda(u)$ proporciona el valor esperado de eventos por unidad de área en un entorno de $u \in \mathbb{R}^2$. Teniendo en cuenta que $f(x) = \mathbf{1}\{x \in A\}$, se puede observar fácilmente la relación entre la función de intensidad $\lambda(\cdot)$ y la de conteo N , establecida como

$$\mathbb{E}[N(X \cap A)] = \int_A \lambda(u) du.$$

Si la función de intensidad $\lambda(\cdot)$ es constante, i.e. $\lambda(\cdot) = \lambda$, se dice que el proceso X es homogéneo, mientras que, en caso contrario, se dice que es inhomogéneo; en este último caso, la distribución espacial varía a lo largo de la región soporte. Para el lector con un mayor interés en conceptos y desarrollos, se aconseja consultar [Møller and Waagepetersen \(2003\)](#); [Illian et al. \(2008\)](#); [Diggle \(2013\)](#) y [Baddeley et al. \(2015\)](#).

En la práctica se suele observar sólo una única realización, y por ello es importante disponer de una estimación de $\lambda(\cdot)$ que pueda imitar la distribución espacial del proceso subyacente, el cual ha generado el patrón observado. Por ello, se consideran diferentes tipos de **estimadores no paramétricos de la intensidad**.

25.2.1. Estimación de la intensidad basada en funciones núcleo

Dos estimadores no paramétricos, basados en **funciones núcleo**, de la función de intensidad ampliamente utilizados en patrones puntuales en \mathbb{R}^2 , vienen dados por

$$\hat{\lambda}_{\sigma}^U(u) = \frac{1}{c_{\sigma,W}(u)} \sum_{i=1}^n \kappa_{\sigma}(u - x_i), \quad u \in W, \tag{25.1}$$

y

$$\hat{\lambda}_{\sigma}^{JD}(u) = \sum_{i=1}^n \frac{\kappa_{\sigma}(u - x_i)}{c_{\sigma,W}(x_i)}, \quad u \in W, \tag{25.2}$$

donde κ_{σ} es una función de densidad de probabilidad en \mathbb{R}^2 con parámetro de suavizado (ancho de banda) σ , y

$$c_{\sigma,W}(u) = \int_W \kappa_{\sigma}(u - v) dv, \quad u \in W, \tag{25.3}$$

es el área del núcleo centrado en $u \in W$, y equivale a un corrector de borde que compensa por la falta de información fuera de W . Hay que recordar que, en la práctica, sólo se observa una realización de X en la región acotada W . Más allá de la elección de σ , el estimador (25.1) es insesgado si la función de intensidad es constante ([Diggle, 1985](#)), mientras que el estimador

(25.2) conserva la masa total (Jones, 1993). Los estimadores (25.1) y (25.2) suelen ser llamados ‘uniformly-edge-corrected’ y ‘Jones-Diggle’ (Rakshit et al., 2019b). En este capítulo, se considera en todo momento la función núcleo Gaussiana (Silverman, 1986).

En términos prácticos, la adecuación de los estimadores basados en núcleos depende del parámetro de suavizado, de forma que un suavizamiento pequeño lleva a un sesgo (por debajo) y varianza alta, mientras que un parámetro de suavizado alto resulta en un sesgo alto y poca varianza. Para un cierto patrón puntual \mathbf{x} , los estimadores (25.1) y (25.2) pueden ser calculados utilizando la función ‘density.ppp’ de ‘spatstat.core’ especificando ‘diggle=FALSE’ y ‘diggle=TRUE’, respectivamente.

25.2.1.1. Selección del parámetro de suavizado

Scott (1992) propuso elegir este parámetro a través de una regla un tanto naive (llamada *rule of thumb*), de la forma

$$(s_x n^{-1/6}, s_y n^{-1/6}),$$

para cada coordenada cartesiana x, y , donde s_x, s_y son las desviaciones típicas de las coordenadas x, y de los eventos. Este procedimiento es útil para análisis exploratorios. La función ‘bw.scott’ de ‘spatstat.explore’ proporciona este estimador. Nótese que, en el caso de Scott, el parámetro de suavizado es, por construcción, un vector de dos componentes para suavizar ambas coordenadas cartesianas.

Cronie and Van Lieshout (2018) propusieron encontrar el parámetro óptimo minimizando

$$CvL(\sigma) = \left(|W| - \sum_{i=1}^n 1/\widehat{\lambda}_\sigma^*(x_i) \right)^2,$$

donde $\widehat{\lambda}_\sigma^*(x_i)$ es un estimador de la intensidad sin corregir (bien sea (25.1) o (25.2) pero sin el término de corrección) evaluado en x_i y con parámetro de suavizado σ . La idea de este estimador proviene de la fórmula de Campbell, ya que

$$\mathbb{E} \left[\sum_{x \in X} 1/\lambda(x) \right] = \int_W (1/\lambda(x)) \lambda(x) du = |W|.$$

Para un patrón puntual \mathbf{x} , la función ‘bw.CvL’ de ‘spatstat.explore’ calcula el parámetro de suavizado mediante el método de Cronie y van Lieshout (se denominará por Cronie–van Lieshout).

25.2.2. Ejemplos prácticos

En esta sección se hace uso de los estimadores de la intensidad anteriormente mostrados y de los diferentes métodos de selección del parámetro de suavizado para analizar la distribución espacial de dos conjuntos de datos: incendios forestales en Nepal (Fig. 25.1), y eventos de crímenes en Medellín, Colombia (Fig. 25.2). En este capítulo, haremos uso de las librerías ‘spatstat, versión 2.3-0,’ (Baddeley and Turner, 2005; Baddeley et al., 2015) para el análisis de patrones puntuales y ‘raster, versión 3.5-15,’ (Hijmans, 2022) para ciertas representaciones gráficas. Nótese

que la librería ‘spatstat’ ha sido recientemente dividida en una familia de sub-librerías ‘spatstat.utils’, ‘spatstat.data’, ‘spatstat.sparse’, ‘spatstat.geom’, ‘spatstat.random’, ‘spatstat.core’, ‘spatstat.linnet’, ‘spatstat.explore’, ‘spatstat.model’, de forma que ‘spatstat’ actúa como una librería paraguas de todas ellas. Los lectores deben estar atentos a posibles futuros cambios en ‘spatstat’ para satisfacer ciertas restricciones de CRAN en relación con los tamaños de sus librerías.

25.2.2.1. Ejemplo 1: Incendios forestales en Nepal

Por cortesía de Ganesh Prasad Sigdel, se dispone de localizaciones georeferenciadas de incendios forestales en Nepal durante 2016, datos cedidos por la institución ICIMOD-Nepal. En 2016, Nepal sufrió 5757 incendios, de los cuales 475 ocurrieron en el distrito de Surkhet, en la provincia de Karnali, en el medio-oeste de Nepal. Se comienza llamando a algunas librerías de **R**, útiles para nuestros propósitos.

```
library("spatstat")
library("raster")
library("CDR")
```

Utilizando los métodos descritos en la Sec. 25.2.1.1, se estima el correspondiente parámetro de suavizado. La regla de Scott proporciona los valores (50253.47 m, 21158.42 m) y el método de validación cruzada de Cronie–van Lieshout estima el valor como 36513.16 m. Mediante el argumento ‘ns’ de ‘bw.CvL’, se puede controlar mejor la búsqueda del parámetro óptimo a través de un grid más fino.

```
data(nepal)
scott_nepal <- bw.scott(nepal) # Scott's rule
CvL_nepal <- bw.CvL(nepal) # Cronie and van Lieshout's criterio
```

Conocido el parámetro de suavizado, se estima la intensidad mediante los estimadores (25.1) y (25.2). La función ‘density.ppp’ proporciona una estimación basada en funciones núcleo para patrones en \mathbb{R}^2 , teniendo en cuenta que, por defecto, esta función hace uso del estimador con corrección uniforme para los bordes (‘uniformly-edge-corrected estimator’) (25.1) con un núcleo Gaussiano. Se fija ‘leaveoneout=FALSE’ para no calcular el estimador *leave-one-out*, mientras que se establece ‘positive=TRUE’ para forzar valores positivos en la densidad. Esto último obedece a que, debido a errores numéricos en el cálculo de la Transformada Rápida de Fourier, se pueden obtener valores negativos en ciertas áreas (ver la ayuda de ‘density.ppp’).

```
d_scott_nepal <- density.ppp(nepal, sigma = scott_nepal, leaveoneout = FALSE, positive
                                = TRUE)
d_cvl_nepal <- density.ppp(nepal, sigma = CvL_nepal, leaveoneout = FALSE, positive =
                                TRUE)
```

Se estima ahora la intensidad mediante el estimador de Jones-Diggle (25.1) escribiendo ‘diggle=TRUE’ en ‘density.ppp’.

```
d_scott_dig_nepal <- density.ppp(nepal, sigma = scott_nepal, leaveoneout = FALSE,
→ positive = TRUE, diggle = TRUE)
d_cvl_dig_nepal <- density.ppp(nepal, sigma = CvL_nepal, leaveoneout = FALSE, positive
→ = TRUE, diggle = TRUE)
```

Tras obtener diferentes estimadores de la intensidad bajo diferentes métodos de selección del parámetro de suavizado, a continuación se muestran estas estimaciones y se comentan sus discrepancias. Para una mejor representación gráfica, se convierten las imágenes de intensidad dadas en la clase ‘im’ a objetos de clase ‘raster’ para luego juntarlas en un ‘RasterStack’. La Fig. 25.1 muestra estas estimaciones, observándose una mayor intensidad en el sur y sur-oeste de Nepal, indicando una clara distribución no uniforme de dicha intensidad, lo que, a su vez, indica un alto grado de inhomogeneidad.

```
sp_int_nepal <- stack(raster(d_scott_nepal), raster(d_cvl_nepal),
→ raster(d_scott_dig_nepal), raster(d_cvl_dig_nepal))
sp_int_nepal <- sp_int_nepal*10^7
names(sp_int_nepal) <- c("scott_gaus_U", "CvL_gaus_U", "scott_gaus_JD", "CvL_gaus_JD")

at <- c(seq(0,1.4,0.2))
pts_nepal <- as.data.frame(nepal)
coordinates(pts_nepal) <- ~ x + y
library("latticeExtra")
spplot(sp_int_nepal, at = at, scales = list(draw = FALSE), col.regions =
→ rev(topo.colors(20)), colorkey = list(labels = list(cex = 3)), par.strip.text =
→ list(cex = 3))+layer(sp.points(pts_nepal,pch=20,col=1))
```

25.2.2.2. Ejemplo 2: Crímenes en Medellín

Medellín es la segunda ciudad con más población en Colombia (DANE, 2019), con un territorio urbano de 105 km², que ha sufrido de múltiples acciones criminales durante muchos años, como es bien conocido. En 2018, la Secretaría de Seguridad de Medellín reportó que el 40% de los ciudadanos se sentía inseguro, proporcionando, a modo de ejemplo, 20607 quejas de robos (Restrepo, 2019). Adicionalmente, el departamento de policía reconocía la necesidad de contratar al menos 2000 policías más para luchar contra los homicidios, robos y micro-tráfico (Monsalve, 2019).

En esta sección, sólo se analiza la distribución espacial de los eventos georeferenciados de crímenes ocurridos en Medellín durante 2005 (Sanabria et al., 2022). En 2005, ocurrieron 910 crímenes, de los cuales el porcentaje de víctimas varones fue del 66%, 28% fueron cometidos durante los fines de semana, el porcentaje de robos fue del 42%, y el de víctimas con edades entre 20 y 40 fue del 60%.

Nótese notar que el conjunto de localizaciones de estos crímenes no necesariamente ocurrió en las calles de la ciudad, y por tanto se considera que el patrón puntual tiene como dominio de definición todo \mathbb{R}^2 .

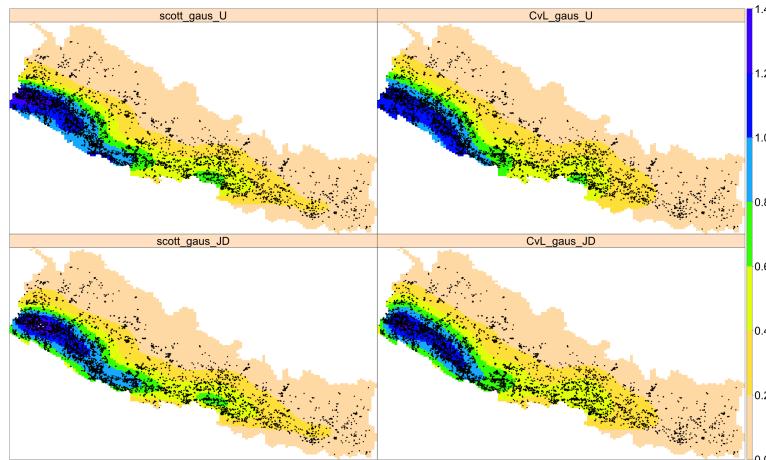


Figura 25.1: Estimación basada en funciones núcleo para los incendios forestales (puntos negros) en Nepal en 2016. Las etiquetas de los nombres comienzan con el método de suavizado, seguido del núcleo utilizado y de la corrección de borde. Los valores de la intensidad indican número de incendios por diez mil km cuadrados. Se usa JD y U para indicar los estimadores de ‘Jones-Diggle’ y ‘uniformly-edge-corrected’.

```
data(medellin)
scott_med <- bw.scott(medellin) # Scott's rule
CvL_med <- bw.CvL(medellin) # Cronie and van Lieshout's criterion
```

La regla de Scott estima el parámetro de suavizado en (691.31m, 954.20m) mientras que el criterio de validación cruzada (CvL) nos lleva a 692.31m. Se hace uso de la función ‘density.ppp’ para obtener los correspondientes estimadores de la intensidad (25.1) y (25.2) bajo los mismos escenarios que en la Sección 25.2.2.1.

```
d_scott_med <- density.ppp(medellin, sigma = scott_med, leaveoneout = FALSE, positive
                           = TRUE)
d_cvl_med <- density.ppp(medellin, sigma = CvL_med, leaveoneout = FALSE, positive =
                           = TRUE)

d_scott_dig_med <- density.ppp(medellin, sigma = scott_med, leaveoneout = FALSE,
                                 positive = TRUE, diggle = TRUE)
d_cvl_dig_med <- density.ppp(medellin, sigma = CvL_med, leaveoneout = FALSE, positive
                               = TRUE, diggle = TRUE)
```

La Fig. 25.2 muestra la intensidad estimada bajo diferentes parámetros de suavizado. Se observa, en general, una distribución no homogénea de los crímenes. Independientemente del método utilizado, se observan dos grandes hotspots en la zona central de Medellín, aunque con diferentes magnitudes. El efecto de la corrección de borde es sólo marginal.

```

sp_int_med <- stack(raster(d_scott_med),raster(d_cvl_med), raster(d_scott_dig_med),
                     raster(d_cvl_dig_med))
sp_int_med <- sp_int_med*10^5
names(sp_int_med) <- names(sp_int_nepal)
at <- seq(0,3,by=0.2)
pts <- as.data.frame(medellin)
coordinates(pts) <- ~ x + y
spplot(all, at = at, scales = list(draw = FALSE), col.regions = rev(topo.colors(20)),
       colorkey = list(labels = list(cex = 3)), par.strip.text = list(cex = 3))+ 
       layer(sp.points(pts,pch=20))

```

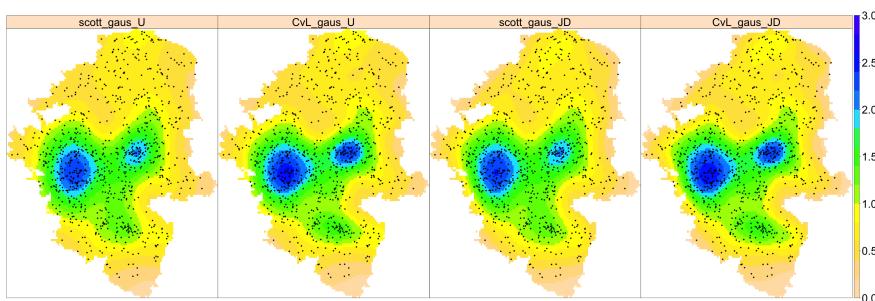


Figura 25.2: Estimación de la intensidad basada en funciones núcleo para los datos de Medellín (puntos negros), durante 2005. Las etiquetas de los nombres comienzan con el método de suavizado, seguido del núcleo utilizado y de la corrección de borde. Los valores de la intensidad indican número de crímenes por cien km cuadrados. Se usa JD y U para indicar los estimadores de ‘Jones-Diggle’ y ‘uniformly-edge-corrected’.

25.2.3. Estimación de la intensidad basada en funciones núcleo en dominios irregulares

Los estimadores (25.1) y (25.2) pueden mostrar deficiencias importantes como no cumplir la condición de que la integral sea el número de puntos, sesgo cerca de las fronteras o presentar suavizamientos artificiales que lleven a resultados inverosímiles en ciertas ocasiones (Baddeley et al., 2022). Estos problemas son más aparentes en caso de dominios irregulares. Como remedio, Baddeley et al. (2022) propusieron estimar la intensidad via una **función núcleo-calor** (*heat kernel*), la cual puede ser definida como una densidad de probabilidad de transición de un movimiento Browniano en W que respeta las fronteras. De hecho, su propuesta, llamada **estimador de difusión**, toma la forma

$$\hat{\lambda}_t(u) = \sum_{i=1}^n \kappa_t(u|x_i), \quad (25.4)$$

donde $t = \sigma^2$ (σ es el parámetro de suavizado en (25.1) y (25.2)) y $\kappa_t(\cdot|x_i)$ es el núcleo-calor. Este estimador es insesgado (bajo homogeneidad) y preserva la masa (es decir, integra el número de

puntos). Baddeley et al. (2022) proponen algunos nuevos métodos de selección del parámetro de suavizado, adaptados a su estimador de difusión, incluyendo el de Cronie–van Lieshout. El estimador de difusión se puede calcular con la función ‘densityHeat.hpp’, y el criterio de Cronie–van Lieshout viene en la función ‘bw.CvLHeat’. Todas estas funciones pertenecen al ‘spatstat.explore’.

A continuación, se utiliza el estimador de difusión para analizar su comportamiento comparándolo, con el estimador (25.1) sobre unos datos de incendios activos en EEUU y América Central (sin considerar las islas) desde el 24 de Febrero al 3 de Marzo 2022¹. Hay que hacer notar que las localizaciones, en este caso, no necesariamente confirman la existencia de un incendio, sino más bien píxeles susceptibles de existencia de incendio, los cuales han sido clasificados por medio de algoritmos preparados para ello. Este formato está relacionado con el contexto de datos en *Near Real-Time (NRT)*.

Los parámetros de suavizado para los estimadores (25.1) y (25.4) siguen el criterio de Cronie–van Lieshout. Nótese que al considerar un área mucho más grande que en los ejemplos precedentes, se considera ‘ns=50’, es decir, se usa un vector de tamaño 50 para buscar el parámetro de suavizado óptimo (por defecto es 16), y ‘dimyx=512’ para obtener imágenes de intensidad con una mejor resolución (por defecto, las imágenes son de tamaño 128×128 píxeles). Los parámetros de suavizado elegidos para calcular (25.1) y (25.4) son 556.3km y 104.9km.

```
data(activefires)
CvL_northcentre <- bw.CvL(activefires, ns = 50)
d_CvL_northcentre <- density.ppp(activefires, sigma = CvL_northcentre, leaveoneout =
  FALSE, dimyx = 512)

heat_CvL_northcentre <- bw.CvLHeat(activefires, ns = 50)
dheat_CvL_northcentre <- densityHeat.hpp(activefires, sigma = heat_CvL_northcentre,
  leaveoneout = FALSE, dimyx = 512)
```

Ambas estimaciones se juntan en un objeto ‘RasterBrick’ que se representa en la Fig. 25.3. Obsérvese que el dominio no es regular pues los estados de Florida, California del Sur y América Central hacen que la región objeto de estudio sea ciertamente irregular. En este caso, sería poco realista si el estimador utilizado proporciona intensidad en zonas como el Golfo de California/Mexico. El mapa de intensidad que se muestra a la izquierda de la Fig. 25.3 muestra que el estimador con corrección uniforme (‘uniformly-edge-corrected’) distribuye la masa total por toda la región, provocando una sobre-suavización. Sin embargo, el mapa de la derecha, construido con el estimador de difusión, muestra una situación más realista, distribuyendo la masa de la intensidad acorde a los sucesos ocurridos.

```
d_northcentre_stack <- stack(raster(d_CvL_northcentre), raster(dheat_CvL_northcentre))
names(d_northcentre_stack) <- c("CvL_gaus_U", "Diffusion")
pts_northcentre <- as.data.frame(activefires)
coordinates(pts_northcentre) <- ~ x + y
d_northcentre_stack <- d_northcentre_stack*10^6
```

¹https://firms.modaps.eosdis.nasa.gov/active_fire/

```
spplot(d_northcentre_stack, scales = list(draw = FALSE), col.regions =
  rev(terrain.colors(20)), colorkey = list(labels = list(cex = 5)), par.strip.text =
  list(cex = 5))+layer(sp.points(pts_northcentre,pch=20,col=1))
```

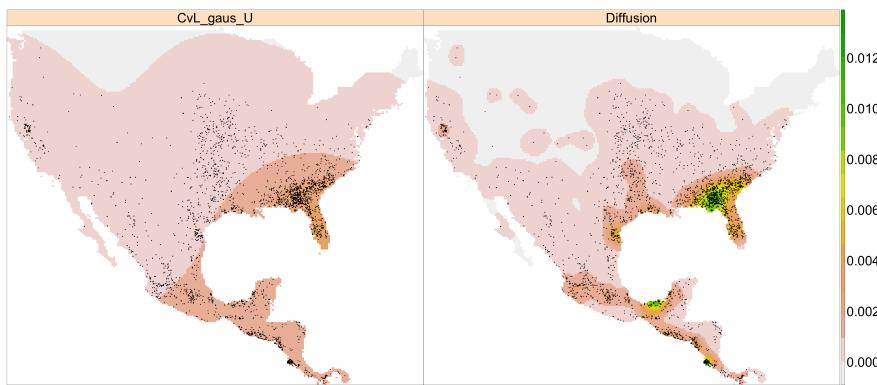


Figura 25.3: Estimación basada en función núcleo para incendios (puntos negros) en EEUU y Centro América (sin las islas) desde el 24 de Febrero hasta el 3 de Marzo de 2022. Izquierdo: estimador con corrección uniforme con núcleo Gaussiano. Derecha: estimador de difusión. El parámetro de suavizado fue obtenido con el criterio de Cronie–van Lieshout. Los valores de la intensidad son fuegos por mil km cuadrados.

25.2.4. Estimadores basados en teselaciones de Voronoi

Como se ha visto, el comportamiento de los estimadores basados en funciones núcleo depende del parámetro de suavizado, e incluso en situaciones en las que hay cambios abruptos en la distribución espacial de los puntos, un único valor constante de este parámetro no puede representar el suavizamiento necesario en toda la región. Para dar una solución a este problema, se propuso un parámetro con variación espacial (adaptable a la estructura espacial) aunque a costa de una mayor complejidad (Davies and Baddeley, 2018; Baddeley et al., 2022). Sin embargo, y como alternativa a esta propuesta, se pueden utilizar estimadores basados en teselaciones de Voronoi, que son no paramétricos (Barr and Schoenberg, 2010).

Para cada $x \in \mathbf{x}$, su celda de Voronoi/Dirichlet \mathcal{V}_x , consistente en todos los $u \in W$ que están más cercanos a x que a cualquier otro elemento $y \in \mathbf{x} \setminus \{x\}$, viene dada por

$$\mathcal{V}_x = \{u \in W : d(x, u) \leq d(y, u) \text{ para todo } y \in \mathbf{x} \setminus \{x\}\}. \quad (25.5)$$

El **estimador basado en teselaciones de Voronoi**, evaluado en cualquier punto arbitrario $u \in W$, es de la forma

$$\hat{\lambda}^V(u) = \sum_{x \in \mathbf{x}} \frac{\mathbf{1}_{\{u \in \mathcal{V}_x\}}}{|\mathcal{V}_x|}. \quad (25.6)$$

El estimador $\widehat{\lambda}^V(u)$ conserva la masa (al igual que $\widehat{\lambda}_\sigma^{\text{JD}}(u)$), y es insesgado si la intensidad real es constante (igual que $\widehat{\lambda}_\sigma^U(u)$), propiedades compartidas por el estimador de difusión. Sin embargo, Moradi et al. (2019) demostraron que $\widehat{\lambda}^V(u)$ tiene una varianza alta, lo que implica una infrasuavización en áreas densas de puntos y una sobre-suavización en áreas con pocos puntos. Por tanto, estos autores proponen corregir el problema de $\widehat{\lambda}^V(u)$ mediante un sub-muestreo de $m \geq 1$ copias re-escaladas \mathbf{x} a través de adelgazamientos independientes (*independent p-thinning*). Su propuesta viene dada por

$$\widehat{\lambda}_{p,m}^V(u) = \frac{1}{m} \sum_{i=1}^m \frac{\widehat{\lambda}_i^V(u)}{p}, \quad u \in W, \quad (25.7)$$

donde $\widehat{\lambda}_i^V(u)$ es el estimador de Voronoi del i -ésimo patrón adelgazado. La idea es que este nuevo estimador balancee mejor la varianza en función de la cantidad de puntos presentes en la subregión, y este efecto se consigue con muestras de menor tamaño procedentes del patrón original. El estimador $\widehat{\lambda}_{p,m}^V(u)$ se conoce como **estimador de remuestreo-suavizado** (*resample-smoothed*), y adicionalmente a las propiedades estadísticas de $\widehat{\lambda}^V(u)$, tiene una varianza bastante más pequeña. En este caso, también se debe seleccionar a priori (m, p) ; sin embargo, Moradi et al. (2019) proponen tanto una *rule-of-thumb* ($m = 400$ y $p \leq 0.2$) como una validación cruzada. Ambos estimadores (25.6) y (25.7) son accesibles por medio de la función ‘densityVoronoi.ppp’ de ‘spatstat.explore’ y en la que los argumentos ‘f’ y ‘nrep’ controlan la probabilidad p y el número de adelgazamientos m . Fijando ‘f=1’ se puede obtener el estimador basado en Voronoi (25.6).

A modo de ejemplo, se estima la intensidad de los incendios en Nepal (Sec. 25.2.2.1) mediante el método de Voronoi resample-smoothed (25.7) considerando diferentes probabilidades de retención para el adelgazamiento correspondiente.

```
d_vor_1_nepal <- densityVoronoi.ppp(nepal, f = 1, nrep = 1)
d_vor_2_nepal <- densityVoronoi.ppp(nepal, f = 0.8, nrep = 400)
d_vor_3_nepal <- densityVoronoi.ppp(nepal, f = 0.6, nrep = 400)
d_vor_4_nepal <- densityVoronoi.ppp(nepal, f = 0.5, nrep = 400)
d_vor_5_nepal <- densityVoronoi.ppp(nepal, f = 0.4, nrep = 400)
d_vor_6_nepal <- densityVoronoi.ppp(nepal, f = 0.2, nrep = 400)
d_vor_7_nepal <- densityVoronoi.ppp(nepal, f = 0.1, nrep = 400)
d_vor_8_nepal <- densityVoronoi.ppp(nepal, f = 0.05, nrep = 400)
```

Las estimaciones obtenidas, al igual que las que proceden de ‘density.ppp’, son de clase ‘im’ y se unen en un objeto ‘RasterBrick’ para su representación gráfica.

```
sp_int_nepal_v <- stack(raster(d_vor_1_nepal), raster(d_vor_2_nepal),
                           raster(d_vor_3_nepal), raster(d_vor_4_nepal), raster(d_vor_5_nepal),
                           raster(d_vor_6_nepal), raster(d_vor_7_nepal), raster(d_vor_8_nepal))
names(sp_int_nepal_v) <- NULL
names <- as.character(sort(c(seq(.2, 1, .2), 0.1, 0.05, 0.5), decreasing = TRUE))
names <- paste("p =", names)
```

```
sp_int_nepal_v <- sp_int_nepal_v*10^7
at <- c(0, 0.3, 0.7, seq(2, 5, 1), 30)

spplot(sp_int_nepal_v, at = at, colorkey = list(labels = list(cex = 3)), col.regions =
  topo.colors(20), scales = list(draw = FALSE), par.strip.text = list(cex = 3),
  names.attr = names)
```

La Fig. 25.4 muestra las intensidades procedentes de los estimadores Voronoi *resample-smoothed* para los incendios de Nepal, y para diferentes probabilidades de retención. Se puede observar un menor suavizamiento y mayor varianza para altas probabilidades. Asimismo, se puede ver que para probabilidades de retención menores que 0.2, el estimador proporciona mejores suavizamientos locales que los basados en suavizamientos fijos.

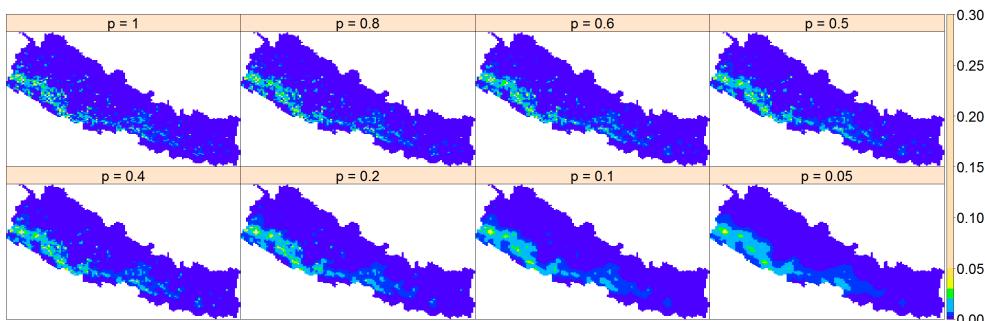


Figura 25.4: Estimaciones de la intensidad de Voronoi resample-smoothed para los incendios en Nepal en 2016 para diferentes probabilidades de retención. La intensidad proporciona el número de incendios por diez mil km cuadrados.

25.2.5. Características de segundo orden: la función K de Ripley

La función de intensidad presentada en las secciones anteriores describe el número esperado de puntos por unidad de espacio, y no tiene en cuenta la estructura de dependencia entre dichos puntos. Esta estructura, sin embargo, viene caracterizada a través de lo que se llaman características de segundo orden. Las funciones de segundo orden determinan la estructura de dependencia espacial (o en su caso espacio-temporal, si interviene el tiempo) inherente al patrón puntual. La literatura ha propuesto varias funciones de segundo orden, de entre las cuales la función K de Ripley es posiblemente la más utilizada. Esta función se define de forma pragmática como el número medio de eventos en un radio r alrededor de cualquier otro evento. Dicho de otra forma, la función $K(r)$ representa el número medio de eventos dentro de un círculo de radio r alrededor de un evento típico del patrón (sin contar dicho evento central). De esta forma, $K(r)$ describe características del proceso de puntos a muchas escalas (tantas como diferentes r se consideren). Esta función puede venir corregida por la intensidad de primer orden en el caso de procesos inhomogéneos. Ambas versiones de la función K vienen implementadas en ‘spatstat’

a través de las funciones ‘Kest’ y ‘Kinhom’ para los casos homogéneo e inhomogéneo. Una propiedad interesante de esta función es que tiene una forma cerrada bajo el caso de aleatoriedad espacial completa, es decir, bajo la situación en la que el patrón de puntos es totalmente aleatorio, sin dependencia espacial alguna (llamado, en este caso, **proceso de Poisson**). Como bajo esta suposición, $K(r) = \pi r^2$ se puede contrastar si un cierto patrón es o no aleatorio, construyendo bandas de confianza sobre la función K evaluada bajo simulaciones de aleatoriedad y evaluando la función K empírica procedente de los datos. La función ‘envelope’ construye tales intervalos de confianza.

También se han utilizado otras funciones para describir y contrastar patrones espaciales; estas funciones están basadas en la distribución de distancias entre puntos que existiría en un patrón de Poisson, como por ejemplo, la función de distribución de distancias al vecino más próximo, la función de distribución de distancias a un punto fijo aleatorio, o la función J , una combinación de las anteriores. Todas estas funciones, incluida la función K , son en cierta forma funciones de distribución ya que, a cada escala o distancia r , todos los pares de puntos separados por una distancia menor que r se usan para estimar el valor de la correspondiente función. En ocasiones, puede ser necesario disponer de una función que caracterice de forma no acumulativa el patrón, es decir, que tenga en cuenta tan sólo los pares de puntos que se encuentran separados por una distancia exactamente igual o similar a la distancia r . La función de correlación de par $g(r)$ (*pair correlation function*) es la herramienta apropiada en este caso (Baddeley et al., 2015).

A continuación se muestra el código y resultados de llevar a la práctica la estimación de la función K (inhomogénea) y los correspondientes intervalos de confianza bajo aleatoriedad para los casos de incendios en Nepal y delitos en Medellín.

```
d_nepal <- density.ppp(nepal,bw.scott,leaveoneout = TRUE)
en_nepal <- envelope(nepal,fun = Kinhom,correction = "border",nsim = 99, simulate =
  expression(rpoispp(d_nepal)), sigma = bw.scott, normpower=2)

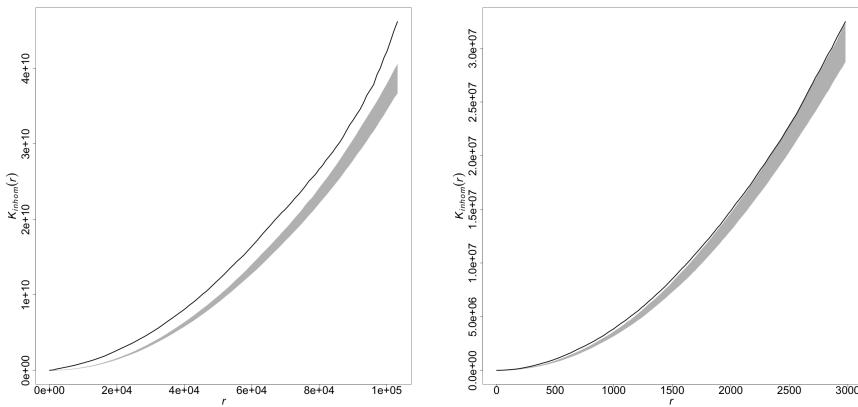
d_med <- density.ppp(medellin,bw.scott,leaveoneout = TRUE)
en_med <- envelope(medellin,fun = Kinhom,correction = "border",nsim = 99, simulate =
  expression(rpoispp(d_med)), sigma = bw.scott, normpower=2)

en_nepal$mmean <- NULL
plot(en_nepal,main="",lwd=3,cex.axis=2.5,cex.lab=2.5,legend=FALSE)

en_med$mmean <- NULL
plot(en_med,main="",lwd=3,cex.axis=2.5,cex.lab=2.5,legend=FALSE)
```

25.3. Patrones puntuales espaciales sobre redes lineales

En los últimos diez años, los patrones de puntos en redes lineales han recibido mucha atención científica. Una red lineal es un conjunto de segmentos (o aristas) unidos por nodos con un formato lineal, tipo una combinación convexa entre dos nodos. La explicación inicial detrás de la consideración de redes lineales, como espacios de estado de algunos procesos puntuales,

Figura 25.5: Funciones K de Ripley para incendios en Nepal y delitos en Medellín

podría estar en el hecho de que los objetos definidos en tales estructuras no pueden usar todo el espacio, y sus movimientos dependen fuertemente de su libertad sobre tales estructuras (Okabe and Sugihara, 2012). En consecuencia, entre otras cosas, la distribución espacial de los puntos, así como la correlación entre ellos, debe estudiarse con respecto a la red subyacente. Sin embargo, no ha sido tan fácil lidiar con este cambio de soporte cuando se pretende adaptar metodologías estadísticas para el análisis de patrones de puntos en redes lineales. Los principales desafíos no fueron solo matemáticos/estadísticos, sino también computacionales (Moradi, 2018; Baddeley et al., 2021).

Una red lineal es una unión de segmentos de línea $l_i = [u_i, v_i] = \{tu_i + (1-t)v_i : 0 \leq t \leq 1\} \subset \mathbb{R}^2$, y una elección común de métrica sobre dicha estructura ha sido inicialmente la distancia de ruta más corta (*shortest-path distance*) $d_L(u, v)$, aunque más tarde Rakshit et al. (2017) propusieron otros tipos de distancias, incluida la distancia euclídea. La idea es que moverse por la red de segmentos implica respetar la geometría de dicha red y por tanto las líneas rectas (que sería el caso de usar distancias euclídeas) no son adecuadas. La distancia de ruta más corta si que permite adaptarse a esta geometría. Sea Y un proceso puntual en una red lineal L , la fórmula de Campbell (25.2) se adapta como

$$\mathbb{E} \left[\sum_{y \in Y} f(y) \right] = \int_L f(z) \lambda(z) d_1 z,$$

donde d_1 denota integración con respecto a la longitud de arco. En este caso, $\lambda(z)$ proporciona el número esperado de puntos por unidad de longitud de L en una vecindad de $z \in L$. Se han desarrollado distintos estimadores de la intensidad para patrones en redes considerando métricas adecuadas y resolviendo ciertos obstáculos matemáticos. El lector puede leer más al respecto en Moradi (2018) y Baddeley et al. (2021); en particular, se recomienda leer sobre el método de estimación no paramétrica basado en convoluciones bi-dimensionales de (Rakshit et al., 2019b). Dada una realización $y = \{y_1, y_2, \dots, y_n\}$ de un proceso puntual Y sobre una

25.3. Patrones puntuales espaciales sobre redes lineales

447

red lineal L , estos autores propusieron

$$\hat{\lambda}_\sigma^U(z) = \frac{1}{c_{\sigma,L}(z)} \sum_{i=1}^n \kappa_\sigma(z - y_i), \quad z \in L, \quad (25.8)$$

con una corrección uniforme, y

$$\hat{\lambda}_\sigma^{JD}(z) = \sum_{i=1}^n \frac{\kappa_\sigma(z - y_i)}{c_{\sigma,L}(y_i)}, \quad z \in L, \quad (25.9)$$

con la corrección de Jones-Diggle, donde κ_σ es una función núcleo bivariante con suavizado σ , y

$$c_{\sigma,L}(z) = \int_L \kappa_\sigma(z - v) d_1 v,$$

es una corrección de borde.

Los dos estimadores anteriores tienen propiedades estadísticas similares a las de sus análogos para patrones de puntos espaciales en \mathbb{R}^2 (es decir, los estimadores (25.1) y (25.2)), y se pueden calcular rápidamente incluso en redes grandes y para grandes anchos de banda (parámetros de suavización). El cálculo rápido se logra mediante la transformada rápida de Fourier (FFT) (Silverman, 1982). Además, Rakshit et al. (2019b) propusieron utilizar las versiones adaptadas de la regla de Scott, a la cual se puede acceder a través de la funciones ‘bw.scott.iso’ de ‘spatstat.linnet’, para obtener un ancho de banda óptimo. Nótese que el cálculo rápido de los estimadores anteriores simplifica aún más el cálculo de los estimadores de intensidad basados en el núcleo adaptativo y el riesgo relativo sobre las estructuras de red (Rakshit et al., 2019b).

También se recuerda que Moradi et al. (2019) propusieron su enfoque de sub-muestreo basado en Voronoi para procesos de puntos generales; para patrones de puntos en redes lineales puede calcularse mediante la función ‘densityVoronoi.lpp’ de ‘spatstat.linnet’.

Como ejemplo práctico para esta sección, se estudia la distribución espacial de delitos callejeros en Valencia. Valencia es la tercera ciudad más grande de España, siendo la capital de la Comunidad Valenciana. El territorio urbano de Valencia encierra un área de 134,65 km², con más de 800000 habitantes en el municipio. El conjunto de datos consta de las ubicaciones de 90247 delitos callejeros como agresión (55610 casos), robo (25342 casos), robo contra la mujer con violencia (454 casos) y otros tipos de delitos (8841 casos). Estos delitos se cometieron entre 2010 y 2020. Sin embargo, en lo que sigue, el análisis únicamente se centra en los datos correspondientes al año 2020, que incluye 6868 casos, de los cuales 4077 son agresiones, 2060 son robos y 66 se relacionan con delitos contra la mujer con violencia. Este conjunto de datos es propiedad de la Generalitat Valenciana (GV), se obtuvieron a través del teléfono de emergencias 112, y se puso a disposición de los autores gracias a un convenio entre GV y la Universidad Jaume I.

A continuación, estimar el parámetro de suavizado utilizando la regla general de Scott, que da 584,1m. La función ‘densityQuick.lpp’ de ‘spatstat.linnet’ se usa para calcular cualquiera de los estimadores (25.8) y (25.9) en los que su valor predeterminado calcula el estimador de borde uniforme corregido (25.8).

```
data(valencia)
scott_valencia <- bw.scott.iso(valencia) # Scott rule
d_scott_valencia <- densityQuick.lpp(valencia, sigma = scott_valencia, leaveoneout =
  FALSE, positive = TRUE, dimyx = 512)
d_scott_valencia <- d_scott_valencia*1000
```

Las imágenes obtenidas son de tipo ‘linim’, y se convierten en objetos de clase ‘im’ antes de pasarlas a objetos ‘raster’.

```
par(mfrow=c(1,2))
plot(valencia$domain>window, lwd = 4)
plot(valencia, pch = 20, main = "", lwd = 4, cex = 1, add = T, cols = "red", col =
  "blue")
plot(raster(as.im(d_scott_valencia)), main = "", axis.args = list(cex.axis = 4),
  legend.width = 2, zlim = c(0,6))
plot(valencia$domain>window, add = TRUE, lwd = 4)
par(mfrow = c(1,1))
```

La Fig. 25.6 muestra la intensidad estimada junto con los eventos de delitos. Dicha intensidad identifica las zonas central y norte de la ciudad de Valencia como áreas de alto riesgo junto con otras zonas de bajo riesgo como son el este y la costa de la ciudad.

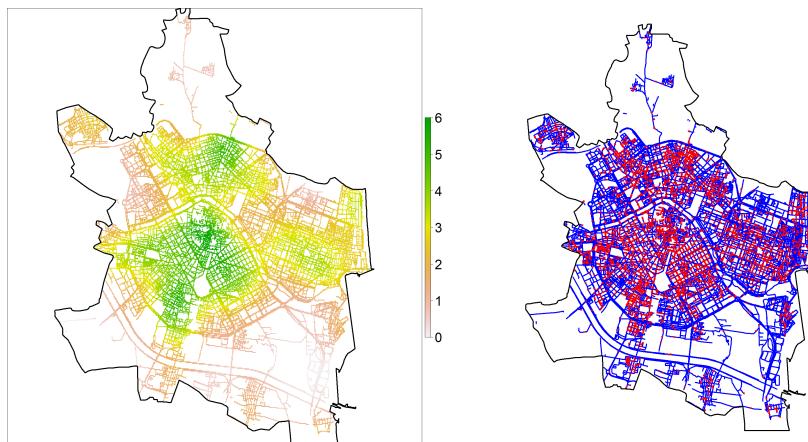


Figura 25.6: Intensidad estimada por función núcleo, usando un estimador borde uniforme corregido (izquierda), para los datos de delitos (puntos rojos) en Valencia durante 2020 (derecha). Los valores de intensidad muestran número de crímenes por km lineal.

```
d_vlc <- densityQuick.lpp(valencia, sigma = scott_valencia, leaveoneout = TRUE,
  positive=TRUE, at = "points", dimyx=512)
d_vlc_im <- densityQuick.lpp(valencia, sigma = scott_valencia, leaveoneout = TRUE,
  positive=TRUE,dimyx=512)
```

25.3. Patrones puntuales espaciales sobre redes lineales

449

Finalmente, se muestra la función K de Ripley y el intervalo de confianza bajo un proceso de Poisson en una red lineal (ver Fig. 25.7 (Ang et al., 2012; Rakshit et al., 2019a)). Se observa que la función K empírica cae dentro de las bandas indicando que el tipo de delitos considerados, en 2020, es compatible con un proceso aleatorio. Obsérvese que al no considerar el tiempo, podemos detectar clusters espaciales que no existen en realidad pues estos desaparecerían con la evolución temporal.

```
sim_vlc <- rpoislpp(lambda = d_vlc_im, L=net_vlc, nsim = 199)
library(spatstat.Knet)
K_vlc <- Knetinhom(valencia, lambda = as.numeric(d_vlc))
r <- K_vlc$r

K_sim <- lapply(X=1:199, function(i){
  sigma <- bw.scott.iso(sim_vlc[[i]])
  lambda <- densityQuick.lpp(sim_vlc[[i]], sigma = sigma, leaveoneout = TRUE,
  ↪ positive=TRUE, at = "points", dimyx=512)
  Ksim <- Knetinhom(sim_vlc[[i]], lambda = as.numeric(lambda), r=r)
  return(Ksim)
})
```

```
K_nsim_df <- as.data.frame(do.call(cbind,d_nsim))
K_nsim_df_est <- K_nsim_df[,seq(3,399,by=2)]

maxn <- function(n) function(x) order(x, decreasing = TRUE)[n]
minn <- function(n) function(x) order(x, decreasing = FALSE)[n]

Kmin <- apply(K_nsim_df_est, 1, function(x)x[minn(5)(x)])
Kmax <- apply(K_nsim_df_est, 1, function(x)x[maxn(5)(x)])
```

```
plot(r, Kmin, type="n", col="grey", ylim=c(0,270),
  ↪ ylab=expression(italic(K[inhom])))
points(r, Kmax, type="n", col="grey")
polygon(c(r, rev(r)), c(Kmax, rev(Kmin)), col = "grey", border = "grey")
points(r, K_vlc$est, type="l")
```

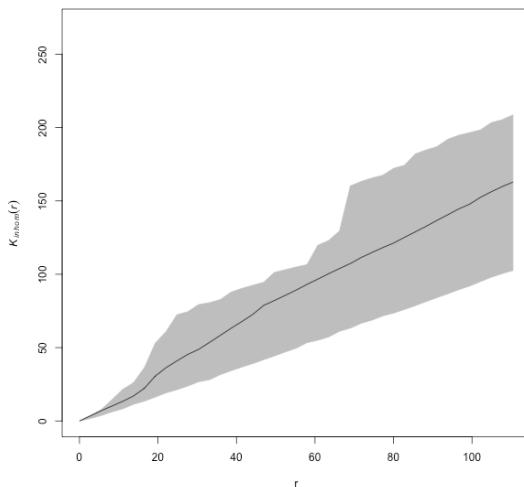


Figura 25.7: Función K para los delitos en Valencia, junto con la envoltura bajo un proceso de Poisson.

RESUMEN. La teoría de procesos puntuales espaciales constituye la base para el análisis de eventos observados geográficamente a través de sus coordenadas (longitud, latitud) en un espacio bi-dimensional. Esta es una de las ramas del campo de la estadística espacial en conjunción con la de procesos estocásticos. De hecho, un proceso puntual espacial es un proceso estocástico cuyas realizaciones consisten en un conjunto numerable de puntos (llamados en muchas ocasiones eventos). Heurísticamente, se trata de un conjunto de datos que se encuentran en una región concreta (área de estudio). Los puntos pueden representar cualquier población espacialmente explícita, como localizaciones de animales, nidos de aves, epicentros de terremotos, galaxias, crímenes, etc. El modelo estadístico más conocido para el análisis de patrones puntuales espaciales es el proceso puntual espacial de Poisson (asociado a la condición de aleatoriedad espacial completa). A partir del modelo de Poisson se construyen modelos más complejos. La modelización pasa por determinar las intensidades de primer y segundo orden que caracterizarán las propiedades básicas del comportamiento de los puntos. En este capítulo se proponen varios estimadores de la función de intensidad de primer orden junto con sus elementos asociados relacionados con funciones núcleo, parámetro de suavizado y correcciones de borde. Se consideran también algunos aspectos de medidas de segundo orden. El capítulo finaliza con aspectos sobre el cambio de soporte del plano euclídeo a redes lineales.

Parte VI

Comunica y colabora

Capítulo 26

Informes reproducibles con R Markdown y Quarto

Emilio L. Cano

Universidad Rey Juan Carlos

26.1. ¿Por qué informes reproducibles?

El resultado final de un proyecto de análisis de datos terminará comunicándose a distintos niveles, tanto *aguas arriba* como *aguas abajo*. Esta comunicación es “la última milla” del flujo de análisis que se esquematizaba en la Fig. ???. Se llama genéricamente “informe” a cualquiera de estos resultados que se pueden producir en distintos formatos de destino. Estos informes estarán compuestos de múltiples elementos como texto, gráficos, resultados numéricos, tablas, etc. Además, es posible que haya que generarlos en distintos formatos (por ejemplo html o pdf, entre otros), para diferentes destinos, como la Web, documentos imprimibles o presentaciones. Finalmente, con una alta probabilidad varias personas intervendrán en el proceso, y la **trazabilidad** (reproducibilidad) del análisis mejorará el proyecto de forma global.

La forma de abordar el problema es típicamente con un enfoque *corta-pegá*, en el que primero se realiza todo el análisis de datos con el software estadístico y después se utilizan los resultados del análisis como base de un informe escrito, posiblemente con algunas iteraciones del proceso si el proyecto tiene cierta envergadura. El software estadístico comercial suele incluir formas de generar resultados listos para integrar en un informe, pero habitualmente bajo este paradigma de incluir el resultado a posteriori ([Leisch, 2002](#)).

Esta forma de trabajar genera un alto coste de mantenimiento (debido a la regeneración manual del informe) a la vez que provoca inconsistencias (por ejemplo entre unos grupos y otros, entre diferentes analistas, etc.), errores, contenidos desactualizados o no reproducibles. Este enfoque es propenso a fallos de organización y gestión de un proyecto, lo se traduce en vulnerabilidades

especialmente en la ejecución de software, simulaciones, etc. Además, cada vez que hay que hacer un cambio, hay que hacerlo en muchos sitios, con la consiguiente pérdida de tiempo y posibles errores.

El enfoque de la **investigación reproducible**¹ supera muchos de los obstáculos a la hora de preparar informes de análisis de datos. El objetivo es unir instrucciones para análisis de datos con datos experimentales de forma que los resultados se puedan volver a obtener automáticamente, entender mejor y verificar.

Un concepto muy relacionado que se utiliza en **R** es la **programación literaria**², mediante la cual se combina un lenguaje de programación como **R** con documentación de todo tipo (por ejemplo comentarios en el código fuente o inclusión de ficheros *readme*).

Con el enfoque de la investigación reproducible, lo que se hace es darle la vuelta al enfoque *corta-pega*, de forma que se escribe el informe a la vez que se realiza el análisis, incrustando el código dentro del propio informe. Obviamente, es necesario un sistema que consolide el informe con los resultados del código, y esto es lo que nos permite **R** y **RStudio** mediante archivos **R Markdown** y su evolución reciente a **Quarto**. El desarrollo de **Quarto** está patrocinado por la empresa Posit, PBC, donde anteriormente crearon **R Markdown** que compartía los mismos objetivos, pero estaba dirigido principalmente a usuarios del lenguaje **R**. El mismo equipo central trabaja tanto en **R Markdown** como en **Quarto**³.

Las ventajas de utilizar un enfoque reproducible se pueden resumir en:

- Si el mismo analista tiene que volver al análisis en el futuro, los resultados se pueden volver a obtener automáticamente de nuevo fácil y comprensiblemente.
- En el caso de que en el proyecto participen más analistas, toda la explicación está a mano.
- Cualquier cambio en un punto del análisis (por ejemplo, añadir una variable a un modelo) se puede realizar de una sola vez y los cambios en los resultados y gráficos se actualizarán automáticamente.
- Los resultados se pueden verificar por terceros en caso necesario. Un caso paradigmático fue el escándalo de los ensayos de cáncer en Duke en 2011⁴. No obstante es un tema que cada vez se demanda más en otros campos fuera de la investigación clínica (por ejemplo en publicaciones de cualquier tipo).

El flujo de trabajo sería el siguiente: los contenidos se encuentran en ficheros de texto plano, con código y texto explicativo. Estos ficheros fuente, se compilán y producen los materiales en los formatos necesarios. Los cambios se hacen una vez, y todos los materiales son actualizados adecuadamente.

¹El objetivo de la investigación reproducible es vincular instrucciones específicas a los análisis y datos experimentales, de modo que los informes puedan recrearse entendidos mejor y verificados. (Kuhn, 2019).

²La programación literaria es una metodología que combina un lenguaje de programación con un lenguaje de documentación (Knuth, 1984).

³<https://quarto.org/about.html>

⁴<http://www.nytimes.com/2011/07/08/health/research/08genes.html>

A continuación se abordará el enfoque reproducible. Para el otro enfoque, simplemente basta copiar los resultados de la consola y los gráficos de la pestaña *Plots* del panel inferior derecho en cualquier editor de documentos.

26.1.1. Markdown, R Markdown, Quarto y RStudio

Markdown es un lenguaje de marcas ligero que fue creado con la intención de ser más legible y fácil de escribir que el código HTML, aunque actualmente se utiliza para otros formatos de salida. Al ser ficheros de texto plano los ficheros se pueden leer bajo cualquier circunstancia, con una sintaxis muy sencilla que permite leerlo directamente por las personas, o ser convertido por un ordenador en otro formato más elaborado, como por ejemplo HTML (página web), pdf o Microsoft Word. En **RStudio**, se pueden crear ficheros **R Markdown**⁵ utilizando esta sintaxis para las explicaciones del análisis, e incluir dentro “trozos” (*chunks*) de código, de forma que, al generar el informe, el resultado de ese código queda incluido en el documento de salida. Así, si una vez terminado el informe se ha olvidado, por ejemplo, incluir un gráfico, sólo hay que añadir las líneas de código que lo crean y volver a generar el informe.

R Markdown ha evolucionado a un nuevo formato denominado **Quarto**⁶, que extiende aún más la funcionalidad de Markdown y está pensado para ser usado con otros lenguajes de programación. En esencia, y a los efectos de este capítulo, hay pocas diferencias.

Para poder utilizar las capacidades de **R Markdown** y **Quarto**, es necesario tener instalado el paquete **knitr** (Xie, 2017), que utiliza también otros paquetes como **rmarkdown**. Aunque **knitr** no forma parte del **tidyverse**, sí es un enfoque moderno de **R** que vino a hacer más fácil la generación de documentos que se hacía en **R** base con la función **Sweave** (Leisch, 2002). Para usar **Quarto** se necesita también el paquete **quarto** (Allaire, 2022) y tener instalado el software **quarto** en el ordenador⁷.

Para crear un nuevo documento **Quarto**, se selecciona *Quarto Document...* en el ícono de nuevo archivo de la barra de herramientas o en el menú *File*. Entonces se abre el cuadro de diálogo *New Quarto Document*. Hay varios tipos de archivos que se pueden crear, que producirán formatos diferentes: *Document* (documento), *Presentation* (presentación de diapositivas) e *Interactive* (aplicación web interactiva con Shiny u Observable JS). De momento se verán los documentos. Se puede crear un archivo **Quarto** vacío si no se quiere crear la estructura. Para que se cree con una estructura mínima, se necesita un título del documento y un autor, que después se podrán cambiar. También se selecciona un formato de salida por defecto, que puede ser HTML (para ver en el navegador), PDF, o Word. Esto también se podrá cambiar después, por lo que la forma más eficiente de trabajar es empezar con HTML, cuya previsualización es más rápida, y cuando esté el resultado final generar el archivo en el formato deseado.

Se puede seleccionar la *Engine* entre **knitr** (**R**) y **Jupyter** (**Python**), y también elegir si se quiere utilizar el editor visual (por defecto). Con el editor visual se pueden utilizar menús para editar el texto y dar el formato Markdown sin esfuerzo. Al hacer clic en el botón *Create* de la ventana *New Quarto Document*, se abre en el editor de **RStudio** un documento quarto (extensión .qmd)

⁵<https://rmarkdown.rstudio.com>

⁶<https://quarto.org>

⁷La instalación es trivial para cualquier sistema desde la web de quarto, <https://quarto.org>.

con una estructura básica a modo de plantilla. Los elementos principales de un archivo **Quarto** aparecen en esta plantilla:

- **Encabezado YAML:** constituyen la configuración del documento, y controlan sobre todo las opciones del formato de salida, es decir, cómo se verá el resultado final. Este encabezado se encuentra entre dos líneas con tres guiones (---), donde se expresan las opciones como `opcion: valor`, y estos valores además se pueden anidar. Dispone de ayuda contextual, de forma que pulsando la combinación de teclas CTRL+ESPACIO aparecen las opciones que se pueden configurar y los posibles valores. Esta parte del documento es constituye el **formato** del documento.
- **Texto formateado:** con una sintaxis muy sencilla, se puede dar formato al texto, como negritas, listas, etc. En el editor visual se puede hacer con los menús y botones de la barra de herramientas del editor.
- **Fragments de código (*chunks*):** al generar el documento, se ejecutará el código dentro de estos fragmentos, y en el documento resultante se mostrará el resultado. En cada fragmento de código aparecen dos botones que sirven para ejecutar todos los *chunks* anteriores y para ejecutar el *chunk* actual. Junto con el texto formateado constituyen el **contenido** del documento.

La barra de herramientas del editor ofrece algunas opciones:

- El botón *Render* convierte (“renderiza” en lenguaje informático) el documento **Quarto** produciendo el archivo de salida configurado. Se puede desplegar un menú para cambiar el formato de salida y otras opciones. Al crear el documento solo aparece el formato de salida elegido, pero se puede cambiar el encabezado para poder convertir el documento a distintos formatos. Por ejemplo si se cambia el encabezado que se ha creado por defecto por el siguiente, se puede generar el archivo de salida en html o word seleccionando en la lista desplegable junto al botón *Render*:

```
---
title: "Título del informe"
format:
  html: default
  docx: default
  editor: visual
---
```

- El botón de opciones permite cambiar la forma en que se mostrarán las salidas al ejecutar el código desde el editor.
- El botón *Insert a new code chunk* nos permite insertar un nuevo fragmento de código.

- Las flechas de navegación permiten moverse entre los *chunks* del documento. También se puede usar el selector de esquema en la parte inferior para ir a un fragmento de código o apartado concreto del documento.
- Desde el menú *Run* se puede ejecutar el código de los distintos *chunks*.
- El menú *Publish* nos permitiría publicar el documento en algún servicio como [RPubs](#)
- El botón *Outline* muestra un esquema para navegar por el documento, donde aparecerán los encabezados formateados con Markdown.
- Se puede cambiar entre el editor visual y el del código fuente con los botones *Source* y *Visual* en la parte superior del editor.

Para generar el documento, se guarda el archivo en cualquier carpeta de nuestro proyecto y se utiliza el ícono de conversión (“renderizado”). La Fig. 26.1 muestra en el panel izquierdo el archivo fuente en el editor visual, con alguna opción adicional añadida a la plantilla por defecto, y en el panel derecho el informe renderizado. Si en vez de pulsarlo directamente se selecciona el triángulo de la derecha, se puede seleccionar el formato de salida (html, pdf o Word) si se han incluido esos formatos en el encabezado YAML como se ha indicado. El formato PDF requiere tener instalada una distribución del sistema de edición libre LaTeX⁸. El archivo de destino, con extensión .html, .pdf o .docx según el caso, quedará guardado en la carpeta donde se encuentre el archivo quarto. Dependiendo de las opciones configuradas, el archivo se abrirá automáticamente en una ventana nueva de **RStudio** (por defecto), o en la pestaña *Viewer* del panel inferior derecho, o en el visor de pdf integrado en **RStudio** (pdf). Para poder abrir archivos .docx será necesario tener instalado **Microsoft Word** o algún otro programa que pueda abrirlo, como **LibreOffice**.

Se puede compilar el informe tantas veces como se quiera con el ícono *Render*. Para trabajo en curso, se recomienda ir previsualizando en formato HTML, y una vez sea definitivo generar el formato de destino final. Para la conversión de formatos, **RStudio** integra la aplicación **pandoc**.

Hay una guía rápida de Markdown (*Markdown Quick Reference*) disponible en el menú de ayuda de **RStudio**, así como enlaces a dos *Cheatsheets*: *R Markdown Cheatsheet* y *R Markdown Reference Guide*. Esta última es la más completa y donde se encuentran todas las opciones disponibles (que sirven para los documentos **Quarto** aunque en la propia web de **Quarto** hay una documentación más completa). En los siguientes apartados se revisan las opciones más habituales que cubren un amplio abanico de proyectos.

26.2. Documentos Quarto

En esta sección se detalla cómo añadir contenido y configuración a un documento Quarto con algunas de las opciones más interesantes para la Ciencia de Datos reproducible.

⁸Se puede instalar una distribución ligera de LaTeX con el paquete `tinytex` ejecutando `tinytex::install_tinytex()` habiendo instalado previamente dicho paquete.

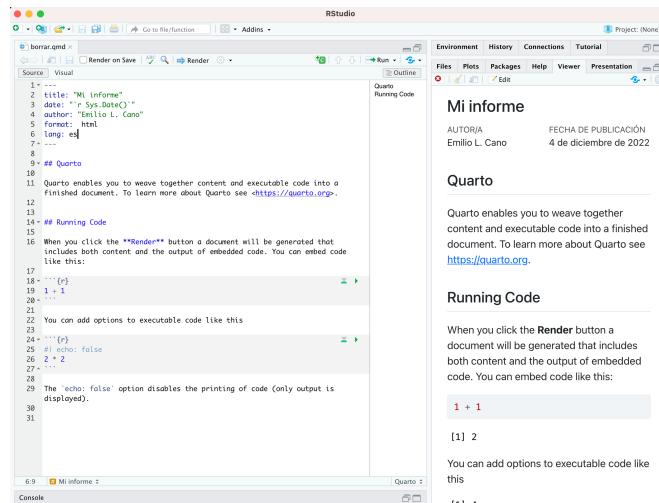


Figura 26.1: Informe Quarto y el documento de salida que produce su conversión (renderizado") con algunas opciones adicionales

26.2.1. Encabezado YAML y configuración

Las opciones de configuración del documento se establecen en este encabezamiento. Al crear el documento con la plantilla, se crea el siguiente encabezado:

```
---
title: "Título del informe"
format: html
editor: visual
---
```

Ya se ha visto cómo se pueden añadir más formatos, poniendo uno en cada línea e “indentando” con el tabulador las distintas opciones. Cada formato a su vez puede incluir opciones, que de nuevo se indican con nuevas líneas que se “indentan” debajo del formato.

La “indentación” se refiere al número de espacios en blanco (o tabulaciones) al principio de cada línea. Las opciones que estén “dentro” de otra, deben tener la misma “indentación” (el mismo número de espacios en blanco al principio de la línea). Véase un ejemplo más completo debajo. En el encabezado YAML la incorrecta “indentación” puede provocar errores al generar el informe.

Además del título (opción `title`), se pueden incluir autor (opción `author`) y fecha (opción `date`). Estos elementos son cadenas de texto que aparecerán al principio del documento de salida. Se debe cuidar que estén entre comillas para evitar posibles errores. El elemento `format` indica el formato de salida.

La cantidad de opciones que se pueden incluir en el encabezado YAML es enorme y no tiene cabida en este capítulo. Algunas de las más utilizadas son `lang` para indicar el idioma del documento (“es” para español), `bibliography` para indicar un fichero bibtex de bibliografía, o `toc` para incluir una tabla de contenidos. Algunas son específicas del formato. Por ejemplo, una muy útil es `reference-doc` para documentos de Word, con la que se puede indicar una plantilla personalizada para usar colores corporativos u otras opciones de diseño del informe⁹. Para documentos html se puede incluir una hoja de estilos con la opción `css`. La lista completa para cada uno de los formatos soportados por **Quarto** se puede consultar en la guía de referencia en <https://quarto.org/docs/reference/>.

El siguiente encabezado YAML fijaría el ancho y el alto de las figuras (en pulgadas) para el formato de salida html, además de las otras opciones comentadas:

```
---
title: "Título del informe"
format:
  html:
    fig-width: 8
    fig-height: 6
    css: estilos.css
  docx:
    toc: true
    reference-doc: plantilla.docx
    pdf: default
lang: "es"
bibliography: bibliografia.bib
---
```

Una explicación detallada del uso de hojas de estilo CSS queda fuera del alcance de este libro. Un ejemplo sencillo para formatear bloques con identificador (nombres precedidos por `#`) y bloques con clase (nombres precedidos por `.`) sería el siguiente:

```
#parrafoazul {
  color: blue;
}

.enfatizar {
  font-size: 1.2em;
}
```

26.2.2. Formateado de texto

Incluir títulos, énfasis en el texto y listas es muy sencillo, y a menudo no se necesita mucho más para realizar un informe. En el informe que se crea con la plantilla ya se ven algunas opciones:

⁹La plantilla se debe crear a partir de un archivo generado con **Quarto**, modificando los estilos del documento y añadiendo elementos como encabezados y pies de página.

- Los encabezados se crean poniendo al principio de la línea tantos símbolos almohadilla (##) como nivel de título se desee (dos almohadillas, apartado, tres almohadillas, subapartado, etc.) Una almohadilla sería para el título del informe, si no se especificara en el encabezado.
- Para poner texto en **negrita**, se incluye entre dos asteriscos a cada lado: ****negrita****.
- Para poner texto en formato **monoespaciado**, tipo código, se pone entre tildes graves (*backticks*, `): `monoespaciado`.
- Los enlaces se crean con: [texto del enlace] (<http://ejemplo.com>).

Existen otras opciones sencillas, que se pueden ver en la *Markdown Quick Reference* del menú *Help*:

- *Cursiva* rodeando el texto con un solo asterisco a cada lado (o guión bajo): cursiva.
- Listas poniendo al principio de la línea un asterisco, guión o signo más:
 - * Primer elemento de la lista
 - * Segundo elemento de la lista
 - + Primer elemento dentro del segundo
 - + Segundo elemento dentro del segundo
 - * ...
- Listas ordenadas poniendo un número y un punto al principio de la línea:
 1. Primer elemento
 2. segundo elemento
 3. ...
- Imágenes de cualquier tipo como: .
- Superíndices^{sup} y subíndices_{sub} con el texto entre símbolos ^ y ~ respectivamente.
- Ecuaciones en formato *LaTeX*, por ejemplo $\sum x_i$ sería \$\\sum x_i\$.
- Saltos de línea (añadiendo más de dos espacios al final de la línea) y saltos de página (tres o más asteriscos, *, o guiones medios, -, en una línea).
- Tablas, usando guiones medios y barras verticales para separar filas y columnas:

Primer encabezado	Segundo encabezado
-----	-----
Contenido de la celda	Contenido de la celda
Contenido de la celda	Contenido de la celda

Con estas opciones se cubren las necesidades de la práctica totalidad de proyectos de análisis. No obstante, dependiendo del formato de salida se pueden añadir otras opciones de formato.

26.2.3. Inclusión de código en el documento

Se pueden crear archivos **Quarto** sin incluir nada de código, simplemente para crear documentos editables fácilmente. Sin embargo, la verdadera potencia de **Quarto** es la posibilidad de incluir código de **R** (y también de otros lenguajes) en los documentos. Como ya se avazó, el código se incluye, principalmente, en forma de *chunks* o bloques de código.

Un *chunk* consta de unos marcadores de inicio y final del *chunk*, entre los cuales se insertan expresiones de **R** que se ejecutarán al generar el documento de salida. El marcador de inicio son tres símbolos de tilde grave seguidos de unas llaves con la letra **r** dentro. El marcador de cierre del *chunk* son de nuevo tres tildes graves, sin más. Y dentro del *chunk* se pueden poner expresiones de **R** de la misma forma en que se trabaja con los *scripts*. Al convertir (“renderizar”) el documento, el código se ejecutará con las opciones que se indiquen como se explica más adelante, y el archivo de salida incluirá el resultado de la ejecución del *chunk*. El siguiente sería un ejemplo de código para incluir gráficos en el informe.

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

En todo caso, no hay que escribir los marcadores de inicio y final, ya que se dispone del atajo de teclado **CTRL+ALT+I** que lo hace automáticamente, o, alternativamente, el ícono *Insert a new code chunk* de la barra de herramientas del editor. Una vez se tiene el cursor dentro de un *chunk*, se puede ejecutar una expresión como del mismo modo que se hace en un *script* (**CTRL+ENTER**), o el *chunk* completo (**MAYUS+CTRL+ENTER**).

A veces es necesario incluir algún resultado de **R** en medio del texto y no como un bloque. En esos casos se puede insertar un bloque en línea poniendo, entre dos tildes graves, la letra **r** como primer carácter, y después una expresión de **R** que se pueda “imprimir” como cadena de texto:

```
`r expresion_de_R`
```

Una opción muy interesante de los informes de **Quarto** es la parametrización. Esta opción es muy útil para informes automatizados que pueden cambiar dependiendo de algún valor, por ejemplo del fichero de datos, la fecha, o cualquier otro valor. Estos parámetros se crean como elementos del encabezado YAML de la forma:

```
params:  
  parametro: valor
```

que después se pueden usar en los *chunks* de código como `params$parametro`. La verdadera potencia de esta característica es cuando se convierte el documento desde un *script* en el que

los parámetros son resultados de algún tipo de operación en los datos (por ejemplo, un informe de análisis de inventario solo de una tienda donde se han producido roturas de stock el día x). En vez de utilizar el botón *Render*, en estos casos se usa la función `quarto_render()`, una de cuyas opciones es `execute_params`, donde se pasarían los valores de los parámetros en forma de lista cuyos elementos tienen el nombre de los parámetros.

26.2.4. Opciones de los bloques de código (*chunks*)

Al renderizar un informe que contiene *chunks* sin configurar ninguna opción, el informe mostrará por defecto el código de entrada y las salidas (textos y gráficos), así como todos los mensajes que se pueden producir.

Opcionalmente, justo después del marcador de inicio del *chunk* se pueden añadir opciones del mismo mediante líneas que comienzan por el llamado *hashpipe*, que es una almohadilla seguida de la barra vertical, `#|` y a continuación la opción y su valor, de la misma forma que se hacía en el encabezado YAML para las opciones del documento, es decir, `opcion: valor`.

El *chunk* que se muestra a continuación tiene como identificador “ejemplo”, y como opciones `echo: false` y `fig.align: 'center'`, lo que indica que el código no se mostrará en el informe final y que el gráfico producido se alinearán en el centro del texto.

```
---
#| label: "Ejemplo"
#| echo: false
#| fig-align: 'center'
plot(cars)
---
```

Las opciones de *chunk* se pueden incluir de forma global en el documento estableciéndolas en el encabezado YAML del mismo, por ejemplo para mantener las opciones anteriores en todos los *chunks* por defecto:

```
---
title: "Mi documento"
format: html
knitr:
  opts_chunk:
    echo: false
    fig-align: 'center'
---
```

Es importante señalar que las opciones establecidas en los *chunks* tienen prioridad a las opciones establecidas en el documento.

Hay varias opciones de *chunk* que tienen que ver con la presentación en la salida. Por defecto, si se produce un error, el proceso se detiene y no se genera el archivo de destino. Este comportamiento, y otras muchas opciones, se pueden configurar como opciones del *chunk*. Las más

habituales son: `error: true` para mostrar los errores y no detener la generación del documento; `warning: false` y `message: false` para no mostrar `warnings` ni mensajes respectivamente; `include: false` para ejecutar el código pero no mostrar ningún tipo de salida; `eval: false` para no ejecutar el código; `results: "hide"` para indicar que no se muestren los resultados (otras opciones son `asis`, `hold` o `markup`); `comment: simbolo` para cambiar el símbolo que se usará como comentario del output (a veces es conveniente simplemente no poner comentario, es decir, `.`). Estas opciones del `chunk` se pueden incluir a nivel global en el encabezado YAML como se ha indicado anteriormente. La lista completa de opciones se encuentra en la *R Markdown reference guide* que está disponible en el menú *Help/Cheatsheets*, o a la [documentación de Quarto sobre opciones de ejecución](#)¹⁰.

Una característica muy cómoda es usar la ayuda contextual al escribir las opciones del `chunk`. Al comenzar a escribir, o pulsando la combinación de teclas **CTRL+ESPACIO**, se muestran las opciones disponibles, y al seleccionar una opción, si tiene varios posibles valores aparecen también para seleccionar.

26.2.5. Referencias cruzadas y formateo de tablas

La salida tabular por defecto de la consola normalmente no es adecuada para un informe. En su lugar, lo que se desea es tener una tabla formateada adecuadamente para el formato de salida. Se pueden incluir en los informes de **Quarto** tablas formateadas de calidad. Para ello, se debe utilizar alguna función que formatee la tabla de acuerdo al formato de salida (HTML, PDF, Word) y, a veces, configurar la opción `results` del `chunk` como '`asis`'. Muchas de estas funciones preparan automáticamente el formato según el fichero de salida que se está generando. Por ejemplo, el siguiente `chunk` generaría una tabla en cualquiera de los formatos de salida usando la función `kable` del paquete `knitr`:

```
```{r}
knitr::kable(TIC2021)
```
```

Hay otros paquetes con multitud de opciones de formato y presentación para las tablas como `xtable`, `flextable`, `kableExtra`, `DT`, o `gt`. Se anima al lector a consultar la documentación de estos paquetes para aprender a crear tablas de calidad que comuniquen adecuadamente los resultados de los análisis.

Tanto las tablas como las figuras se deben referenciar adecuadamente en los informes. Para ello se utilizan las **referencias cruzadas** de los informes **Quarto**. Para poder referenciar un gráfico creado en un `chunk`, es necesario: i) que el `chunk` tenga una etiqueta (`label: 'etiqueta'`); ii) que el `chunk` tenga una opción `fig-cap` para el título de la figura. Entonces el gráfico se puede referenciar en cualquier lugar del documento **Quarto** simplemente escribiendo `@etiqueta`. Por ejemplo, el siguiente `chunk` crearía el gráfico de la figura 26.2, y en el texto se referenciaría como “Figura `@fig-tic`”.

¹⁰<https://quarto.org/docs/computations/execution-options.html>

```
```{r}
#| label: "fig-tic"
#| fig-cap: "Ventas vs. % empresas con banda ancha"
TIC2021 |>
 ggplot(aes(ebroad, esales)) +
 geom_point() +
 geom_smooth(method = "lm")
```

```

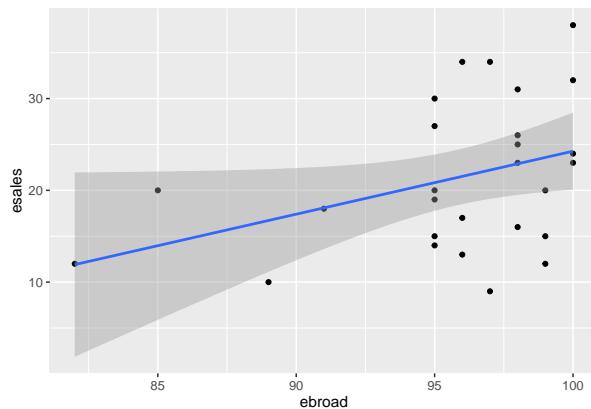


Figura 26.2: Ventas vs. % Empresas con banda ancha

En cuanto a las tablas, igualmente el *chunk* que las crea debe tener una etiqueta. El título de la tabla en este caso lo proporcionará la propia función que la crea. A modo de ejemplo, el siguiente *chunk* crearía la Tabla 26.1 ya formateada con la función `flextable()` del paquete homónimo ([Gohel and Skintzos, 2022](#)), y en el texto se referenciaría como “Tabla @tab-tic”.

```
```{r}
#| label: "tab-tic"
#| fig-cap: "Contaminación media NOx según tipo de estación"
library(dplyr)
library(flextable)
contam_mad |>
 filter(nom_abv == "NOx") |>
 group_by(tipo) |>
 summarise(Media = mean(daily_mean, na.rm = TRUE), n = n(),
 Desv.Tip = sd(daily_mean, na.rm = TRUE),
 Perdidos = sum(is.na(daily_mean))) |>
 flextable() |>
 set_caption("Contaminación media NOx según tipo de estación.") |>
 autofit()
```

```

Tabla 26.1: Contaminación media NOx según tipo de estación.

| tipo | Media | n | Desv.Tip | Perdidos |
|------------|----------|--------|----------|----------|
| Fondo | 64.11108 | 49,656 | 61.48603 | 70 |
| Suburbanas | 32.87574 | 12,414 | 32.13426 | 39 |
| Tráfico | 81.27392 | 33,104 | 68.13974 | 83 |

26.3. Otros formatos

El formato **Quarto** es solo uno de los que se pueden utilizar para aplicar la reproducibilidad que motiva este capítulo. Como se ha comentado, es la nueva generación de **R Markdown**, archivos que pueden seguir creándose con **RStudio**. En **R** base se pueden crear archivos **Sweave**, con sintaxis **LaTeX** para la narrativa y bloques de código **R**. También es posible crear archivos **R** **HTML**, con la narrativa en **HTML**. Los identificadores de los bloques son ligeramente distintos, así como las opciones disponibles, aunque la filosofía es la misma. **R Markdown** y ahora **Quarto** han ido desplazando estos otros formatos al ser más versátil (puede generar cualquiera de los otros formatos, y además otros como **.docx**).

En cuanto a los formatos de salida, hay una cantidad de opciones muy interesante que queda fuera de los objetivos de este libro. A continuación se relacionan algunos de ellos, si bien se puede consultar el libro de [Xie et al. \(2019\)](#) y la documentación de **Quarto** para ver detalles.

- **Notebooks:** es un tipo especial de salida **HTML**, más indicado cuando se quieren ir probando cosas guardando el resultado parcial de lo ejecutado en el **html**.
- **Presentaciones:** es posible crear presentaciones **PowerPoint** y usar una plantilla, como se vio con los documentos de **Word**. Se utiliza una sintaxis muy sencilla y se puede incluir código y resultados igual que en un informe. Otros formatos de presentaciones son **Reveal JS** y **beamer** (**LaTeX**) y con **Quarto**, además, **ioslides**, **slidify** y **xaringan** (**HTML**).
- **Tableros (dashboards):** pueden ser estáticos, usando el paquete **flexdashboard**, útiles para comunicar resultados en un par de pantallazos.
- **Shiny:** aplicaciones web interactivas que responden a inputs del usuario (*reactive*). Estas aplicaciones se tratan en detalle en el Cap. [27](#).
- **Websites:** websites sencillos con páginas enlazadas en el mismo directorio.
- **Blog:** directamente como proyecto **Quarto**, o con el paquete **blogdown**, se pueden generar sitios web completos al estilo de un blog con páginas.
- **Libros:** directamente como proyecto **Quarto**, o con el paquete **bookdown**, se pueden crear libros en varios formatos. El material de este libro está creado con **bookdown**.

- **Tutoriales:** aparecieron en **RStudio** 1.3; se pueden crear documentos interactivos con preguntas y navegación usando sintaxis **R Markdown**.
- **Tufte Handouts:** un tipo especial de documento con anotaciones al margen que comunica muy bien.

Resumen

- En la comunicación de resultados, es esencial seguir un enfoque reproducible.
- **R Markdown** y su evolución **Quarto** es el formato más versátil para crear informes reproducibles que permitan una trazabilidad de los análisis.
- **RStudio** permite trabajar eficientemente con **R Markdown** y **Quarto** a través de ayuda y opciones.
- El encabezado YAML del informe contiene la configuración global, que puede incluir parámetros para automatización.
- La narrativa del informe se escribe en Markdown, con una sintaxis extremadamente sencilla.
- El código se puede incluir en forma de bloques (*chunks*) o en línea.
- En las opciones del *chunk* se puede personalizar la forma en que se ejecutará y mostrará en el informe.
- Los informes **R Markdown** y **Quarto** se pueden generar en formatos HTML, PDF y Word, entre otros.
- En los informes **Quarto** se pueden hacer referencias cruzadas a tablas, figuras y otros elementos del documento.
- Hay otros formatos más elaborados que merece la pena explorar.

Capítulo 27

Creación de aplicaciones web interactivas con Shiny

Aurora González Vidal

27.1. Introducción

Shiny es un paquete de **R** que permite crear aplicaciones web interactivas que cuentan con todos los elementos de **R**. Shiny se ha convertido en un referente ya que, para aquellos que tienen conocimiento de **R**, es muy sencillo crear una aplicación en cuestión de horas (Winston et al., 2020). Para crear una aplicación mínima, no se necesitan conocimientos de HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) o JavaScript y sus dependencias. Además, no es necesario pensar en elementos técnicos para hacerla accesible en la web como, por ejemplo, el puerto, ya que Shiny se encarga de esos detalles si no se cambian las opciones por defecto. Ésas son algunas de las razones principales por las cuales Shiny se ha vuelto tan popular a lo largo de los años, ya que se pueden crear pruebas de concepto de un producto, mostrar algoritmos o presentar resultados de investigación con claridad a través de interfaces de usuario accesibles, reproducibles y amigables (Fay et al., 2021).

El primer paso para usar Shiny consiste en instalar el paquete que está disponible en CRAN:

```
install.packages("shiny")
```

Para asegurarse de que la versión instalada es igual o superior a la 1...5.0. hay que usar `packageVersion("shiny")`. A continuación, se puede cargar el paquete y ver algunos ejemplos que se incluyen directamente en el mismo utilizando distintas opciones para el argumento `example`.

```
library("shiny")
runExample(example = "01_hello")
# otras: 02_text, 03_reactivity, 04_mpg, 05_sliders, 06_tabsets, 07_widgets, 08_html,
→ 09_upload, 10_download, 11_timer.
```

27.2. Componentes mínimos de una aplicación Shiny y disposición básica

Las aplicaciones **Shiny** tienen dos componentes:

1. Una interfaz de usuario **ui**, que es un script y
2. Un **server** que es un script de servidor o secuencia de comandos de servidor.

Estos componentes pueden encontrarse en el mismo script o estar separadas en dos scripts con nombres fijos: **ui.R** y **server.R**. En este caso, se ha elegido la segunda opción para ilustrar los ejemplos con mayor claridad. Una aplicación **Shiny** es un directorio que contiene estos scripts y otros ficheros adicionales (conjuntos de datos, fichero donde se definen funciones no dinámicas, etc.).

El código mínimo para crear una aplicación con un título, panel lateral y panel principal es el que sigue:

■ **ui.R**

```
shinyUI(fluidPage(
  titlePanel("TÍTULO", # panel de encabezado TÍTULO
  sidebarPanel(), # panel lateral
  mainPanel() # panel principal
))
```

■ **server.R**

```
shinyServer(function(input, output) {})
```

La aplicación se puede lanzar de dos maneras diferentes. La primera mediante el comando **runApp()**, que tiene como argumento la ruta del directorio que almacena los ficheros que componen la aplicación.

```
library(shiny)
runApp("ruta al directorio")
```

La segunda es lanzar la aplicación directamente desde Rstudio mediante el botón RunApp que aparece en cualquiera de los dos scripts `ui.R`, `server.R` reemplazando al Run habitual.

En este capítulo, además de ver los distintos componentes de **Shiny**, se construye una aplicación para la visualización de algunos gráficos presentados en el Cap. 34. Los datos relacionados se encuentran en el paquete CDR del libro.

Además del `ui.R` y el `server.R`, puede ser muy útil tener un fichero donde recoger las funciones, paquetes y datos necesarios para el funcionamiento de la aplicación. Este fichero se puede cargar mediante la función `source` desde el `ui.R` o desde el `server.R`. Una recomendación es denominarlo `source.R` para mejor organización y legibilidad.

27.3. Diseño de una aplicación *Shiny*

Shiny incluye una serie de opciones para el diseño o la disposición de los distintos componentes de una aplicación. En esta sección se ven dos muy componentes sencillos:

- `sidebarLayout()`: para colocar un `sidebarPanel()`, es decir, un panel lateral de entradas junto a un `mainPanel()` de contenido de salida.
- `tabsetPanel()` y `navlistPanel()` para la segmentación de diseños.

Hasta ahora se ha utilizado el primero, sin introducirlo específicamente, para mostrar distintos ejemplos, por ser el más sencillo.

27.3.1. Diseño de las páginas: `fluidPage()`

Un diseño de página fluido `fluidPage()` consiste en filas que a su vez incluyen columnas. Las filas tienen como propósito asegurar que sus elementos aparezcan en la misma línea (si el navegador tiene el ancho adecuado). El objetivo de las columnas es definir cuánto espacio horizontal, dentro de una cuadrícula de 12 unidades de ancho, deben ocupar sus elementos. Las páginas fluidas escalan sus componentes en tiempo real para llenar todo el ancho disponible del navegador.

Una `fluidPage()` tiene 2 argumentos: `headerPanel()` con el título de la aplicación, y `sidebarLayout()`, que es un punto de partida útil para la mayoría de las aplicaciones. Éste a su vez tiene 2 argumentos más: `sidebarPanel()`, que es una barra lateral para las entradas y `mainPanel()`, un gran área principal para la salida.

```
shinyUI(fluidPage(
  headerPanel("Evolución del paro"), # panel de encabezado
  sidebarLayout(
    sidebarPanel( # panel lateral
      radioButtons(
        "vble", "Variable", # botones circulares: nombre y etiqueta
        c(
```

```

        "sexo" = "sexo",
        "tramo_edad" = "tramo_edad",
        "tiempo_búsqueda_empleo_agregado" = "tiempo_búsqueda_empleo_agregado",
        "sector" = "sector",
        "tiempo_búsqueda_empleo" = "tiempo_búsqueda_empleo"
    ), "sexo"
)
),
mainPanel(
    plotOutput("gra1")
)
)
))
```

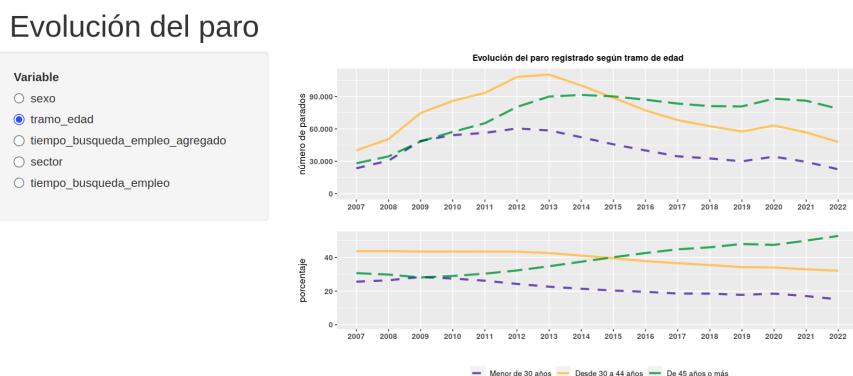


Figura 27.1: Aplicación shiny con sidebarPanel posicionado por defecto a la izquierda

La barra lateral puede posicionarse a la izquierda (por defecto) o a la derecha del área principal. Por ejemplo, para posicionar la barra lateral a la derecha se debe utilizar `position = 'right'` como se aprecia en la Fig. 27.2, donde el resto de argumentos son igual que para generar la Fig. 27.1.

```
shinyUI(fluidPage(  
  headerPanel("Evolución del paro"),  
  sidebarLayout(position = "right", ...)  
)
```

Las funciones `radioButtons()` y `plotOutput()` se introducirán en detalle en las respectivas secciones de este capítulo.

27.3. Diseño de una aplicación *Shiny*

471

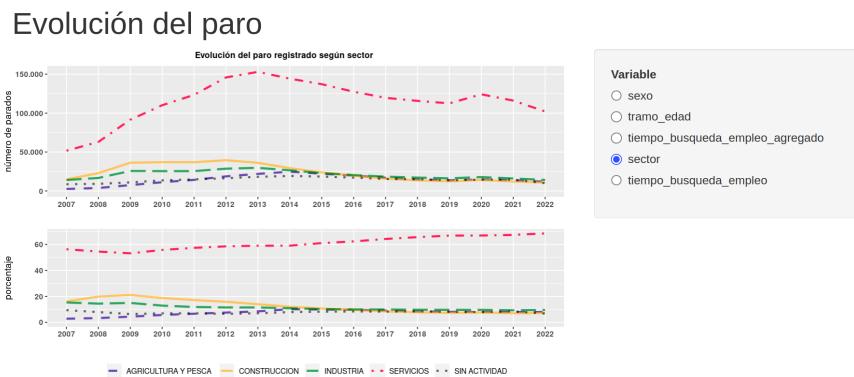


Figura 27.2: Aplicación shiny con sidebarPanel a la derecha

27.3.2. Segmentación de diseños: `tabsetPanel()` y `navlistPanel()`

Para subdividir el panel principal en varias secciones discretas, es decir, crear pestañas, se puede usar `tabsetPanel()` y `tabPanel()` como sigue:

```
mainPanel(
  tabsetPanel(
    tabPanel(
      "Elección con botones circulares",
      radioButtons(
        "vble", "Variable", # botones circulares: nombre y etiqueta
        c(
          "sexo" = "sexo",
          "tramo_edad" = "tramo_edad",
          "tiempo_búsqueda_empleo_agregado" = "tiempo_búsqueda_empleo_agregado",
          "sector" = "sector",
          "tiempo_búsqueda_empleo" = "tiempo_búsqueda_empleo"
        ), "sexo"
      ),
      plotOutput("gra1")
    ),
    tabPanel("Tramo edad fijo", plotOutput("gra2")),
    tabPanel("Tiempo búsqueda empleo fijo", plotOutput("gra3"))
  )
)
```

En el ejemplo de la Fig. 27.3, se aprecia que hay 3 ventanas y se muestra la tercera que es la evolución del paro considerando el tiempo de búsqueda de empleo y que no es una gráfica reactiva sino estática.

`navlistPanel()` es una alternativa a `tabsetPanel()` cuando existan muchas separaciones. Un `navlist` presenta los distintos componentes como una lista de la barra lateral en lugar de utilizar pestañas y no se hace en el `mainPanel`.

Evolución del paro

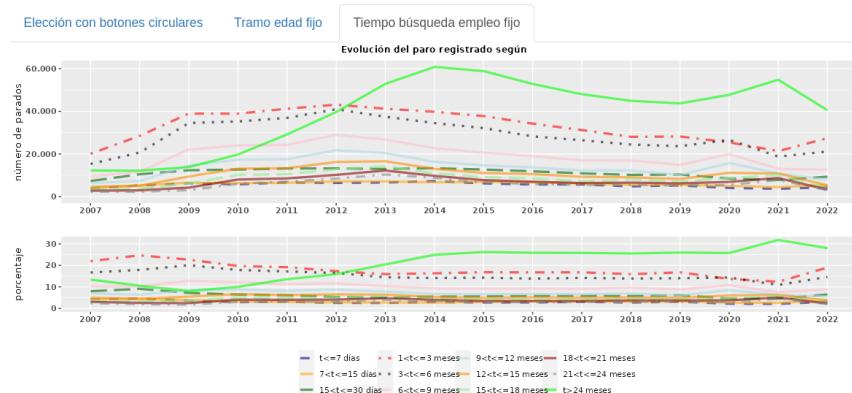


Figura 27.3: Aplicación shiny con varias ventanas

```
ui <- fluidPage(
  titlePanel("Application Title"),
  navlistPanel(
    "Header A",
    tabPanel("Component 1"),
    tabPanel("Component 2"),
    "Header B",
    tabPanel("Component 3")
  )
)
```

27.4. Elementos para la introducción de datos

Para que el usuario de la aplicación Shiny introduzca datos manualmente, hay diversos elementos que se enumeran a continuación:

- **Control deslizante:** un control deslizante permite que el usuario seleccione entre un intervalo de valores moviendo un control de posición por una pista. En Shiny se crea con la función `sliderInput()` que tiene, entre otros, los siguientes argumentos:
 - `inputId`: la entrada que se utiliza para acceder al valor
 - `label`: etiqueta o nombre que aparece en la interfaz
 - `min`: el mínimo del control deslizante
 - `max`: el máximo del control deslizante
 - `value`: el valor inicial
 - `step`: el intervalo entre cada valor seleccionable. NULL significa que va de uno en uno

27.4. Elementos para la introducción de datos

473

- **animate**: booleano que indica si los valores se cambian automáticamente para animar la aplicación

```
sliderInput(inputId, label, min, max, value, step = NULL, animate = FALSE)
```

Sus características incluyen:

- La posibilidad de introducir un único valor y rangos
- Formatos personalizados (por ejemplo, para entradas relativas al dinero)
- Pueden ser animados y recorrer los valores de forma automática (argumento **animate**)

Algunos ejemplos son:

```
sliderInput("enteros", "Enteros:", min = 0, max = 1000, value = 500)
sliderInput("decimales", "Decimales:", min = 0, max = 1, value = 0.5, step = 0.1)
sliderInput("rango", "Rango:", min = 1, max = 1000, value = c(200, 500))
sliderInput("animacion", "Animacion:", 10, 200, 10, step = 10, animate =
  animationOptions(loop = T))
```

- **Botón circular**: un botón circular es un tipo de selector que proporciona una lista de opciones entre las cuales solo se puede seleccionar una. En **Shiny** se crean con la función **radioButtons** que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
radioButtons(inputId, label, choices, selected = NULL)
```

Un ejemplo donde la variable “sexo” está elegida por defecto se puede ver en el primer trozo de código de la subsección [27.3.1 sidebarLayout\(\)](#).

- **Selección múltiple**: un cuadro de selección múltiple es un tipo de selector que proporciona una lista de opciones entre las cuales se pueden seleccionar varias. En **Shiny** se crean con la función **selectInput** que tiene, entre otros, los siguientes argumentos autoexplicativos:

```
selectInput(inputId, label, choices, multiple = FALSE)
```

```
selectInput("año", "Año:",
  c(
    "año1" = "2007",
    "año2" = "2013",
    "año3" = "2019",
    "año4" = "2022"
  ),
  multiple = TRUE
)
```

■ checkboxGroupInput

Muy similar al anterior, este componente crea un grupo de casillas que se pueden utilizar para alternar varias opciones de forma independiente.

```
checkboxGroupInput(inputId, label, choices, multiple = FALSE)
```

```
checkboxGroupInput(  
  "variable", "Variables to show:",  
  c(  
    "año1" = "2007",  
    "año2" = "2013",  
    "año3" = "2019",  
    "año4" = "2022"  
  )  
)
```

■ Entrada numérica

```
numericInput("obs", "Número de observaciones:", 10)
```

■ Entrada de texto

```
helpText("aclaraciones")
```

Otras opciones de entrada que se invita al lector a analizar se relacionan con las fechas: `dateInput()`, `dateRangeInput()` y con un área de texto: `textAreaInput()`.

■ Lectura de ficheros de datos

También es posible introducir información a través de la lectura de ficheros de datos, con la función `fileInput()`. Se pueden combinar valores por defecto de la función utilizada para la lectura de datos con algunos de los elementos anteriores para definir las características del dataset (separador, decimal, cabecera). En el siguiente ejemplo se utiliza la función `read.csv()`, y se da a elegir si tiene cabecera o no con el `checkboxInput()`, así como el tipo de decimal con el `radioButtons`. Otra particularidad del ejemplo es que se asume que los datos están separados por punto y coma, tal y como se aprecia en el argumento `sep` del `read.csv()`.

```
shinyUI(fluidPage(  
  headerPanel("Lectura de datos"),  
  sidebarPanel(  
    h4("Cargar fichero CSV"),  
    fileInput("file1", "",
```

27.4. Elementos para la introducción de datos

475

Elementos para la introducción de datos

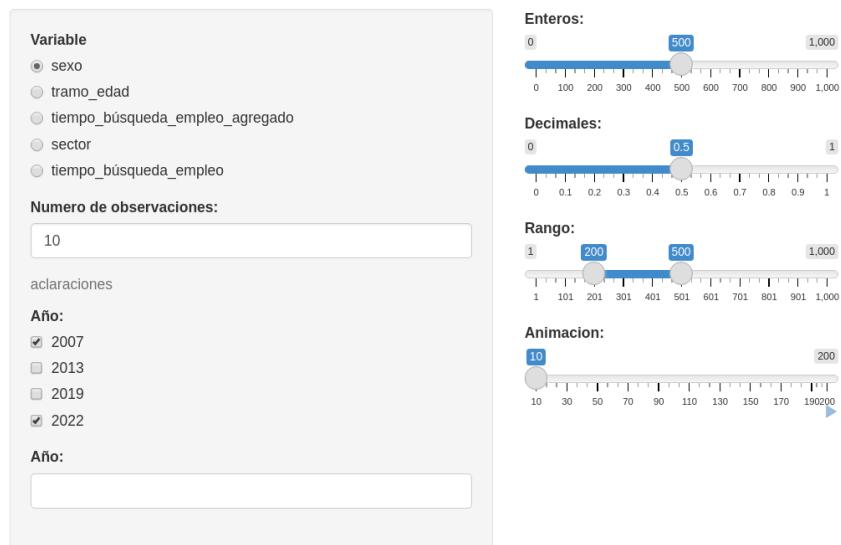


Figura 27.4: Distintos elementos para la introducción de datos

```

accept = c("text/csv", "text/comma-separated-values", "text/plain", ".csv")
),
checkboxInput("header", "Header (el csv tiene nombres de variables)", TRUE),
radioButtons(
  "dec", "Separador de decimales:",
  c(
    "Punto" = ",",
    "Coma" = "."
  )
),
mainPanel(
  tabsetPanel(
    tabPanel(
      "CSV",
      h4("Vista del fichero CSV"),
      tableOutput("contents")
    )
  )
))
shinyServer(function(input, output) {
  output$contents <- renderTable({

```

```

inFile <- input$file1

if (is.null(inFile)) {
  return(NULL)
}

read.csv(inFile$datapath, header = input$header, dec = input$dec, sep = ";")
})
}

```

Lectura de datos

| año | sector | sexo | edad | tiempo_búsqueda_empleo | parados |
|------|---------------|--------|------|------------------------|---------|
| 2022 | SIN ACTIVIDAD | HOMBRE | 16 | <= 7 DIAS | 7.00 |
| 2022 | SIN ACTIVIDAD | MUJER | 16 | <= 7 DIAS | 8.25 |
| 2022 | SIN ACTIVIDAD | HOMBRE | 16 | > 7 <= 15 DIAS | 10.25 |
| 2022 | SIN ACTIVIDAD | MUJER | 16 | > 7 <= 15 DIAS | 6.25 |
| 2022 | SIN ACTIVIDAD | HOMBRE | 16 | > 15 <= 30 DIAS | 16.25 |
| 2022 | SIN ACTIVIDAD | MUJER | 16 | > 15 <= 30 DIAS | 8.50 |
| 2022 | SIN ACTIVIDAD | HOMBRE | 16 | > 1 <= 3 MESES | 38.50 |
| 2022 | SIN ACTIVIDAD | MUJER | 16 | > 1 <= 3 MESES | 28.00 |
| 2022 | SIN ACTIVIDAD | HOMBRE | 16 | > 3 <= 6 MESES | 24.00 |
| 2022 | SIN ACTIVIDAD | MUJER | 16 | > 3 <= 6 MESES | 16.75 |
| 2022 | SIN ACTIVIDAD | HOMBRE | 16 | > 6 <= 9 MESES | 7.75 |

Figura 27.5: Lectura de datos

27.5. Elementos para visualización (salida)

Tras la introducción de ciertos parámetros en el `ui.R`, éstos se pueden utilizar en el script `server.R` mediante la expresión `input`. El código de **R** que construye el objeto basado en esos datos se desarrolla en el servidor, y para generar dicho objeto se utilizan las funciones `renderX`, donde X es el tipo de objeto a devolver. Por último, este objeto se referencia nuevamente en el `ui.R` en el lugar que se desea mostrar (panel) a través de la expresión `XOutput`.

El hecho de colocar una función en `ui` le dice a **Shiny** dónde mostrar su objeto. A continuación, hay que decirle a **Shiny** cómo construir el objeto. Esto se hace proporcionando el código **R** que construye el objeto en la función del servidor.

En concreto, algunas posibilidades se pueden ver en la Tabla ??.

- **Gráficos:** para generar la aplicación de la Fig. ??, se utiliza `renderPlot` en el `server.R` como sigue:

| Server | Ui | Crea |
|-------------|--------------------|----------------|
| renderImage | imageOutput | Imagen |
| renderPlot | plotOutput | Gráfico |
| renderTable | tableOutput | Tabla |
| renderText | textOutput | Texto |
| | htmlOutput | HTML |
| | verbatimTextOutput | Texto verbatim |

```
source("source.R")
shinyServer(function(input, output) {
  output$gra1 <- renderPlot({
    graf_evol(input$vble)
  })
})
```

Se ha llamado `gra1` a la variable que es el gráfico y que se crea con `renderPlot()`. En el interior se utiliza una función denominada `graf_evol()` que es compleja y se crea en `elsource.R` que se carga al principio. Lo que interesa de esta función es que tiene como único argumento el nombre de la variable de interés cuya evolución se desea mostrar. Ésta puede ser una de las diversas opciones que se dan a través del `radioButton` denominado `vble` que ha sido creado anteriormente y que, como vemos, se utiliza `input$vble` para invocar a la selección realizada en la interfaz de usuario.

- **Tablas:** para mostrar la tabla de la Fig. 27.5 se utiliza `renderTable()` en el server y `tableOutput()` en el ui.

```
shinyServer(function(input, output) {
  output$contents <- renderTable({
    inFile <- input$file1
    if (is.null(inFile)) {
      return(NULL)
    }
    read.csv(inFile$datapath, header = input$header, dec = input$dec, sep = ";")
  })
})
```

27.6. Reactividad

La programación reactiva es un paradigma de programación que se encarga de los flujos de datos y la propagación de los cambios. Ésto significa que cuando un flujo de datos es emitido

por un componente, el cambio se propagará a otros componentes.

El modelo de reactividad que utiliza Shiny es el siguiente: hay una fuente reactiva, un conductor reactivo y un punto final de la reactividad (Wickham, 2021). La fuente reactiva suele ser lo que el usuario introduce y el punto de parada lo que se muestra por pantalla. A lo que el usuario introduce se accede con el objeto `input` y a lo que se muestra por pantalla con el objeto `output`. Un ejemplo que ya se ha usado es el siguiente:

```
output$gra1 <- renderPlot({
  graf_evol(input$vble)
})
```

El objeto `output$gra1` es un punto final de la reactividad, y usa la fuente reactiva `input$vble`. Cuando `input$vbles` cambia, a `output$gra1` se le notifica que necesita ejecutarse de nuevo.

27.6.1. Conductores reactivos y control de la reactividad

También es posible crear componentes reactivos que conecten los inputs y los outputs. En el siguiente ejemplo se ha creado un objeto reactivo `datos` que genera datos que siguen una distribución que el usuario selecciona a través del radioButton `dist` y cuya muestra tiene tantos elementos como el usuario haya especificado en el numericInput `obs`.

Shiny, además, permite controlar la reactividad a través de los actionButtons. Se pueden modificar las entradas sin obtener una respuesta hasta que se apriete dicho botón.

Se ha creado un panel nuevo dentro del mainPanel y éste contiene, además del plot previo, un `actionButton` con la etiqueta `Presiona`.

```
shinyUI(fluidPage(
  headerPanel("Controlar reactividad"),
  sidebarPanel(
    radioButtons("dist", "Tipo de distribucion:",
      c(
        "Normal" = "norm",
        "Uniforme" = "unif",
        "Log-normal" = "lnorm",
        "Exponencial" = "exp"
      ),
      selected = "Exponencial"
    ),
    numericInput("obs", "Numero de observaciones:", 10),
  ),
  mainPanel(
    tabPanel(
      "Histograma distribucion RadioButton",
      "Plot", plotOutput("plot"), actionButton("botonReac", "Presiona")
    )
  )
))
```

```
)  
))
```

A continuación, se hace referencia a ese botón para cada una de las expresiones reactivas que se aislarán con la función `isolate()`:

```
shinyServer(function(input, output) {  
  datos <- reactive({  
    if (input$botonReac == 0) {  
      return(dist(rexp(input$obs)))  
    }  
    isolate({  
      dist <- switch(input$dist,  
        norm = rnorm,  
        unif = runif,  
        lnorm = rlnorm,  
        exp = rexp,  
        rnorm  
      )  
  
      dist(input$obs)  
    })  
  })  
  
  output$plot <- renderPlot({  
    if (input$botonReac == 0) {  
      return(NULL)  
    }  
    isolate({  
      hist(datos(),  
        main = paste("r", input$dist, "(", input$obs, ")"), sep = "")  
    })  
  })  
})
```

27.7. Publicación de la aplicación en la web

Después del desarrollo de una aplicación Shiny, suele ser interesante publicarla para su explotación científica o empresarial. Rstudio ofrece diversas soluciones que se analizarán, con distintos niveles de complejidad y libertad, para poder publicar la aplicación web: (i) shinyapps.io, (ii) Shiny Server y (iii) RStudio Connect.

A continuación se introduce, muy brevemente, cada uno de ellos, y se proporcionan los enlaces para que el lector pueda indagar en profundidad.

- Shinyapps.io

Rstudio ofrece un servicio de hospedaje denominado Shinyapps.io que permite subir la aplicación directamente desde la sesión de **R** a un servidor que se mantiene por Rstudio. Hay un control casi completo sobre la aplicación, incluyendo la administración del servidor. Tiene distintos planes, desde gratuito hasta profesional, siendo el primero más restringido en cuanto a servicios (número de aplicaciones, horas activo...) y el último más completo (autenticación, personalización, etc).

Lo único que se necesita es:

- Un entorno de desarrollo de **R**, como RStudio IDE.
- La última versión del paquete **rsconnect**.

En la web shinyapps.io en el apartado “Dashboard” se realiza el registro. Shinyapps.io genera de forma automática un token que el paquete **rsconnect** utiliza para acceder a la cuenta.

```
rsconnect::setAccountInfo(name = "<ACCOUNT>", token = "<TOKEN>", secret = "<SECRET>")
```

Para desplegar la aplicación se utiliza **deployApp()** como sigue:

```
library(rsconnect)
deployApp()
```

Existen opciones gratuitas, que carecen de ciertas ventajas, como la posibilidad de restringir el acceso a la aplicación shiny, es decir, la aplicación no será privada con el plan gratuito aunque sí existen las opciones de autentificación con otros planes. Para más información sobre este método, consultese la página <https://shiny.rstudio.com/articles/shinyapps.html>.

- Shiny Server

Shiny server construye un servidor web diseñado para hospedar aplicaciones **Shiny**. Es gratuito, de código abierto y está disponible en GitHub.

Para usar el **Shiny** Server, es necesario tener un servidor Linux que tenga soporte explícito para Ubuntu 12.04 or superior (64 bit) y CentOS/RHEL 5 (64 bit). Aunque no se esté utilizando una distribución con soporte explícito, también se puede utilizar, si bien construyéndolo desde el paquete fuente.

En el mismo **Shiny** Server se pueden hospedar múltiples aplicaciones **Shiny**. Para ver instrucciones detalladas para su instalación y configuración, se recomienda la guía **Shiny** Server <https://docs.rstudio.com/shiny-server>.

La seguridad y privacidad quedarán supeditadas a los conocimientos del usuario, ya que dependerán de su propio servidor.

- RStudio Connect

Cuando **Shiny** se utiliza en entornos con fines lucrativos, existen herramientas de servidor que se pueden comprar y que vienen equipadas con los programas habituales de un servidor de pago:

- Soporte SSL
- Herramientas de administrador
- Soporte prioritario
- Privilegios de usuario
- Opción con Docker

Para tener dichas herramientas de servidor, la plataforma de publicación RStudio Connect puede ser una solución. Esta herramienta permite compartir aplicaciones **Shiny**, informes RMarkdown, cuadros de mando, gráficos, Jupyter Notebooks y más. Con RStudio Connect se puede programar la ejecución de informes y políticas de seguridad flexibles.

Además, RStudio Connect permite seleccionar privilegios de usuario en aplicaciones Shiny. La aplicación Shiny puede reconocer a un usuario basándose en la información de inicio de sesión y ofrecerle contenido personalizado, de manera que se puede controlar quién ve qué contenido y cuándo lo ve.

27.8. Extensiones de Shiny

Shiny es una herramienta totalmente expansible. Lo que se ha mostrado en este capítulo hasta ahora es un aperitivo en relación a todas las posibilidades que existen en el mundo de **Shiny**. Hay repositorios que recopilan información sobre paquetes que proveen de mejoras a las aplicaciones **Shiny** en su estilo y funcionalidad ([Xiao, 2018](#), [Gilmore et al. \(2017\)](#)). En esta sección se mencionan algunos de ellos pero, sobre todo, se recomienda al lector visitar dichos repositorios para una mayor profundidad en este tema.

- **shinydashboard**, **shinydashboardPlus** y **flexdashboards**

En temas de estilo, merece la pena destacar estos tres paquetes. Los dos primeros presentan una serie de plantillas predefinidas para la creación de las aplicaciones **Shiny**, de manera que los colores combinan y los elementos visuales tienen cierta armonía.

Por su parte, **flexdashboards** tiene como base un documento *R Markdown* y los distintos niveles del mismo definen los paneles de la aplicación a crear.

- **shinyWidgets**

Este paquete ofrece *widgets*¹ personalizados y diversos componentes para mejorar las aplicaciones. Se pueden reemplazar los **checkboxes** por **switch buttons**, añadir colores a los

¹El término “customer churn” se suele traducir como perdida de clientes o rotación de clientes. Se compone de las palabras inglesas “change” (en castellano cambio) y “turn” (en castellano abandonar)

`radioButtons` y al grupo de casillas de verificación (`checkboxGroupInput`), etc. Cada widget tiene un método de actualización para cambiar el valor de una entrada del server.

- `shinycssloaders`

Cuando una salida de `Shiny` (un gráfico, una tabla, etc.) se está calculando, permanece visible pero en gris. Si hay procesos algo más complejos, pueden tardar en mostrarse. Utilizando `shinycssloaders`, se puede añadir una rueda de carga (*spinner*) a las salidas en lugar de hacerlas grises. Envuelviendo una salida `Shiny` en `withSpinner()`, el *spinner* aparecerá automáticamente mientras la salida se recalcula.... Hay ocho tipos de animación incorporadas y personalizables en color y tamaño, pero también se pueden cargar otras animaciones.

- Visualizaciones interactivas

Paquetes como `heatmaply` o `leaflet` se pueden combinar perfectamente con `Shiny` para crear mapas de calor y mapas geográficos interactivos, y utilizarlos en las aplicaciones.

RESUMEN

`Shiny` es un paquete de `R` que permite crear aplicaciones web interactivas requiriendo únicamente conocimientos de `R`. En la primera parte de este capítulo se muestran los elementos básicos de una aplicación `Shiny`: user interface (`ui.R`) y servidor (`server.R`), así como los posibles diseños en relación a los componentes que una aplicación puede tener: barra lateral, paneles discretos, paneles de navegación, etc. A continuación, se repasan los elementos de introducción de datos en una aplicación `Shiny`, incluyendo la carga de conjuntos de datos y también los elementos de salida, como gráficas y tablas. También se aborda el modelo reactividad, es decir, cómo al cambiar algo en los parámetros de entrada de forma dinámica cambia la salida y cómo controlar éste proceso y se muestran distintas opciones para la publicación de las aplicaciones `Shiny`. Por último, se mencionan algunas de las posibles extensiones al paquete.

Capítulo 28

Git y GitHub R

Michal Kinel

28.1. ¿Qué es Git y GitHub?

Git es un sistema de control de versiones distribuido, diseñado para registrar y rastrear los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo ([Chacon, 2009](#)). Al utilizar Git, se pueden ver y restaurar versiones anteriores de un archivo, así como fusionar cambios realizados por diferentes personas en una única versión actualizada.

Por otro lado, GitHub es una plataforma de alojamiento de código online que utiliza Git como sistema de control de versiones subyacentes. Esta plataforma permite a los desarrolladores compartir y colaborar en proyectos de software, alojando el código fuente en la nube ([Astigarraga and Cruz-Alonso, 2022](#)). Además de alojar repositorios de Git, GitHub ofrece herramientas adicionales como seguimiento de problemas, solicitudes de extracción, y wiki de proyectos, lo que la hace una herramienta popular para el desarrollo de software colaborativo y de código abierto.

El uso de Git y GitHub se ha extendido a otros campos más allá del desarrollo de software, como la ciencia de datos, la documentación técnica y la colaboración en general. Su popularidad se debe en gran parte a la facilidad de uso, la flexibilidad y la capacidad de trabajar en proyectos de software colaborativos de manera eficiente y segura tanto.

28.2. ¿Por qué usar Git y GitHub?

Git y GitHub son herramientas para el desarrollo de software moderno, y su uso se ha extendido a otros campos como la ciencia de datos, la documentación técnica y la colaboración entre los desarrolladores. A continuación, se presentan tres ventajas clave de uso de Git y GitHub:

1. **Control de versiones y colaboración eficiente** Git es un sistema de control de versiones distribuido que permite registrar y rastrear los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo. Esto es especialmente útil cuando se trabaja en proyectos de software colaborativos, donde múltiples personas pueden estar editando el mismo archivo al mismo tiempo. Con Git, se pueden ver y restaurar versiones anteriores de un archivo, y también fusionar cambios realizados por diferentes personas en una única versión actualizada. Además, GitHub ofrece herramientas adicionales como seguimiento de problemas, solicitudes de extracción, y wiki de proyectos.
2. **Mejora la eficiencia y la seguridad en el desarrollo de software** El uso de Git y GitHub permite a los desarrolladores trabajar de manera más eficiente y segura en proyectos de software. Al utilizar un sistema de control de versiones como Git, los desarrolladores pueden colaborar de manera más efectiva y reducir el riesgo de conflictos o errores en el código. Además, GitHub ofrece características como la integración continua y la entrega continua (CI/CD), que automatizan y agilizan el proceso de desarrollo de software.
3. **Fomenta la transparencia y la comunidad** GitHub es una plataforma de alojamiento de código abierto, lo que significa que los proyectos alojados en ella son de acceso público y pueden ser revisados y mejorados por otros desarrolladores. Esto fomenta la transparencia de código abierto como privado dentro de una empresa. Además, GitHub cuenta con una gran comunidad de desarrolladores que pueden ofrecer soporte y retroalimentación a otros miembros de la comunidad.

28.3. Instalación y/o actualización de R y RStudio

R es un lenguaje de programación utilizado en la estadística y la ciencia de datos para realizar análisis, modelado y visualización de datos. **RStudio**, por otro lado, es un entorno de desarrollo integrado (IDE) que proporciona una interfaz gráfica de usuario para trabajar con R. Instalar o actualizar **R** y RStudio es un proceso relativamente sencillo, y se pueden seguir los siguientes pasos:

1. Descargar e instalar **R** Lo primero que se debe hacer es descargar **R** desde la [página oficial de R](#). Dependiendo del sistema operativo, se debe elegir la versión correcta de **R** para descargar. Una vez que se haya descargado el archivo, se debe seguir el asistente de instalación para instalar **R** en el equipo.
2. Descargar e instalar RStudio: una vez instalado **R**, se puede proceder a instalar RStudio desde su [página oficial](#). Al igual que con **R**, se debe elegir la versión adecuada para el sistema operativo y seguir el asistente de instalación para instalar RStudio en el equipo.
3. Actualización de **R** y RStudio Para actualizar **R**, se debe abrir **R** y ejecutar el siguiente comando en la consola:

```
install.packages("installr")
library(installr)
updateR()
```

Esto instalará el paquete `installr` y actualizará automáticamente **R** a la última versión disponible. Para actualizar RStudio, se debe abrir RStudio y verificar si hay una actualización disponible en el menú “Help” -> “Check for Updates”. Si hay una actualización disponible, se debe seguir el asistente de actualización para instalar la última versión de RStudio.

En resumen, la instalación o actualización de **R** y RStudio es un proceso sencillo que se puede realizar siguiendo los pasos mencionados anteriormente. Mantener estas herramientas actualizadas es importante para asegurarse de tener acceso a las últimas características y correcciones de errores.

28.4. Configuración de Git y GitHub

Configurar Git y GitHub es un paso importante antes de comenzar a trabajar en proyectos de software colaborativos. Se pueden seguir los siguientes pasos para configurar Git y GitHub:

1. Instalar Git . En primer lugar, es necesario instalar Git en el equipo. Git puede descargarse desde la página oficial de [Git](#). Una vez que se haya descargado el archivo, se debe seguir el asistente de instalación para instalar Git en el equipo. Además, en la página oficial se encuentra un manual completo sobre el uso de Git.
2. Configurar Git. Una vez que se ha instalado Git, se debe configurar el nombre de usuario y la dirección de correo electrónico para que los cambios que se realicen en los repositorios estén correctamente etiquetados. Para hacer esto, se debe abrir la línea de comandos, Git Bash o la terminal de RStudio, y ejecutar los siguientes comandos:

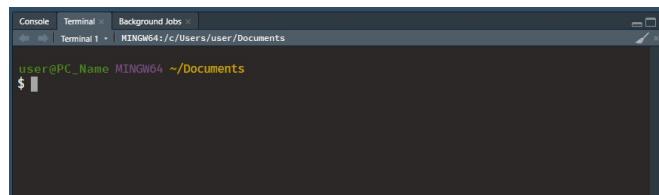


Figura 28.1: Terminal de RStudio

```
$ git config --global user.name "Su Nombre"
$ git config --global user.email "su.correo@ejemplo.com"
```

Esto configurará el nombre de usuario y la dirección de correo electrónico de forma global en Git.

3. Crear una cuenta en GitHub. Para utilizar GitHub, es necesario crear una cuenta en la página oficial de GitHub (<https://github.com/join>). Una vez que se haya creado la cuenta, se debe iniciar sesión en GitHub.
4. Configurar la clave SSH (protocolo Secure Shell), una credencial de acceso para el protocolo de red que permite el acceso remoto a través de una conexión cifrada. Para autenticar las conexiones con GitHub de manera segura (véase capítulo 10 de (Jenny Bryan, 2021)), se recomienda configurar una clave SSH en el equipo y agregarla a la cuenta de GitHub. Para ello, se debe abrir la línea de comandos, Git Bash o la terminal de RStudio, y ejecutar los siguientes comandos:

```
$ ssh-keygen -t rsa -b 4096 -C "su.correo@ejemplo.com"
```

Esto generará una clave SSH. A continuación, se debe agregar la clave SSH al agente de SSH:

```
$ eval "$(ssh-agent -s)"  
$ ssh-add ~/.ssh/id_rsa
```

Finalmente, se debe copiar la clave SSH al portapapeles:

```
$ clip < ~/.ssh/id_rsa.pub
```

y agregarla a la cuenta de GitHub siguiendo las instrucciones en la página de configuración de la cuenta de GitHub:

- En la esquina superior derecha de la página del inicio, haga clic en la foto del perfil y, luego, en “Settings” (Configuración).
- En la sección “Access” de la barra lateral, haga clic en “SSH and GPG keys”.
- Haga clic en “New SSH key” para agregar la clave SSH.

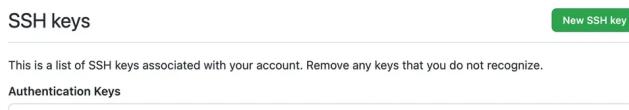


Figura 28.2: Llaves SSH en GitHub

- En el campo “Title” (Título), agregue una etiqueta descriptiva para la clave nueva. Por ejemplo, si está utilizando un portátil personal, puede llamar a esta clave “Portátil personal”.
- Seleccione el tipo de clave, ya sea de autentificación o de firma. Para obtener más información sobre la firma de una confirmación, consulte [aquí](#).

28.4. Configuración de Git y GitHub

487



Figura 28.3: Añadir llave SSH en GitHub

- Pegue su clave pública en el campo “Key”.
- Haga clic en “Add SSH key” para agregar la clave SSH.
- Si se le solicita, confirme su contraseña en GitHub. Para obtener más información, véase [Modo sudo](#).

Además de utilizar una clave SSH para autenticar las conexiones con GitHub, también se puede utilizar la autenticación basada en token de acceso personal, PAT , de GitHub. Esta forma de autenticación es recomendada por GitHub como una forma segura de autenticar conexiones, especialmente cuando se trabaja con aplicaciones y herramientas que requieren acceso a repositorios de GitHub. Para más información sobre el token de acceso personal, PAT, consulte el [capítulo 9](#) de ([Jenny Bryan, 2021](#)).

A continuación, se describen los pasos para utilizar la autenticación basada en token de acceso personal de GitHub:

1. Generar el token PAT: existen dos librerías `usethis` y `gitcreds` que facilitan la generación del PAT y almacenarlo, para ello, se introduce en la consola de RStudio:

```
library(usethis)
usethis::create_github_token()
```

2. Seguir las instrucciones en GitHub: a continuación se abrirá el sitio web de GitHub, se ingresa mediante el usuario y contraseña, con el cuadro para generación del PAT, *New personal access token (classic)*. En *Note* se introduce una nota identificativa al igual que en el procedimiento anterior y se selecciona el tiempo de validez del PAT en la pestaña *Expiration*, dejando las demás opciones por defecto. Se hace clic en *Generate token* para crear el token. En la nueva ventana se copia el token para posteriormente introducir en la consola:

```
library(gitcreds)
gitcreds::gitcreds_set()
```

En *password* se pega el token copiado anteriormente.

3. Para verificar que el nuevo PAT está configurado se introduce en la consola:

```
gitcreds::gitcreds_get(use_cache = FALSE)
```

Si la autentificación fue correcta se generará una salida similar a la siguiente:

```
<gitcreds>
  protocol: https
  host     : github.com
  username: mi_usuario
  password: <-- hidden -->
```

Una vez conectado RStudio y GitHub mediante SSH o PAT se puede proceder con la creación del repositorio en Git.

28.5. Conectar Git y GitHub con Rstudio

28.5.1. Rstudio primero

Este apartado se centra en el enfoque de creación de un nuevo proyecto en un ordenador local para posteriormente subirlo a GitHub, en remoto.

Una vez instalados y configurado Git en nuestro sistema y con la cuenta de GitHub hay que seguir los siguientes pasos para conectar Git y GitHub con RStudio:

1. Configurar Git en RStudio: Una vez que Git está instalado en el sistema, se debe configurar Git en RStudio. Para ello, se debe ir a la pestaña “Tools” en la barra de menú principal, seleccionar “Global Options” y luego seleccionar “Git/SVN”. Desde allí, se debe configurar la ubicación del ejecutable de Git en el sistema.
2. Verificar la versión de Git, introduciendo en la Terminal:

```
$ git --version
```

Si la salida es la versión de Git entonces la instalación fue ejecutada correctamente.

28.5. Conectar Git y GitHub con Rstudio

489

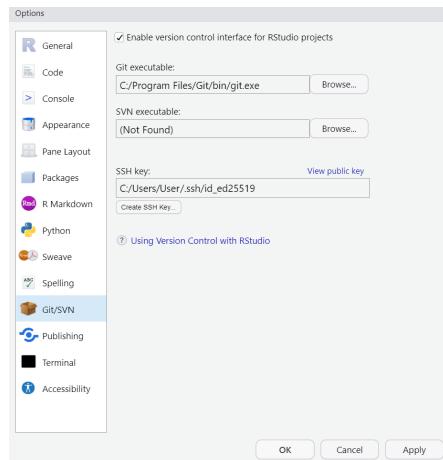


Figura 28.4: Tools de Rstudio

3. Crea un proyecto nuevo desde “File” -> “New project”

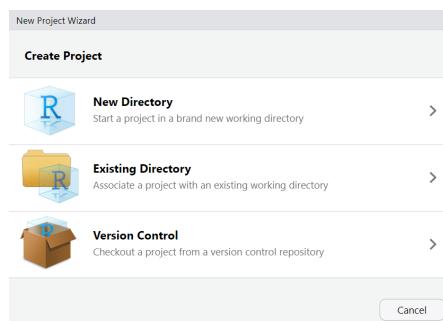


Figura 28.5: Nuevo proyecto de Rstudio

4. En el siguiente cuadro se procede dando clic en “New directory” y en la siguiente ventana se rellenan los datos como el nombre del proyecto y se marca la opción “Create a git repository” para crear un nuevo proyecto con repositorio de Git.

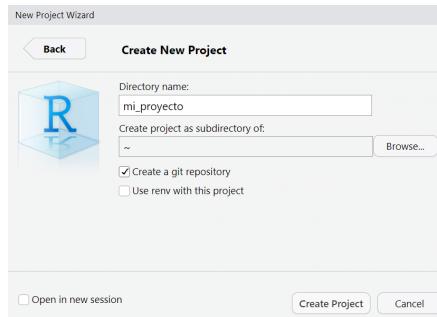


Figura 28.6: Nuevo proyecto en un directorio nuevo

5. En el ícono de Git en la parte superior se accede a la ventana de revisión de cambios, se añaden los ficheros pinchando en los ticks, se añade el mensaje de confirmación y se hace clic en “commit”.

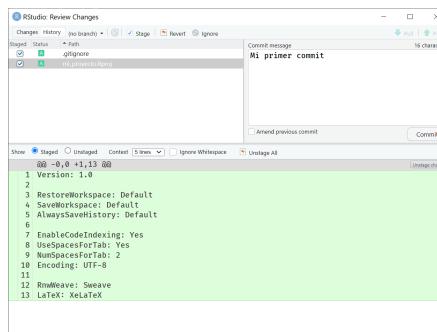


Figura 28.7: Revisión de cambios

6. Alternativamente se puede utilizar la pestaña de Git, marcando los ficheros modificados o creados y confirmando mediante clic en “commit” tras el cual se abrirá el cuadro de diálogo anterior.

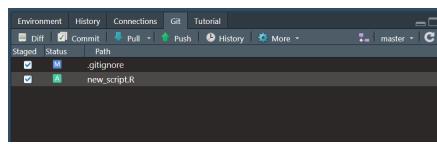


Figura 28.8: Revisión de cambios

7. Para subir los cambios realizados en el proyecto recién configurado en RStudio, habiendo configurado Git y GitHub en los pasos anteriores, se ejecuta en la consola los siguientes pasos

```
library("usethis")
usethis::use_github()
```

La función `usethis::use_github()` en sus valores por defecto creará un repositorio público con el nombre de proyecto en la cuenta asociada. Para ver más opciones acuda a la ayuda de la función, ejecutando en la consola `?usethis::use_github`.

28.5.2. GitHub Primero

Este apartado se centra en el enfoque de creación de un nuevo proyecto en un ordenador local a partir de un repositorio disponible en GitHub, en remoto. Antes de todo hay que verificar si se tiene instalado Git, basta introduciendo en la terminal de RStudio al igual que en el apartado anterior.

```
$ git --version
```

Cuando la salida de la terminal arroje la versión de Git entonces la instalación fue correcta. En el caso de que la salida no arroje la versión vuelva la Sec. 28.4 o consulte el manual de la página oficial de Git en: <https://git-scm.com>.

A continuación, se describe paso a paso sobre cómo conectar GitHub y RStudio a partir de un proyecto ya existente en GitHub y con Git configurado previamente:

1. Abra RStudio y seleccione la opción “New Project” en la pestaña “File” del menú principal. Posteriormente haga clic en la opción “Version Control”.

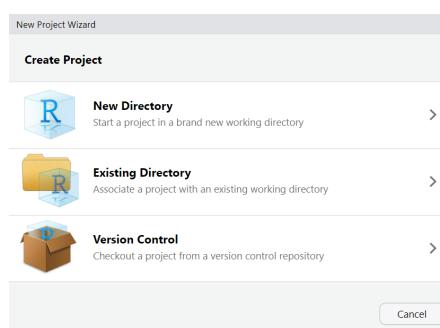


Figura 28.9: Nuevo proyecto de Rstudio

2. En la ventana emergente que aparece, elija “Git”.

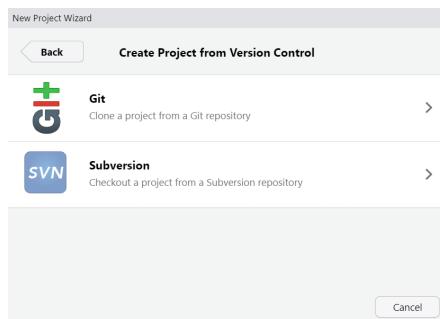


Figura 28.10: Crear proyecto desde control de versiones

3. En la siguiente ventana, pegue la URL del repositorio que deseé clonar y presione “Create Project”. RStudio le preguntará en qué carpeta desea guardar el proyecto, una vez que hayas elegido una ubicación, el proyecto se clonará en tu computadora.

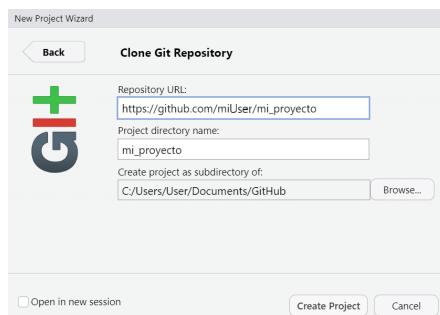


Figura 28.11: Nuevo proyecto desde un repositorio de Git

Los cambios realizados en el repositorio se realizan de la misma forma que en el apartado anterior.

28.6. Flujo de trabajo general de Git y GitHub en RStudio

A continuación se describe un flujo básico de trabajo, comenzando desde RStudio:

1. **Iniciar un repositorio local:** Lo primero que hay que hacer es inicializar un repositorio local en RStudio. Para ello, abra RStudio y seleccione la opción “New Project” en la

28.6. Flujo de trabajo general de Git y GitHub en RStudio

493

pestaña “File” del menú principal. Luego, seleccione la opción “New Directory” y elija una ubicación para su proyecto. A continuación, seleccione “Version Control” y luego “Git”. RStudio le preguntará si desea inicializar un repositorio en este directorio; haga clic en “Yes”. Tal y como se ha descrito en el punto 1 de la Sec. 28.5.1.

2. **Añadir archivos al repositorio:** Ahora, debe añadir los archivos de su proyecto al repositorio. Para ello, haga clic en la pestaña “Git” en la parte superior derecha de RStudio, y luego seleccione los archivos que desea agregar al repositorio. Haga clic en el botón “Stage”, y los archivos seleccionados pasarán a la sección “Staged” en la parte inferior de la pestaña “Git”. Si desea agregar todos los archivos del proyecto al repositorio, haga clic en el botón “Stage All”.
3. **Hacer un “commit” de los cambios:** Una vez que los archivos están en la sección “Staged”, debe hacer un “commit” para registrar los cambios. Para hacerlo, escriba un mensaje breve que describa los cambios que ha realizado en la sección “Commit message”. Luego, haga clic en el botón “Commit”. Los cambios se registrarán en el repositorio local.
4. **Crear una rama (opcional):** Si desea trabajar en una nueva función o corregir un error sin afectar la rama principal (master o main), debe crear una nueva rama. Para ello, haga clic en el botón “New Branch” en la pestaña “Git”. Escriba un nombre para la nueva rama y haga clic en “Create”. Ahora ya es posible hacer cambios en los archivos en la nueva rama sin afectar la rama principal.
5. **Subir los cambios al repositorio remoto:** Una vez que ha hecho un “commit” o confirmación de sus cambios, es hora de subirlos al repositorio remoto en GitHub. Para ello, haga clic en el botón “Push” en la pestaña “Git”. Los cambios se subirán al repositorio remoto en GitHub, que fue configurado en la Sec. 28.4.
6. **Solicitar un pull request (opcional):** Si trabaja en un proyecto colaborativo con otros usuarios, debe solicitar un “pull request” antes de fusionar los cambios en la rama principal. Para hacerlo, haga clic en la pestaña “Pull Requests” en la interfaz de GitHub. Luego, haga clic en el botón “New Pull Request” y siga las instrucciones para crear la solicitud.
7. **Fusionar los cambios en la rama principal (opcional):** Si trabaja en una nueva rama y desea fusionar los cambios en la rama principal, debe crear una solicitud de “pull request”. Si la solicitud es aceptada por el propietario del repositorio, los cambios se fusionarán en la rama principal.

Repository local y remoto:

- **Repository local** en Git es una copia completa de un proyecto que se encuentra en el equipo del usuario. Con un repositorio local, los usuarios pueden trabajar en un proyecto sin conexión a Internet y luego enviar los cambios al repositorio remoto cuando estén conectados.
- **Repository remoto** en GitHub es una versión en línea del proyecto que está almacenada en los servidores de GitHub. Los usuarios pueden clonar un repositorio remoto a su equipo para tener una copia local del proyecto y trabajar en ella. Los cambios realizados en la copia local pueden ser enviados al repositorio remoto para compartirlos con otros usuarios.

En resumen, el flujo de trabajo general de Git y GitHub en RStudio implica inicializar un repositorio local, añadir archivos al repositorio, hacer un “commit” de los cambios, crear una nueva rama si es necesario, subir los cambios al repositorio remoto en GitHub.

Todas las operaciones se pueden realizar desde la terminal de RStudio. Aquí hay algunos de los comandos más comunes que se utilizan en Git:

- **git init:** Este comando se utiliza para crear un nuevo repositorio de Git. Se ejecuta en el directorio raíz del proyecto y establece la estructura necesaria para que Git rastree los cambios en el código fuente.
- **git clone:** Este comando se utiliza para clonar un repositorio existente de Git. Es útil cuando se desea trabajar en un proyecto que ya está en GitHub o en otro servicio de alojamiento de repositorios de Git.
- **git add:** Este comando se utiliza para agregar archivos nuevos o modificados al área de preparación “Stage” de Git. La preparación es el primer paso para confirmar los cambios en Git.
- **git commit:** Este comando se utiliza para confirmar los cambios realizados en el repositorio de Git. Los cambios confirmados se guardan en la base de datos de Git y se etiquetan con un mensaje que describe los cambios.
- **git push:** Este comando se utiliza para enviar los cambios confirmados a un repositorio remoto, como GitHub. Esto actualiza el repositorio remoto con los cambios realizados en el repositorio local.
- **git pull:** Este comando se utiliza para actualizar el repositorio local con los cambios realizados en el repositorio remoto. Es útil cuando se está trabajando en un proyecto colaborativo y otros colaboradores han realizado cambios en el repositorio remoto.
- **git branch:** Este comando se utiliza para crear, listar y eliminar ramas en el repositorio de Git. Las ramas son una forma de trabajar en diferentes versiones del proyecto sin afectar la rama principal.

28.6. Flujo de trabajo general de Git y GitHub en RStudio

495

- **git merge:** Este comando se utiliza para fusionar ramas diferentes del repositorio de Git. Esto se utiliza comúnmente cuando se trabaja en diferentes ramas y se desea integrar los cambios realizados en una rama en la rama principal.
- **git status:** Este comando se utiliza para verificar el estado del repositorio de Git. Proporciona información sobre los archivos que se han modificado y los archivos que se han agregado al área de preparación.
- **git log:** Este comando se utiliza para ver un registro detallado de los cambios confirmados en el repositorio de Git. Muestra información como el autor del cambio, la fecha y la descripción del cambio.

Para conocer más a fondo la mecánica de Git es muy recomendable el manual ([Chacon, 2009](#)) o la hoja resumen proporcionada por GitHub, disponible en https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf.

Resumen

- Git es un sistema de control de versiones distribuido utilizado para rastrear cambios en archivos a lo largo del tiempo, mientras que GitHub es una plataforma de alojamiento de código que utiliza Git como su sistema de control de versiones subyacente.
- La instalación y configuración de Git y GitHub es sencilla y permite una colaboración eficiente y un control de versiones en el desarrollo de software.
- Conectar GitHub y RStudio implica configurar las credenciales de Git, hacer cambios en los archivos y enviar los cambios al repositorio de GitHub.
- El flujo de trabajo general en Git y GitHub implica inicializar un repositorio local, agregar archivos, comprometer cambios, crear una nueva rama si es necesario, enviar cambios al repositorio remoto, solicitar una solicitud de extracción si se trabaja en colaboración y fusionar cambios en la rama principal.

Capítulo 29

Geoprocесamiento en nube

Dominic Royé

Fundación de la Investigación del Clima

29.1. Introducción

Cuando se plantea un problema basado en datos desde diversos proveedores, habitualmente implica la descarga de grandes volúmenes. La actual proliferación de servicios de Open Data, despliegues de sensores y diversas fuentes incluyendo los satélites dificulta su procesamiento en equipos personales. El gran crecimiento en grandes volúmenes de datos espacio-temporales de tipo vectorial o ráster lleva a la necesidad en trabajar con servicios en nube para ahorrar tiempo computacional y espacio de almacenamiento. En la actualidad existen diferentes servicios de geoprocесamiento en nube que ayudan a hacer análisis online sin necesidad de descargar los datos ni preocuparse por el rendimiento computacional. Uno de estos servicios es *Google Earth Engine* (GEE), donde se combina un catálogo de varios petabytes de imágenes satelitales y conjuntos de datos geoespaciales multidimensionales (vectorial y ráster) de alta resolución con capacidades de análisis a escala planetaria. Este servicio gratuito para uso no comercial incluye incluso la posibilidad en crear aplicaciones.

GEE consiste en una API con bibliotecas de cliente para JavaScript y Python, que traducen los análisis geoespaciales y hacen posible acceder a los datos. No es necesario descargar grandes volúmenes de datos ni configurar la computación. Para el lenguaje de R se puede hacer uso del paquete `rgee` que hace puente entre **R** y la API GEE.

Se hará uso del dataset con el nombre “NOAA CDR OISST v02r01”, una interpolación óptima diaria de temperatura de la superficie del mar (OISST, por sus siglas en inglés) con una resolución de 1/4 grados (27 km). Los datos los proporciona la National Oceanic and Atmospheric Administration (NOAA) con campos completos de temperatura del océano construidos mediante la combinación de observaciones ajustadas por sesgo de diferentes plataformas

(satélites, barcos, boyas) en una cuadrícula global regular, con lagunas estimadas por interpolación (https://developers.google.com/earth-engine/datasets/catalog/NOAA_CDR_OISST_V2_1) (Reynolds et al. (2008)).

El objetivo de este capítulo es mostrar el potential del uso de APIs directamente en **R**. El resultado se empleará en el Cap. 38.

29.2. Sintaxis de Google Earth Engine

Con ayuda de GEE se preprocesan los datos de tal manera que el resultado son las anomalías estivales en forma de ráster para cada año entre 1981 y 2022. El primer paso consiste en crear el usuario en earthengine.google.com. Además, es necesario instalar *CLI* de *gcloud* (<https://cloud.google.com/sdk/docs/install?hl=es-419>).

Antes se deben conocer algunos conceptos fundamentales sobre la sintaxis en GEE. En general, el lenguaje nativo es Javascript el que se caracteriza por la forma combinando funciones y variables usando el punto, el que se sustuye por el \$ en R. Todas las funciones GEE en **R** empiezan por el prefijo ee_* (`ee_print()`, `ee_image_to_drive()`). Los términos más relevantes son los siguientes:

- *ImageCollection*: serie temporal de imágenes.
- *Geometry*: dato vectorial.
- *Functions*: `map()` aplica funciones sobre *ImageCollections*, `ee.Date()` define una fecha, `filterDate()` permite filtrar por fecha una *ImageCollection*, `select()` selecciona una banda, etc.

Muchas funciones son similares a las de `tidyverse`.

Se puede obtener más ayuda en <https://r-spatial.github.io/rgee/reference/rgee-package.html> y en la propia página de GEE.

29.3. Primeros pasos

Después de darse de alta en GEE y haber instalado *CLI gcloud* en el sistema operativo, se crea un entorno virtual de Python con todas las dependencias de GEE usando la función `ee_install()`.

```
library(rgee)

ee_install() # crear entorno virtual de Python; ¡sólo una vez!
ee_check() # comprobar si todo está correcto
```

Antes de pasar a programar con la sintaxis propia de GEE, se debe autenticar e inicializar GEE empleando la función `ee_Initialize()`.

```
ee_Initialize(drive = TRUE) # autenticar e inicializar GEE
ee_user_info() # inf sobre usuario
```

Hay que tener en cuenta que, únicamente cuando se envían tareas, GEE ejecuta el cálculo en los servidores enviando todos los objetos creados. En la mayoría de los pasos se crean objetos *EarthEngine*, que se usan una vez que se construyó un mapa interactivo, la exportación o la impresión en consola de un objeto.

Por ejemplo, se puede seleccionar la banda NDVI del producto MODIS MOD13A2 e imprimir los metadatos del primer día disponible con `ee_print()`. Existe un límite 5000 elementos que se podrían ver usando esta función.

```
# imageCollection NDVI
img <- ee$ImageCollection('MODIS/006/MOD13A2')$select('NDVI')

# metadatos
ee_print(img$first())
```

29.4. Cálculo de anomalías

29.4.1. Definiciones previas

Los datos NOAA CDR OISST contienen la temperatura superficial de los océanos a nivel global, por eso, se fija un rectángulo que cubre la extensión del Mar Mediterráneo como objeto de estudio.

```
# extensión del Mar Mediterráneo
geom <- ee$Geometry$Polygon(coords = list(
  c(-6.046418548121442, 46.733937391710846),
  c(-6.046418548121442, 29.680544334046786),
  c(42.469206451878556, 29.680544334046786),
  c(42.469206451878556, 46.733937391710846)
),
proj = "EPSG:4326",
geodesic = FALSE)

geom #vemos que es un objeto EarthEngine de tipo geometría
str(geom) # construcción javascript
```

En el siguiente paso se define el período de interés, desde el año 1982 hasta el 2022.

```
startDate <- ee$Date('1982-01-01') # fecha inicio
endDate <- ee$Date('2023-01-16') # fecha final
```

Se puede acceder a todas las colecciones (*ImageCollection*) indicando su identificación. Además, se filtran y se recortan los datos con respecto al periodo y a la extensión fijada. Finalmente se selecciona la banda o variable de interés “sst” (*surface sea temperature*).

```
collection_era5 <- ee$ImageCollection("NOAA/CDR/OISST/V2_1")$filterDate(startDate, endDate)$filterBounds(geom)$select('sst')
```

Finalmente, se procede a calcular el número de años en el período fijado.

```
number0fyears <- endDate$difference(startDate, 'years')$round()
```

29.4.2. Promedio estival

Después de las anteriores definiciones se crea una nueva colección con el promedio estival de cada año durante el periodo objeto de estudio. Para ello se crea una lista de los años sobre la que se mapea otra función. Esta función se debe pasar con `ee_utils_pyfunc()`, que traduce una función R a una de Python.

En la función personalizada se filtran los meses de verano, se calcula el promedio y se multiplica por 0,01, un factor de escala. Cuando se crean nuevas colecciones es importante fijar la nueva fecha con `set()`.

```
yearly <- ee$ImageCollection(
  ee>List$sequence(0, number0fyears$subtract(1L))$map(ee_utils_pyfunc(function(dayOffset) {
    yr = startDate$advance(dayOffset, 'years')$get('year')
    start = ee$Date$fromYMD(yr, 12L, 1L)
    end = ee$Date$fromYMD(yr$add(1L), 2L, 28L)
    return(collection_era5$filterDate(start, end)$mean()$multiply(0.01)$set('system:time_start', start$millis()))
  })))
)
```

En el siguiente paso se calcula la temperatura media estival entre 1982 y 2010, como período de referencia para las anomalías.

```
msst <- collection_era5$filterDate('1982-01-01', '2010-12-31')$  
filter(ee$Filter$calendarRange(12L, 2L, 'month'))$  
mean()$  
multiply(0.01)
```

Se aplica otra función personalizada sobre las medias estivales de todos los años, en la que se resta la temperatura del periodo de referencia, obteniéndose así las diferencias entre la temperatura media de cada año en el periodo estival y la temperatura media global del periodo estival en el periodo 1982-2010.

```
anom <- yearly$map(ee_utils_pyfunc(function (im) {  
  return(im$subtract(msst)$set('system:time_start',  
                                im$get('system:time_start')))  
}))
```

Es puede crear un mapa interactivo de un año concreto aplicando la función `Map.addLayer()` (Fig. 29.1). En este paso es la primera vez que GEE calcula lo que se ha creado anteriormente.

```
# metadatos  
ee_print(anom$first()) # año 1982  
  
# mapa interactiva del año 1982  
Map$setCenter(9, 40, 5) # centrar mapa en el mediterráneo con nivel de zoom 5  
  
# crear mapa con leyenda  
Map$addLayer(  
  eeObject = anom$first(),  
  visParams = list(  
    palette = rev(RColorBrewer::brewer.pal(11, "RdBu")),  
    min = -3,  
    max = 3  
  ),  
  name = "MED_SST"  
) +  
Map$addLegend(list(min = -3, max = 3,  
                  palette = rev(RColorBrewer::brewer.pal(11, "RdBu"))),  
             name = "SST Anomaly",  
             position = "bottomright",  
             bins = 4)
```

Hasta este momento, no se ha enviado una tarea para que GEE la realice. Para ello hay que exportar las anomalías de cada año en formato `geotiff`. La función `ee_imagecollection_to_local()` facilita la exportación de todas las capas de una `ImageCollection`. En cambio, la función `ee_image_to_drive()` exporta datos individuales de una única imagen a `Google Drive`. El argumento `scale` indica con qué resolución se exporta. Aunque la resolución de los datos originales es de 27 km, se fija una resolución de 2 km, lo que implica una interpolación en la exportación por razones de estética en la visualización.

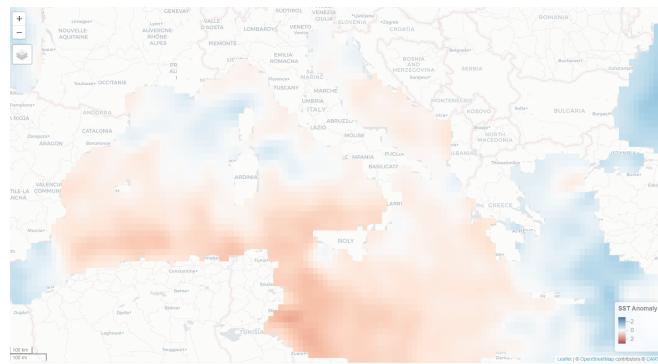


Figura 29.1: Mapa interactivo del mar Mediterráneo (1982)

```
tmp <- tempdir() # carpeta temporal o cualquier otra ruta

ic_drive_files_2 <- ee_imagecollection_to_local(
  ic = anom,
  region = geom,
  fileFormat = "GEO_TIFF",
  scale = 2000,
  lazy = FALSE,
  dsn = file.path(tmp, "rast_"), # prefijo del archivo
  add_metadata = TRUE
)
```

Este ejemplo ha mostrado una pequeña parte de la capacidad del geoprocésamiento en manejar grandes volúmenes de datos sin que implique su descarga ni el cálculo localmente. Pero también es posible manejar datos vectoriales u otros datos de alta resolución. Incluso se puede llegar a crear aplicaciones basadas en GEE.

Resumen

El crecimiento de volúmenes de datos en la actualidad lleva a la necesidad de emplear servicios de geoprocésamiento en nube que ayuden a hacer análisis online sin necesidad de descargar los datos ni preocuparse por el rendimiento computacional. Uno de estos servicios es *Google Earth Engine*, donde se combina un catálogo de imágenes satelitales y conjuntos de datos geoespaciales multidimensionales de alta resolución con capacidades de análisis a escala planetaria. En este ejemplo se ha mostrado cómo procesar la temperatura superficial del mar calculando las anomalías estivales de la cuenca mediterránea desde 1982 a 2022.

Parte VII

Casos de estudio en ciencia de datos

Capítulo 30

Análisis de una red criminal

F. Liberatore^a, L. Quijano-Sánchez^b, M. Camacho-Collados^c

^aCardiff University, ^bUniversidad Autónoma de Madrid, ^cMinisterio del Interior

30.1. Introducción

En este capítulo se plantea la idea de realizar un **análisis de una red de crimen organizado**. Para ello, se estudia la red derivada de un *dataset* real, relativo a la operación *Oversize*. El estudio se llevará a cabo usando la librería `igraph`.

30.2. El conjunto de datos *Oversize*

Los datos que se van a analizar se han obtenido de la operación *Oversize* (Berlusconi et al., 2016) (Tribunale di Milano, Ufficio del giudice per le indagini preliminari, 2006) (Tribunale di Lecco, 2a Sezione Penale, 2009), un proceso italiano contra un grupo mafioso. La investigación duró del 2000 al 2006, y se enfocó en más de 50 sospechosos involucrados en tráfico internacional de drogas, homicidios y robos. El juicio empezó en el 2007 y duró hasta el 2009, cuando se dictó la sentencia y los principales sospechosos fueron condenados con penas de 5 a 22 años de cárcel. La mayoría de los sospechosos eran afiliados de la '*Ndrangheta*, una mafia de Calabria (una región del sur de Italia) con ramificaciones en otras regiones y en el extranjero.

En particular, se va a estudiar la red obtenida de las escuchas telefónicas. Los datos hacen referencia a todas las conversaciones telefónicas transcritas por la policía y consideradas relevantes. En esta red, los nodos representan sospechosos (los datos son anónimos y los nombres asignados en la red se han generado de forma aleatoria). Las aristas conectan los sospechosos que han tenido al menos una conversación telefónica relevante al caso durante la investigación.

30.3. Creación de la red mafiosa

El dataset `Oversize_nodes` contiene el listado de nodos con sus propiedades, en este caso el nombre (ficticio) del sospechoso. `Oversize_edges` contiene las aristas del grafo, representadas como parejas de nodos, a su vez identificados por su ID. A partir de estos datasets la librería `igraph` permite crear un grafo, tal y como se ilustra a continuación.

```
library('igraph')
library('CDR')
data(oversize_edges, oversize_nodes)
net <- graph_from_data_frame(d=oversize_edges,
                             vertices=oversize_nodes,
                             directed=F)
net
#> IGRAPH a3dadd3 UN-- 182 247 --
#> + attr: name (v/c)
#> + edges from a3dadd3 (vertex names):
#> [1] Casto Ben          --Gustavo Mango
#> [2] Casto Ben          --Metrofane Abbatiello
#> [3] Uranio Natoli     --Fidenziano Marcellino
#> [4] Lancilotto Di Biasi--Romolo Gemignani
#> [5] Lancilotto Di Biasi--Fidenziano Marcellino
#> [6] Senesio Rabito     --Pacifico Caliri
#> [7] Senesio Rabito     --Michelangelo Piccininni
#> [8] Romolo Gemignani   --Alighiero Mazzarella
#> + ... omitted several edges
```

La vista previa del grafo indica lo siguiente:

- El grafo es no dirigido (`UN`) y está compuesto por 182 nodos y 247 aristas.
- El único atributo es el nombre de los nodos (`attr: name (v/c)`).
- Finalmente, se proporciona una previsualización de un subconjunto de aristas, indicando para cada una los dos nodos conectados (ej. *Casto Ben --Gustavo Mango*).

30.4. Visualización de la red mafiosa

Para hacerse una idea de que aspecto tiene el grafo, se procede a su visualización, usando el comando `plot()` de **R**.

```
plot(net, asp=0)
```

30.4. Visualización de la red mafiosa

507

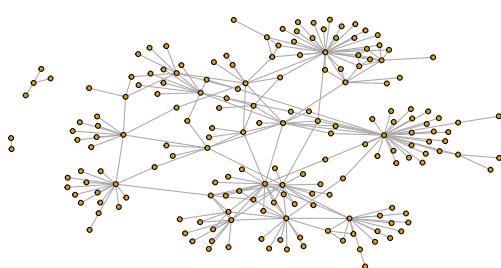


Como se puede apreciar, el resultado no es muy claro. Todos los nodos tienen el mismo tamaño y se solapan entre ellos. Además, se muestran los nombres de todos los actores dentro de la red, lo cual dificulta ulteriormente su interpretación.

Se puede mejorar esta presentación usando unos parámetros de `plot()`, específicos de `igraph`. En particular:

- `vertex.size`: determina el tamaño de los nodos.
 - `vertex.label`: define el texto asociado a cada nodo. Por defecto se asume que es su nombre. En el ejemplo de abajo, se excluyen los nombres de la visualización.

```
plot(net, vertex.size=2, vertex.label=c(''), asp=0)
```



En la Fig. ?? se puede ver cómo el grafo permite una mejor valoración de la distribución de los actores dentro de la red. Por ejemplo, hay dos grupos pequeños (de cuatro y dos actores) completamente desconectados de la red principal.

30.5. Importancia de los actores (delincuentes)

Las medidas de centralidad permiten asignar un valor a cada actor que establece su importancia relativa a los demás. Existen diversas medidas, cada una con sus características y finalidad. En este ejemplo se van a usar las siguientes:

- **Grado:** número de aristas que llegan al nodo o salen de él. Cuanto más alto sea este valor, más vecinos tendrá el nodo.
- **Intermediación:** cuantifica el número de veces que un nodo se encuentra en el camino más corto entre otros actores. Cuanto más alto este valor, más información pasará por el nodo.

```
dgr <- degree(net) # Centralidad de grado
btwn <- betweenness(net) # Centralidad de intermediación
```

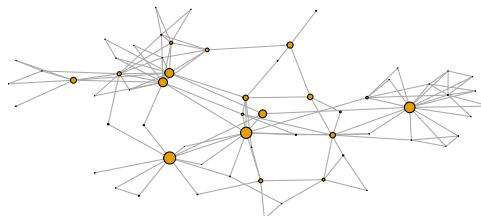
A continuación, se muestran los actores con los valores más altos en cada medida de centralidad.

```
head(sort(dgr, decreasing = T))
#>      Gustavo Mango      Pacifico Caliri Metrofane Abbatiello
#>            32                  31                      18
#>      Olindo Iacona      Arturo Gizzi      Guido Minervini
#>            17                  16                      16
```

```
head(sort(btwn, decreasing = T))
#>      Gustavo Mango      Bino Lana      Pacifico Caliri
#>        4602.167      4292.902      4056.435
#>      Olindo Iacona Metrofane Abbatiello      Donato Di Santi
#>        3397.907      3387.931      2978.427
```

Las medidas de centralidad se pueden usar para mejorar la visualización del grafo. Primero, se filtran todos los nodos que tengan grado menor que dos, ya que representan actores muy marginales en la red. Luego, se representa el tamaño de cada nodo en función de su valor de intermediación, escalando con un tamaño máximo de cinco.

```
vertex_filter <- dgr > 1 # detección actores marginales
scaled_btwn = 0.1 + 4.9*btwn/max(btwn) # Escalado del tamaño del nodo en función de la
#> → intermediación
net2 = induced.subgraph(net, which(vertex_filter)) # creación subgrafo
plot(net2,
     vertex.size=scaled_btwn[vertex_filter],
     vertex.label=c(''),
     rescale=T,
     asp = 0) # visualización subgrafo
```



Como se puede apreciar en la Fig. ??, gracias a las medidas de centralidad se puede tener una mejor idea de cómo se configura la red respecto a sus actores más importantes.

30.6. Identificación de comunidades de la mafia

A continuación, se procede a identificar las comunidades existentes en el grafo de la operación *Overdrive*. `igraph` proporciona una gran variedad de algoritmos para la detección de comunidades en redes sociales. En el siguiente ejemplo, se usa el algoritmo Louvain (Blondel et al., 2008) que es el más popular.

```
louvain_partition <- cluster_louvain(net) # Ejecucion del algoritmo Louvain
net$community <- louvain_partition$membership # Asignacion de las comunidades al grafo
unique(net$community) # Visualizacion de las comunidades encontradas
#> [1] 1 2 3 4 5 6 7 8 9
```

El algoritmo identifica distintas comunidades, cada una con su número asignado.

30.7. Visualización de comunidades de la mafia

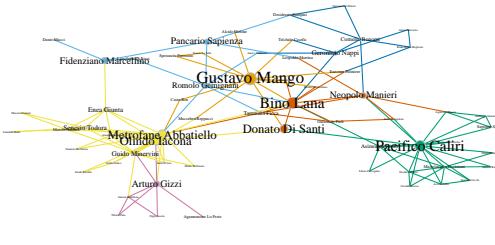
Se procede a visualizar las comunidades detectadas en el subgrafo, representando cada una de ellas en un color distinto. Además, para mejorar la calidad de la información representada, se resalta la importancia de cada actor representando los nodos asociados y sus nombres en tamaños proporcionales a su centralidad en toda la red.

```
V(net2)$size <- scaled_btwn[vertex_filter] # Tamaño del nodo en funcion de su
#>   centralidad
V(net2)$frame.color <- "grey"
```

```
V(net2)$color <- net$community[vertex_filter] # Color del nodo en función de su
#> comunidad
V(net2)$label <- V(net2)$name
V(net2)$label.cex <- (1+scaled_btwn[vertex_filter])/6 # Escalado del nombre en función
#> de su centralidad
V(net2)$label.color <- 'black'

# Definición del color de las aristas en función de la comunidad de origen
edge.start <- ends(net2, es = E(net2), names = F[,1])
E(net2)$color <- V(net2)$color[edge.start]

plot(net2, asp=0) # Los resultados puede ser distintos con cada ejecución
```

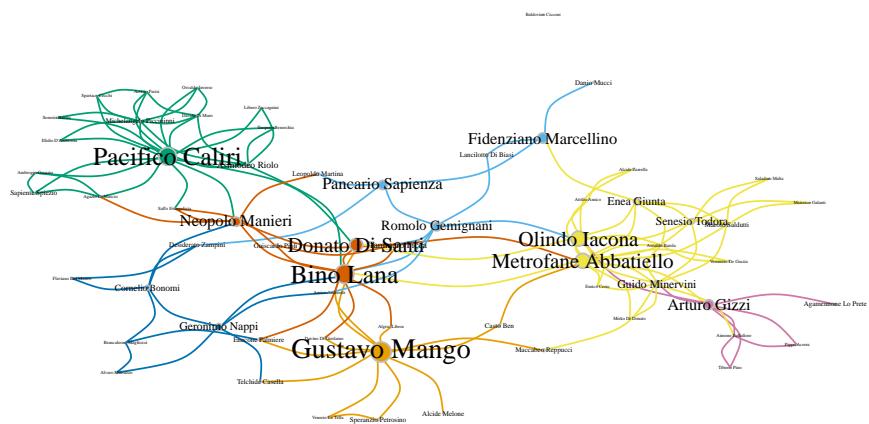


Se puede mejorar aún más el aspecto del grafo. Para ello, se va a experimentar con una disposición diferente de los nodos. En este ejemplo, se usa el algoritmo Fruchterman-Reingold ([Fruchterman and Reingold, 1991](#)). Además, se aplica un efecto de curvatura a las aristas asignando un valor positivo al parámetro `edge.curved`. El resultado se puede ver en la Fig. ??.

```
l1 <- layout_with_fr(net2) # algoritmo Fruchterman-Reingold
plot(net2,
     layout=l1,
     asp = 0,
     edge.curved=0.5) # Los resultados pueden ser distintos con cada ejecución
```

30.7. Visualización de comunidades de la mafia

511



Finalmente, se puede exportar el grafo como PDF usando la función `pdf()` de **R**.

```
pdf('grafo_final.pdf')
plot(net2, layout=11, asp = 0, edge.curved=0.5) # Los resultados puede ser distintos
#> con cada ejecucion
dev.off()
#> pdf
#> 2
```

Como se ha podido observar tras las acciones anteriores, en la red se aprecian siete distintas comunicades. Tres destacan por su importancia, lideradas por Gustavo Mango, Bino Lana y Pacifico Caliri. Bino Lana, en particular, tiene especial relevancia ya que actúa como un puente entre Gustavo Mango y Pacifico Caliri.

Capítulo 31

Optimización de inversiones publicitarias

Carlos Real Ugena

Deloitte

31.1. Metodologías para optimizar las inversiones publicitarias

Uno de los principales retos a los que se enfrentan los departamentos de marketing de cualquier compañía es cuantificar el impacto de la publicidad en el negocio. Esta medición es clave en la optimización de las inversiones que destinan a cada medio publicitario, existiendo múltiples métodos para medir el ROI (*Return On Investment*), es decir, el retorno que se obtiene por cada euro invertido en publicidad. Antes de revisar un ejemplo práctico, es necesario entender bien las características que presentan cada uno de ellos. Los métodos de cuantificación del impacto de la publicidad en el negocio, según la investigación llevada a cabo por la consultora Gartner¹, se pueden clasificar en cuatro grandes grupos:

\index{ROI (*Return On Investment*)}

1. **Marketing Mix Modeling (MMM)**: modelos de series temporales que sirven para estimar la contribución del marketing u otras variables explicativas a las ventas desde un punto de vista estratégico.
2. **Multitouch Attribution (MTA)**: modelos de atribución que valoran cada punto de contacto (*touchpoint*) del recorrido del cliente (*customer journey*), asignando a cada uno

¹ Artículo “Gartner Identifies Four Methods for Measuring Marketing’s Impact.” disponible en <https://www.gartner.com/en/newsroom/press-releases/2020-03-04-gartner-identifies-four-methods-for-measuring-marketing-s-impact>

un peso en la conversión (venta, descarga de folleto, etc). Son modelos tácticos que normalmente se centran en el canal online.

3. **Holdout Testing o Experiments (EXP)**: modelos causales que evalúan el impacto de una campaña publicitaria a partir de una muestra de control y otra de test.
4. **Unified Measurement Approaches (UMA)**: combinación de los modelos anteriores (MMM+MTA+EXP) con el objetivo de tener una visión unificada de los resultados.

En la comparativa (Fig. 31.1) que se muestra a continuación se resume el objetivo de cada uno de los modelos, así como las preguntas que permiten responder:

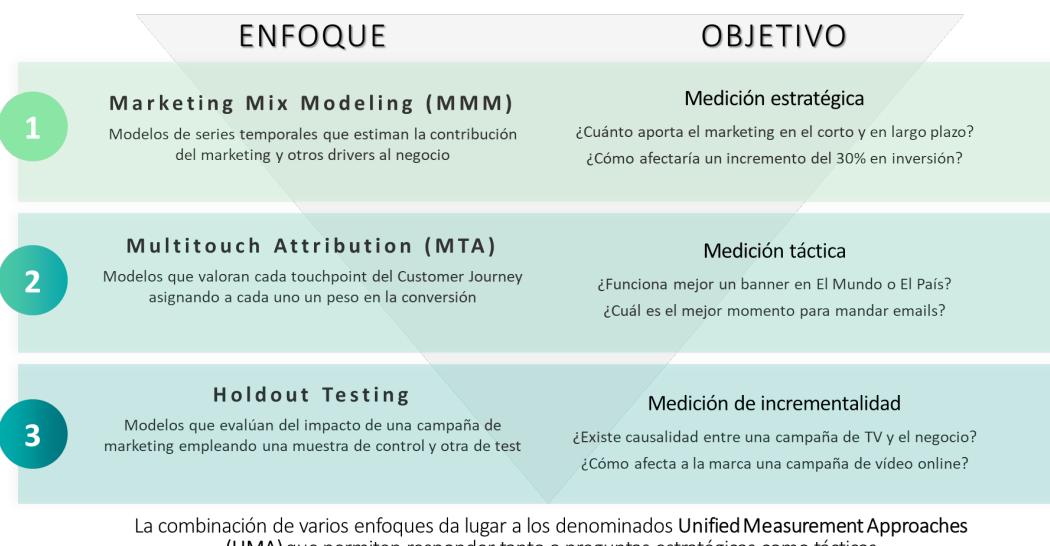


Figura 31.1: Comparativa de metodologías de medición

En los últimos años se han producido una serie de cambios en el entorno de la industria del marketing digital enfocados a garantizar el más estricto control de la privacidad de los usuarios. Desde el lanzamiento de la nueva ley de protección de datos “GDPR” en 2018, hasta la reciente confirmación por parte de Google de la prohibición de uso de cookies de tercera parte en “Chrome”, la posibilidad de acceder a datos de identificación personal para la medición y optimización del impacto de las campañas de publicidad digital está cada vez más limitada.

Esta situación está provocando que el primer tipo de enfoque, el MMM, esté siendo el gran beneficiado, dado que es una técnica que no depende del acceso a datos a nivel de individuo. El ejemplo más claro sobre el protagonismo que está alcanzando el MMM es que grandes compañías como Meta, Google y Uber están desarrollando soluciones *open-source* basadas en técnicas de *Machine Learning* que integran grandes avances y mejoras sobre las metodologías tradicionales. Entre las metodologías que han desarrollado, hay claras diferencias en las bases teóricas sobre las que se rigen, así como diferencias en tiempos de computación, lenguaje en el que se han

desarrollado o capacidades funcionales. Se detalla a continuación las principales características de cada una de ellas:

- **Robyn**²: desarrollado por Meta, Robyn ([Facebook Marketing Science \(2021\)](https://facebookexperimental.github.io/Robyn/)) está pensado para *datasets* con gran cantidad de variables independientes dado que trabaja con regresiones *ridge* (cubiertas en el Cap. 1), las cuales están pensadas para lidiar con problemas de multicolinealidad, muy presentes en este tipo de análisis. Cabe destacar que, entre las pruebas que realiza, ofrece una serie de *outputs* visuales avanzados que permiten al usuario seleccionar el que mejor se adapta al contexto de negocio y necesidad. Requiere tiempos de cómputo alto, pudiendo llegar hasta las tres horas, y permite hacer optimizaciones de presupuesto.
- **Lightweight**³: desarrollado por Google, sus fundamentos teóricos se basan en modelos bayesianos. La particularidad de esta solución es la posibilidad de incluir datos geográficos para su posterior segregación. Lightweight también considera distintos tipos de *adstock* (recuerdo publicitario), así como la tendencia y estacionalidad de la serie a explicar. El uso de esta herramienta es más sencillo, aunque los resultados son expuestos en un notebook y no posee *outputs* más elaborados como el resto de soluciones. Por último, también permite realizar optimizaciones de presupuesto.
- **Orbit**⁴: desarrollado por Uber, se basa en modelos bayesianos. Permite al usuario medir los retornos a lo largo del tiempo, lo cual hace que sea un modelo adecuado para compañías con grandes picos de ventas. Incluye análisis de estacionalidad mediante la descomposición de la serie en series de Fourier. Es el modelo más complejo de desarrollar, sin embargo, el tiempo de ejecución es bajo. Cabe destacar que es la librería más estable y, por lo tanto, se podría utilizar para realizar reportes con mayor frecuencia. No incluye la posibilidad de optimizar presupuestos.

31.2. Robyn como alternativa *open-source* en R

Robyn está ganando en popularidad debido a las mejoras continuas que están aplicando desde Meta, así como gracias a su adaptabilidad a la realidad del anunciante. Además, existe un paquete en **R** validado y subido al CRAN que lo sitúa como una gran alternativa de código abierto en **R** para iniciarse en el campo de la medición y optimización del retorno de las inversiones publicitarias. La fórmula empleada es la siguiente:

$$y_t = \beta_0 + S\text{Curve}(x_j) + \beta_{hol}hol_t + \beta_{seat}seat_t + \beta_{trend}trend_t + \dots + \beta_{ETC}ETC_t + \epsilon_t \quad (31.1)$$

donde:

²Dirección Web Robyn: <https://facebookexperimental.github.io/Robyn/>.

³Dirección Web Lightweight: https://github.com/google/lightweight_mmm.

⁴Dirección Web Orbit: <https://www.uber.com/en-ES/blog/orbit/>.

y_t = ventas en el instante t

t = instante de las variables (por ejemplo, semanas)

j = subíndice del medio (por ejemplo, TV o Display)

β_0 = intercepto

$x_{decay_{t,j}} = x_{t,j} + \theta_j x_{decay_{t,j-1}}$ (adstock, es decir, el recuerdo publicitario)

x_j = inversión publicitaria en cada medio j

θ_j = tasa fija de decrecimiento en cada medio j

$$SCurve(x, j) = \beta_j \times \frac{x_{decay_{t,j}}^\alpha}{x_{decay_{t,j}}^\alpha + \gamma^\alpha} \text{ transformación no lineal de curva en S} \quad (31.2)$$

α, γ = hiperparámetros que definen la S-Curve

γ : implementada en la SCurve, donde $\gamma_{tran} = cuantil(x_{decay_j}, \gamma)$

β_j = coeficientes de cada medio j

hol = festivos

sea = estacionalidad

$trend$ = tendencia

ETC = resto de variables independientes (precio, promociones, etc)

ϵ_t = término de error en el instante t

Las variables de medios no suelen presentar un efecto lineal sobre la variable de negocio que se modeliza, sino que la publicidad tiende a presentar rendimientos decrecientes no lineales. Para modelizar este efecto se utiliza la función biparamétrica de *S Curve* o curva de rendimientos decrecientes. Esta curva permite optimizar los repartos presupuestarios en todos los canales de medios, ya que su forma en “S” indica tanto el umbral a partir del cual los resultados del gasto presupuestario mejoran significativamente el objetivo (por ejemplo, ventas), como en qué punto está saturando y, por lo tanto, perdiendo eficacia.

A continuación, se aplica Robyn sobre un ejemplo con información simulada del sector hotelero, donde el objetivo es predecir el número de reservas de un hotel en función de una serie de predictores (entre ellos, las inversiones publicitarias). Toda la información sobre cómo aplicar esta metodología se puede consultar en Github⁵. La versión utilizada en este ejemplo práctico es la 3.6.3, disponible en el CRAN o descargable desde Github⁶. Este ejercicio traza el camino más corto que se puede seguir hasta llegar a los principales *outputs* y a la interpretación de los mismos. Sin embargo, para conocer en detalle la metodología, se recomienda seguir profundizando a través de la realización de pruebas adicionales más complejas.

Para comenzar, se carga el paquete Robyn, comprobando que se está trabajando con la versión 3.6.4 y forzando el uso de multicore al utilizar Rstudio:

⁵Metodología de Robyn disponible en <https://github.com/facebookexperimental/Robyn>

⁶Versión 3.6.3 de Robyn disponible en <https://github.com/facebookexperimental/Robyn/releases/tag/v3.6.3>

31.2. Robyn como alternativa open-source en R

517

```
library(Robyn)
packageVersion("Robyn")
Sys.setenv(R_FUTURE_FORK_ENABLE = "true")
options(future.fork.enable = TRUE)
```

Se carga una librería de Python llamada `nevergrad` (?), necesaria en el proceso de estimación de los hiperparámetros, invocándola desde R. Hay varias opciones para llevar a cabo este proceso, una de ellas es instalar primero el paquete `reticulate` (?) y, a continuación, `nevergrad` vía `pip`:

```
install.packages("reticulate")
library(reticulate)
virtualenv_create("r-reticulate")
use_virtualenv("r-reticulate", required = TRUE)
py_install("nevergrad", pip = TRUE)
py_config()
```

En caso de encontrar alguna dificultad al cargar `nevergrad`, existen distintas alternativas para su instalación que se pueden consultar en Github⁷.

Posteriormente, se carga la tabla que recoge la información simulada del sector hotelero con la que se medirá el impacto de la publicidad:

```
library('CDR')
data('hotel_tablonsemanal')
head(hotel_tablonsemanal)
```

Se detalla a continuación la información que contiene cada variable:

1. *semana*: lunes de la semana de referencia.
2. *reservas*: número de reservas que ha conseguido la cadena hotelera en cada una de las semanas bajo análisis.
3. *turismo*: número de pernoctaciones.
4. *covid_mov*: movilidad desde el comienzo de la pandemia.
5. *notoriedad*: conocimiento espontáneo de la marca a lo largo del tiempo.
6. *temperatura*: temperatura media.
7. *tv_grps20* y *tv_inv*: métrica de impactos (GRPs) e inversión realizada en TV.
8. *resto_off_inv*: resto de inversiones offline realizadas.
9. *paidsearch_imp* y *paidsearch_inv*: métrica de impactos (impresiones) e inversión realizada en Paid Search.
10. *display_imp* y *display_inv*: métrica de impactos (impresiones) e inversión realizada en Display.

⁷ Alternativas a la instalación de `nevergrad` disponibles en <https://github.com/facebookexperimental/Robyn/issues/189>.

11. *onlinevideo_imp* y *onlinevideo_inv*: métrica de impactos (impresiones) e inversión realizada en Online Video
12. *competidores*: inversión realizada por la competencia.
13. *eventos*: recoge eventos que tienen impacto en la serie de reservas. En este caso se considera el Black Friday.

El objetivo de este ejercicio es estimar el impacto de cada una de las variables detalladas sobre las reservas, poniendo especial foco en las variables de medios —TV, Resto Medios Offline, Paid Search, Display y Online Video— para medir sus efectos y, posteriormente, optimizar sus inversiones.

Con este fin, se cargan los festivos de una tabla auxiliar:

```
data('festivos')
head(festivos)
```

Se fija dónde se guardarán los resultados:

```
robyn_object <- "data/MyRobyn.RDS"
ruta_outputs <- "data"
```

Y se define la configuración inicial del modelo:

```
hotel_tablonsemanal$eventos <- hotel_tablonsemanal$eventos |> replace_na("na")
InputCollect <- robyn_inputs(
  dt_input = hotel_tablonsemanal,
  dt_holidays = festivos,
  date_var = "semana", # tiene que tener este formato "2020-01-01"
  dep_var = "reservas", # sólo una variable dependiente
  dep_var_type = "conversion", # "revenue" o "conversion". En nuestro caso son
  → reservas.
  prophet_vars = c("trend", "season", "holiday"), #
  → "trend=tendencia", "season=estacionalidad", "weekday=día de la semana" &
  → "holiday=festivos"
  prophet_signs = c("default", "default", "default"),
  prophet_country = "ES", # seleccione un país. España en nuestro caso
  context_vars = c("eventos", "turismo", "covid_mov", "notoriedad", "temperatura",
  → "competidores"), # variables de contexto que no sean medios
  context_signs = c("default", "positive", "negative", "positive", "default",
  → "negative"), # signos fijados a priori (por ejemplo, el turismo debe tener un
  → signo positivo puesto que se está analizando reservas de hotel)
  paid_media_spends = c("display_inv", "onlinevideo_inv", "paidsearch_inv",
  → "resto_off_inv", "tv_inv"), # variables de inversión
  paid_media_signs = c("positive", "positive", "positive", "positive", "positive"),
  paid_media_vars = c("display_imp", "onlinevideo_imp", "paidsearch_imp",
  → "resto_off_inv", "tv_grps20"), # variables de impacto si están disponibles. Si no
  → están disponibles utilice el coste como en el caso de Resto_off_inversion
  factor_vars = c("eventos"), # especifique variables que son factores. En nuestro
  → caso, sólo la variable de Eventos
```

31.2. Robyn como alternativa open-source en R

519

```
window_start = "2018-10-01", # fecha de inicio del modelo. En nuestro caso, octubre
  ↵ 2010 porque previamente no se tienen datos de reservas
window_end = "2021-09-27", # fecha fin del modelo
adstock = "geometric" # tipo de adstock. Seleccione el adstock geométrico para
  ↵ reducir tiempos de cómputo
)
print(InputCollect)
```

Después, se obtienen los nombres de los hiperparámetros a ajustar:

```
hyper_names(adstock = InputCollect$adstock, all_media = InputCollect$all_media)
```

A continuación, se definen los rangos en los que se moverán los hiperparámetros que definen la *S-curve* de cada medio. En este ejemplo se utilizan los mismos límites para α y γ , que controlan la forma de la curva y el punto de inflexión, respectivamente. Por otro lado, θ_j define el *adstock*, que se puede acotar teniendo en cuenta los intervalos recomendados por Meta: TV (entre 0,3 y 0,8), Resto Medios Offline (entre 0,1 y 0,4) y Digital (entre 0 y 0,3). En este caso, se deja θ libre entre 0 y 0,99 para todos los medios:

```
hyperparameters <- list(
  display_inv_alphas = c(0.0001, 3), display_inv_gammas = c(0.3, 1), display_inv_thetas
  ↵ = c(0, 0.99),
  onlinevideo_inv_alphas = c(0.0001, 3), onlinevideo_inv_gammas = c(0.3, 1),
  ↵ onlinevideo_inv_thetas = c(0, 0.99),
  paidsearch_inv_alphas = c(0.0001, 3), paidsearch_inv_gammas = c(0.3, 1),
  ↵ paidsearch_inv_thetas = c(0, 0.99),
  resto_off_inv_alphas = c(0.0001, 3), resto_off_inv_gammas = c(0.3, 1),
  ↵ resto_off_inv_thetas = c(0, 0.99),
  tv_inv_alphas = c(0.0001, 3), tv_inv_gammas = c(0.3, 1), tv_inv_thetas = c(0, 0.99)
)
```

Se añaden los hiperparámetros al *input* para ajustar los modelos:

```
InputCollect <- robyn_inputs(InputCollect = InputCollect, hyperparameters =
  ↵ hyperparameters)
print(InputCollect)
```

Y se ejecuta el modelo definiendo el número de iteraciones y *trials*. En este ejercicio, el número de iteraciones será 2000 y el de *trials* 10, obteniendo un total de 20000 posibles soluciones del modelo:

```
OutputModels <- robyn_run(
  InputCollect = InputCollect,
  iterations = 2000,
  trials = 10,
```

```

outputs = FALSE # se desactivan los outputs que se guardarán después
)
print(OutputModels)

```

Los mejores resultados de la modelización según el frente de Pareto (conjunto de óptimos de Pareto que minimizan el error cuadrático medio normalizado (*Normalized Root Mean Square Error*, NRMSE) y la descomposición de la suma cuadrática de la distancia⁸ (*Decomposition Root Sum of Squared Distance*, DECOMP.RSSD)) se guardan en la carpeta seleccionada:

```

OutputCollect <- robyn_outputs(
  InputCollect, OutputModels,
  pareto_fronts = 3,
  csv_out = "all",
  clusters = TRUE,
  plot_pareto = TRUE,
  plot_folder = ruta_outputs #ruta para exportar los resultados
)
print(OutputCollect)

```

Para finalizar, se revisan los distintos modelos obtenidos y el resumen gráfico proporcionado por Robyn. Se recomienda profundizar en la interpretación de uno de los modelos obtenidos con buen ajuste. A través del *one-pager* (Fig. 31.2), se puede consultar toda la información relativa al impacto de la publicidad, incluyendo métricas relativas a la bondad del ajuste como el coeficiente de determinación o R2 y el NRMSE en la parte superior:

Los principales *outputs* a evaluar de los modelos se visualizan en los gráficos del *one-pager*:

1. **Descomposición en cascada de la respuesta por predictor** (*Response Decomposition Waterfall by Predictor*): representa el peso de cada una de las variables en el modelo. En el ejemplo, el peso de la publicidad es del 54,5% (suma de los pesos de las variables de inversiones).
2. **Respuesta real vs. estimada** (*Actual vs. Predicted Response*): muestra el ajuste del modelo. En este caso, la línea del ajuste (azul) y la real (naranja) se aproximan bastante, indicando que el modelo es capaz de explicar la mayor parte de la variabilidad de la serie de reservas.
3. **Cuota de gasto frente a cuota de efecto con CPA total** (*Share of Spend vs Share of Effect with total CPA*): muestra la relación entre la cuota de inversión y la cuota de contribución generada por las inversiones publicitarias. En este ejercicio, se puede observar que tanto Online Video como Display obtienen un aporte mayor de lo esperado por su cuota de inversión, lo que hace que el Coste Por Adquisición o CPA (valores en azul) sean menores para estos medios. La TV se encuentra en el polo opuesto, con el máximo CPA, es decir, es el medio menos eficiente a la hora de generar reservas.

⁸Indicador clave de Robyn que representa la diferencia entre la cuota de gasto y la cuota de efecto para las variables de medios.

31.2. Robyn como alternativa open-source en R

521

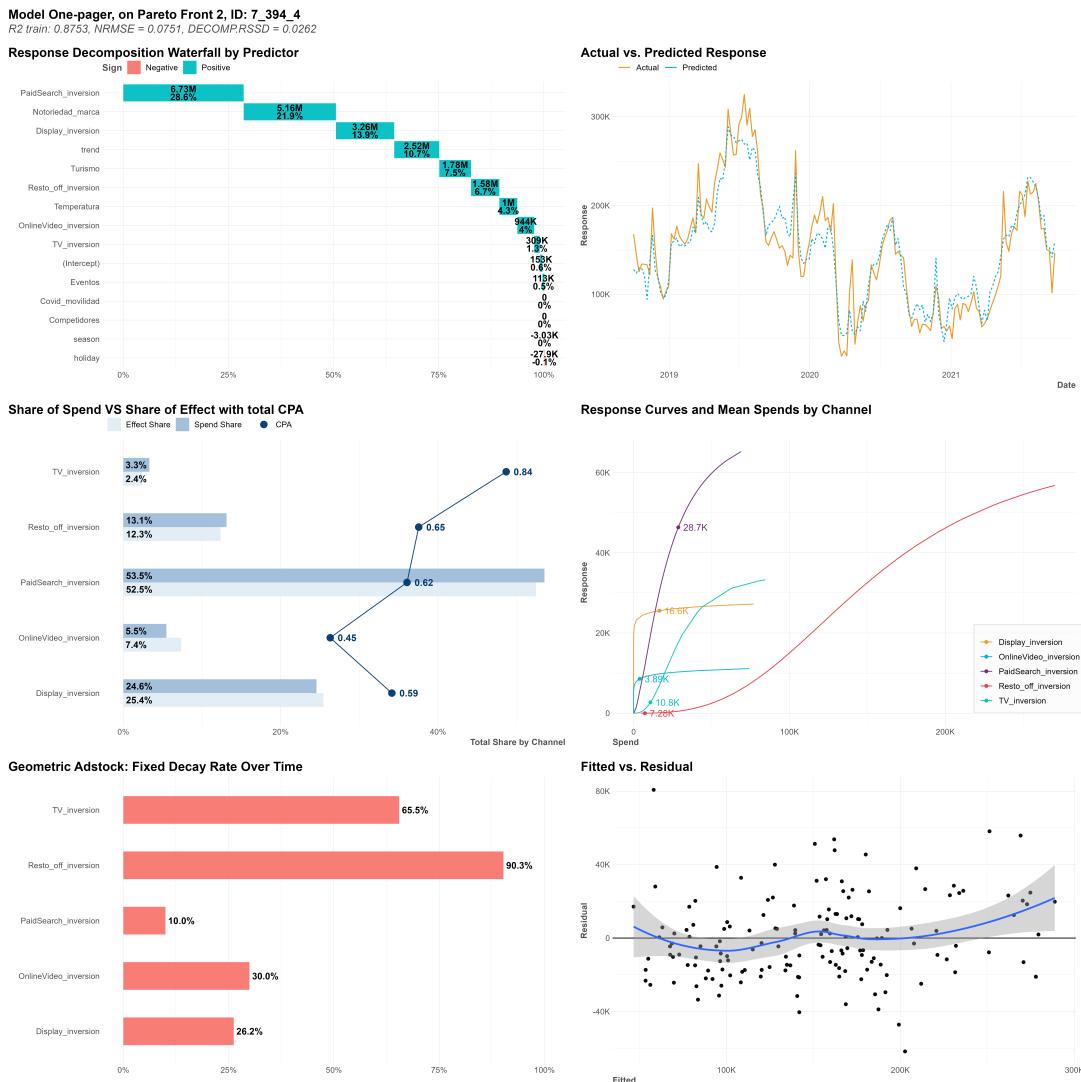


Figura 31.2: One-pager de resultados de Robyn

4. **Curvas de respuesta y gastos medios por canal** (*Response Curves and Mean Spends by Channel*): muestra la relación no lineal existente entre las inversiones y las reservas. Es el input principal a la hora de optimizar las inversiones.
5. **Adstock geométrico: Tasa constante de decrecimiento en el tiempo** (*Geometric Adstock: Fixed Decay Rate Over Time*): muestra el efecto recuerdo (*adstock*) de cada medio publicitario. Se puede observar que los medios offline (TV y Resto Medios Offline) son aquellos que presentan un efecto recuerdo más prolongado. No se debe olvidar que la configuración del modelo permite fijar el rango de valores que puede tomar el *adstock* en cada uno de los medios.
6. **Ajustados frente a residuos** (*Fitted vs. Residual*): muestra la nube de puntos para los datos ajustados (eje x) y los residuos (eje y). Este gráfico debe mostrar que los puntos están aleatoriamente ubicados alrededor del eje horizontal.

Y, ¿cómo se pueden optimizar las inversiones en medios empleando estos resultados? En primer lugar, se pueden utilizar las curvas obtenidas (parámetros guardados en el fichero *all_hyperparameters.csv*) para simular distintos escenarios y seleccionar el que genere un mayor número de reservas. La segunda opción es utilizar la documentación de código ⁹ incluida en el Step 5 para generar el reparto óptimo de inversión basado en un presupuesto pre-establecido.

En este capítulo se dan los primeros pasos en la medición y optimización del ROI de las inversiones publicitarias. Se anima al lector a que siga profundizando en este campo tan apasionante y complejo para basar las decisiones futuras de inversión en análisis desarrollados en **R**.

⁹Documentación de Robyn disponible en <https://github.com/facebookexperimental/Robyn/blob/main/demo/demo.R>

Capítulo 32

¿Cómo tuitea Elon Musk?

Mariluz Congosto

Universidad Carlos III de Madrid

32.1. Introducción

El objetivo de este caso de uso es arrojar luz, de manera objetiva, sobre un **fenómeno social** de interés general: el uso de Twitter por parte de **Elon Musk**. Este multimillonario adquirió la red social el 28 de octubre de 2022 y, desde entonces, ha tomado decisiones drásticas, como reducir la plantilla y lanzar nuevos servicios de pago de manera apresurada. Su constante cambio de rumbo queda reflejado en su actividad frenética en Twitter, donde es un usuario muy activo.

Este caso de estudio aborda, una vez descargados los *tweets* mediante las **APIs de Twitter**¹, cómo adaptarlos mediante minería de textos (20) para su **análisis y visualización**. Se representa el contenido de estos mensajes mediante nubes de palabras, *scatters plots* y *timelines*. Este conjunto de gráficas ofrecerán distintas vistas de los datos que, sin duda, ayudarán a comprender los cambios de comunicación desde que adquirió Twitter.

32.2. Análisis visual de los *tweets* de Elon Musk

Las librerías que se utilizan son las siguientes:

```
library("tidyverse")    # manejo de datos  
library("tidyverse")    # manejo de datos
```

¹El término “customer churn” se suele traducir como perdida de clientes o rotación de clientes. Se compone de las palabras inglesas “change” (en castellano cambio) y “turn” (en castellano abandonar)

```
library("lubridate")    # manejo de fechas
library("scales")        # manejo de escalas numéricas
library("tidytext")      # manejos de textos
library("ggwordcloud")   # creación de nube de palabras
library("RColorBrewer")   # paleta de colores
```

El rango de fechas elegido para delimitar temporalmente los tweets de Elon Musk va desde el 16-06-2022 hasta el 22-12-2022. Este rango es adecuado para visualizar la actividad e impacto del perfil de Musk antes y después de la adquisición de Twitter.

Se cargan los datos de la librería CDR del libro.

```
tweets_user <- CDR::elon_musk |>
  # Cambiar a formato fecha
  mutate(created_at = as.POSIXct(created_at, format = "%Y-%m-%dT%H:%M:%S", tz = "UTC"
  ↵ ))
```

Una vez obtenidos los datos, se les puede dar forma. Los datos incluyen fechas, textos, tipos de *tweets* y métricas que pueden ser representados mediante gráficos. A continuación, se definen unos parámetros generales a todas las gráficas: la fecha de la compra de Twitter, el color de los distintos tipos de mensajes (*original*, *quoted*, *reply*, *retweeted*).

```
# Fecha en la que Elon Musk compró Twitter
compra_twitter <- as.POSIXct("2022-10-28")

# Se ordena la leyenda del tipo de tweet
order_tipo_tweet <- c("original", "quoted", "reply", "retweeted")
tweets_user$tipo_tweet <- factor(tweets_user$tipo_tweet, levels = order_tipo_tweet)

# Se define el color de los elementos de la leyenda del tipo de tweet
my_color <- c("retweeted" = "purple", "reply" = "blue", "quoted" = "green", "original"
  ↵ = "red")
```

32.2.1. ¿Cuáles son los temas más recurrentes?

Para representar los términos más frecuentes en los tweets de Elon Musk se utiliza una nube de palabras. Esta representación gráfica se crea mediante la librería **ggwordcloud**, que funciona en el entorno **ggplot**.

El texto de los mensajes se encuentra en la variable **full_text**, la cual se limpia eliminando las URLs y los *handles* de los usuarios. Además, se añade una columna para distinguir los textos anteriores y posteriores a la compra de Twitter.

Posteriormente, se descomponen los textos en palabras independientes, se eliminan las *stop words* y se calcula la frecuencia de aparición de cada palabra.

32.2. Análisis visual de los tweets de Elon Musk

525

```

data(stop_words) # Descarga las stop words de la librería tidytext

corpus_text <- tweets_user |>
  mutate(text_plain = gsub("http\\S+\\s*", "", full_text)) |> # Quita las URL
  mutate(text_plain = gsub("RT @\\w+", "", text_plain)) |> # Quita los RTs
  mutate(text_plain = gsub("&", "&", text_plain)) |> # Rectifica el &
  mutate(text_plain = gsub("@\\w+", "", text_plain)) |> # Quita las menciones
  # Crea una columna para distinguir el periodo antes/después de la compra
  mutate(periodo = ifelse(created_at < compra_twitter,
    "Antes de la compra", "Después de la compra")) |>
  select(text_plain, periodo) |>
  unnest_tokens(word, text_plain) |> # Convierte las frases en un conjunto de palabras
  anti_join(stop_words) |> # Elimina las stop words
  group_by(word, periodo) |> # Agrupa por palabras
  summarise( freq = n(), .groups = "drop" ) |> # Calcula la frecuencia de cada palabra
  ungroup() |>
  arrange(desc(freq)) # Ordena de mayor a menor frecuencia de aparición

# print (corpus_text) # Descomentar para ver el resultado final

```

Para generar la nube de palabras de Elon Musk, se utiliza la función `geom_text_wordcloud_area()`. Esta función toma como entrada la lista de palabras y su frecuencia, y genera una comparación entre antes y después de la compra de Twitter (Fig. 32.1).

La proporción del tamaño de las palabras en la nube está en función de su frecuencia y se utiliza la librería `RColorBrewer` para definir la paleta de colores.

El resultado muestra que, antes de la compra de Twitter, Musk centraba su atención en sus empresas y en la guerra de Rusia-Ucrania. Sin embargo, tras la adquisición, su temática se relaciona con su nueva propiedad.

```

paleta <- brewer.pal(8, "Dark2")

ggplot() +
  geom_text_wordcloud_area(
    data = corpus_text |>
      top_n(300),
    aes(label = word, group = periodo, size = freq, color = freq),
    angle = 0.35
  ) +
  scale_size_area(max_size = 24) + # tamaño de las letras según frecuencia
  scale_color_gradientn(colors = paleta) +
  facet_wrap(~periodo) + # desdobra gráfica
  theme_minimal() +
  theme (strip.text = element_text(color = "grey50", size = 18))

```

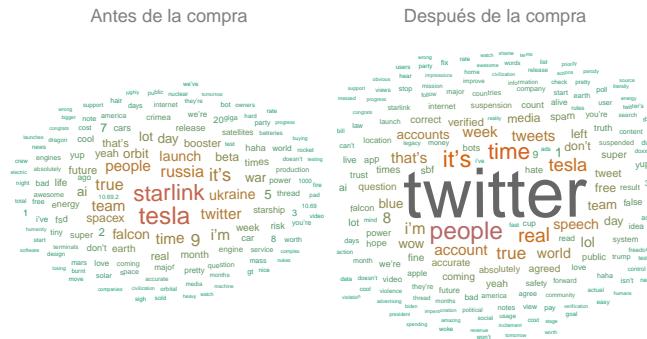


Figura 32.1: Palabras más frecuentes de Elon Musk en Twitter, antes y después de su compra

32.2.2. ¿Quiénes son los usuarios con los que más conversa?

Es posible visualizar con quiénes ha conversado Elon Musk con mayor frecuencia. Para ello, se pueden utilizar las respuestas que ha dado a otros usuarios en Twitter. Estas respuestas se obtienen de la variable `full_text`.

Para identificar con quiénes ha interactuado más Musk, se extraen los *handles* de los comentarios y se añade una columna para distinguir las menciones antes y después de la adquisición de Twitter. A continuación, se calcula la frecuencia de aparición de cada *handle*.

```

data(stop_words)
corpus_menciones <- tweets_user |>
  # Extrae los handles de los comentarios con una expresión regular "@\\w+"
  mutate(mentions = ifelse(tipo_tweet == "reply", str_extract(full_text, "@\\w+"), NA))
  |>
  # Crea una columna para distinguir el periodo antes/después de la compra
  mutate(periodo = ifelse(created_at < compra_twitter,
    "Antes de la compra", "Después de la compra")) |>
  filter(!is.na(mentions)) |>      # elimina las filas vacías
  select(mentions, periodo) |>    # selecciona menciones y periodo
  group_by(mentions, periodo) |>
  summarise( freq = n(), .groups = "drop" ) |> # calcula freq. de palabra
  ungroup() |>
  arrange(desc(freq)) # ordena de mayor a menor freq. de aparición

# print (corpus_menciones) # Descomentar para ver el resultado final

```

Una vez que los datos han sido procesados, se utiliza la función `geom_text_wordcloud_area()` para generar la nube de palabras correspondiente a las menciones en los tweets de Elon Musk.

Para ello, se toma la lista de menciones y su frecuencia, y se utiliza la misma operación que se realizó con la nube de palabras anterior.

El resultado (Fig. 32.2) muestra que algunos interlocutores se mantienen, otros pierden protagonismo y aparecen otros nuevos. Se mantienen @BillyM2k (comediante) y @WholeMars-Blog (relacionado con temas de Marte). Pierden protagonismo @teslaownersSVm, @EvaFoxU, @PPathole y @Teslarati (relacionados con Tesla). Ganan protagonismo @stillgray (*influencer*), @micsolana (capital riesgo) y @Jason (emprendedor).

```
paleta <- brewer.pal(8, "Dark2")
ggplot() +
  geom_text_wordcloud_area( # dibuja la nube de palabras
    data = corpus_menciones |> top_n(50),
    aes(label = mentions, size = freq, color = freq), angle = 0.35
  ) +
  scale_size_area(max_size = 12) +
  scale_color_gradientn(colors = paleta) +
  facet_wrap(~periodo) +
  theme_minimal() +
  theme (strip.text = element_text(color = "grey50", size = 18))
```



Figura 32.2: Usuarios con los que dialoga Elon Musk antes y después de la compra de Twitter

32.2.3. ¿Cuál es su rutina de publicación?

Para analizar la distribución horaria de los tweets de Elon Musk, se examina la frecuencia de publicación de *tweets* cada hora de cada día. Dado que la residencia declarada de Musk es Austin (Texas), se ajustará la hora de los tweets al huso horario de esta ciudad, ya que la hora proporcionada por Twitter está en GMT.

Debido a que los datos abarcan un período largo, desde junio hasta diciembre, se acotarán a 15 días antes y después de la compra de Twitter. Es importante tener en cuenta que la fecha de creación de los tweets (`created_at`) se presenta en formato fecha-hora, y cada día consta de 86.400 segundos (60 segundos * 60 minutos * 24 horas).

```

tweets_user_hour <- tweets_user |>
  # cambia al huso horario de Texas
  mutate(created_at = lubridate::with_tz(created_at, "US/Central")) |>
  # filtra los tweets anteriores a la compra de Twitter
  filter(created_at >= (compra_twitter - (60 * 60 * 24 * 15))) |>
  filter(created_at <= (compra_twitter + (60 * 60 * 24 * 15))) |>
  # crea una nueva columna para la fecha
  mutate(time_in_days = as.POSIXct(floor_date(created_at, "day"))) |>
  # crea una nueva columna para la hora
  mutate(hour_tweet = hour(created_at)) |>
  # agrupa el número de tweets por tipo y hora
  group_by(time_in_days, hour_tweet, tipo_tweet) |>
  # calcula el número de tweets por día, hora y tipo
  summarise( num_tweets = n(), .groups = "drop" ) |>
  ungroup()

# print (tweets_user_hour) # Descomentar para ver el resultado final

```

A continuación, se recalcan los días de la semana que son festivos en color rojo para apreciar si hay distinta rutina.

```

festivos <- tweets_user |>
  # cambia al huso horario de Texas
  mutate(created_at = lubridate::with_tz(created_at, "US/Central")) |>
  # filtra los tweets anteriores a la compra de Twitter
  filter(created_at >= (compra_twitter - (60 * 60 * 24 * 15))) |>
  filter(created_at <= (compra_twitter + (60 * 60 * 24 * 15))) |>
  # crea una columna con el tiempo en días
  mutate(time_in_days = floor_date(created_at, "1 day")) |>
  # agrupa por día
  group_by(time_in_days) |>
  # calcula el número de tweets por día
  summarise( num_tweets = n(), .groups = "drop" ) |>
  ungroup() |>
  # crea una columna con el día de la semana
  mutate(week_day = wday(time_in_days)) |>
  # crea una columna para colorear los días según sean festivos o no
  mutate(festivo = ifelse(wday(time_in_days) == 7 |
    (wday(time_in_days) == 1), "red", "black"))

# print (festivos) # Descomentar para ver el resultado final

```

Finalmente, se representa un gráfico de dispersión (*scatter plot*) con las coordenadas de las horas del día (eje X) y los días seleccionados (eje Y), utilizando la función `geom_point()`. El tamaño del punto es proporcional al número de *tweets* en esa hora y día, y el color el tipo de tweet (*original, reply, quoted* y *retweeted*). Se marca una línea horizontal con la función `geom_hline()` en la fecha de compra de Twitter y se crea un eje X doble para que sea más fácil ver las horas debido a la altura de la gráfica.

32.2. Análisis visual de los tweets de Elon Musk

529

La Fig. 32.3 muestra que no hay una rutina clara en la publicación de tweets de Elon Musk. Esto podría deberse a que viaja mucho. La mayoría de sus mensajes son comentarios y han aumentado considerablemente desde la compra de Twitter. El máximo número de *tweets* por hora fue 10.

```
ggplot() +
  geom_point(
    data = tweets_user_hour,
    aes(
      x = hour_tweet,
      y = time_in_days,
      size = num_tweets,
      color = tipo_tweet
    ),
    alpha = 0.5
  ) +
  # separa las fechas antes y después de la compra
  geom_hline(aes(yintercept = compra_twitter), linetype = 2) +
  # define una etiqueta de tiempo por día
  scale_y_datetime(
    date_labels = "%d-%b-%y(%a)", # formato fecha (día semana abreviado)
    date_breaks = "1 day", # una marca de tiempo cada día
    expand = c(0, 0, 0.02, 0.02)
  ) + # ajustes de márgenes
  # define una etiqueta para cada hora
  scale_x_continuous(
    breaks = seq(0, 23, 1), # crea un vector de 0 a 23
    sec.axis = dup_axis() # duplica el eje X
  ) +
  labs( x = "", y = "", color = "", size = "N. tweets") +
  # ajusta las leyendas en dos filas para que no se trunquen
  guides(color = guide_legend(nrow = 2, override.aes = list(size = 4))) +
  theme_minimal() +
  # indica la posición de la leyenda y el color de las fechas
  theme(
    panel.grid.major.x = element_line(),
    legend.position = "top",
    axis.text.y = element_text(colour = festivos$festivo)
  )
```

32.2.4. ¿Cuál es su *timeline* de publicación?

Ahora se analiza cómo se distribuyen los *tweets* en el tiempo por tipo de tweet. Se resaltará la fecha de compra de Twitter con una anotación para facilitar la comparación de la frecuencia anterior y posterior a esta fecha.

Se crea una columna con la fecha redondeada a días, se agrupan los *tweets* por fecha y el tipo de tweet y se calcula su número para cada día.

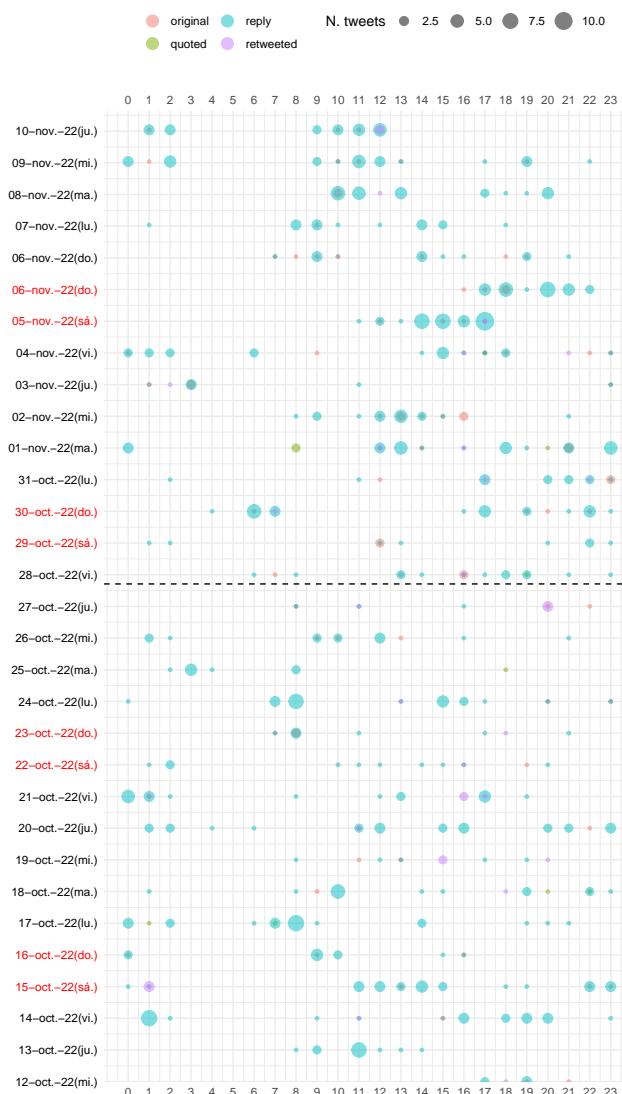


Figura 32.3: Rutina de publicación de Elon Musk. (huso horario de Texas)

32.2. Análisis visual de los tweets de Elon Musk

531

```

tweets_user_day <- tweets_user |>
  # Creamos una columna con el tiempo en días
  mutate(time_in_days = floor_date(created_at, "1 day")) |>
  # Agrupamos el número de tweets por día y tipo
  group_by(time_in_days, tipo_tweet) |>
  # calculamos el número de tweets por día y tipo
  summarise( num_tweets = n(), .groups = "drop" ) |>
  ungroup()

# print (tweets_user_day) # descomentar para ver el resultado

```

En la Fig. 32.4 se puede observar un incremento en el número de publicaciones después de la compra de Twitter. De hecho, se publicaron casi el doble de tweets en comparación con el periodo anterior a la adquisición de la plataforma. Asimismo, se puede ver, al igual que en la Fig. 32.3, que la mayoría de los *tweets* de Elon Musk fueron comentarios.

```

ggplot(data = tweets_user_day) +
  geom_col(aes(x = time_in_days, y = num_tweets, fill = tipo_tweet), alpha = 0.7) +
  geom_vline(aes(xintercept = compra_twitter), linetype = 2) + # compra de Twitter
  geom_label( # señala el evento
    aes(
      x = compra_twitter - (60 * 60 * 24 * 25),
      y = max(num_tweets),
      label = "Elon Musk\ncompra Twitter"
    ),
    color = "gray45"
  ) +
  geom_curve( # flecha con curva para señalar el evento
    aes(
      x = compra_twitter - (60 * 60 * 24 * 10),
      y = max(num_tweets),
      xend = compra_twitter,
      yend = max(num_tweets) * 0.80
    ),
    arrow = arrow(length = unit(0.08, "inch")), linewidth = 0.5,
    color = "gray20", curvature = -0.3
  ) +
  scale_x_datetime( # ajusta la escala de tiempo y su formato
    date_labels = "%d\n%b",
    date_breaks = "2 week"
  ) +
  scale_y_continuous( # ajusta el formato del eje Y
    name = "Num. Tweets por día",
    labels = label_number(scale_cut = cut_short_scale())
  ) +
  scale_color_manual(values = my_color) + # aplica colores definidos
  labs( x = "", y = "Num. Tweets por día", fill = "") +
  theme_minimal() +
  theme(legend.position = "top")

```

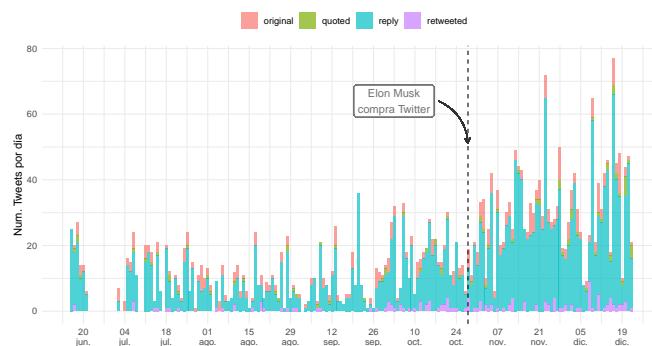


Figura 32.4: Publicación de Tweets por día de Elon Musk

32.2.5. ¿Cuál es el impacto de sus tweets?

Para comparar los *tweets* propios publicados (sin *retweets*) y el impacto que reciben (*retweets* recibidos), se utilizará una gráfica de doble escala. Dado que ambas variables tienen diferentes órdenes de magnitud, este tipo de gráfica permitirá una mejor comparación. Además, se incluirá una anotación con la fecha de compra de Twitter para distinguir los cambios antes y después de este evento.

En esta gráfica se podrá ver cómo se van superponiendo capas de dibujo.

Se preparan los datos en dos `data.frames` y se calcula la relación de escala:

- `tweets_propios_day` con los *tweets* propios por día y los mensajes originales/hora:

```
tweets_propios_day <- tweets_user |>
  # crea una columna con el tiempo en días
  mutate(time_in_days = floor_date(created_at, "1 day")) |>
  filter(tipo_tweet != "RT") |>  # elimina los retweets
  group_by(time_in_days) |>  # agrupa los tweets por día
  summarise( num_tweets = n(), .groups = "drop" ) |>
  ungroup()

# print (tweets_propios_day) # Descomentar para ver el resultado
```

- `tweets_RT_day` con los *retweets* recibidos por día:

32.2. Análisis visual de los tweets de Elon Musk

533

```

tweets_RT_day <- tweets_user |>
  # crea una columna con el tiempo en días
  mutate(time_in_days = floor_date(created_at, "1 day")) |>
  filter(tipo_tweet != "RT") |>
  group_by(time_in_days, tipo_tweet) |>
  summarise( num_tweets = sum(retweet_count), .groups = "drop" ) |>
  ungroup()

# print(tweets_RT_day) # descomentar para ver el resultado

```

- Se calculan las escalas:

```

# máximo número de tweets propios
max_tweets <- max(tweets_propios_day$num_tweets, na.rm = TRUE)
# máximo número de retweets recibidos
max_RT <- max(tweets_RT_day$num_tweets, na.rm = TRUE)
ajuste_escala <- max_RT / max_tweets # Ajuste de escala
# print (ajuste_escala) # descomentar para ver el ajuste
my_color <- c("Num. original tweets" = "steelblue4", "RTs" = "red4")

```

La Fig. 32.5 muestra un incremento masivo de los *retweets* recibidos desde la compra de Twitter, siendo el día que tomó posesión, el que generó el mayor pico: 800K RTs.

```

ggplot() +
  # pinta la evolución de los tweets propios/día
  # Pinta el área que representa los tweets propios por día
  geom_area(
    data = tweets_propios_day,
    aes(x = time_in_days, y = num_tweets), fill = "steelblue4",
    alpha = 0.5
  ) +
  # pinta el borde del área por estética
  geom_line( data = tweets_propios_day,
    aes(x = time_in_days, y = num_tweets, color = "Num. original tweets")
  ) +
  # pinta la evolución de los RTs/día
  geom_line(
    data = tweets_RT_day,
    aes(x = time_in_days, y = num_tweets / ajuste_escala, color = "RTs")
  ) +
  # marca la linea de la compra de Twitter por Elon Musk
  geom_vline(aes(xintercept = compra_twitter), linetype = 2) +
  # anota el evento
  geom_label(
    data = tweets_propios_day,
    aes(
      x = compra_twitter - (60 * 60 * 24 * 25),

```

```

    y = max(num_tweets) * .95,
    label = "Elon Musk\ncompra Twitter"),
    color = "grey50"
) +
# dibuja una flecha con curva para señalar el evento
geom_curve(
  data = tweets_propios_day,
  aes(
    x = compra_twitter - (60 * 60 * 24 * 10),
    y = max(num_tweets),
    xend = compra_twitter,
    yend = max(num_tweets) * 0.90
  ),
  arrow = arrow(length = unit(0.08, "inch")), linewidth = 0.5,
  color = "gray20", curvature = -0.3
) +
# ajusta la escala de tiempo y su formato
scale_x_datetime(
  date_labels = "%d\n%b",
  date_breaks = "2 week"
) +
# doble escala, derecha: tweets propios, izquierda: retweets
scale_y_continuous(
  name = "Num. Original tweets por día",
  labels = label_number(scale_cut = cut_short_scale()),
  sec.axis = sec_axis(
    trans = (~ .. * ajuste_escala), name = "RTs por día",
    labels = label_number(scale_cut = cut_short_scale())
  )
) +
scale_color_manual(values = my_color) +
labs(x = "", color = "") +
theme_minimal(base_family = "sans") +
theme(
  legend.position = "top",
  axis.title.y = element_text(color = "steelblue4", size = 12),
  axis.title.y.right = element_text(color = "red4", size = 12),
  axis.text.y = element_text(color = "steelblue4"),
  axis.text.y.right = element_text(color = "red4")
)

```

32.2. Análisis visual de los tweets de Elon Musk

535

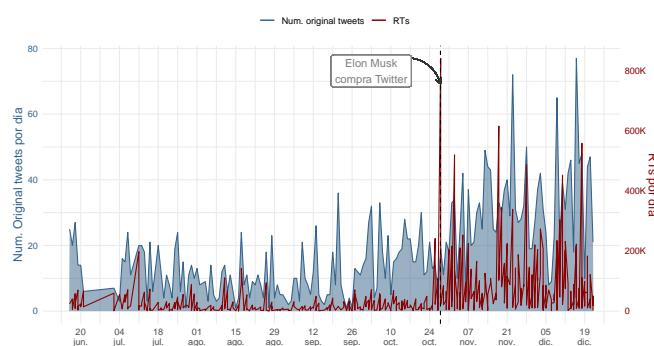


Figura 32.5: Tweets vs. retweets de Elon Musk

Capítulo 33

Análisis electoral: de Rstudio a su periódico

Borja Andrino Turón

EL PAÍS

33.1. Motivación

El uso de **R** en el entorno profesional ha llegado también a los periódicos. Cada vez es más habitual encontrar en los medios analistas de datos que lo utilizan en su día a día. En EL PAÍS, muchos de los contenidos que se publican en la Unidad de Datos surgen de un notebook de RStudio. A continuación, se muestra un análisis sobre las últimas elecciones andaluzas, de RStudio a su periódico favorito.

33.2. Obtención de los datos

Los datos electorales no siempre son igual de accesibles. Los de las elecciones que dependen del Ministerio del Interior se publican en el portal [Infoelectoral](#). En el caso de las elecciones andaluzas, los resultados a nivel de mesa se han publicado en los portales de cada convocatoria, aunque pueden encontrarse entre los contenidos del libro para replicar estos análisis.

En primer lugar se compondrá un diccionario de municipios que se usará para filtrar y agrupar los resultados por provincia. Primero se escraperá de la web del INE la relación de códigos de provincia con la librería rvest. Se lee el código html de la página y se buscan los elementos table con clase miTabla. A continuación, se usa la función html_table para convertir las tres tablas en un objeto tibble. La información con los nombres de municipios y provincias se leerá en la web del INE.

```

pacman::p_load(CDR, ggplot2, dplyr, rvest, lubridate, sf, ggtext,
  rio, janitor, here, purrr, stringr, scales)

url_provincias <-
  "https://www.ine.es/daco/daco42/codmun/cod_provincia.htm"

cod_provincias <-
  read_html(url_provincias) |>
  html_nodes("table.miTabla") |>
  html_table() |>
  map_df(as_tibble) |>
  rename(codigo_prov = 1, name_prov = 2) |>
  mutate(codigo_prov = str_pad(codigo_prov, width = 2, pad = "0", side = "left"))

url_municipios <-
  "https://www.ine.es/daco/daco42/codmun/codmun20/20codmun.xlsx"

cod_municipios <-
  import(url_municipios, skip = 1) |>
  clean_names() |>
  transmute(codigo_prov = cpro,
            codigo_mun = str_glue("{codigo_prov}{cmun}"),
            name_mun = nombre) |>
  left_join(cod_provincias) |>
  select(codigo_mun, name_mun, name_prov)

```

Se añade la información sobre municipios al dataset de elecciones que tiene los datos de cada sección censal.

```

datos_elecciones <-
  datos_elecciones |>
  left_join(cod_municipios) |>
  select(codigo_secc, codigo_mun, name_mun, name_prov, convocatoria, everything())

```

33.3. Transformación y primeros gráficos

En el primer gráfico se mostrará la evolución de los votos a partidos de izquierda y de derecha en toda Andalucía desde 2015. Primero se calculan los votos válidos en cada convocatoria. Como en la estructura de datos ese dato está repetido para cada combinación de convocatoria-sección-partido se usará la función `distinct` antes de agrupar y sumar los votos validos de todas las secciones.

```

datos_elecciones_validos_total <-
  datos_elecciones |>
  distinct(convocatoria, codigo_secc, .keep_all = T) |>

```

33.3. Transformación y primeros gráficos

539

```

mutate(region = "Andalucía") |>
group_by(convocatoria, region) |>
summarise(validos = sum(validos), .groups = "drop")

datos_elecciones_validos_provs <-
  datos_elecciones |>
  distinct(convocatoria, codigo_secc, .keep_all = T) |>
  mutate(region = name_prov) |>
  group_by(convocatoria, region) |>
  summarise(validos = sum(validos), .groups = "drop")

datos_elecciones_validos <-
  datos_elecciones_validos_total |>
  bind_rows(datos_elecciones_validos_provs)

```

Ahora se calcula la suma de votos de cada bloque en cada convocatoria. En este caso, como cada fila tiene el dato de votos de un partido distinto no es necesaria la función `distinct`.

```

datos_bloques_total <-
  datos_elecciones |>
  mutate(region = "Andalucía") |>
  group_by(convocatoria, region, bloque) |>
  summarise(votos_bloque = sum(votos_partido), .groups = "drop")

datos_bloques_provs <-
  datos_elecciones |>
  mutate(region = name_prov) |>
  group_by(convocatoria, region, bloque) |>
  summarise(votos_bloque = sum(votos_partido), .groups = "drop")

datos_bloques <-
  datos_bloques_total |>
  bind_rows(datos_bloques_provs) |>
  left_join(datos_elecciones_validos) |>
  mutate(votos_bloque_pc = votos_bloque / validos)

```

A continuación, se realiza el gráfico con los datos que se han calculado antes. Se definen los colores que representan a cada bloque, las fechas para poder etiquetar en el gráfico los ticks del eje x y se programa el gráfico.

```

title <-
  "Evolución de voto a partidos de <b style='color:#457b9d;'>derecha</b>, <b
  ↵  style='color:#e63946;'>izquierda</b><br>y <b style='color:#676767;'>otros</b>
  ↵  desde 2015"

convocatorias_dates <-
  datos_bloques |>

```

```

distinct(convocatoria) |>
  pull(convocatoria)

datos_bloques |>
  filter(region == "Andalucía") |>
  ggplot(aes(x = convocatoria, y = votos_bloque_pc,
             color = bloque, group = bloque)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_hline(yintercept = 0, width = 0.2) +
  scale_color_manual(values = colors_bloques) +
  scale_x_date(breaks = convocatorias_dates, date_labels = "%Y") +
  scale_y_continuous(labels = percent) +
  labs(title = title, x = "Convocatoria", y = "Votos bloque",
       caption = "Fuente: Junta de Andalucía") +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_markdown(margin=margin(0,0,30,0)))

```

Evolución de voto a partidos de **derecha** **izquierda**
y otros desde 2015

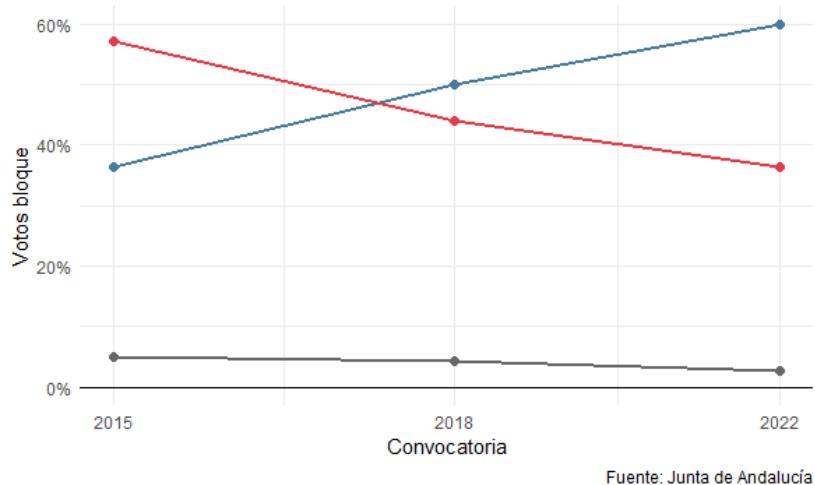


Figura 33.1: Evolución del voto en Andalucía

Replicar la Fig. 33.1 para cada provincia no es complicado. Sólo se descartarán los datos de toda Andalucía y se usará la función `facet_wrap()` que realizará el mismo gráfico con el mismo estilo para cada provincia.

33.3. Transformación y primeros gráficos

541

```
datos_bloques |>
  filter(region != "Andalucía") |>
  ggplot(aes(x = convocatoria, y = votos_bloque_pc,
             color = bloque, group = bloque)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_hline(yintercept = 0, width = 0.2) +
  scale_color_manual(values = colors_bloques) +
  scale_x_date(breaks = convocatorias_dates, date_labels = "%Y") +
  scale_y_continuous(labels = percent) +
  labs(title = title, x = "Convocatoria", y = "Votos bloque",
       caption = "Fuente: Junta de Andalucía") +
  facet_wrap(~region, ncol = 4) +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_markdown(margin=margin(0,0,10,0)))
```

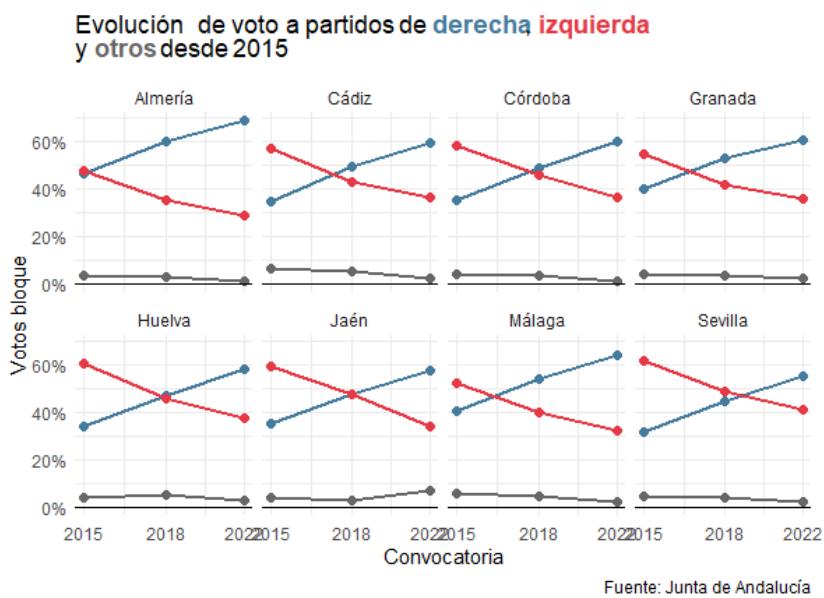


Figura 33.2: Evolución del voto provincial

La Fig. 33.2 cuenta una historia complementaria al primero. El giro no se ha producido igual en toda Andalucía, no es igual el de Almería que el de Sevilla. Para intentar buscar nuevas diferencias territoriales se explorarán los mapas de ganadores a nivel municipal. Se procede de igual manera que con los datos de provincias, salvo que en este caso se agrega a partir de la columna `codigo_mun`. Para calcular el ganador se agrupa por esta columna y se usa la función `slice_max()`, que tomará para cada municipio la fila del partido con el mayor número de votos.

```

datos_elecciones_validos_muns <-
  datos_elecciones |>
  distinct(convocatoria, codigo_secc, .keep_all = T) |>
  group_by(convocatoria, codigo_mun) |>
  summarise(validos = sum(validos),
            .groups = "drop")

datos_bloques_muns <-
  datos_elecciones |>
  group_by(convocatoria, codigo_mun, bloque) |>
  summarise(votos_bloque = sum(votos_partido), .groups = "drop") |>
  left_join(datos_elecciones_validos_muns) |>
  mutate(votos_bloque_pc = votos_bloque / validos, 1)

# Ahora calculamos los ganadores
datos_winners_muns <-
  datos_bloques_muns |>
  group_by(convocatoria, codigo_mun) |>
  slice_max(votos_bloque, n = 1, with_ties = F) |>
  select(convocatoria, codigo_mun,
         winner = bloque, votos_bloque_pc)

```

Para realizar el gráfico se tomará el objeto `sf` con los recintos de los municipios andaluces y se les añadirá los datos de ganadores calculados anteriormente con la función `left_join()`. Se usa el color del bloque para el relleno y el porcentaje de votos que suma el bloque ganador para la transparencia, de forma que de un vistazo se pueden encontrar feudos de uno u otro bloque.

```

map_munis |>
  left_join(datos_winners_muns) |>
  mutate(convocatoria = year(convocatoria)) |>
  ggplot() +
  geom_sf(aes(fill = winner, alpha = votos_bloque_pc),
         size = 0.01) +
  scale_fill_manual(values = colors_bloques) +
  facet_wrap(~convocatoria, ncol = 1) +
  labs(title = title,
       caption = "Fuente: Junta de Andalucía") +
  coord_sf(label_graticule = "", ndiscr=0) +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_markdown(margin=margin(0,0,10,0)))

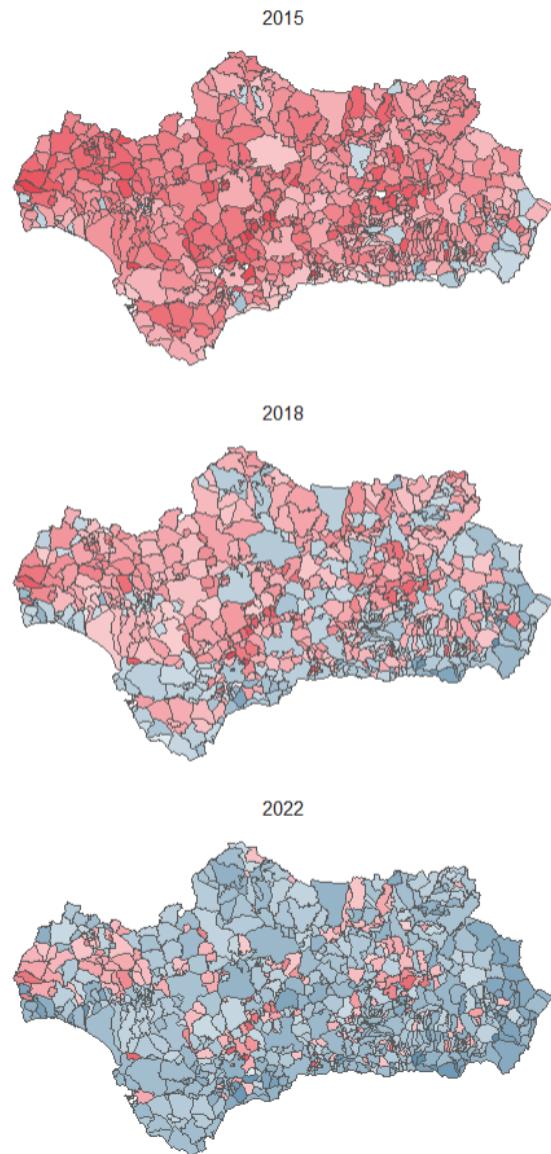
```

En los mapas se encuentran nuevas historias. En 2015 la derecha era fuerte en la costa de Almería y Málaga. Su presencia creció en 2018, aunque la izquierdas seguía ganando el interior de la comunidad. En 2018 el dominio del bloque de derechas se extiende por casi todo el territorio, en especial en las zonas donde ya era fuerte en 2015.

33.3. Transformación y primeros gráficos

543

Evolución de voto a partidos de **derecha, izquierda** y otros desde 2015



Fuente: Junta de Andalucía

Figura 33.3: Resultados de las elecciones andaluzas

Capítulo 34

Crisis: impacto en el paro de Castilla-La Mancha

Isidro Hidalgo Arellano^a y Ángel Jiménez Rojas^b

^{a,b}Observatorio del Mercado de Trabajo de Castilla-La Mancha

34.1. Planteamiento

En los últimos 15 años el mundo ha sufrido dos grandes períodos de **crisis económica**: en **2008**, de tipo financiero; y en **2020**, a causa de la pandemia de **COVID-19**. Uno de los parámetros socioeconómicos que se ven más afectados por este tipo de procesos es el paro registrado. El paro registrado se define como el conjunto de los demandantes inscritos en las oficinas de empleo, una vez excluidos los inscritos sin disponibilidad para trabajar y los demandantes no parados, tales como estudiantes, desempleados en formación, etc. (Toharia, 2012). **Castilla-La Mancha**, comunidad autónoma interior de España, no ha sido ajena a las crisis económicas mencionadas, por lo que en este trabajo se quiere analizar el impacto de las mismas en la estructura del **paro registrado** de la región. Para ello, se utilizan las siguientes variables explicativas: **sexo** y **edad** de la persona desempleada, **sector de actividad económica de procedencia** y **tiempo de búsqueda de empleo**. El conjunto de datos utilizado comprende la **media anual del paro registrado en la comunidad autónoma de Castilla-La Mancha** desagregado según estas variables, a lo largo de los años que van desde 2007 a 2022.

Para el análisis se usan las librerías y objetos (paletas de colores para los gráficos) siguientes:

```
library(CDR)
library(tidyverse)
library(ggpubr)
paleta_heatmaps <- c(rgb(.7,.1,.0,.5),   rgb(.13,.22,.58,1))
paleta_lineas <- c("blue4", "orange", "darkgreen")
```

Para cargar el conjunto de datos, `parados_clm`, incluido en el paquete CDR, y mostrar la estructura de la `tibble` se usa:

```
data("parados_clm")
parados_clm
# A tibble: 92,215 × 8
#   anyo    sexo    edad sector t_bus_e    tramo_edad t_bus_e_agr parados
# <dbl> <fct> <dbl> <fct> <dbl> <fct> <dbl> <dbl>
# 2007 hombre 16 agricu t<=7 días <30 años t<=6 meses 0.66666667
# 2018 mujer 36 sinact t<=7 días 30-44 años t<=6 meses 1.66666667
# 2012 mujer 30 agricu t<=7 días 30-44 años t<=6 meses 5.33333333
# 2022 mujer 49 constr t<=7 días >44 años t<=6 meses 0.75000000
# 2007 mujer 54 indust t<=7 días >44 años t<=6 meses 1.50000000
# ... with 92,210 more rows
```

34.2. Evolución del paro medio anual en Castilla-La Mancha

Para ver el paro medio anual en función del tiempo, se construye un gráfico de evolución. Para ello, se representa el paro medio por año, marcando los años que suponen un máximo o mínimo en la serie:

```
resumen <- parados_clm |>
  group_by(anyo) |>
  summarise(parados = sum(parados)) |>
  mutate(anyo = as.numeric(as.character(anyo)))
anyos <- c(2007, 2013, 2019, 2020, 2022)
paro_anyos <- resumen |>
  filter(anyo %in% anyos) |>
  select(parados) |>
  mutate(parados = round(parados, 0))
puntos <- data.frame(anyos, paro_anyos)

graf <- ggplot(resumen, aes(anyo, parados)) +
  geom_line(linewidth = 2, col = paleta_lineas[1], alpha = 0.5) +
  xlab("") + ylab("número de parados") +
  geom_point(puntos, mapping = aes(x = anyos, y = parados,
    shape = "circle filled", size = 1, fill = paleta_lineas[1],
    alpha = 0.5)) +
  theme(legend.position = "none",
    axis.title = element_text(face="bold", size = 10),
    axis.text = element_text(face="bold", size = 10),
    strip.text = element_text(size = 9, face = "bold")) +
  scale_y_continuous(labels = function(x) format(x, big.mark = ".",
    scientific = FALSE))
graf
```

34.3. Evolución del paro medio anual en función de la edad y el sexo

547

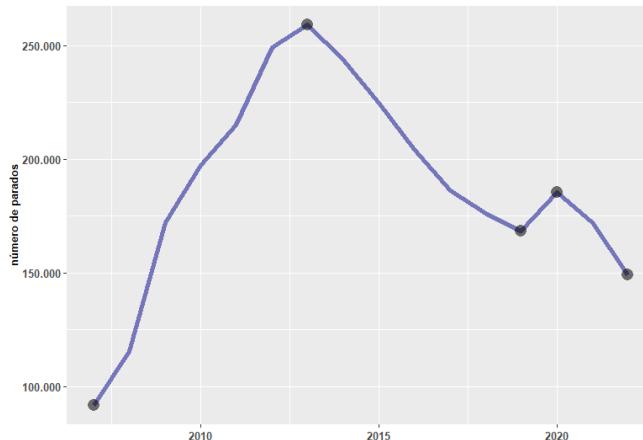


Figura 34.1: Evolución del paro medio anual en CLM

De un primer análisis visual de la Fig. 34.1 se toman como puntos de referencia los años previos a las crisis: 2007 y 2019, y el último año, 2022. Se puede observar que, si bien la crisis de la **COVID-19** ha tenido profundos efectos sectoriales, principalmente en turismo, comercio y restauración, la **crisis de 2008** tuvo un impacto enorme y generalizado en toda la economía, por lo que su efecto en el paro registrado fue devastador, multiplicando por un factor mayor de 3 la cifra total de paro en la región desde 2007. Sin embargo, a partir del año 2013 el paro registrado inicia una tendencia a la baja muy pronunciada que aún hoy continúa, después de haber repuntado ligeramente por la crisis de la COVID-19.

34.3. Evolución del paro medio anual en función de la edad y el sexo

Para ver cómo ha cambiado la estructura del paro registrado en función de la **edad** y el **sexo** de los parados se pueden utilizar diferentes gráficos. En este análisis, se usan mapas de calor y gráficos de distribución de densidad. Para hacer un mapa de calor que permita comparar dos variables simultáneamente, se construye la siguiente función:

```
heatmap_anyos <- function(var1, var2, inicio = 2007, intermedio = 2019,
                           fin = 2022){
  tabla <- select(parados_clm, anyo, var1, var2, parados) |>
    filter(anyo %in% c(inicio, intermedio, fin))
  names(tabla) <- c("anyo", "var1", "var2", "parados")
  tabla <- tabla |>
    group_by(anyo, var1, var2) |>
    summarise(parados = sum(parados))
  graf <- ggplot(tabla, aes(x = var1, y= var2, fill = parados)) +
```

```
geom_raster() +
scale_fill_gradientn(colours = paleta_heatmaps) +
facet_wrap(~ anyo) +
labs(x = "", y = "") +
theme(axis.text = element_text(size = 10, face = "bold"),
      axis.title = element_text(size = 10, face = "bold"),
      strip.text = element_text(size = 10, face = "bold"))
return(graf)}
```

Si se lanza la función `heatmap_anyos()` para las variables `edad` y `sexo`, tomando como años comparativos 2007, 2019 y 2022, se obtiene:

```
heatmap_anyos("sexo", "edad")
```

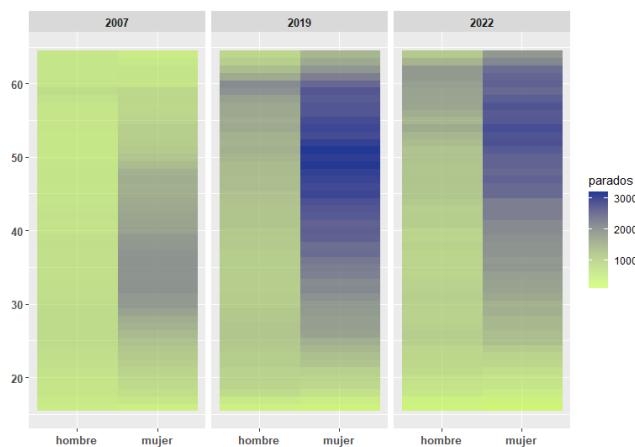


Figura 34.2: Paro medio anual según edad y sexo en 2007, 2019 y 2022

En la Fig. 34.2 se puede apreciar que en los dos procesos críticos se ha producido un **desplazamiento del paro hacia los intervalos de mayor edad**, siendo este cambio más pronunciado en las **mujeres**.

El mapa de calor es muy útil para una primera impresión de estos cambios, pero si se desea observar detalladamente cómo ha cambiado la distribución del paro según el `sexo` y la `edad`, es mejor programar la función `densidad_compara()`, que proporciona mayor nivel de detalle: produce un cuadro de gráficos comparando la distribución de la edad, para cada estrato de la variable elegida, para tres años diferentes (2007, 2019 y 2022 por defecto). Los parámetros `alpha` y `size` permiten ajustar tamaño y opacidad de las líneas, mejorando la apariencia general del gráfico.

```
densidad_compara <- function(variab, inicio = 2007, medio = 2019,
                                fin = 2022){
```

34.3. Evolución del paro medio anual en función de la edad y el sexo

549

```

tabla <- select(parados_clm, anyo, variab, edad, parados) |>
  filter(anyo %in% c(inicio, medio, fin))
names(tabla) <- c("anyo", "variable", "edad", "parados")
tabla <- tabla |>
  group_by(anyo, edad, variable) |>
  summarise(parados = sum(parados))# />

graf <- ggplot(tabla, aes(x = edad, y = parados, color = anyo,
                           fill = anyo)) + geom_line(alpha=0.6, size = 1) +
  facet_wrap(~ variable, ncol = dim(table(tabla$variable))[1]) +
  ylab("número de parados") + labs(color="año") +
  scale_color_manual(values = paleta_lineas) +
  scale_y_continuous(labels = function(x) format(x,
                                                 big.mark = ".",
                                                 scientific = FALSE)) +
  theme(strip.text = element_text(size = 10, face = "bold"),
        axis.title = element_text(size = 10, face = "bold"),
        axis.text = element_text(size = 10, face = "bold"))
return(graf)

```

Ejecutando esta función para la variable `sexo` se obtiene:

```
densidad_compara("sexo")
```

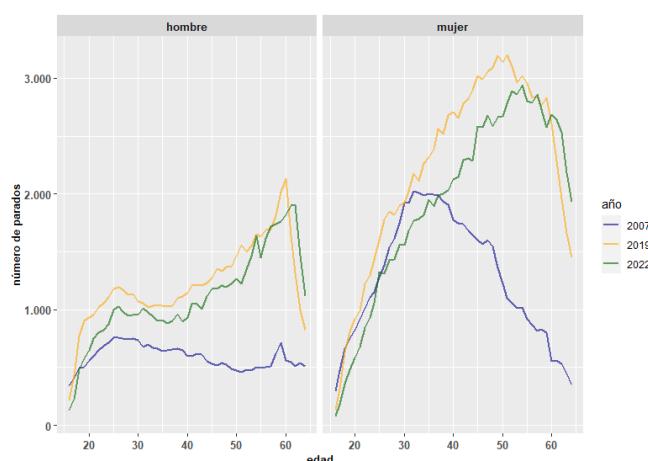


Figura 34.3: Distribución del paro medio anual por edad y sexo (2007, 2019 y 2022)

En la Fig. 34.3 se observa que en 2007, antes de ambas crisis, los hombres parados presentan **dos máximos**, en torno a 25 y 60 años, mientras que las mujeres desempleadas tienen una distribución bastante centrada entre 30 y 40 años. En cambio, en 2022 se aprecia el desplazamiento de la distribución de los parados de ambos sexos hacia los estratos de edad **mayores**

de 50 años. Este desplazamiento es algo más intenso en las mujeres.

Se observa igualmente que comparando las distribuciones de las mujeres de 2019 y 2022, la crisis de la COVID-19 ha incrementado entre 5 y 10 años la distribución de la edad de las mujeres paradas. Este desplazamiento es inferior en los hombres, donde supone menos de 5 años.

34.4. Evolución del paro medio anual según el tiempo de búsqueda de empleo

Se define el **tiempo de búsqueda de empleo** como el tiempo transcurrido ininterrumpidamente desde la última inscripción de la persona en el paro registrado (Pérez Infante, 2006).

Si, para simplificar, se agregan los doce intervalos que considera la estadística de paro registrado para el tiempo de búsqueda de empleo en tan solo cuatro, ejecutando la función `densidad_compara()` para la variable `t_bus_e_agr` se obtiene:

```
densidad_compara("t_bus_e_agr")
```



Figura 34.4: Distribución del paro medio anual por edad y tiempo de búsqueda de empleo

En la Fig. 34.4 se pone de manifiesto que el tramo con mayor incremento de número de parados es el correspondiente a más de 24 meses de búsqueda de empleo (**paro de muy larga duración**), ya que la crisis financiera de 2008 les redujo su probabilidad de encontrar empleo. Se puede afirmar también que los dos períodos de crisis han provocado la creación de un **paro estructural de larga duración**.

Se deja al lector ejecutar la función `heatmap_anyos()` para las variables `sexo` y `t_bus_e`, tomando como años comparativos 2007, 2019 y 2022. Observará en el gráfico resultante que el incremento en el paro de muy larga duración es más intenso en el colectivo de las mujeres. El código a utilizar es:

34.5. Evolución del paro medio anual según sexo, edad y sector de procedencia 551

```
heatmap_anyos("sexo", "t_bus_e")
```

34.5. Evolución del paro medio anual según sexo, edad y sector de procedencia

La variable **sector de procedencia** es un tanto particular, ya que, cuando un parado lleva mucho tiempo buscando empleo ininterrumpidamente, “ pierde” el sector de procedencia y se clasifica automáticamente en la rúbrica “sin actividad”. A la hora de analizar esta variable, por tanto, es importante tener en cuenta que una parte de los parados ubicados en la rúbrica “sin actividad”, realmente tuvieron un trabajo hace mucho tiempo.

La visualización de los cambios producidos en estas variables con un mapa de calor, se puede llevar a cabo ejecutando de nuevo la función `heatmap_anyos()` obteniendo la Fig. 34.5:

```
heatmap_anyos("sexo", "sector")
```

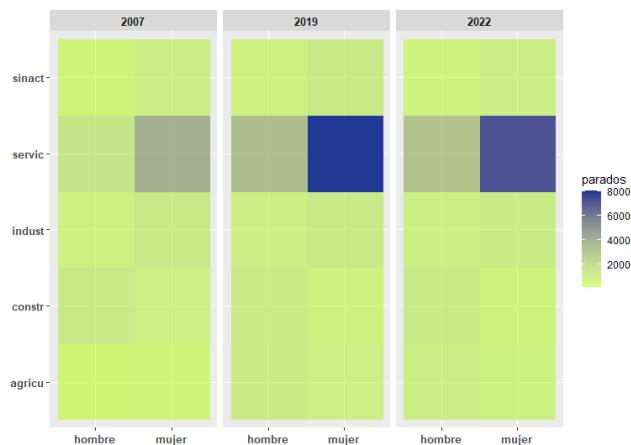


Figura 34.5: Paro medio anual según sexo y sector de procedencia

En la Fig. 34.5 se aprecia el incremento del paro registrado en el sector **servicios**, especialmente en el colectivo femenino.

Ejecutando la función `densidad_compara()` para la variable **sector** se obtiene:

```
densidad_compara("sector")
```

Como se observa en la Fig. 34.6, las diferencias a lo largo del tiempo del número de parados por sector de actividad económica revelan algunas particularidades interesantes. **Industria** y **construcción** se comportan de modo similar: hay un fuerte desplazamiento en edad desde 2007,

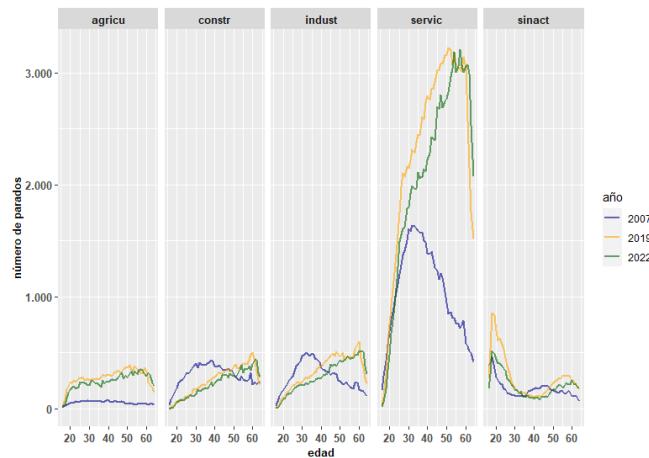


Figura 34.6: Distribución del paro medio anual por edad y tiempo de búsqueda de empleo

pero se mantiene el volumen de paro en ambos sectores a lo largo de los 15 años de estudio. El paro en el sector **agropecuario** y en el sector **servicios** también presenta desplazamiento en edad, pero además se ha incrementado notablemente en estos 15 años; ambos efectos son mucho más evidentes en el sector **servicios**. Finalmente, en el colectivo **sin actividad** se aprecian dos características: en primer lugar, los parados menores de 30 años suponen el mayor volumen en este colectivo, como era de esperar, ya que la población joven que accede al mercado laboral por primera vez, no cuenta con experiencia previa; en segundo lugar, desde 2007 a 2019, y algo menos desde 2019 a 2022, hay un incremento de volumen de paro en los **mayores de 45 años** que, con toda probabilidad, corresponde a los parados de larga duración de mayor edad. En todos los sectores se aprecia el descenso del volumen total de paro registrado desde 2019 a 2022, a pesar de la crisis sanitaria de la COVID-19.

34.6. Conclusiones

La crisis de 2008 tuvo un gran impacto en el paro registrado de Castilla-La Mancha, multiplicándolo por un factor mayor de 3 desde 2007. Sin embargo, a partir del año 2013 el paro registrado inicia una tendencia a la baja muy pronunciada que aún hoy continúa, después de haber sufrido un rebote debido a la crisis de la COVID-19.

La estructura interna de la población parada en la región ha cambiado sustancialmente atendiendo a las variables analizadas. En efecto, la población mayor de 45 años, las mujeres, los parados de larga duración y el sector servicios son los grandes perjudicados por ambos procesos de crisis.

Capítulo 35

Segmentación de clientes en el comercio minorista

Jaime Fierro Martín^a, Rocío González Martínez^a y Cristina Sánchez Figueroa^b

^aAnalyticae, SL, ^bUniversidad Nacional a Distancia

35.1. Motivación y conceptos clave

Los comercios minoristas (*retailers*) se mueven en un entorno turbulento y necesitan acercarse a sus clientes para asegurar su supervivencia. Su producto, o servicio, es nexo clave en dicho proceso. En este contexto, conocer el **perfil de los clientes** permitirá detectar en qué momento de su ciclo de vida con la empresa se encuentran y desarrollar propuestas de valor que convengan en cada momento.

Segmentar se define como el proceso de dividir a los clientes actuales o potenciales, en diferentes grupos o segmentos consistentes en individuos con características y niveles similares de interés (véase el Cap. 13 para una explicación detallada de las técnicas del cluster no jerárquico). Es un proceso creativo e iterativo con el fin de satisfacer con mayor acierto las necesidades de los clientes, proporcionando una ventaja competitiva y sostenible a la compañía. La segmentación viene dada por las necesidades de los clientes, no de la compañía, y debería ser revisada periódicamente.

Este caso práctico de negocio está basado en un proyecto real impulsado por el departamento de marketing de una empresa del sector *retail* que necesitaba mejorar el conocimiento de sus clientes, agrupándolos en función de su comportamiento de compra. Los resultados obtenidos fueron clave para definir la estrategia de marketing relacional de la compañía.

35.2. El modelo *Recency, frequency, monetary* tradicional

El **modelo RFM** es una técnica popular que se utiliza para analizar el comportamiento de compra de los clientes: cómo compran, su frecuencia de compra y cuánto gastan. Es un método útil para enriquecer la segmentación de los clientes en varios grupos que permitan la personalización e identificación de los clientes más proclives a responder a las promociones. El análisis RFM depende de las medidas de actualidad (*recency*) (R), frecuencia (*frequency*) (F) y valor monetario (*monetary*) (M), que son tres importantes variables relacionadas con la compra que influyen en las posibilidades de compra futura de los clientes.

El **modelo RFM tradicional** categoriza el valor de las variables dividiéndolas en quintiles, a partir de los cuales se calcula una puntuación única que representa el valor del cliente. Sin embargo, no es muy preciso. Si el intervalo de frecuencia de compras se fija entre 0 y 20, en términos de negocio podría interpretarse como que un cliente con una sola compra será igual que otro que tenga 20. Por ello, los enfoques de conjuntos clásicos pueden resultar poco funcionales (Martínez et al., 2019). En este caso práctico, se propone una mejora en la definición de los intervalos mediante la aplicación del **ranking de percentiles**. Este método, que se ha denominado modelo RFM extendido, proporciona un método robusto para tratar los valores atípicos (*outliers*), y además normaliza las variables entre 0 y 1 para evitar la diferencias de peso entre las variables, permitiendo así la correcta implementación del **algoritmo de segmentación**.

35.3. El modelo *Recency, frequency, monetary* extendido

Los autores de este caso práctico recomiendan seguir una metodología de gestión de proyectos. La **metodología CRISP-DM** (Chapman et al., 2000), presentada en el Cap. @ref(metodología) es un estándar ampliamente utilizado en los proyectos de ciencia de datos.

Una vez definido el problema (mejorar el conocimiento que una empresa de comercio minorista tiene de sus clientes, agrupándolos en función de su comportamiento de compra), la recopilación y comprensión de los datos (primera etapa del modelo CRISP-DM) se establece como etapa esencial para el desarrollo del proyecto.

35.3.1. Recopilación y comprensión de los datos

Hoy en día, la mayoría de las empresas de *e-commerce* y comercio minorista tradicional cuentan con sistemas que permiten registrar los datos básicos de cada una de sus ventas (fecha, artículo, cantidad e importe), asociados a un código único de cliente. La información contenida en estos datos de compra atesora gran valor, ya que carecen del sesgo y subjetividad propias de otras informaciones obtenidas mediante encuestas de opinión, estudios de mercado, entrevistas y grupos de discusión, etc. Estos datos suelen encontrarse en las plataformas ERP (*Enterprise Resource Planning*) de gestión de pedidos y ventas, o CRM (*Customer Relationship Management*) de las empresas.

35.3. El modelo Recency, frequency, monetary extendido

555

El lector es, o será, consciente de que la fase de extracción, carga y limpieza de los datos es la más exigente del proyecto, y donde se empleará gran parte de los recursos y tiempo de todo el proyecto. **R** cuenta con gran cantidad de paquetes y recursos que facilitan la extracción desde diferentes tipos de bases de datos.

Para este caso práctico serán necesarias las siguientes librerías:

```
library("lubridate")
library("factoextra")
library("ggpubr")
library("CDR")
data("datos_retail")
```

Cualquier tipo de estudio o proyecto de ciencia de datos requiere familiarizarse con los datos y determinar si presentan suficiente exactitud, completitud, consistencia, credibilidad y actualidad (Muñoz-Reja et al., 2018). Los datos de transacciones de venta registrados por las empresas pueden contener datos atípicos (p.ej. valores perdidos, inexactos, outliers, etc.). Para determinar la acción a tomar, o no, de limpieza o corrección de los datos de partida, es esencial conocer el negocio y las consecuencias que éstas operaciones tendrán en el resultado final de la segmentación.

El conjunto de datos de muestra contiene 200.000 observaciones correspondientes a transacciones de compra. Las siguientes variables iniciales están explicadas en el set de datos:

```
head(datos_retail)
#> # A tibble: 6 x 4
#>   id_ticket fecha importe_venta codigo_socio
#>   <chr>     <date> <dbl> <chr>
#> 1 num_1646673 2021-10-30 12.4 id_1076134
#> 2 num_2762559 2021-12-03 38.8 id_0552641
#> 3 num_0309422 2022-01-07 67.8 id_0537369
```

35.3.2. Cálculo de las variables del modelo RFM

Identificadas las variables iniciales, es necesario calcular los factores clave del Modelo RFM:

- La variable actualidad, *recency* (R), es el intervalo de tiempo transcurrido desde la última compra de un cliente hasta la fecha de elaboración del modelo RFM.
- La variable frecuencia, *frequency* (F), se obtiene agrupando las compras por cliente y contando el número total de tickets únicos.
- La variable valor monetario, *monetary* (M), se calcula sumando todos los importes de venta por cliente.

```

fecha_estudio_rfm <- ymd("2022-08-01")

rfm <- datos_retail |>
  group_by(codigo_socio) |>
  summarise(
    frecuencia = n_distinct(id_ticket),
    monetario = sum(importe_venta, na.rm = TRUE),
    fecha_transaccion_reciente = first(fecha, order_by = desc(fecha))
  ) |>
  mutate(actualidad = time_length(interval(start = fecha_transaccion_reciente, end =
  fecha_estudio_rfm), unit = "days"), .keep = "unused")

```

`head(rfm) # el lector puede ver las variables del Modelo RFM`

35.3.3. Breve análisis exploratorio de las variables del modelo RFM

Del análisis puede concluirse que:

- 107.929 clientes han realizado una media de 1,85 compras, con un importe medio total de 70,56€ y 450,4 días de media desde la última compra hasta la fecha de realización del estudio, con una fuerte asimetría positiva de los valores *frequency* y *monetary* (ver Fig. 35.1).
- Se detecta una gran estacionalidad de las compras, como se puede apreciar en la agrupación de las observaciones de *recency*. Teniendo en cuenta la fecha en la que se realiza el análisis, los valores obtenidos en la variable *recency*, se pueden interpretar como el periodo de ventas de la campaña navideña.

```

set.seed(12345)
plot_data <- rfm |>
  slice_sample(n = 2000) |>
  pivot_longer(!codigo_socio, names_to = "variable", values_to = "valor")
plot_data |>
  ggplot(aes(x = variable, y = valor)) +
  geom_boxplot(outlier.shape = NA, color = "red") +
  geom_jitter(alpha = 1 / 10) +
  facet_wrap(~variable, ncol = 6, scales = "free") +
  theme(strip.text.x = element_blank(), text = element_text(size = 9))

```

35.3.4. Cálculo del ranking de percentiles

Los valores de ranking son relativos entre clientes y no pueden ser utilizados para objetivos de negocio, basados en valores absolutos de puntuación por cliente.

35.3. El modelo Recency, frequency, monetary extendido

557

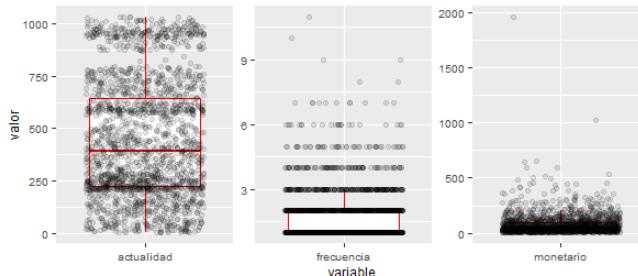


Figura 35.1: Box-plot

```
rfm_rank <- rfm |>
  mutate(across(.cols = c("frecuencia", "monetario"), percent_rank, .names =
    ~ "rank_{.col}")) |>
  mutate(across(.cols = c("actualidad"), ~ percent_rank(-.x), .names = "rank_{.col}"))
  # menor recency indica mayor puntuación en rank
```

Se podría decir que el análisis RFM combina tres atributos clave de los clientes para construir un ranking que permite agruparlos de forma útil para el negocio. Así, a si un cliente que compró en una fecha reciente (Recency) se le otorgan más puntos. Si compró muchas veces (Frequency), también se le coloca más arriba en el ranking. Finalmente, si gastó más en el total de sus compras (Monetary), también puntúa más alto. Combinando estos tres parámetros, se obtiene un ranking RFM. Para la elaboración de este ranking se parte del concepto de percentil . La idea es asignarle a cada cliente una puntuación según las tres variables o factores clave del modelo RFM, de modo que los mejores clientes serán los que tengan una puntuación mayor.

```
head(rfm_rank) # el lector puede ver la puntuación del ranking
```

Una vez que se tienen los rankings de percentiles en las tres variables para todos los clientes, se procede a su clusterización mediante el método k-means.

35.3.5. Modelado: RFM mediante k-means

El modelo establecido debe proporcionar una segmentación de clientes con sentido de negocio. En este caso práctico se opta por el algoritmo de clustering estándar que presenta la ventaja de ser muy intuitivo y permite trabajar con grandes conjuntos de datos. Como el lector ha podido comprobar, existen otros muchos algoritmos de aprendizaje no supervisado que pueden ser empleados.

El número óptimo de *clusters* (o segmentos, en la jerga del marketing) es uno de los retos a la hora de aplicar técnicas de *clustering*. No existe una manera exclusiva de encontrar el número adecuado de clusters. Se trata de un proceso subjetivo que depende de los datos, del tipo de *clustering* empleado y, en este caso, de que el número elegido tenga sentido y utilidad en el

negocio. Existen numerosos métodos para facilitar la elección del número de *clusters*; entre ellos destacan el *Elbow method*, el *Average silhouette method* y el *Gap statistic method*, que gracias a la función `fviz_nbclust()` del paquete `factoextra`, se pueden calcular con facilidad para realizar una buena elección.

Con el método Elbow, número óptimo de *clusters* se calcula como sigue:

```
set.seed(123)
muestra_clusters <- rfm_rank |>
  slice_sample(n = 5000) |>
  dplyr::select(matches("rank"))

fviz_nbclust(x = muestra_clusters, FUNcluster = kmeans, method = "wss", k.max = 10)
```

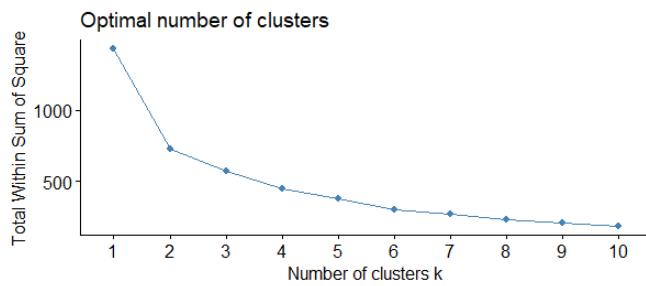


Figura 35.2: Número óptimo de clusters

En la Fig. 35.2 se observa que la varianza total *intra-cluster* apenas mejora a partir del cuarto cluster.

El algoritmo de clustering k-means se entrena con las variables R-F-M normalizadas con el ranking. La salida de la función `kmeans()` del paquete base `stats` es un objeto que, entre otros componentes, ofrece un vector numérico indicativo del *clusters* al que pertenece cada uno de los clientes.

```
set.seed(123)
km_fit <- kmeans(x = rfm_rank[, 5:7], centers = 4, nstart = 10)

clientes_segmentos <- rfm_rank |>
  mutate(segmento = km_fit$cluster)

head(clientes_segmentos) # el lector puede ver el segmento al que pertenece el cliente
```

35.3.6. Descriptivos e interpretación de los segmentos

Los segmentos obtenidos deben tener sentido y utilidad de negocio. Para ello es imprescindible proporcionar los estadísticos descriptivos de cada segmento y proceder a su interpretación de

35.3. El modelo Recency, frequency, monetary extendido

559

perfil de cliente.

```
descriptivo_segmentos <- clientes_segmentos |>
  group_by(segmento) |>
  summarise(across(c("monetario", "frecuencia", "actualidad"),
    .fns = mean, .names = "md_{.col}")
  ), n_clientes = n() |>
  ungroup() |>
  relocate(segmento, n_clientes)

head(descriptivo_segmentos)
#> # A tibble: 4 x 5
#>   segmento n_clientes md_monetario md_frecuencia md_actualidad
#>   <int>     <int>      <dbl>        <dbl>        <dbl>
#> 1 1         23551     36.2       1.07       239.
#> 2 2         23632     77.0       2.26       567.
#> 3 3         28809     128.       3.11       188.
#> 4 4         31937     39.3       1.00       757.
```

Interpretación de los segmentos:

- 1-Nuevos probando: segmento que agrupa nuevos clientes que están realizando compras desde hace poco tiempo y tienen un gran potencial de desarrollo. Es un segmento de clientes con interés para la empresa.
- 2-No podemos perder: se trata de los clientes ‘churn’¹ que fueron buenos clientes en términos monetarios y de frecuencia pero que hace tiempo que no realizan nuevas compras. La compañía debe hacer un esfuerzo en recuperar estos clientes para convertirlos al segmento TOP.
- 3-Top: reúne a los mejores clientes de la empresa. Son clientes que compran con frecuencia, están activos y aportan ventas a la compañía. Es el segmento de clientes con mayor interés para la empresa.
- 4-Una compra: segmento formado por aquellos clientes que han realizado una sola compra hace tiempo. Presentan frecuencia, actualidad y valor monetario bajo. Se trata de un segmento de clientes con escaso interés para la compañía.

```
segmentos_descriptivo <- clientes_segmentos |>
  mutate(segmento = case_when(
    segmento == 1 ~ "1_Nuevos probando",
    segmento == 2 ~ "2_No perder",
    segmento == 3 ~ "3_Top",
    segmento == 4 ~ "4_Una compra"
  )) |>
  group_by(segmento) |>
  summarise(
```

¹El término “customer churn” se suele traducir como perdida de clientes o rotación de clientes. Se compone de las palabras inglesas “change” (en castellano cambio) y “turn” (en castellano abandonar)

```

across(
  .cols = where(is.numeric),
  .fns = mean
),
  n_clientes = n()
) |>
ungroup() |>
relocate(segmento, n_clientes)

table_dot_plot <- segmentos_descriptivo |>
# select(starts_with("rank")) |>
pivot_longer(cols = c("rank_monetario", "rank_frecuencia", "rank_actualidad"),
  names_to = "Variable RFM", values_to = "Puntuación")

ggdotchart(
  table_dot_plot,
  x = "Variable RFM", y = "Puntuación",
  group = "segmento", color = "segmento", palette = "jco",
  add = "segment", position = position_dodge(0.3),
  sorting = "none", facet.by = "segmento", dot.size = 5,
  rotate = TRUE, legend = "none"
)
)

```

La Fig 35.3 muestra cada uno de los segmentos indicados. El ranking obtenido ayuda a identificar las diferencias en los tipos de clientes y es útil para decidir a qué segmentos enfocarse y qué estrategias usar para cada uno.

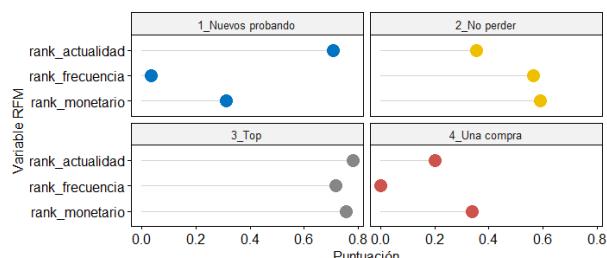


Figura 35.3: Lollipop de variables RFM

35.3.7. Puesta en producción

Calculado el modelo RFM k-means, la compañía puede incorporar periódicamente los datos de los clientes nuevos, o actualizados. De este modo, los segmentos de clientes se actualizarán y, más allá de las acciones de *marketing mix* que realicen las compañías gracias a la segmentación, podrán analizarse las migraciones de clientes entre los diferentes segmentos en el periodo estudiado. La función `cl_predict()` facilita la actualización periódica de los segmentos con el modelo entrenado.

Capítulo 36

Análisis de datos en medicina

Alberto M. Borobia^a y María Jiménez-González^a

^aHospital Universitario La Paz - IdiPAZ

36.1. Justificación

La aplicación de la estadística en la investigación clínica ha sido una de las herramientas clave en los últimos dos años. La pandemia mundial causada por la enfermedad por coronavirus (COVID-19) es una enfermedad infecciosa provocada por el virus SARS-CoV-2. Drante el año 2020, más de 13 millones de casos diagnosticados en España arrojaban un diagnóstico claro: se necesita más investigación.

El primer apartado de este capítulo señala la importancia de la identificación de los sesgos (en concreto, del sesgo de selección) que aparecen en los estudios de investigación no aleatorizados. Tras ello, se abordará un ejemplo práctico de una de las aplicaciones más significativas de la bioestadística: el análisis de supervivencia, con el que se resuelven preguntas tan importantes cómo: ¿qué factores de riesgos están asociados a la mortalidad provocada por coronavirus?.

36.2. Introducción al uso de datos en investigación clínica y ensayos clínicos

En este capítulo, y a modo ilustrativo del ámbito de la investigación clínica, se abordarán tres análisis a partir de los datos:

- Un análisis relativo a la eliminación de sesgos, o más concretamente, a la eliminación del sesgo de selección.
- Un análisis relativo a la estimación e interpretación de las curvas de supervivencia.
- Un análisis relativo a la estimación e interpretación de la Regresión de COX.

36.2.1. ¿Qué es un ensayo clínico?

En la investigación clínica existen dos tipos de estudios: *estudios observacionales* y *ensayos clínicos*.

Los ensayos clínicos aleatorios se definen como el diseño experimental óptimo para proporcionar evidencia, eficacia y seguridad de una intervención (Liu et al., 2020). Los tratamientos estudiados o investigados son asignados aleatoriamente en grupos que garantizan que las diferencias en los resultados después del tratamiento reflejen los efectos del mismo (Rosenbaum, 2005). Cuando estas condiciones ideales no son posibles (falta de recursos, financiación, tiempo, etc), se definen como estudios observacionales.

Previo a la puesta en marcha de un ensayo clínico, es imprescindible la redacción de un **Protocolo** y un **Plan de Análisis Estadístico (PAE)**.

- El protocolo, elaborado por los investigadores del estudio, precisa y justifica los métodos y planes del proceso que se llevará a cabo en el ensayo clínico (Rivera et al., 2020).
- El PAE detalla las características principales del eventual análisis estadístico de los datos, que deben describirse en la sección estadística del protocolo (Lewis, 1999).

Los documentos anteriormente mencionados, y el resto de directrices necesarias para un ensayo clínico, están regulados por la “Conferencia Internacional sobre armonización de requisitos técnicos para el registro de productos farmacéuticos para uso humano” (sus siglas ICH en inglés).

36.2.2. Limitaciones de los estudios observacionales

En el apartado anterior, se puso de manifiesto la importancia de los ensayos clínicos aleatorizados. Sin embargo, la posible falta de recursos, financiación, tiempo o materiales, dificultan la puesta en marcha y realización de los mismos.

En consecuencia, la puesta en práctica de la investigación puede no ser la ideal. Los estudios observacionales, sin embargo, son una herramienta elemental en circunstancias no tan óptimas, ya que permiten analizar e investigar (contra viento y marea).

La limitación principal de los estudios observacionales es que introducen sesgos en el análisis. Los ensayos clínicos tienen como principal objetivo eliminar el **sesgo de selección**: cuando los sujetos no son asignados aleatoriamente, por ejemplo, los resultados diferentes pueden reflejar estas diferencias iniciales en lugar de los efectos de los tratamientos (Rosenbaum, 2005).

36.2.3. Índice de propensión (*propensity score*)

Una solución aconsejable y recomendable ante los sesgos “escondidos” en los estudios observacionales es la técnica *propensity score* o índice de propensión. Esta técnica de emparejamiento equilibra las covariables observadas sesgadas ajustando por su índice de propensión, eliminando presumiblemente el sesgo. Habitualmente, el índice de propensión se obtiene a partir de un

36.2. Introducción al uso de datos en investigación clínica y ensayos clínicos 563

modelo de regresión cuya variable dependiente corresponde a la intervención o el resultado principal (por ejemplo, la muerte) y las variables independientes o covariables corresponden a las variables que puedan tener un efecto confusor en la variable dependiente [molina2015indices].

36.2.4. Ejemplo práctico en R de un estudio observacional

El *dataset* sintético `datos_observacional` reproduce los datos de un hipotético estudio observacional sobre una enfermedad **X**. El objetivo del estudio es estudiar los factores de riesgo asociados a la mortalidad causada por esa enfermedad.

```
#> # A tibble: 5 x 8
#>   ID fecha_hospitalizacion sexo   edad comorbilidades      fecha~1 exitus
#>   <dbl> <dttm>          <chr> <dbl> <chr>           <chr>    <dbl>
#> 1     1 2015-04-17 00:00:00 Mujer     76 1 o más comorbilidades 17/04/~     1
#> 2     2 2015-03-21 00:00:00 Mujer     64 1 o más comorbilidades 31/03/~     0
#> 3     3 2015-04-09 00:00:00 Hombre    65 1 o más comorbilidades 16/04/~     0
#> 4     4 2015-04-04 00:00:00 Hombre    77 1 o más comorbilidades 13/04/~     0
#> 5     5 2015-03-24 00:00:00 Mujer     66 1 o más comorbilidades 27/03/~     0
#> # ... with 1 more variable: fecha_exitus <dttm>, and abbreviated variable name
#> #   1: fecha_alta
```

En la literatura, se ha evidenciado que las mujeres de mayor edad y con una o más comorbilidades tienen más riesgo de fallecer (`exitus`) por la enfermedad **XX**. En investigación, la estructura de los resultados en un paper o en un informe estadístico, independientemente de la revista o PAE, comienza en el mismo punto: una tabla resumen de las características basales de la población objeto de estudio. El paquete `tableone` (sencillo juego de palabras) integra funciones específicas para la creación de dichas tablas. La función principal de este paquete es `CreateTableOne()`.

```
library("tableone")
my_vars <- c("sexo", "edad", "comorbilidades")
nonnormal <- c("edad")
factor_vars <- c("sexo", "comorbilidades")

# crea la tabla
tab1 <- CreateTableOne(
  vars = my_vars, factorVars = factor_vars,
  strata = "exitus", data = datos_observacional
)

# imprime la tabla
tab1 <- print(tab1,
  showAllLevels = TRUE, formatOptions = list(big.mark = ","),
  exact = "stage", nonnormal = nonnormal
)
```

Tabla 36.1: Características basales de la población

	level	Vivo	Exitus	p-valor	test
n		79	21		
sexo (%)	Hombre	28 (35.4)	2 (9.5)	0.042	
	Mujer	51 (64.6)	19 (90.5)		
edad (median [IQR])		64.00 [53.00, 73.00]	82.00 [72.00, 85.00]	<0.001	nonnorm
comorbilidades (%)	1 o más comorbilidades	43 (54.4)	18 (85.7)	0.018	
	No	36 (45.6)	3 (14.3)		

Para presentar la tabla de resultados formateada basta con usar la función `kable()`:

```
knitr::kable(tab1,
  caption = "Características basales de la población",
  col.names = c("level", "Vivo", "Exitus", "p-valor", "test")
)
```

En la Tabla 36.1 y acorde a la bibliografía existente, se confirma el sesgo de selección a través del desequilibrio de la variable principal (`exitus`) en las variables `sexo`, `edad` y `comorbilidades`, evidenciado a través de la significación de éstas. Un argumento que motiva la aplicación, en este caso, de la técnica *propensity score* se fundamenta en la viabilidad de, por ejemplo, un modelo multivariante (como puede ser un modelo de predicción). La recogida de datos de un estudio observacional, como el de este ejemplo, normalmente viene dada por la disponibilidad de la población: sujetos ingresados en el Hospital por la enfermedad (en nuestro caso, coronavirus). Por tanto, esta muestra seleccionada recogerá pacientes con pronóstico más grave que la población general (mujer de mayor edad con una o más comorbilidades).

El paquete `MatchIt` integra las funciones principales para el ajuste de la técnica *propensity score*, concretamente la función `matchit()` integra la teoría de (Ho et al., 2007) para el emparejamiento óptimo de los grupos estudiados. Los argumentos más importantes de esta función son: - `formula`: modelo de regresión que estudia la relación entre la variable principal de estudio (`exitus`) con las variables sesgadas (`sexo`, `edad` y `comorbilidades`). - `method`: especifica el método de *matching*. - `distance`: especifica el método para la estimación del índice de propensión.

La función `get_matches()` empareja, posteriormente, las coincidencias que resultan del `MatchIt`.

Nota: es imprescindible que los casos del `dataset` estén completos.

```
library("MatchIt")
match <- matchit(exitus ~ edad + as.factor(sexo) + as.factor(comorbilidades),
  method = "nearest", distance = "mahalanobis",
  data = datos_observacional
```

36.2. Introducción al uso de datos en investigación clínica y ensayos clínicos 565

Tabla 36.2: Características basales de la población aplicando la técnica de propensity score

	level	Vivo	Exitus	p-valor	test
n		21	21		
sexo (%)	Hombre	2 (9.5)	2 (9.5)	1.000	
	Mujer	19 (90.5)	19 (90.5)		
edad (median [IQR])		72.00 [69.00, 84.00]	82.00 [72.00, 85.00]	0.182	nonnorm
comorbilidades (%)	1 o más comorbilidades	18 (85.7)	18 (85.7)	1.000	
	No	3 (14.3)	3 (14.3)		

```
)
datos_observacional_match <- get_matches(match, datos_observacional)
```

Para comprobar que el sesgo evidenciado en estudios anteriores ha desaparecido, se reproduce la tabla anterior.

```
tab1_corregida <- CreateTableOne(
  vars = my_vars, factorVars = factor_vars,
  strata = "exitus", data = datos_observacional_match
)

# se imprime en el objeto tab1_corregida
tab1_corregida <- print(tab1_corregida,
  showAllLevels = TRUE, formatOptions = list(big.mark = ","),
  exact = "stage", nonnormal = nonnormal
)
```

Se formatea la salida de la tabla:

```
knitr::kable(tab1_corregida,
  caption = "Características basales de la población aplicando la técnica de propensity
  ← score",
  col.names = c("level", "Vivo", "Exitus", "p-valor", "test")
)
```

En la Tabla @ref(tab:tab1_corregida) se observa que el sesgo de selección existente en la muestra se ha resuelto equilibrando las variables (aunque reduciendo la muestra). Tras este paso previo, podría realizarse un análisis estándar de esta muestra intentando aproximarse lo máximo posible a un estudio aleatorizado.

36.3. Análisis de supervivencia

Durante la pandemia ocasionada por el SARS-CoV-2, la pregunta principal de los investigadores clínicos se centró en un mismo objetivo: factores de riesgo asociados a la mortalidad causada por COVID-19. El análisis de supervivencia ha permitido a los investigadores intentar explicar las causas más factibles que producen esa mayor probabilidad de fallecer. El análisis de supervivencia permite estudiar los factores de riesgo asociados a la mortalidad. La ventaja principal de este análisis frente a un análisis estándar (como puede ser una regresión logística) se centra en la integración en la variable respuesta del evento y del tiempo hasta el evento, que tiene como consecuencia la interpretación de “riesgo” y no de “probabilidad” en los resultados.

El *dataset* utilizado, `datos_supervivencia` está incluido en el paquete `CDR` y está formado por 301 pacientes, 101 diagnosticados con infección por SARS-CoV-2 y 100 exitus.

```
head(datos_supervivencia, 5)
#> # A tibble: 5 x 7
#>   id EXITUS_TIME DIAG_COVID EXITUS N_COMORBIDITIES SEX     EDAD
#>   <dbl>      <dbl>    <dbl>    <dbl>        <dbl> <chr>    <dbl>
#> 1 262         0        1        1            5 Hombre    83
#> 2 236         1        1        1            5 Hombre    72
#> 3 170         11       0        0            2 Mujer     65
#> 4 204         11       1        1            4 Hombre    80
#> 5 46          14       1        1            5 Hombre    90
```

36.3.1. Estimación y comparación de curvas de supervivencia

La **función (o curva) de supervivencia** estuda la probabilidad de que el paciente o sujeto, sobreviva a un tiempo **X**. El estimador más común utilizado para el ajuste de la función de supervivencia es el estimador no paramétrico **Kaplan-Meier** y su función escalonada. Una vez generadas estas curvas de supervivencia, existen diferentes métodos (paramétricos y no paramétricos) para su comparación. En este apartado, se utiliza la prueba de **Mantel-Cox** (o test **Log-Rank**) para el contraste de funciones.

Los paquetes `survival` y `survminer` integran las funciones principales de la técnica:

- La función `Surv()`, de la librería `survival`, crea un objeto de supervivencia formado por el evento (exitus) y el tiempo hasta la ocurrencia del evento.
- La función `survfit()`, de la librería `survival`, estima la función de supervivencia mediante el método *Kaplan-Meier* del objeto `Surv` y los factores de riesgo asociados.
- La función `ggsurvplot()`, genera el gráfico de la curva de supervivencia (basada en la librería `ggplot2`). El argumento principal de la función es la función de supervivencia estimada, `survfit()`. Los argumentos más importantes (y recomendables) a la hora de graficar la función de supervivencia son:
 - `pval`: muestra el p-valor correspondiente a la comparación a través del test *Log-Rank*.

36.3. Análisis de supervivencia

567

- `conf.int`: muestra los intervalos de confianza de la(s) curva(s) de supervivencia.
- `risk.table`: añade el número de sujetos (absoluto o relativo) en riesgo en cada momento del periodo objeto de estudio.

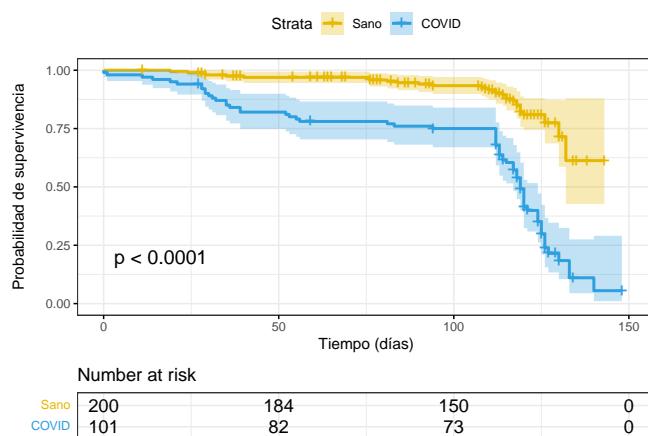
Se cragan los paquetes

```
library("survival")
library("survminer")
```

Se ajusta el modelo y, posteriormente, se representa:

```
fit <- survfit(Surv(EXITUS_TIME, EXITUS) ~ DIAG_COVID,
  data = datos_supervivencia
)

ggsurvplot(fit,
  data = datos_supervivencia,
  pval = TRUE,
  conf.int = TRUE,
  ggtheme = theme_bw(),
  palette = c("#E7B800", "#2E9FDF"),
  xlab = "Tiempo (días)",
  ylab = "Probabilidad de supervivencia",
  legend.labs = c("Sano", "COVID"),
  # añade tabla de supervivencia
  risk.table = TRUE,
  tables.height = 0.2,
  tables.theme = theme_cleantable()
)
```



En la Fig. ??, dónde el eje X corresponde al tiempo en días y el eje Y a la probabilidad de supervivencia, se observa que la probabilidad de supervivencia de las personas expuestas a COVID es significativamente menor (p -valor < 0.001) a las personas sanas. La mediana de supervivencia (línea trazada desde el 0.5 del eje Y, correspondiente al 50 % de la probabilidad de supervivencia) corresponde a los 120 días, es decir, el 50 % de los sujetos diagnosticados por COVID y objeto de estudio sobrevivieron, al menos, 120 días.

Por tanto, se puede concluir que se ha encontrado evidencia sobre el aumento de mortalidad asociada a la enfermedad COVID-19.

36.4. Regresión de COX

La regresión de Cox¹ o modelo de riesgos proporcionales es una técnica utilizada para el estudio del efecto de covariables sobre el tiempo hasta la ocurrencia de un evento (exitus, recaída, progresión, etc). La regresión de Cox es realmente una Regresión en la que la variable dependiente es siempre una función de riesgo o supervivencia (están íntimamente relacionadas) y los predictores son una función del tiempo y una función de las variables consideradas como explicativas. En general, se suele expresar así:

$$h(t, x_1, x_2, \dots, x_p) = h_0(t) + g(x_1, x_2, \dots, x_p),$$

y más concretamente,

$$h(t, x_1, x_2, \dots, x_p) = h_0(t) + e^g(x_1, x_2, \dots, x_p),$$

donde g , normalmente, indica una combinación lineal de las covariables o variables explicativas. Es, por tanto, una técnica semi-paramétrica.

La función principal para el ajuste de un modelo de regresión de Cox es `coxph()`. Esta función, al igual que la función `survfit()`, está formada por un objeto `Surv` y las covariables del modelo.

```
fit_cox <- coxph(Surv(EXITUS_TIME, EXITUS) ~ DIAG_COVID + EDAD + SEX + N_COMORBIDITIES,
                    data = datos_supervivencia
)
```

El *output* principal de una regresión de Cox,

$$h(t, x_1, x_2, \dots, x_p),$$

¹Es importante distinguir la regresión de Cox de la regresión logística (véase el Cap. ??). La Regresión logística relaciona la variable dependiente dicotómica con un conjunto de variables independientes sin contemplar el tiempo o contemplándolo sólo de forma estática, viendo en un punto fijo del tiempo si el suceso estudiado ha acontecido o no, pero no teniendo en consideración en qué momento ha sucedido. La Regresión de Cox proporciona un análisis más fino. No analiza, en un instante de tiempo dado, si un acontecimiento de interés ha sucedido o no, sino cuándo ha sucedido, si es que ha sucedido, y lo hace teniendo en cuenta el comportamiento de una o varias variables independientes. Es por ello que la regresión logística trabaja con la *odds ratio* y la regresión de Cox con la *hazard ratio*.

36.4. Regresión de COX

569

son las **razones de riesgos** o ***hazard ratios (HR)**. Es decir, la relación entre las dos funciones de riesgo en función de los cambios operados en las variables explicativas. En concreto, la exponencial del coeficiente estimado para la va

El *output* principal de una regresión de Cox , $h(t, x_1, x_2, \dots, x_p)$, son las razones de riesgos o hazard ratios (HR). Es decir, la relación entre las dos funciones de riesgo en función de los cambios operados en las variables explicativas. En concreto, la exponencial del coeficiente estimado para la variable explicativa X_i indica el incremento en el riesgo de fallecer cuando la variable explicativa aumenta en una unidad y las demás permanecen constantes. Esta razón de riesgos oscila entre 0 a ∞ , siendo el intervalo [0,1] una relación de riesgo bajo y [1, ∞] una relación de riesgo alto.

Nota

- Los HR localizados entre 1 y 2 se interpretan en porcentaje. Es decir, HR = 1.5 indica a un aumento del riesgo del 50 %.
- Los HR localizados entre 2 e ∞ se interpretan en “veces”. Es decir, HR = 3 indica a un aumento del riesgo de 3 veces.
- Los HR localizados entre 0 y 1 se interpretan como una reducción del riesgo del $(1 - HR) \times 100\%$. Es decir, HR = 0.8 indica a una disminución del riesgo del 20 %.

```
summary(fit_cox)
#> Call:
#> coxph(formula = Surv(EXITUS_TIME, EXITUS) ~ DIAG_COVID + EDAD +
#>       SEX + N_COMORBIDITIES, data = datos_supervivencia)
#>
#>   n= 271, number of events= 100
#>   (30 observations deleted due to missingness)
#>
#>           coef  exp(coef)    se(coef)      z Pr(>|z|)    
#> DIAG_COVID     1.3023581  3.6779594  0.5184547  2.512  0.0120 *  
#> EDAD          0.0006006  1.0006008  0.0116113  0.052  0.9587    
#> SEXMujer      -1.1256901  0.3244285  0.2360183 -4.770 1.85e-06 *** 
#> N_COMORBIDITIES 0.1643743  1.1786554  0.0774043  2.124  0.0337 *  
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>           exp(coef) exp(-coef) lower .95 upper .95    
#> DIAG_COVID      3.6780      0.2719     1.3314    10.1605  
#> EDAD            1.0006      0.9994     0.9781     1.0236  
#> SEXMujer        0.3244      3.0823     0.2043     0.5153  
#> N_COMORBIDITIES 1.1787      0.8484     1.0127     1.3717  
#>
#> Concordance= 0.815  (se = 0.025 )
#> Likelihood ratio test= 130.1  on 4 df,   p=<2e-16
```

```
#> Wald test          = 117.9  on 4 df,   p=<2e-16
#> Score (logrank) test = 165.9  on 4 df,   p=<2e-16
```

De la compleja salida del modelo, deben resaltarse y explicarse los siguientes puntos:

- Apartado 1: Fórmula del modelo, tamaño muestral y número de eventos.
- Apartado 2: Tabla con los coeficientes del modelo y su p-valor.
- Apartado 3: Tabla con los Hazard Ratio (exponencial de los coeficientes de la tabla anterior) y sus intervalos de confianza (lower .95 y upper .95).
- Apartado 4: parámetros de bondad de ajuste del modelo.

Por tanto, de este modelo se pueden concluir las siguientes interpretaciones:

- Un paciente diagnosticado de COVID-19 tiene 3.6 veces más riesgo de fallecer que un paciente sano.
- Una mujer tiene un 67.6 % menos riesgo de fallecer que un hombre.
- Por cada comorbilidad, el riesgo de fallecer aumenta un 17.9 %.

36.5. Conclusión

Ha sido necesaria una pandemia mundial para que la sociedad empiece a dar visibilidad y reconocimiento no sólo a la bioestadística, sino a la investigación clínica y a la necesidad de gestionar el uso masivo de datos en salud. A pesar de los múltiples estudios y experiencias pasadas que llamaban a la prudencia y a la acción concreta si se daba una situación similar, el mundo ha sido incapaz de actuar convenientemente. Esto último se ve reflejado en el mínimo aumento de inversión, reconocimiento y notoriedad no sólo en investigación o desarrollo, sino en el apoyo a la ciencia.

Es, quizás, la paradoja más extraña pero que representa el dicho popular:

La sociedad no avanzará si la ciencia no lo hace primero.

Capítulo 37

Messi y Ronaldo: dos ídolos desde la perspectiva de los datos

Borja Andrino Turón

EL PAÍS

37.1. Motivación

El uso de estadísticas avanzadas en los deportes, especialmente en el fútbol, ha despegado en los últimos años. Una buena señal de su irrupción es la apuesta de algunos medios deportivos — como FiveThirtyEight o The Athletic — por contenidos basados en el análisis y la visualización de estas estadísticas para explicar las fortalezas y debilidades de jugadores y equipos. Además, la generación de estadísticas avanzadas, como los goles esperados, la amenaza o el valor con balón están comenzando a sustituir a las métricas tradicionales en la narración y las crónicas de los encuentros.

37.2. Las estadísticas y el fútbol

En el presente capítulo se usarán estadísticas de la web especializada Fbref.com para visualizar el dominio de Cristiano Ronaldo y Lionel Messi durante más de 15 años. Para usar estos datos podríamos usar técnicas de *web scraping* esta página web o usar la librería `worldfootballR`, desarrollada por Jason Zivkovic. La librería permite obtener datos de diferentes plataformas.

La publicación y explotación de estadísticas avanzadas es reciente, de las últimas seis temporadas, con lo que para analizar las carreras completas de estos dos jugadores tendremos que conformarnos, de momento, con métricas tradicionales.

En la Fig. 37.1 se ve la evolución de las principales cifras que definen a un atacante: los goles y las asistencias. Esta estandarización nos permite poder comparar ambos jugadores independientemente del número de minutos, aunque se ha añadido un filtro de al menos 1.000 minutos jugados en la temporada para evitar ruido.

Para realizar el gráfico, se toman los datos originales y se filtran para que solo aparezcan los jugadores seleccionados, en las temporadas con muestra suficiente. A continuación, se seleccionan las columnas que se usarán en el *plot* y se giran las dos métricas para poder añadirlas en un único `geom_line()`.

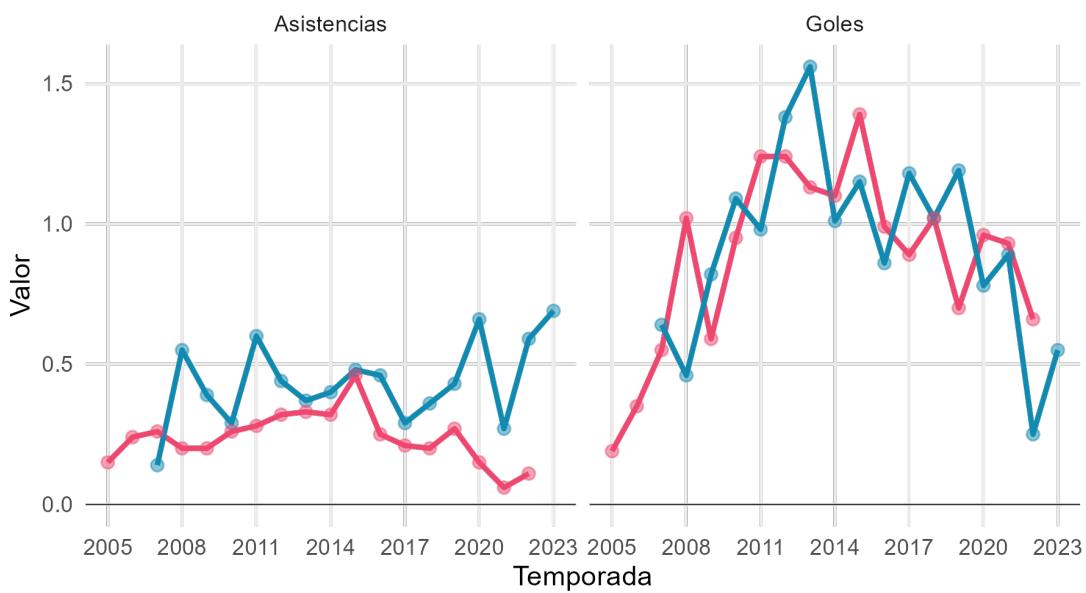
```
pacman::p_load(CDR, tidyverse, janitor, ggbeeswarm, here,
                 patchwork, ggtext, ggrepel)

datos_players |>
  filter(player %in% c("Cristiano Ronaldo", "Lionel Messi"),
         min_playing > 1000) |>
  select(season_end_year, player, Goles = gls_per, Asistencias = ast_per) |>
  pivot_longer(c(Goles, Asistencias), names_to = "metric", values_to = "value") |>
  ggplot(aes(x = season_end_year, y = value, color = player)) +
  geom_line(size = 1) +
  geom_point(size = 2, alpha = 0.5) +
  scale_color_manual(values = c("Lionel Messi" = "#118ab2",
                                "Cristiano Ronaldo" = "#ef476f")) +
  scale_x_continuous(breaks = seq(2005, 2023, 3)) +
  geom_hline(yintercept = 0, size = 0.2) +
  facet_wrap(~metric) +
  labs(title = "Evolución de los goles y asistencias por cada 90 minutos de<br><b>Cristiano Ronaldo</b> y <b>Lionel Messi</b>",
       x = "Temporada", y = "Valor", caption = "Fuente: Fbref.com") +
  theme_minimal() +
  theme(legend.position = "none",
        panel.grid.minor = element_blank(),
        plot.title = element_markdown(margin=margin(0,0,10,-30),
                                       size=12))
```

La Fig. 37.1 arroja un dato increíble, durante 10 años, tener a Messi o Cristiano en el campo significaba contar en ese partido con un gol y casi media asistencia.

Pero la visualización solo nos habla de estos dos futbolistas. Para compararlos con otros jugadores se puede calcular el percentil de goles y asistencias por 90 minutos, temporada a temporada, de los jugadores que hayan jugado más de 1.000 minutos (véase Fig. 37.2). El resultado de nuevo es impactante: durante 13 temporadas Messi y Cristiano han estado entre el 1% de jugadores con más goles. Además, el argentino ha terminado la temporada entre el 1% con más asistencias en 9 ocasiones.

Evolución de los goles y asistencias por cada 90 minutos de Cristiano Ronaldo y Lionel Messi



Fuente: Fbref.com

Figura 37.1: Evolución de goles y asistencias por 90 minutos de Cristiano y Messi desde 2005

```

percentiles_to_plot <-
  datos_players |>
  clean_names() |>
  filter(min_playing > 1000) |>
  select(season_end_year, player, min_playing, gls_per, ast_per) |>
  group_by(season_end_year) |>
  mutate(across(c(gls_per, ast_per), ntile, 100,
                .names = "{.col}_centil")) |>
  ungroup() |>
  mutate(highlighted_player = if_else(player %in%
                                         c("Cristiano Ronaldo", "Lionel Messi"),
                                         T,
                                         F)) |>
  select(season_end_year, player, highlighted_player,
         Goles = gls_per_centil, Asistencias = ast_per_centil)

percentiles_to_plot |>
  pivot_longer(c(Goles, Asistencias), names_to = "metric", values_to = "value") |>
  ggplot(aes(x = season_end_year, y = value, group = season_end_year)) +
  geom_jitter(aes(alpha = highlighted_player, color = player)) +
  scale_color_manual(values = c("Lionel Messi" = "#118ab2",
                                "Cristiano Ronaldo" = "#ef476f")) +
  geom_hline(yintercept = 0, size = 0.1) +
  labs(title = "Percentil de goles y asistencias por cada 90<br>minutos de <b>Temporada</b> para <b>Cristiano Ronaldo</b> y <b>Lionel Messi</b>",
       x = "Temporada", y = "Percentil", caption = "Fuente: Fbref.com") +
  facet_wrap(~metric, scales = "free") +
  scale_x_continuous(breaks = seq(2005, 2023)) +
  scale_alpha_manual(values = c(0.01, 1)) +
  coord_flip() +
  guides(alpha = "none") +
  theme_minimal() +
  theme(legend.position = "none",
        panel.grid.minor = element_blank(),
        plot.title = element_markdown(margin=margin(0,0,0,-30), size=12))

```

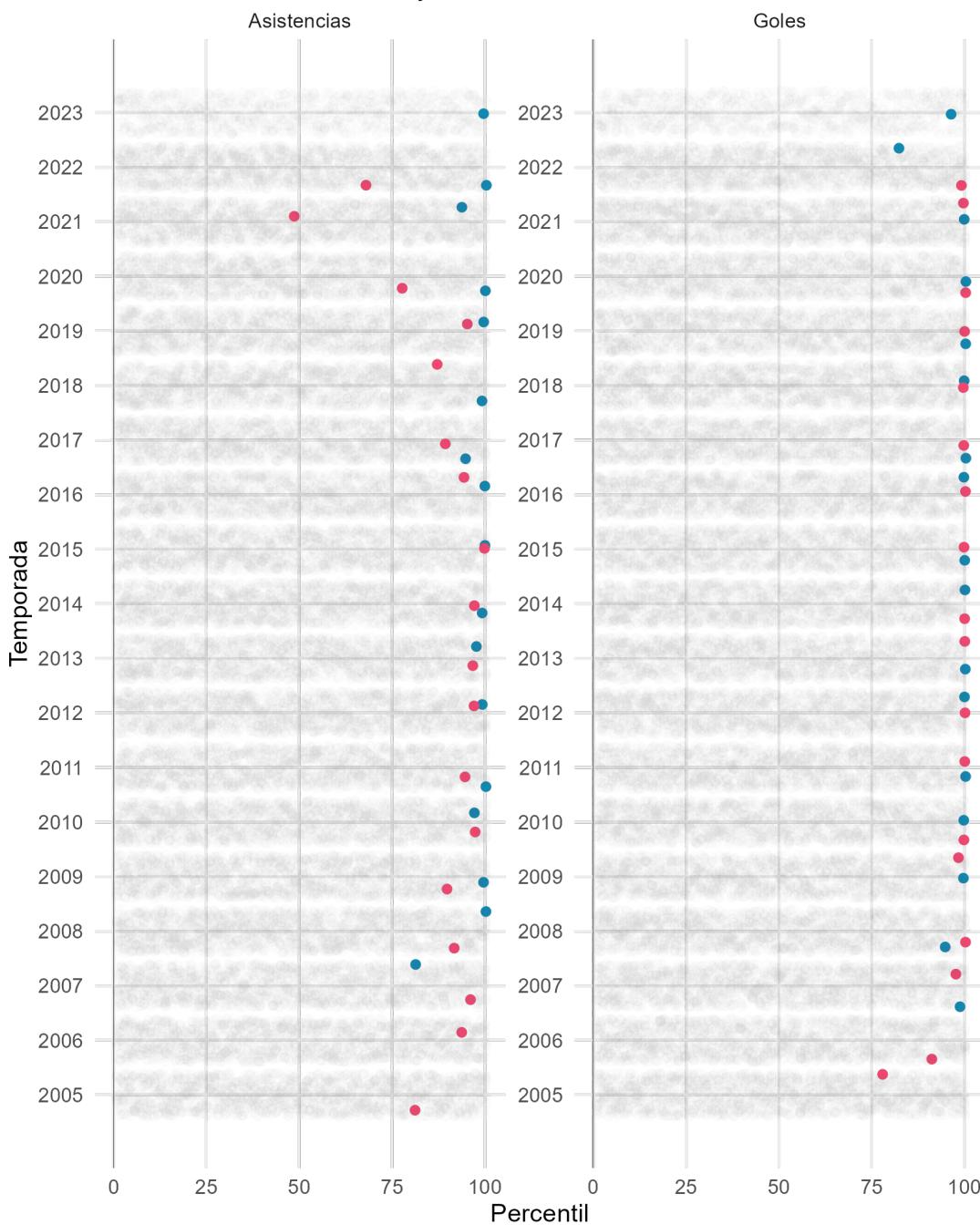
Desde la temporada 2017/18 en esta web publican estadísticas avanzadas de jugadores por partido y temporada. En la Fig. 37.3 se representan los goles esperados (miden cómo de probable es el gol dado un disparo) y las asistencias esperadas (suma de los goles esperados que suman los pases que desembocan en un tiro) por 90 minutos de los jugadores con más de 1.000 minutos. De nuevo el gráfico destaca a nuestros dos protagonistas, que se sitúan en el arco más alejado del origen de coordenadas, donde se juntan los jugadores con mejores números de asistencias y goles esperados.

```

expected_data <-
  datos_players |>
  clean_names() |>

```

Percentil de goles y asistencias por cada 90 minutos de **Cristiano Ronaldo** y **Lionel Messi**



Fuente: Fbref.com

Figura 37.2: Percentil de goles y asistencias por 90 minutos cada temporada desde 2005

```

filter(season_end_year >= 2018,
       min_playing > 1000,
       x_g_per > 0 | x_ag_per > 0) |>
mutate(highlighted_player = if_else(player %in% c("Cristiano Ronaldo", "Lionel
→ Messi"),
                                     T,
                                     F),
       label = if_else(player %in% c("Cristiano Ronaldo", "Lionel Messi"),
                      as.character(season_end_year),
                      NA_character_))

expected_data |>
  select(season_end_year, player, highlighted_player, label,
         Goles = x_g_per, Asistencias = x_ag_per) |>
  ggplot(aes(x = Asistencias, y = Goles)) +
  geom_point(aes(alpha = highlighted_player,
                 color = player)) +
  geom_text_repel(aes(label = str_sub(label, 3, 4))) +
  scale_color_manual(values = c("Lionel Messi" = "#118ab2",
                                "Cristiano Ronaldo" = "#ef476f")) +
  geom_hline(yintercept = 0, size = 0.2) +
  geom_vline(xintercept = 0, size = 0.2) +
  scale_alpha_manual(values = c(0.1, 1)) +
  labs(title = "Goles y asistencias esperadas por 90 minutos cada temporada de<br><b
→ style='color:#ef476f;'>Cristiano Ronaldo</b>, <b style='color:#118ab2;'>Lionel
→ Messi</b> y el resto de jugadores",
       x = "Asistencias esperadas", y = "Goles esperados", caption = "Fuente:
       → Fbref.com") +
  guides(alpha = "none") +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_markdown(margin=margin(0,0,10,-30),
                                       size=12),
        legend.title = element_blank())

```

La métrica de goles esperados permite también hablar de efectividad. Cuando un jugador suma más goles con sus disparos de lo que era esperable su efectividad es alta; cuando por el contrario el jugador termina anotando menos goles de los que se preveían por sus disparos su efectividad es baja. En la Fig. ?? se muestra para cada jugador y temporada esta relación. Se vuelve a observar cómo Cristiano y Messi destacan en la generación de goles esperados, aunque hay una ligera diferencia: entre 2018 y 2021 la efectividad del argentino fue mayor que la del portugués. Los puntos de Cristiano se sitúan sobre la línea que representa lo esperado: mismo número de goles que probabilidad de que los disparos acaben en gol. Los de Messi se sitúan por encima, ha anotado más goles que los que sus disparos hacían prever.

```

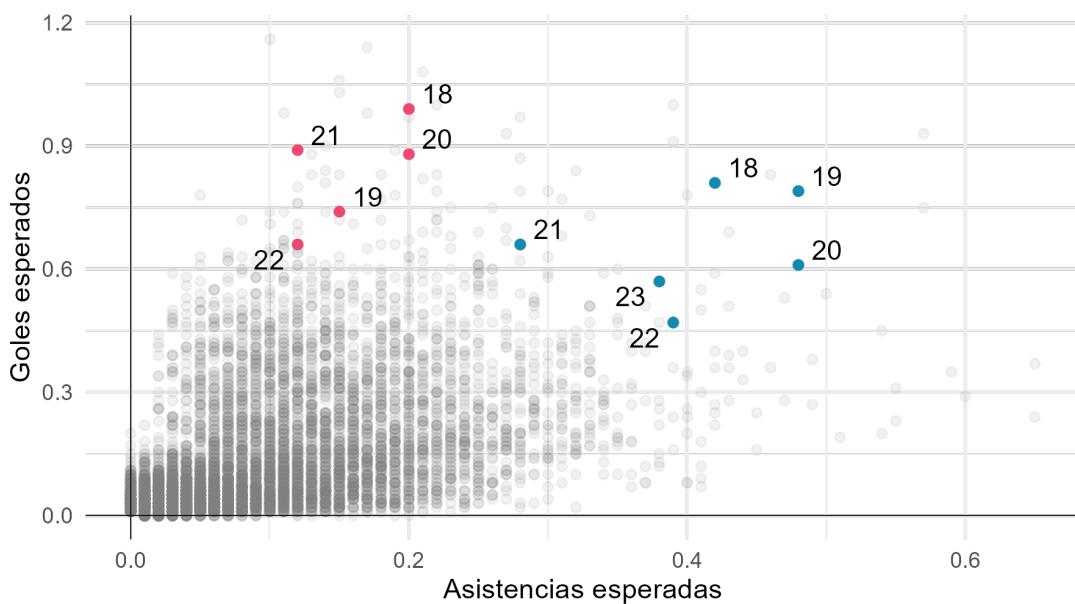
expected_data |>
  select(season_end_year, player, highlighted_player, label,

```

37.2. Las estadísticas y el fútbol

577

Goles y asistencias esperadas por 90 minutos cada temporada de **Cristiano Ronaldo**, **Lionel Messi** y el resto de jugadores



Fuente: Fbref.com

Figura 37.3: Goles y asistencias por jugador y temporada

```

Goles = gls_per, `Goles esperados` = x_g_per) |>
ggplot(aes(x = `Goles esperados`, y = Goles)) +
geom_point(aes(alpha = highlighted_player,
               color = player)) +
geom_text_repel(aes(label = str_sub(label, 3, 4))) +
scale_color_manual(values = c("Lionel Messi" = "#118ab2",
                             "Cristiano Ronaldo" = "#ef476f")) +
geom_hline(yintercept = 0, size = 0.2) +
geom_vline(xintercept = 0, size = 0.2) +
geom_abline(slope = 1) +
geom_text(x = 1, y = 1.4,
          label = "Por encima de la línea\nlos jugadores más efectivos",
          size = 3, hjust = 1, vjust = 0.5) +
geom_curve(x = 1.01, y = 1.4, xend = 1.2, yend = 1.2,
           size = 0.2, curvature = -0.25, arrow = arrow(length = unit(0.02, "npc")))
           +
scale_alpha_manual(values = c(0.1, 1)) +
scale_x_continuous(limits = c(0, 1.5)) +
scale_y_continuous(limits = c(0, 1.5)) +
labs(title = "Goles esperados y conseguidos por 90 minutos cada temporada de<br><b>Cristiano Ronaldo</b>, <b>Lionel Messi</b> y el resto de jugadores",
     x = "Goles esperados", y = "Goles", caption = "Fuente: Fbref.com") +
guides(alpha = "none") +
theme_minimal() +
theme(legend.position = "none",
      plot.title = element_markdown(size=12),
      legend.title = element_blank())

```

Con estos gráficos se puede hacer una primera evaluación de los datos de estos dos grandes jugadores (y de cualquier otro) y quizás no logremos contestar a la pregunta de quién ha sido el mejor, aunque para algunos con esto ya esté claro.

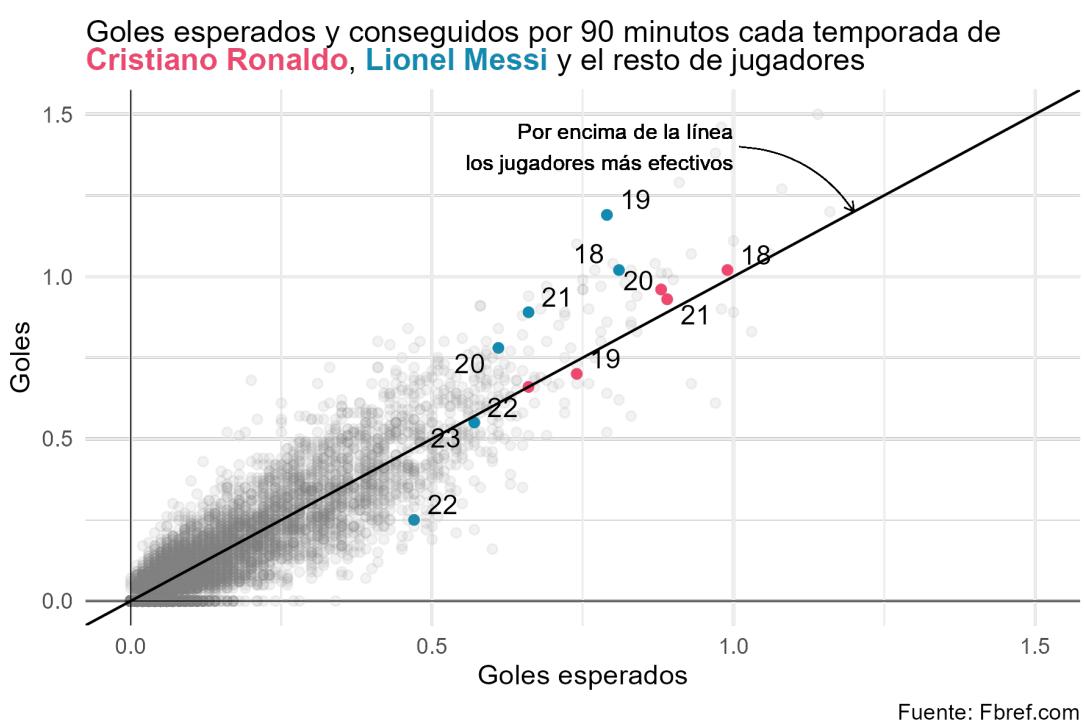


Figura 37.4: Goles esperados y anotados por jugador y temporada

Capítulo 38

Un dato sobre el cambio climático

Dominic Royé

Fundación de la Investigación del Clima

38.1. Introducción

La temperatura media global en la superficie ha aumentado en 1.1 °C desde la era preindustrial (1880-1900). A pesar de parecer un leve incremento en la temperatura, implica un aumento significativo en el calor acumulado del sistema tierra. Cuando se combinan el aumento de la temperatura con respecto a la superficie terrestre y el océano, la tasa promedio es de 0,08 °C por década desde 1880. Sin embargo, la tasa promedio de aumento desde 1981 ha sido más del doble: con 0,18°C. Los océanos se caracterizan por una menor tasa de calentamiento debido a su capacidad calorífica. No obstante, son los océanos los que absorben la mayoría del calor adicional del planeta debido al cambio climático¹.

Entre todas las regiones, la región mediterránea se está calentando un 20% más rápido que el promedio mundial. Este lugar representa actualmente el punto crítico más importante del cambio climático, donde se percibe un significativo aumento de las vulnerabilidades. La temperatura de las aguas superficiales en el Mediterráneo ha estado subiendo 0,34°C cada década desde principios de los 80 (Cramer et al. (2020)).

En este caso práctico con datos sobre el cambio climático se tratan las anomalías de la temperatura superficial del mar Mediterráneo en los meses estivales desde 1982 a 2022. Se hará uso del dataset con el nombre “NOAA CDR OISST v02r01”, una interpolación óptima diaria de temperatura superficial del mar (OISST, por sus siglas en inglés) con una resolución de 1/4 grados (27 km). Los datos los proporciona la *National Oceanic and Atmospheric Administration*

¹<https://www.ncei.noaa.gov/news/global-climate-202112>

(NOAA) con campos completos de temperatura del océano construidos mediante la combinación de observaciones ajustadas por sesgo de diferentes plataformas (satélites, barcos, boyas) en una cuadrícula global regular, con lagunas estimadas por interpolación (https://developers.google.com/earth-engine/datasets/catalog/NOAA_CDR_OISST_V2_1). El geoprocесamiento en nube está explicado en el Cap. @ref{geoproc}.

38.2. Consideraciones iniciales

La información espacio-temporal es clave en muchas disciplinas, especialmente en la climatología o la meteorología, y ello hace necesario disponer de un formato que permita una estructura multidimensional. Además, es importante que ese formato tenga un alto grado de compatibilidad de intercambio y pueda almacenar un elevado número de datos. Estas características llevaron al desarrollo del estándar abierto netCDF (*Network Common Data Form*). El formato netCDF es un estándar abierto de intercambio de datos científicos multidimensionales que se utiliza con datos de observaciones o modelos, principalmente en disciplinas como la climatología, la meteorología y la oceanografía. Se trata de un formato espacio-temporal con una cuadrícula regular o irregular. La estructura multidimensional en forma de matriz (array) permite usar no sólo datos espacio-temporales, sino también multidimensionales. Los datos multidimensionales en formato *geotiff* son menos comunes, pero también se pueden llegar a usar. Además, es posible crear objetos multidimensionales importando múltiples archivos ráster.

38.3. Paquetes

El manejo de datos en formato netCDF o múltiples archivos ráster es posible a través de varios paquetes de forma directa o indirecta. Destaca el paquete **ncdf4**, específicamente diseñado para esto, del que hacen uso también otros paquetes de forma oculta. El manejo con **ncdf4** es algo complejo, particularmente por la necesidad de gestionar la memoria RAM cuando se tratan con grandes conjuntos de datos o también por la forma de manejar la clase *array*. Otro paquete muy potente es **terra**, clave en el trabajo con datos ráster y permite usar sus funciones también para el manejo del formato netCDF.

```
# paquetes
library("tidyverse")
library("sf")
library("terra")
library("lubridate")
library("fs")
library("patchwork")
library("giscoR")
library("scales")
library("rmapshaper")
library("RColorBrewer")
library("CDR")
```

38.4. Visualización de mapas “pequeños múltiples”

Una forma muy efectiva para mostrar cambios espacio-temporales son los mapas de pequeños múltiples, donde se representan en una rejilla para cada año las anomalías observadas, lo que permite una comparación sencilla.

38.4.1. Datos

Se importa el polígono del Mar Mediterráneo para limitar los datos al área de interés.

```
data("med_limit")
```

A continuación, se importan todos los años empleando la función `dir_ls()` del paquete `fs` y la función `rast()` de `terra`. La primera función crea un vector de todos los archivos ubicados en la carpeta “data”. Finalmente se renombran todas las capas con los correspondientes años. Es importante que se garantice el correcto orden de los archivos. Siempre que se haya realizado el geoprocесamiento en nube de las anomalías (Cap. @ref{geoproc}) se puede usar la alternativa: `anom <- dir_ls("data-cc", regexp = "tif") |> rast()`.

```
# importar
anom <- dir_ls(system.file("external/data-cc/", package = "CDR"), regexp = "tif") |>
  rast()

# renombrar las capas
names(anom) <- 1982:2022
```

38.4.2. Preparación de los datos

Después de importar el polígono del Mar Mediterráneo, es necesario recortar y enmascarar el área de interés. Para ello, se usa primero la función `crop()` con los límites del Mar Mediterraneo y se pasa el resultado a la función `mask()`. El paquete `terra` necesita los datos vectoriales en su propia clase `SpatVector`, por eso, se pasa con la función `vect()`, que lo convierte de la clase `sf` a `SpatVector`. Finalmente, se reproyectan los rásters a *ETRS89-extended / LAEA Europe* con el código EPSG:3035.

```
anom <- crop(anom, med_limit) |> mask(vect(med_limit))
anom <- project(anom, "EPSG:3035")
```

En la Fig. 38.1 se puede ver el resultados de los primeros años.

```
plot(anom)
```

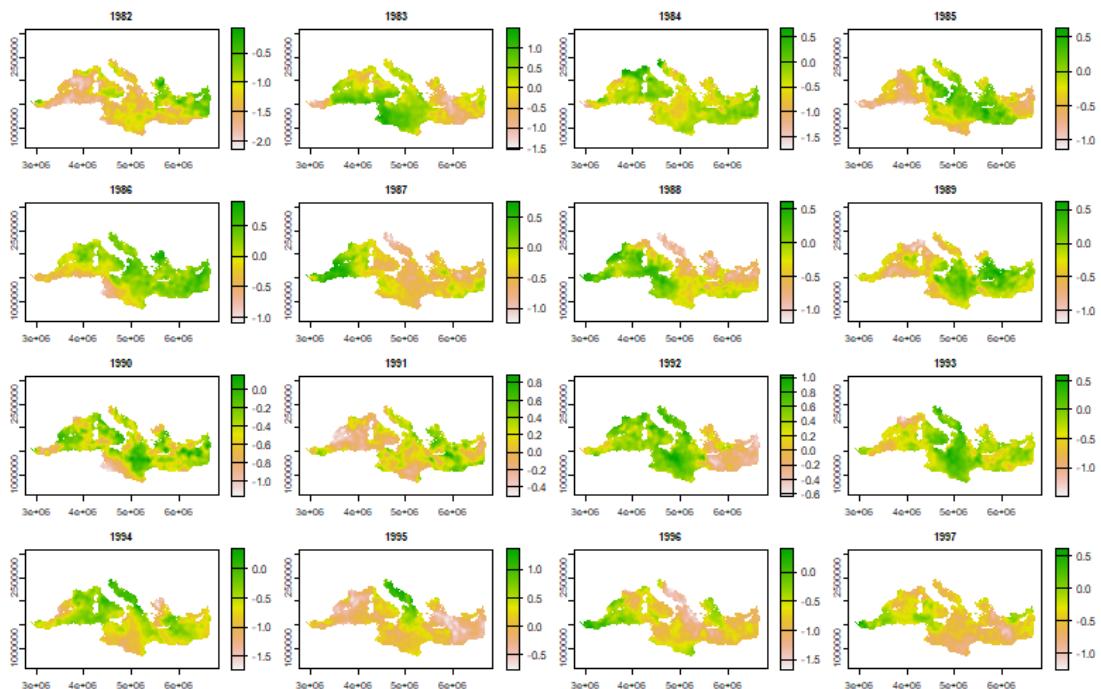


Figura 38.1: Selección de anomalías 1982 a 1997 de los datos brutos.

38.4. Visualización de mapas “pequeños múltiples”

585

Un ráster consiste en latitud, longitud y un único o múltiples valores, también llamados capas. Para poder visualizarlo en `ggplot2`, es necesario convertirlo en un `data.frame`. En este caso, se obtienen 41 columnas para las anomalías, además de las primeras dos que corresponden a la longitud y latitud.

No obstante, es necesario realizar cambios en las distribución de las variables. Ahora mismo se tiene la misma variable, la anomalía, distribuida en muchas columnas, no obstante la estructura adecuada debe ser un conjunto total de dos columnas: una que represente las anomalías y una segunda que contenga los años. Para conseguirlo, se hace uso de `pivot_longer()`, indicando el total de columnas que deben ser fusionadas y los nombres de las dos columnas resultantes.

```
df <- as.data.frame(anom, xy = TRUE)
df <- pivot_longer(df, 3:length(df),
                   names_to = "yr",
                   values_to = "anom")
```

Se añaden los años de la década de los 80 que faltan (1980, 1981) y se limitan las anomalías a un rango entre -2 y +2.

```
df <- bind_rows(df, filter(df, yr == "1982") |>
                  mutate(yr = "1981", anom = NA),
                  filter(df, yr == "1982") |>
                  mutate(yr = "1980", anom = NA)
                ) |>
  mutate(anom2 = case_when(anom > 2 ~ 2,
                           anom < -2 ~ -2,
                           TRUE ~ anom))
```

Previo a la construcción del gráfico, se estima la media de la anomalía global para toda la cuenca mediterránea de cada año. Estos datos se añadirán como texto a cada mapa. Con el objetivo de obtener las coordenadas de la posición en la proyección EPSG:3035, se fija un punto vectorial que se reprojeta.

```
# media global
med_anom <- global(anom, fun = "mean", na.rm = TRUE)
med_anom <- rownames_to_column(med_anom, "yr")

# posición
pos_global <- st_point(c(34.24, 41.5)) |>
  st_sfc(crs = 4326) |>
  st_transform(3035) |>
  st_coordinates()
```

38.4.3. Construcción del gráfico de múltiples mapas

En el primer paso se definen los estilos partiendo de `theme_void()`, configurando los títulos, la leyenda y el color de fondo en `theme()`.

```
theme_SST_facet <- function(base_family = "Bahnschrift",
  base_size = 11,
  base_line_size = base_size/22,
  base_rect_size = base_size/22) {

  theme_void(base_family = base_family, base_size = base_size,
    base_line_size = base_line_size, base_rect_size = base_rect_size) +
  theme(strip.text = element_text(colour = "white",
    face = "bold",
    size = 12,
    margin = margin(b = 15)),
  legend.text = element_text(colour = "white"),
  legend.position = "top",
  legend.justification = .48,
  plot.margin = margin(20, 20, 20, 20),
  plot.title = element_text(colour = "white", size = 30, hjust = .5),
  plot.subtitle = element_text(colour = "white", size = 15, hjust = .5,
    margin = margin(t = 5, b = 5)),
  plot.caption = element_text(colour = "white", size = 10, hjust = 0),
  plot.background = element_rect(fill = "grey10", colour = NA),
  panel.spacing = unit(2, "lines"),
  panel.background = element_rect(fill = "grey10", colour = NA))
}
```

Para representar datos ráster en forma de xyz se utiliza `geom_tile()` o `geom_raster()` en `ggplot2`. La última geometría requiere una rejilla regular. Para este primer ensayo, se filtra sólo el año 2003, y además se añade, con `geom_sf()`, el límite del Mar Mediterráneo. La función `coord_sf()` permite fijar una proyección para objetos `sf`, y por último, se cambia el estilo definido anteriormente.

```
filter(df, yr == "2003") |>
  ggplot() +
  geom_tile(aes(x, y, fill = anom2)) +
  geom_sf(data = med_limit,
    fill = NA, colour = "white", size = .1) +
  coord_sf(crs = 3035) +
  theme_SST_facet()
```

Siguiendo el ejemplo, se modifica la gama de colores con `scale_fill_gradientn()`, en la que se pasa la paleta de colores, los extremos de valores, se reajustan los valores a una escala divergente se definen las etiquetas y sus posiciones. Dentro de la función `guides()`, se cambia el ancho y altura de la barra colores empleando la función `guide_colorbar()`.

Las geometrías `geom_point()` y `geom_text()` añadirán la información de la anomalía global. La posición se pasa de forma directa en `aes()`; además, se definen el color y el tamaño de texto. El objetivo es situar el texto a la derecha del punto. Por esa razón, es necesario un ajuste en longitud indicando un valor correspondiente en el argumento `nudge_x` en la unidad del sistema de coordenadas (SC). Recuérdese que el SC está en metros.

38.4. Visualización de mapas “pequeños múltiples”

587

La función `number()` del paquete `scales` facilita formatear las cifras con un decimal y los símbolos negativo y positivo.

```
# gama de colores
rdbu_pal <- rev(brewer.pal(11, "RdBu"))

# mapa 2003
filter(df, yr == "2003") |>
ggplot() +
  geom_tile(aes(x, y, fill = anom2)) +
  geom_sf(data = med_limit,
          fill = NA,
          colour = "white",
          size = .1) +
  geom_point(data = filter(med_anom, yr == "2003"),
             aes(x = pos_global[1,1], y = pos_global[1,2], fill = mean),
             size = 3.5, shape = 21, colour = "white") +
  geom_text(data = filter(med_anom, yr == "2003"),
            aes(x = pos_global[1,1], y = pos_global[1,2],
                label = number(mean, .1, style_positive = "plus")),
            size = 3.5, nudge_x = 70000, colour = "white") +
  scale_fill_gradientn(colours = rdbu_pal, na.value = NA,
                        values = rescale(c(-2, 0, 2)),
                        limits = c(-2, 2),
                        breaks = c(-2, -1.5, -1, -0.5, 0,
                                   .5, 1, 1.5, 2),
                        labels = c("< -2.0", "-1.5", "-1.0", "-0.5", "0.0",
                                   "0.5", "1.0", "1.5", "> 2.0")) +
  guides(fill = guide_colorbar(barwidth = 20,
                               barheight = .5)) +
  coord_sf(crs = 3035) +
  theme_SST_facet()
```

En los datos se ha añadido dos años con valores perdidos (1980 y 1981) con el objetivo de obtener por cada fila 10 años, evitando que el *facet grid* empiece por 1982 sin posibilidad de mantener en cada fila la década correspondiente. No obstante, para que los límites de la cuenca mediterránea no aparezca en las facetas de los años 1980/81, se debe repetir la geometría para todos los años.

```
med <- slice(med_limit, rep(1, 41)) |>
  dplyr::select(geometry) |>
  mutate(yr = as.character(1982:2022))
```

A continuación, se construye todo el gráfico con todas las facetas de mapas. Lo único nuevo es la función `facet_wrap()`, en la que se indica la variable por la que se crean las facetas. A diferencia de `facet_grid()`, esta variante permite fijar el número de filas y/o columnas. Además, se pasa una función menor en la función `labeller()` en el mismo argumento. Esta

función permite modificar las etiquetas de las facetas (aquí únicamente el texto de los años 1980/81).

```
# paso 1
g <- ggplot(df) +
  geom_tile(aes(x, y, fill = anom2)) +
  geom_sf(data = med, fill = NA, colour = "white", size = .1) +
  geom_point(data = med_anom,
             aes(x = pos_global[1,1], y = pos_global[1,2], fill = mean),
             size = 3.5, shape = 21, colour = "white") +
  geom_text(data = med_anom,
            aes(x = pos_global[1,1], y = pos_global[1,2],
                label = number(mean, .1, style_positive = "plus")),
            size = 3.5, nudge_x = 700000, colour = "white") +
  scale_fill_gradientn(colours = rdbu_pal,
                        na.value = NA, values = rescale(c(-2, 0, 2)),
                        limits = c(-2, 2),
                        breaks = c(-2, -1.5, -1, -0.5, 0,
                                   .5, 1, 1.5, 2),
                        labels = c("< -2.0", "-1.5", "-1.0", "-0.5", "0.0",
                                   "0.5", "1.0", "1.5", "> 2.0")) +
  guides(fill = guide_colorbar(barwidth = 20,
                               barheight = .5)) +
  facet_wrap(yr ~ .,
             ncol = 10,
             labeller = labeller(yr = function(lab){
               ifelse(lab %in% c("1980", "1981"), "", lab)}))
```

Finalmente, se combinan el objeto con definiciones finales, como los títulos, el sistema de coordenadas y el estilo. Es importante indicar `clip = “off”`, dado que en caso contrario se cortan visualmente los valores de las anomalías globales al encontrarse fuera de los límites de cada mapa.

```
# paso 2
g <- g + labs(title = "ANOMALÍA ESTIVAL DE LA TEMPERATURA DE SUPERFICIE DEL\nMar
→ Mediterráneo", subtitle = "Periodo de referencia 1982-2010.", fill = "") +
  coord_sf(crs = 3035, clip = "off") +
  theme_SST_facet()
```

A priori, no sería necesario una ampliación del resultado. No obstante, en ocasiones se requiere un mapa de orientación.

38.4.4. Mapa de orientación

A través del paquete `giscoR` se obtienen los límites administrativos, de los que únicamente se queda con una selección. También se limita la extensión a aproxidamente la de la cuenca

38.4. Visualización de mapas “pequeños múltiples”

589

mediterránea. La función `ms_innerlines()` del paquete `rmapshaper` facilita la obtención de los límites compartidos o interiores de los países seleccionados. Los nombres de los países, en forma de código ISO-3, se incluyen con ayuda de `geom_sf_text()`.

```
# límites de países
countries_med <- gisco_get_countries() |>
  filter(ISO3_CODE %in% c("ESP", "MAR", "FRA", "ITA",
    "GRC", "TUR", "DZA", "TUN",
    "LBY", "EGY", "ALB")) |>
  st_crop(xmin = -6, xmax = 36, ymin = 28, ymax = 45)

# límites internos
innerlimit <- ms_innerlines(countries_med)

# mapa
insetp <- ggplot() +
  geom_sf(data = med, size = .4, colour = NA, fill = "grey90") +
  geom_sf(data = innerlimit, size = .2, colour = "white") +
  geom_sf_text(data = countries_med,
    aes(label = ISO3_CODE),
    size = 2, colour = "white", fontface = "bold", nudge_y = .1) +
  coord_sf(crs = 3035, expand = FALSE) +
  theme_void() +
  theme(plot.background = element_blank(),
    panel.background = element_blank())
```

38.4.5. Exportar mapa final

El mapa de orientación se insertará en el gráfico, como elemento adicional, en la esquina derecha-arriba. El paquete `patchwork` puede ayudar a crear composiciones de distintos gráficos. La función empleada `inset_element()` indica la posición relativa en `xmin`, `ymin`, `xmax`, y `ymax`. Es importante recordar que cualquier modificación del tamaño de impresión (veáse `height` y `width` en `ggsave()`), puede llevar a ajustes en la posición. El argumento `align_to = "full"` permite posicionar sobre todo el lienzo.

```
# paso 3
p_final <- g + inset_element(insetp, 0, .75, .25, .95,
  align_to = "full")

ggsave("sst_anom_med2.png",
  p_final,
  bg = "grey10",
  height = 10,
  width = 20,
  unit = "in",
  type = "cairo-png",
  dpi = 400)
```

El resultado final, como gráfico de múltiples mapas, puede verse en la Fig. 38.2. Los mapas muestran claramente el efecto del calentamiento global, siendo los año 2003 y 2022 de mayor anomalía positiva. Destaca el hecho de que no ha habido un año con temperaturas más bajas de lo normal desde el año 1997.

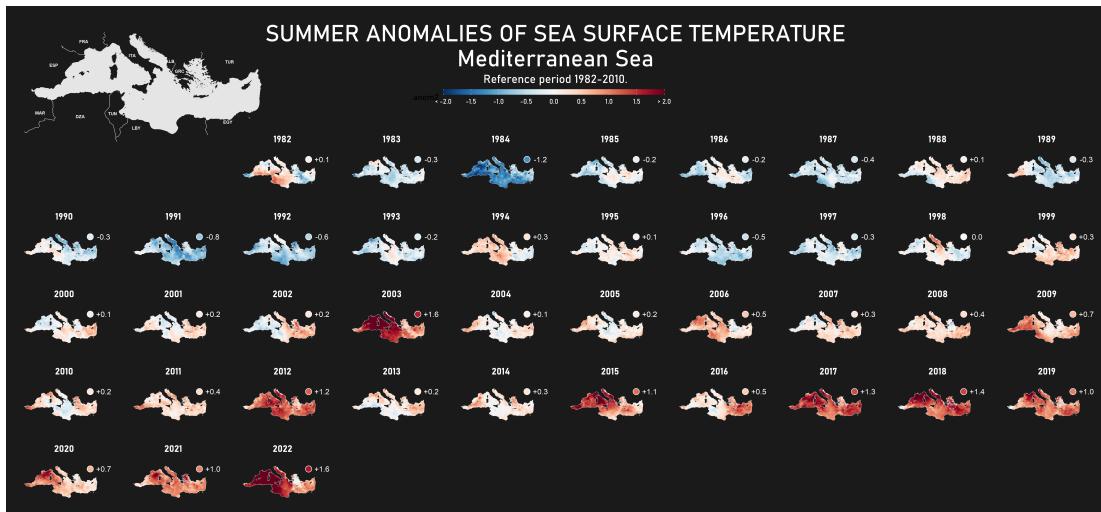


Figura 38.2: Anomalía estival de la temperatura de superficie del mar

Capítulo 39

Predicción de consumo eléctrico con redes neuronales

Jose Manuel Sanz Candales

Red Eléctrica de España

39.1. Introducción

Red Eléctrica, como Operador del Sistema, tiene como principal misión garantizar la continuidad del suministro eléctrico en España. Para ello, entre otras muchas tareas, se desarrollan, evolucionan y mantienen algoritmos de previsión del consumo eléctrico y de la producción con las principales energías renovables (eólica y solar) para distintos horizontes (próximas horas, días, meses, años, etc.) y a distintas escalas temporales (anual, horario, quinceminal).

Este caso de uso se sitúa en el departamento de Ciencia de Datos del Operador del Sistema. Es el principio del año 2018, y el área de planificación de la empresa solicita una **predicción de el consumo eléctrico en España** para el año actual y el siguiente (2018 y 2019).

IMPORTANTE: Este desarrollo no está previsto en el presupuesto del año, por lo que tanto el software como los datos de entrada deben ser, a ser posible, gratuitos.

39.2. Datos de entrada

Respecto a los datos de entrada para el modelo, se requiere tanto una serie histórica de la variable a predecir así como de otras variables que sean capaces de explicar adecuadamente el comportamiento del consumo eléctrico. En este caso es necesario utilizar un modelo de aprendizaje supervisado de regresión, dado que el consumo eléctrico es una variable numérica continua.

En cuanto a la serie histórica de consumo eléctrico anual, Red Eléctrica, en su Web corporativa, publica datos estadísticos accesibles de forma abierta. Sin embargo, el histórico publicado comienza en 2012 y sería conveniente tener un periodo de tiempo más amplio para un entrenamiento adecuado de los modelos potencialmente candidatos a ser utilizados. Se realiza una búsqueda de otras fuentes y, afortunadamente, se encuentra que el Instituto para la Diversificación y Ahorro de la Energía (en adelante IDAE) publica datos desde 1990, con agregación anual, del consumo final de energía eléctrica en miles de toneladas equivalentes de petróleo -ktep-.

Como los datos se deben entregar en MWh, las unidades de la predicción resultante se tendrán que convertir con el coeficiente que indican en la Web del IDAE (1 MWh = 0,086 tep), pero en los modelos se utilizarán las unidades originales porque más adelante se comprobará que dichas unidades resultan muy útiles para ver cómo se relaciona el consumo eléctrico con la variable predictora que se va a utilizar.

De este modo se consigue, por tanto, una parte de los datos necesarios para entrenar los modelos predictivos: **la serie histórica de nuestra variable target (o variable a predecir)**.

Para completar el conjunto de datos del modelo se necesitan, además, **las features o variables explicativas**. Se sabe que, históricamente, las variaciones interanuales de el consumo eléctrico dependen del comportamiento de la economía de una forma directa: si la economía crece, también crece el consumo eléctrico. Como indicador del comportamiento de la economía se decide tomar el PIB per cápita que se puede encontrar en el siguiente enlace, disponible de forma pública en Expansión - datos macro: <https://datosmacro.expansion.com/pib/espana>.

Adicionalmente, se utilizarán otras dos variables explicativas, relacionadas con el mercado inmobiliario y con el empleo, respectivamente. Dado que no son públicas, están anonimizadas y escaladas entre 0 y 1 (dividiendo todos los valores de cada variable entre el mayor de su serie). Debido a que se dispone de datos de estas dos variables desde el año 2000, el dataset comienza en este año.

Una vez definido el dataset de entrada para los modelos (como se verá a continuación, es un conjunto de datos muy pequeño y sencillo), se puede comenzar a construir el modelo en **R**.

39.3. Modelización

En la siguiente celda de código se lee el conjunto de datos, `consumoelectricoanual_2`, del paquete CDR, se convierte al formato `data.table` a `data.frame` y se visualizan sus primeras 3 filas:

```
library(CDR)
df <- CDR::consumoelectricoanual_2
class(df) <- class(as.data.frame(df))
head(df, 3)
#>   Año    PIB Consumo     Inmob     Empleo
#> 1 2000 15.97 16.205 0.7525093 0.6561190
#> 2 2001 17.20 17.279 0.7687356 0.6832698
#> 3 2002 18.09 17.671 0.7836143 0.7104178
```

En este caso, ya se dispone de los datos reales de 2018 y 2019 (esto permitirá validar la precisión del modelo), pero en un caso real, a principios de 2018 el PIB per cápita de 2018 y 2019 será una predicción. Lógicamente, el consumo eléctrico anual también será desconocido, ya que es lo que se necesita predecir. Es decir, se supone que los datos de consumo eléctrico de 2018 y 2019 para el modelo que se va a construir no existen, y no se pueden utilizar ni para entrenar ni para evaluar la precisión del modelo (para ello habrá que utilizar datos pasados).

La siguiente celda de código proporciona la matriz de varianzas-covarianzas de las variables PIB, Consumo, Inmob, Empleo:

```
cormat <- round(cor(df[c("PIB", "Consumo", "Inmob", "Empleo")]), 2)
head(cormat)
#>           PIB Consumo Inmob Empleo
#> PIB     1.00   0.81  0.90   0.93
#> Consumo 0.81   1.00  0.64   0.71
#> Inmob    0.90   0.64  1.00   0.99
#> Empleo   0.93   0.71  0.99   1.00
```

Como se puede observar en la matriz anterior, la correlación entre la variable a predecir y las distintas variables explicativas es fuerte y positiva (es decir, cuando crece una también crece la otra). Si, además, se visualiza la gráfica entre PIB y Consumo en el tiempo, se apreciará de forma aún más clara esta intensa correlación:

```
library("ggplot2")
library("reshape2")
df_m <- melt(df[,c("Año","PIB","Consumo")], id.vars = "Año")
options(repr.plot.width = 15, repr.plot.height = 8)
ggplot(df_m, aes(Año, value, col = variable)) +
  geom_line(size = 2.5)
```

En la Fig. 39.1 se observa que las curvas que representan la evolución temporal de ambas variables están prácticamente superpuestas, pero desde 2006 líneas se separan. ¿A qué puede deberse? Uno de los principales motivos probablemente sean las medidas de eficiencia energética que se han ido introduciendo en los últimos lustros (iluminación led, electrodomésticos, dispositivos con menor consumo, etc.).

Una vez se han explorado los datos (en este caso ha sido muy breve, pero es muy habitual en proyectos reales que la exploración y limpieza de los datos requiera en torno al 80 % del tiempo), se procederá a dividir el conjunto de datos (desde 2000 hasta 2017, ya que 2018 y 2019 son los años a los que se pretende dar respuesta, por lo que se supone que no es conocido todavía) en dos partes:

- (I) entrenamiento+validación (90 % de las filas)
- (II) test (10 % restante)

Previamente a esto, se debe escalar también la variable PIB a valores entre 0 y 1, para que la red neuronal funcione de forma correcta:

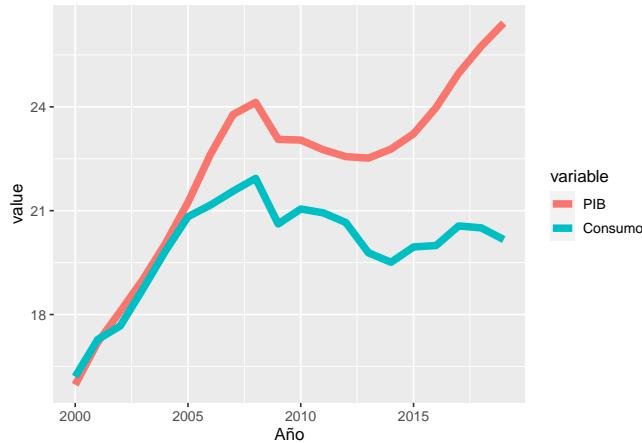


Figura 39.1: Evolución del PIB y el Consumo

```
# estandarizar la variable explicativa "PIB" entre 0 y 1
df$PIB = df$PIB/max(df$PIB)
set.seed(123)
df_aux <- df[df$Año < 2018, ]
n <- nrow(df_aux)
trainIndex <- sample(1:n, size = round(0.85 * n), replace = FALSE)
df_train <- df_aux[trainIndex, ]
df_test <- df_aux[-trainIndex, ]
df_test
```

Ahora se deberían probar distintos modelos de machine learning y comparar sus resultados para determinar cuál es el más preciso para este conjunto de datos. En este ejemplo, por simplicidad no se incluye este proceso de prueba y comparación entre distintos modelos, que en el caso de uso real da lugar elegir una red neuronal simple o perceptrón multicapa (véase Cap. 18), también conocido por su acrónimo en inglés MLP (*Multi Layer Perceptron*), que se utilizará más adelante en este capítulo al obtener los mejores resultados.

Para elegir los hiperparámetros que mejor resultado obtienen para el modelo se van a utilizar dos técnicas que son *grid search* (para probar distintas combinaciones de hiperparámetros) y *cross validation* (para entrenar y validar aprovechando todos los registros del conjunto de entrenamiento-validación).

En este caso de uso, se van a hacer distintas pruebas combinando el número de neuronas por cada capa oculta. En concreto, en la primera y la segunda capa oculta se va a dejar un número constante de neuronas (5), y es en la tercera capa oculta donde se va a probar con 4 neuronas y 6 neuronas. Es decir, se entrenará un modelo con 5 neuronas en cada capa oculta y otro con 5 neuronas en las dos primeras capas y 6 neuronas en la tercera capa.

En la siguiente celda, se importan los paquetes necesarios (neuralnet y caret), se construye la estructura de la red en la variable ‘grid’ y se define el número de *folds* (en cuántas partes se

divide en conjunto de entrenamiento para entrenar y validar con todos los datos del conjunto) de la validación cruzada. Por último, se entrena el modelo.

El proceso está muy simplificado para que sea fácil de entender. No obstante, lo habitual en la práctica es probar más opciones de *grid search* y hacer una división mayor del conjunto de datos para *cross validation* (es bastante habitual entre 5 y 10 folds):

```
# lee paquetes
library("neuralnet")
library("caret")
# define la estructura de la red
grid <- expand.grid(layer1 = c(5), layer2 = c(5), layer3 = c(4,5))
# establece semilla para que los resultados del entrenamiento sean siempre los mismos
set.seed(123)
# define el número de folds en validación cruzada
train_control <- trainControl(method = "cv",
                                 number = 2,
                                 verbose = TRUE)
# entrenar el modelo
model <- train(Consumo ~ PIB+Inmob+Empleo,
                data = df_train,
                trControl = train_control,
                method = "neuralnet",
                tuneGrid = grid)
```

Para mostrar los resultados se aplica la función ‘print’ sobre la variable ‘model’, cuya salida es el texto comentado debajo de la línea de ‘print’.

```
print(model)
#> Neural Network
#>
#> 15 samples
#> 3 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (2 fold)
#> Summary of sample sizes: 8, 7
#> Resampling results across tuning parameters:
#>
#>   layer3    RMSE      Rsquared   MAE
#>   4        0.9713547  0.7631649  0.8521702
#>   5        0.9452518  0.72532010.8165410
#>
#> Tuning parameter 'layer1' was held constant at a value of 5
#> Tuning parameter 'layer2' was held constant at a value of 5
#> RMSE was used to select the optimal model using the smallest value.
#> The final values used for the model were layer1 = 5, layer2 = 5 and layer3 = 5.
```

El modelo con mejor resultado (menor error en el conjunto de entrenamiento / validación) es

el que tiene 3 capas con 5 neuronas cada una. Con este modelo, se predice el consumo eléctrico con el modelo entrenado para la parte del conjunto de datos que se habían reservado para test (años 2006 y 2007), para comprobar que el modelo generaliza bien (es decir, para datos nuevos los resultados de las predicciones tienen un error del orden de los que resultan del entrenamiento del modelo). Para ello, se calcula, por ejemplo, el MAE de las predicciones para dicho conjunto de test:

```
df_test[c("Año", "Consumo")]
#> Año      Consumo
#> 7 2006 21.163
#> 8 2007 21.564
#> 18 2017 20.559

predict(model, df_test)
#>       7       8       18
#> 21.50816 21.83445 20.12596

MAE_test = (abs(21.163-21.50816)+abs(21.564-21.83445)+abs(20.559-20.12596))/3
MAE_test
```

El modelo seleccionado ya está listo para realizar predicciones de consumo eléctrico para los años solicitados (2018 y 2019). Como se avanzó anteriormente, el objetivo del conjunto de test es comprobar la precisión del modelo con datos totalmente desconocidos para él (es decir, no utilizados en la fase de entrenamiento-validación), principalmente para asegurar que el modelo funciona bien para datos distintos a los utilizados en el entrenamiento.

Para predecir el consumo eléctrico anual para 2018 y 2019, que es el dato que solicitaron desde el área de planificación de la empresa. Simplemente se utiliza la función `predict()` del modelo. Previamente, añade una columna en el conjunto de datos original -df- que contendrá los valores predichos, para más adelante comprobar gráficamente el valor predicho frente al real que, en este caso ficticio, ya es conocido:

```
df["Prediccion_MLP"] <- NA
```

Ahora se hace la predicción del año 2018 y se añade el resultado a esta nueva columna del conjunto de datos:

```
df_pred_2018 <- df[df$Año == 2018, ]
df$Prediccion_MLP[df$Año == 2018] <- predict(model,
  → df_pred_2018[c("PIB", "Inmob", "Empleo")])
```

Se hace también la predicción para el año 2019 y se visualizan las predicciones añadidas al conjunto de datos para ambos años:

39.3. Modelización

597

```
df_pred_2019 <- df[df$Año == 2019, ]
df$Prediccion_MLP[df$Año == 2019] <- predict(model,
  ~ df_pred_2019[c("PIB", "Inmob", "Empleo")])
tail(df, 3)
#>   Año Consumo Prediccion_MLP
#> 18 2017 20.559      NA
#> 19 2018 20.504  20.22787
#> 20 2019 20.166  20.16409
```

Estos son los datos que se entregarían como resultado de la petición de información (convertidos a MWh aplicando el coeficiente que se mencionó en la Secc. 39.2).

Claro está, en el momento que se entregan las predicciones para 2018 y 2019 todavía no se sabría cómo de precisas han sido, pero a principios de 2020 sí es posible calcular la bondad del modelo seleccionado, y es lo que se hará en las siguientes celdas:

```
df_m_mlp <- melt(df[c("Año", "Consumo", "Prediccion_MLP")], id.vars = "Año")
ggplot(df_m_mlp, aes(Año, value, col = variable)) +
  geom_point(size = 2) +
  geom_line()
```

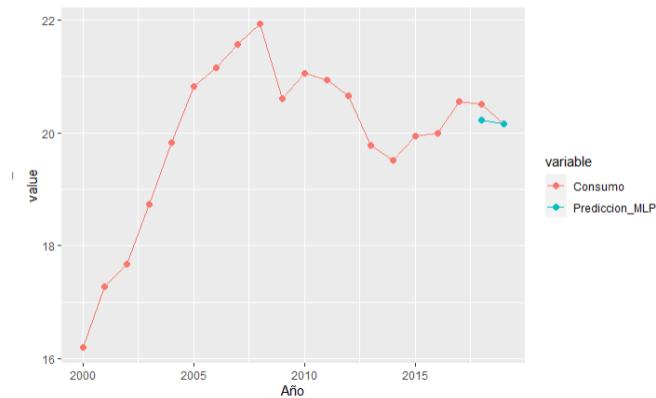


Figura 39.2: Consumo y predicción del modelo de red neuronal MLP

En la Fig. 39.2, las predicciones (puntos azules) tienen unos errores del orden de los que se habían visto en el conjunto de test cuando se hizo el entrenamiento de los modelos, por lo que parece que no hay sobreentrenamiento en el modelo.

Capítulo 40

Implementación de un sistema experto en el ámbito pediátrico de atención primaria

Arturo Peralta^{a,b}, José Ángel Olivas^a y Eusebio Angulo^a

^aUniversidad Internacional de Valencia, ^bUniversidad Internacional de la Rioja

40.1. Introducción

Sin lugar a duda, el análisis de situaciones complejas para la evaluación y toma de decisiones es un proceso para el que tradicionalmente se requiere el apoyo de un especialista dispuesto a poner en uso todo su conocimiento. Sin embargo, el desarrollo de sistemas automáticos capaces de modelar el conocimiento que un experto podría tener sobre un ámbito concreto, y de procesarlo para alcanzar una respuesta adecuada a una consulta relacionada, resulta cada día más extendido como mecanismo de ayuda. A este tipo de herramientas se les denomina Sistemas Expertos (SE).

En este capítulo se introducen los conceptos teóricos fundamentales de la Ingeniería del Conocimiento, los componentes y el funcionamiento de los SE para, posteriormente, presentar cómo su aplicación puede apoyar en el proceso de evaluación clínica en el ámbito pediátrico de atención primaria. Finalmente, se incluye una sencilla implementación en **R** del SE enfocado a la ayuda en esta problemática.

Capítulo 40. Implementación de un sistema experto en el ámbito pediátrico de atención primaria
600

40.2. Marco teórico

Un **Sistema Experto (SE)** es un programa de ordenador que trata de emular el comportamiento de una persona experta en un dominio de conocimiento específico ante un problema que se plantea en dicho dominio y cómo llega a su solución.

La **Ingeniería del Conocimiento** se ocupa, entre otras cosas, del proceso de especificación, análisis y desarrollo de un sistema experto ([Martínez R., 2005](#)).

Los principales componentes de un SE son: (i) La **Base de Hechos (BH)**, que contiene la definición del entorno sobre el que se van a resolver problemas. Hace el papel de “ojos” del SE. (ii) La **Base de Conocimientos (BC)**, que contiene la información del dominio específico, convenientemente representado, capaz de resolver problemas. También puede considerar la representación de incertidumbre. (iii) El **Motor de Inferencias**, que es el proceso de razonamiento que usa el SE. Combina hechos y conocimiento para emitir una conclusión. (iv) El **Interfaz de entrada/salida** para comunicarse con los usuarios y/o expertos.

En la Fig. 40.1 se muestra un diagrama con los principales componentes de un SE.

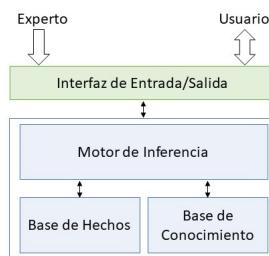


Figura 40.1: Componentes de un SE

Las principales limitaciones en la construcción de SE vienen dadas porque el conocimiento experto humano es experiencia compilada, es heurístico, esto es, basado en experiencia y en reglas prácticas. Es incompleto, impreciso e incierto, y a veces inconsistente y con errores o imprecisiones. Es por ello que las limitaciones de todo SE pueden ser que no conoce lo que conoce ni por qué, carece de imaginación, emociones, inteligencia innata, sentido común, etc. Tiene poco conocimiento de sí mismo, del usuario y del contexto de cada interacción y capacidad de razonamiento limitada por su estrategia de construcción.

En este contexto, los **sistemas de producción** son modelos de cálculo que han probado su eficiencia en la Ingeniería del Conocimiento tanto en el desarrollo de algoritmos de búsqueda como en el modelado de problemas del dominio humano. Sus componentes principales son:

- 1) Las **reglas de producción**: son la forma más extendida de representar el conocimiento, constan de Condiciones (Hipótesis) y Acciones (Conclusiones) y tienen la forma:

```
Si (If)
Condición 1
```

```
y Condición 2  
...  
y Condición n  
Entonces (Then)  
  Conclusión 1  
  Conclusión 2  
...  
y Conclusión m
```

Es la forma más extendida de representar el conocimiento. Ejemplo de Regla de producción:

```
Si  
  ha fallado la bombilla  
  y hay una de repuesto  
  y está útil  
Entonces  
  cambiar la bombilla por la de repuesto  
  y seguir trabajando
```

- 2) **Memoria de trabajo:** contiene una descripción del estado actual del mundo o entorno de la aplicación en cada paso del proceso de razonamiento. Esta descripción es un modelo que servirá para asociar los antecedentes de las reglas con las observaciones del mundo, con el objetivo de seleccionar o producir las acciones apropiadas. En el momento en que se cumplen todas las condiciones de una regla se produce el “disparo” de la misma, ejecutándose la acción. Esta operación alterará el contenido de la memoria de trabajo.
- 3) **Ciclo de reconocimiento y actuación:** es el procedimiento de control de un sistema de producción. Es un procedimiento de feedforward o hacia adelante. La memoria de trabajo se inicializa con la descripción del problema. Los modelos almacenados en la memoria de trabajo se tratan de superponer en las condiciones de las producciones. Tras ello, se crea un conjunto “conflicto”, es decir, un subconjunto de producciones cuyas condiciones se cumplen. Se escoge una producción y se “dispara” o se activa. La acción de la regla es “disparada” cambiando el contenido de la memoria de trabajo. Se repite todo el proceso descrito con la memoria de trabajo modificada. El proceso continúa hasta que no haya condiciones en las reglas que cumplan el contenido de los modelos de la memoria de trabajo.

Una de las principales ventajas de los sistemas de producción en los SE es la separación del conocimiento y del control. Se pueden hacer cambios fáciles de reglas sin cambiar el control y viceversa. Otra es la modularidad de las reglas de producción y la independencia del lenguaje de programación usado.

40.2.1. Razonamiento

El razonamiento se define como el proceso de obtención de inferencias o conclusiones a partir de unos hechos u observaciones reales o asumidos y de un conocimiento previo. La inferencia es

Capítulo 40. Implementación de un sistema experto en el ámbito pediátrico de atención primaria
602

el proceso por el que a partir de unos hechos conocidos se obtienen conclusiones acerca de otros desconocidos ([Fleitas, 2017](#)) y Begu04. La realización de este tipo de procesamiento es llevada a cabo por el denominado **motor de inferencia**.

El razonamiento automático ya se utilizaba en los 50 en juegos. En 1963 se presentó el sistema “*General Problem Solver*” capaz de hacer inferencias lógicas (Newel y Simon).

Tipos de razonamiento en SE:

- **Forward chaining** (encadenamiento hacia delante, deductivo, progresivo, dirigido por datos o hechos): Síntomas → Causas
- **Backward chaining** (encadenamiento hacia atrás, inductivo, regresivo, dirigido por metas u objetivos): Síntomas ← Causas

Pasos del motor de inferencia:

1. Elaboración de un **conjunto conflicto** con todas las reglas cuyas condiciones se cumplen.
2. **Detección** (filtro) de reglas pertinentes o **selección** de reglas a partir de unos hechos. Se trata de obtener de la Base de Conocimiento (BC) el conjunto de reglas aplicables en una situación determinada o estado de la Base de Hechos (BH).
3. **Aplicación de reglas o resolución del conflicto.** Consiste en seleccionar una regla del conjunto conflicto y dispararla (ejecutar su conclusión). Se altera la BH o memoria de trabajo incluyendo el consecuente de la regla “disparada”.
4. **Vuelta a 1** hasta que el conjunto **conflicto** esté vacío.

Ciclo de “razonamiento hacia delante”:

1. Parte de unas observaciones (**hechos**).
2. A partir de los hechos observados, se **seleccionan las reglas** cuyas condiciones están relacionadas con estos.
3. Las reglas seleccionadas son examinadas para ver si cumplen todas sus condiciones. Aquellas que las verifican constituyen el “**conjunto conflicto**”.
4. Del total de reglas que forman el conjunto conflicto se selecciona una sola y se activa (se “dispara”). La selección de una regla del conjunto conflicto se denomina “**resolución del conflicto**”
5. La activación de la regla provocará la **aparición de otros hechos** que se añaden a los observados y se **actualiza** la base de hechos.
6. Volver al **paso 2** hasta analizar todos los hechos observados y deducidos.

En la Fig. 40.2 se muestra el algoritmo correspondiente al ciclo de inferencia hacia adelante.

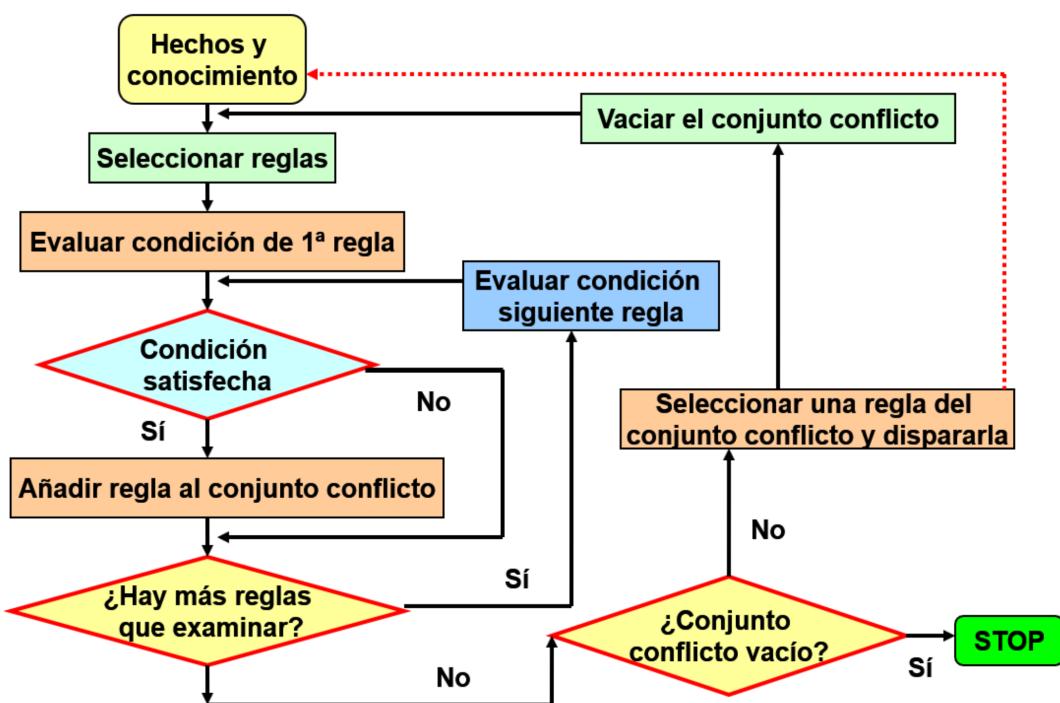


Figura 40.2: Ciclo de razonamiento hacia delante

40.3. Sistema experto para el ámbito pediátrico en atención primaria

En la actualidad, uno de los principales problemas a los que se enfrentan los profesionales de la sanidad en el ámbito pediátrico de atención primaria en España es la falta del tiempo suficiente para realizar una evaluación clínica del estado del paciente. La necesidad de un mayor número de médicos especialistas y la aparición de picos de demanda motivados fenómenos como el COVID, o de modo estacional por otras enfermedades recurrentes, favorecen esta situación.

Ante esta problemática, los centros de salud tratan de optimizar sus recursos mediante diferentes vías, poniendo especial interés en realizar procesos de triaje que les permitan priorizar la atención a los pacientes según su nivel de urgencia. Para ello, en España se utilizan escalas como el MTS (Manchester Triage System), el SET (Sistema Español de Triage) y el CPTAS (Canadian Pediatric Triage and Acuity Scale) [Soler2010] para establecer el tiempo que un paciente puede esperar para recibir atención médica en base a sus síntomas y evolución.

Sin embargo, realizar un correcto proceso de triaje, además de requerir de un gran conocimiento experto, hace necesario un tiempo para una evaluación clínica que a veces resulta difícil dedicar. En este contexto, se plantea el desarrollo de un SE capaz de ayudar en el proceso de evaluación médica, con el objetivo de facilitar el proceso de triaje.

El primer paso para el desarrollo de un SE es la definición de un conjunto de reglas que modelen el conocimiento con el que se nutrirá. Para ello, es posible recurrir al apoyo de expertos, capaces de definir su conocimiento como reglas, o la aplicación de mecanismos de extracción de conocimiento a partir del procesamiento de conjuntos de datos y sucesos.

En este ejemplo, se recopilaron y procesaron un conjunto de datos relativos a los motivos de consulta pediátrica en un centro de salud, donde la escala de triaje utilizada fue el CPTAS. Los datos fueron anonimizados, seleccionando únicamente aquellos campos que pudieran resultar clave para la extracción de conocimiento y la conformación de reglas. Adicionalmente, se contó con el apoyo de profesionales especialistas del ámbito pediátrico para revisar y complementar algunas de las reglas extraídas. Un extracto de los datos utilizados para el proceso de extracción de reglas se muestra en la Tabla 40.1.

Tabla 40.1: . Ejemplo de datos de motivos de consulta y triaje.

Sexo	Edad	Tiempo Evolución	Causa	Triaje
Hombre	1-3 años	25-72 horas	Dermatológica	5 (120 minutos)
Mujer	1-3 años	25-72 horas	Respiratoria	3 (30 minutos)
Hombre	4-6 años	7-12 horas	Gastrointestinal	5 (120 minutos)
Hombre	4-6 años	2-6 horas	Ocular	4 (60 minutos)
Mujer	4-6 años	13-24 horas	Fiebre	5 (120 minutos)

40.3. Sistema experto para el ámbito pediátrico en atención primaria

605

A partir del procesamiento de un total de 400 visitas médicas mediante un algoritmo de extracción de reglas de asociación como “Magnum Opus”, basado en la definición original de [Webb \(2011\)](#), y mediante la aplicación del conocimiento de experto proporcionado por un panel de pediatras, se extrajeron un conjunto de reglas con suficiente calidad. A continuación, se muestra un ejemplo de un conjunto de reglas con 10 de ellas.

- R1: Si Causa Ginecologica o Edad mayor de 12 años → Tiempo de Evolución mayor de 73h
- R2: Si Causa Ginecologica → Sexo Mujer
- R3: Si Edad menor de 7 días o Causa Fiebre → Tiempo de evolución de 1h
- R4: Si Edad mayor de 12 años y Tiempo de Evolución mayor de 73h → Causa Respiratoria
- R5: Si Tiempo Evolución mayor de 73h y Causa Ocular → Triaje 1
- R6: Si Causa Respiratoria y Sexo Mujer → Triaje 3
- R7: Si Tiempo de Evolución es 2-6h o Causa Neurológica → Triaje 2
- R8: Si Causa Respiratoria y Tiempo de Evolución es 2-6h → Triaje 4
- R9: Si Tiempo de Evolución es 13-24h y Causa Ginecológica y Sexo Mujer → Triaje 5
- R10: Si Tiempo de Evolución mayor de 73h → Triaje 4

A continuación, se muestra el código en **R** para la implementación de un SE capaz de procesar reglas como las anteriores, realizando una ejecución para obtener el valor de triaje correspondiente. Para este ejemplo se considera una niña mayor de 12 años de edad, cuyo motivo de consulta es ginecológico con un tiempo de evolución superior a 73 horas.

Es importante señalar que, habitualmente, un SE partirá de una base de hechos compuesta por decenas o cientos de reglas. No obstante, para simplificar el siguiente fragmento de código, se considera únicamente la carga de 10, reglas a modo de ejemplo.

Se declaran las reglas que conformarán la base conocimiento del SE:

1. La BC del SE contiene 10 reglas.
2. Cada regla se modela como una lista de antecedentes y un consecuente.
3. La relación entre los consecuentes se modela con el atributo “operador” del siguiente modo: 1 = Y lógico, 0 = O lógico, -1 = no hay operaciones

```
r1 <- list(
  antecedentes = list("Causa_Ginecologica", "Edad_>12a"),
  consecuente = list("TiempoEvolucion_>73h"), operador = 0
)
r2 <- list(
```

Capítulo 40. Implementación de un sistema experto en el ámbito pediátrico de atención primaria

```

antecedentes = list("Causa_Ginecologica"),
consecuente = list("Sexo_Mujer"), operador = -1
)
r3 <- list(
  antecedentes = list("Edad_<7d", "Causa_Fiebre"),
  consecuente = list("TiempoEvolucion_1h"), operador = 0
)
r4 <- list(
  antecedentes = list("Edad_>12a", "TiempoEvolucion_>73h"),
  consecuente = list("Causa_Respiratoria"), operador = 1
)
r5 <- list(
  antecedentes = list("TiempoEvolucion_>73h", "Causa_Ocular"),
  consecuente = list("Triaje_1"), operador = 1
)
r6 <- list(
  antecedentes = list("Causa_Respiratoria", "Sexo_Mujer"),
  consecuente = list("Triaje_3"), operador = 1
)
r7 <- list(
  antecedentes = list("TiempoEvolucion_2-6h", "Causa_Neurologica"),
  consecuente = list("Triaje_2"), operador = 0
)
r8 <- list(
  antecedentes = list("Causa_Respiratoria", "TiempoEvolucion_2-6h"),
  consecuente = list("Triaje_4"), operador = 1
)
r9 <- list(
  antecedentes = list("TiempoEvolucion_13-24h", "Causa_Ginecologica", "Sexo_Mujer"),
  consecuente = list("Triaje_5"), operador = 1
)
r10 <- list(
  antecedentes = list("TiempoEvolucion_>73h"),
  consecuente = list("Triaje_4"), operador = -1
)

# r1 regla completa
# r1[1] lista antecedentes
# r1[[1]] [1] primero de los antecedentes
# r1[2] lista consecuentes
# r1[[2]] [1] primero de los consecuentes

b_hechos <- list("Causa_Ginecologica", "Edad_>12a", "TiempoEvolucion_>73h")

```

Se inicializa la BC con el conjunto de Reglas y la BH con la circunstancia a evaluar.

```

# Se añaden todas las reglas a la Base de Conocimiento
b_conocimiento <- list(r1, r2, r3, r4, r5, r6, r7, r8, r9, r10)

```

40.3. Sistema experto para el ámbito pediátrico en atención primaria

607

Se define una función para comprobar la existencia de un número en una lista. Esta función será usada por el motor del SE.

```
# Función para comprobar si una lista contiene un número

contiene <- function(numero, lista) {
  existe <- FALSE
  if (length(lista) > 0) {
    for (i in 1:length(lista)) {
      if (numero == lista[[i]]) existe <- TRUE
    }
  }
  return(existe)
}
```

Se implementa el motor del SE.

El algoritmo ejecuta un bucle en el que, en cada iteración, evalúa las reglas disponibles contenidas en la *BC*. Considerando los items de la *BH*, si una regla puede ser “disparada”, se añade al *Conjunto Conflicto*. La regla “disparada” en una iteración será la primera disponible en el *Conjunto Conflicto*. El *Conjunto Conflicto* se inicializa en cada iteración. El consecuente de la regla “disparada” se añade a la *BH*. Cada regla solo puede “dispararse” una vez, por lo que se actualiza una lista de reglas “disparadas”. El algoritmo finaliza cuando el *Conjunto Conflicto* queda vacío, al haber sido “disparadas” todas las reglas o no existir más candidatas a ser “disparadas”.

```
# Motor del Sistema Experto

AlgoritmoSE <- function(b_hechos, b_conocimiento) {
  c_conflicto <- list()
  r_disparadas <- list()
  condicion <- TRUE
  iteracion <- 0
  while (condicion) {
    c_conflicto <- list()
    cat("Iteración: ", iteracion, "\n")
    iteracion <- iteracion + 1
    for (i in 1:length(b_conocimiento)) {
      if (!contiene(i, r_disparadas)) {
        if (b_conocimiento[[i]][[3]][[1]] == 1) {
          r_disparada <- TRUE
        } else {
          r_disparada <- FALSE
        }
        for (j in 1:length(b_conocimiento[[i]][[1]])) {
          antecedente <- FALSE
          for (k in 1:length(b_hechos)) {
            if (b_conocimiento[[i]][[1]][[j]] == b_hechos[[k]]) {
```

Capítulo 40. Implementación de un sistema experto en el ámbito pediátrico de atención primaria
608

```

    if (b_conocimiento[[i]][[3]][[1]] == 0 || b_conocimiento[[i]][[3]][[1]]
        ↵ == -1) r_disparada <- TRUE
    antecedente <- TRUE
  }
}
if (b_conocimiento[[i]][[3]][[1]] == 1) {
  if (!antecedente) r_disparada <- FALSE
}
}
if (r_disparada) {
  cat("Regla", i, "añadida a Conjunto conflicto\n")
  c_conflicto[length(c_conflicto) + 1] <- i
}
}
}
if (length(c_conflicto) > 0) {
  # str("Conjunto conflicto:")
  # str(c_conflicto)
  r_disparadas[length(r_disparadas) + 1] <- c_conflicto[1]
  cat("Regla", r_disparadas[[length(r_disparadas)]], "disparada\n")
  b_hechos[length(b_hechos) + 1] <-
  ↵ b_conocimiento[[r_disparadas[[length(r_disparadas)]]]][[2]][[1]]
  str("Base de hechos:")
  str(b_hechos)
  cat("Consecuente:",
      ↵ b_conocimiento[[r_disparadas[[length(r_disparadas)]]]][[2]][[1]], "\n")
} else {
  condicion <- FALSE
}
}
}

```

Ahora considerese, por ejemplo, una posible paciente con más de 12 años de Edad, que acude a consulta por causa Ginecológica con un Tiempo de Evolución de los síntomas mayor de 73h. ¿Cuál será el triaje correspondiente? Para conocerlo, se inicializa la *BH* con la situación propuesta (*Causa_Ginecologica*, *Edad_>12a* y *TiempoEvolucion_>73h*) y se ejecuta el motor del SE.

```
# Se inicializa la base de hechos con la situación propuesta (paciente de más de 12
→ años, por causa ginecológica con tiempo de evolución mayor de 73h)
b_hechos <- list("Causa_Ginecologica", "Edad_>12a", "TiempoEvolucion_>73h")

# Se lanza la ejecución del motor del Sistema Experto
AlgoritmoSE(b_hechos, b_conocimiento)
```

El resultado del algoritmo, en este caso (motivo de consulta ginecológico y mayor de 12 años y con un tiempo de evolución de los síntomas mayor de 73h), es un triaje de valor 4. Es decir,

40.3. Sistema experto para el ámbito pediátrico en atención primaria

609

el tiempo de espera máximo para recibir atención médica debería ser inferior a 60 minutos. La Tabla 40.3 muestra el proceso realizado por el algoritmo en cada iteración.

Table: . Proceso de ejecución del Sistema Experto.

Como puede observarse, el algoritmo finaliza tras 5 iteraciones, al alcanzar un conjunto conflicto vacío, dando como resultado un valor de 4 para el triaje. A continuación, se describe el proceso ejecutado en cada una de las iteraciones.

- **Iteración 0:** La Base de Hechos se inicializa con las condiciones establecidas en el ejemplo de consulta médica considerado, es decir, **Causa_Ginecologica**, **Edad_>12a** y **TiempoEvolucion_>73h**. En el Conjunto Conflicto se incluyen aquellas reglas que podrían ser lanzadas con los elementos contenidos en la BH es decir, las reglas R1, R2, R4, R10. Se dispara la regla R1, al ser la primera de la lista de reglas del Conjunto Conflicto. El consecuente de la regla disparada (**TiempoEvolucion_>73h**) se añade a la BH para la siguiente iteración, aunque en este caso no es necesario porque ya lo contiene. La iteración finaliza estableciendo como conclusión el consecuente de la regla disparada, es decir, **TiempoEvolucion_>73h**.
- **Iteración 1:** El Conjunto Conflicto se inicializa con las reglas que podrían ser lanzadas, excluyendo las ya ejecutadas (R1), a partir de los elementos incluidos en la Base de Hechos tras la iteración anterior, es decir, las reglas R2, R4 y R10. Se lanza la primera de las reglas del Conjunto Conflicto, es decir, R2. El consecuente de la regla disparada (**Sexo_Mujer**) se añade a la BH. La iteración finaliza estableciendo como conclusión el consecuente de la regla disparada, es decir, **Sexo_Mujer**.
- **Iteración 2:** El Conjunto Conflicto se inicializa con las reglas que podrían ser lanzadas, excluyendo las ya ejecutadas (R1 y R2), a partir de los elementos contenidos en la BH tras la iteración anterior, es decir, las reglas R4 y R10. Se lanza la primera de las reglas del Conjunto Conflicto, es decir, R4. El consecuente de la regla disparada (**Causa_Respiratoria**) se añade a la BH. La iteración finaliza estableciendo como conclusión el consecuente de la regla disparada, es decir, **Causa_Respiratoria**.
- **Iteración 3:** El Conjunto Conflicto se inicializa con las reglas que podrían ser lanzadas, excluyendo las ya ejecutadas (R1, R2 y R4), a partir de los elementos contenidos en la BH tras la iteración anterior, es decir, las reglas R6 y R10. Se lanza la primera de las reglas del Conjunto Conflicto, es decir, R6. El consecuente de la regla disparada (**Triaje_3**) se añade a la BH. La iteración finaliza estableciendo como conclusión el consecuente de la regla disparada, es decir, **Triaje_3**.
- **Iteración 4:** El Conjunto Conflicto se inicializa con las reglas que podrían ser lanzadas, excluyendo las ya ejecutadas (R1, R2, R4 y R6), a partir de los elementos contenidos en la BH tras la iteración anterior, en este caso únicamente R10. Se lanza por tanto la regla R10. El consecuente de la regla disparada (**Triaje_4**) se añade a la BH. La iteración finaliza estableciendo como conclusión el consecuente de la regla disparada, es decir, **Triaje_4**.
- **Iteración 5:** El Conjunto Conflicto queda vacío, al no ser posible incluir en él reglas no disparadas aún y que pudieran ser ejecutadas a partir de los elementos contenidos en

Capítulo 40. Implementación de un sistema experto en el ámbito pediátrico de atención primaria
610

la BH. Por tanto, el algoritmo finaliza concluyendo como resultado el consecuente de la última regla lanzadas, es decir, **Triage_4**.

Sin lugar a duda, la implementación del algoritmo, las reglas y el modelado considerado para este caso de estudio resulta una simplificación intencionada del problema, con el único objetivo de facilitar su comprensión desde una perspectiva académica y docente.

En un escenario de aplicación real, el volumen y la complejidad de las reglas debería ser mayor, considerando además la posibilidad de incorporar otras características que permitieran modelar de un modo más completo el estado de salud del paciente y su motivo de consulta.

Actualmente resultan innumerables los ámbitos donde la aplicación de SE puede ser de ayuda. En este caso de estudio, el objetivo ha sido facilitar y mejorar el proceso de triaje del nivel de urgencia en el ámbito pediátrico en atención primaria. Pero, en este mismo contexto, podría valorarse su uso como herramienta de apoyo para, por ejemplo, el diagnóstico de enfermedades o la prescripción de tratamientos.

La tecnología actual y lenguajes de programación como R facilitan la implementación de SE de un modo rápido y sencillo. Por tanto, pueden ser considerados como herramienta de ayuda para situaciones en las que aplicar conocimiento experto resulte clave para la solución de problemas.

40.3. Sistema experto para el ámbito pediátrico en atención primaria

611

Tabla 40.2: Proceso de ejecución del Sistema Experto

It	Base Hechos	Conjunto conflicto	Regla disparada	Conclusión
0	<i>Causa_Ginecologica</i> <i>Edad_ < 12a</i> <i>TiempoEvoluc_ > 73h</i>	R1, R2, R4, R10	R1	<i>TiempoEvoluc_ > 73h</i>
1	<i>Causa_Ginecologica</i> <i>Edad_ < 12a</i> <i>TiempoEvoluc_ > 73h</i> <i>Sexo_Mujer</i>	R2, R4 R10	R2	<i>Sexo_Mujer</i>
2	<i>Causa_Ginecologica</i> <i>Edad_ > 12a</i> <i>TiempoEvoluc_ > 73h</i> <i>Sexo_Mujer</i> <i>Causa_Respiratoria</i>	R4, R10	R4	<i>Causa_Respiratoria</i>
3	<i>Causa_Ginecologica</i> <i>Edad_ > 12a</i> <i>TiempoEvoluc_ > 73h</i> <i>Sexo_Mujer</i> <i>Causa_Respiratoria</i> <i>Triaje_3</i>	R6, R10	R6	<i>Triaje_3</i>
4	<i>Causa_Ginecologica</i> <i>Edad_ > 12a</i> <i>TiempoEvoluc_ > 73h</i> <i>Sexo_Mujer</i> <i>Causa_Respiratoria</i> <i>Triaje_3</i> <i>Triaje_4</i>	R10	R10	<i>Triaje_4</i>
5	<i>Causa_Ginecologica</i> <i>Edad_ > 12a</i> <i>TiempoEvoluc_ > 73h</i> <i>Sexo_Mujer</i> <i>Causa_Respiratoria</i> <i>Triaje_3</i> <i>Triaje_4</i>	Ø		

Capítulo 40. Implementación de un sistema experto en el ámbito pediátrico de atención primaria

Capítulo 41

El procesamiento del lenguaje natural para tendencias de moda en textil

Ambrosio Nguema Ansue

41.1. Introducción

El Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés), abarca una amplia gama de técnicas y algoritmos, entre los que se encuentra el modelado de temas. El modelado de temas no es un modelo de predicción en sí mismo. En cambio, es una técnica de aprendizaje no supervisado que tiene como objetivo descubrir estructuras ocultas (temas) dentro de un conjunto de documentos o textos aunque está relacionado con el NLP, no son lo mismo, el modelado de temas es una de las muchas técnicas que forman parte del NLP. La relación entre ambos radica en que el modelado de temas utiliza enfoques del NLP para analizar y procesar el lenguaje en los textos, pero se enfoca en una tarea específica: extraer temas. En este capítulo, exploraremos cómo el modelado de temas y otras técnicas de NLP pueden aplicarse al análisis de tendencias en el mundo de la moda. El modelado de temas aplicado a la industria textil puede proporcionar información valiosa sobre las preferencias y opiniones de los clientes, lo que puede mejorar la toma de decisiones y la experiencia del cliente en el ámbito del comercio electrónico de ropa.

41.2. Análisis de tendencias de moda en textil

El conjunto `clothes` de datos incluido en el paquete `CDR` de reseñas y calificaciones de ropa de comercio electrónico para mujeres contiene 23.486 entradas relacionadas con la edad y la

614 Capítulo 41. El procesamiento del lenguaje natural para tendencias de moda en textil

revisión dada por el cliente y sus opiniones sobre la ropa de mujer de varios minoristas.

```
library("CDR")
library("readr")
library("tidyverse")
library("tidytext")
```

Las variables incluidas pueden verse con la ejecutando `names(clothes)` y una descripción de las variables con el comando `??clothes`. El primer registro presenta la siguiente estructura la información:

```
head(clothes)[1, ]
#>   ID Age Title                               Review Rating
#> 1 767 33 <NA> Absolutely wonderful - silky and sexy and comfortable 4
#>   Recommend Liked Division      Dept      Class
#> 1           1          0 Initmates Intimate Intimates
```

El conjunto de datos consta de 23.486 entradas que incluyen información acerca de la edad del cliente, las calificaciones otorgadas y las opiniones sobre la ropa comprada en comercios electrónicos para mujeres. Los datos se organizan en columnas, algunas de las cuales contienen valores enteros y otras almacenan caracteres. Todas las columnas con valores enteros están completas, mientras que algunas columnas de caracteres presentan valores faltantes (NA). La variable con la mayor cantidad de valores NA Título.

En el presente capítulo, se explora la aplicación de técnicas de análisis de texto en un conjunto de datos de reseñas y calificaciones de ropa de comercio electrónico para mujeres. En primer lugar, se realiza un análisis del porcentaje de reseñas y calificaciones en cada departamento, destacando los departamentos con mayor y menor porcentaje. Además, se lleva a cabo un análisis de bigramas para identificar las frases más comunes asociadas con diferentes calificaciones. Finalmente, se utiliza el modelado de temas con *Latent Dirichlet Allocation* (LDA) para explorar las características clave de las revisiones en el departamento de Tendencias. Los resultados del análisis proporcionan información valiosa para las empresas sobre el grupo demográfico objetivo, las preferencias de los clientes y las características clave de las prendas.

```
library("ggplot2")
clothes |>
  dplyr::count(Dept) |>
  dplyr::mutate(prop = n / sum(n)) |>
  ggplot(aes(x = Dept, y = prop * 100)) +
  geom_bar(stat = "identity", fill="blue") +
  xlab("Department Name") +
  ylab("Percentage of Reviews/Ratings (%)") +
  geom_text(aes(label = round(prop * 100, 2)), vjust = -0.25)
```

Los tops y vestidos son los departamentos que cuentan con la mayoría de las reseñas y calificaciones en el conjunto de datos, mientras que las chaquetas y la sección de tendencias tienen

41.2. Análisis de tendencias de moda en textil

615

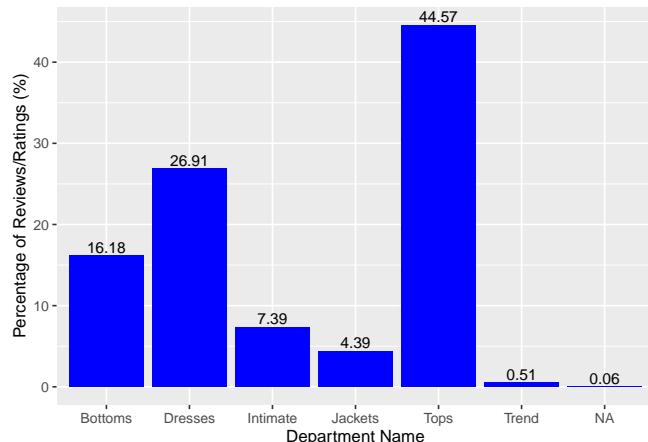


Figura 41.1: Percentage of Reviews by Department

la menor cantidad. Dado que la sección de tendencias presenta una mezcla de ropa que puede pertenecer a otros departamentos, y solo representa un 0,51% del conjunto de datos, se ha decidido excluir esta sección del análisis.

```
clothes |>
  filter(!is.na(Dept), Dept != "Trend") |>
  mutate(Dept = factor(Dept)) |>
  group_by(Dept, Rating) |>
  summarize(n = n()) |>
  mutate(perc = n / sum(n)) |>
  ggplot(aes(x = Rating, y = perc * 100, fill = Dept)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~Dept) +
  ylab("Percentage of reviews (%)") +
  geom_text(aes(label = round(perc * 100, 2)), vjust = -.2) +
  scale_y_continuous(limits = c(0, 65))
```

Se ha observado que en todos los departamentos, la calificación de 5 estrellas es la más común. A pesar de tener una menor cantidad de reseñas en general, las chaquetas tienen la mayor proporción de calificaciones de 5 estrellas en su categoría. Una posible razón de esto es que las chaquetas suelen ser más fáciles de ajustar a diferentes formas corporales en comparación con vestidos y blusas, que pueden ser más difíciles de adaptarse correctamente, especialmente cuando se compran en línea.

```
clothes |>
  filter(!is.na(Age), !is.na(Dept), Dept != "Trend") |>
  select(ID, Age, Dept) |>
  mutate(Age_group = cut(Age, breaks = c(18, 29, 39, 49, 59, 69, 79, 89, 99))) |>
  mutate(Age_group = as.character(Age_group)) |>
```

616 Capítulo 41. El procesamiento del lenguaje natural para tendencias de moda en textil

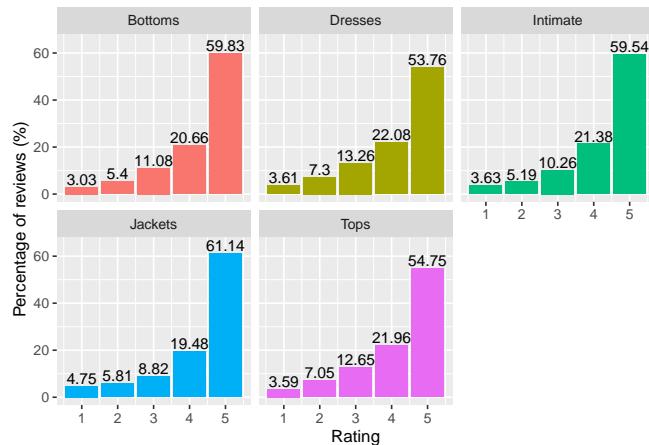


Figura 41.2: Percentage of reviews in each department

```

mutate(Age_group = factor(Age_group, levels = c("18-29", "30-39", "40-49", "50-59",
  ↪ "60-69", "70-79", "80-89", "90-99"))) |>
  mutate(Dept = factor(Dept, levels = rev(c("Tops", "Dresses", "Bottoms", "Intimate",
  ↪ "Jackets")))) |>
  filter(Age < 80) |>
  group_by(Age_group, Dept) |>
  summarize(n = n()) |>
  ggplot(aes(Dept, n, fill = Age_group)) +
  geom_bar(stat = "identity", fill="red") +
  facet_wrap(~Age_group, scales = "free") +
  xlab("Department") +
  ylab("Number of Reviews") +
  geom_text(aes(label = n), hjust = -0.1) +
  scale_y_continuous(expand = c(.1, 0)) +
  coord_flip() +
  scale_fill_manual(values = hcl.colors(8))

```

Se ha observado que la tendencia en la distribución de reseñas por departamento (es decir, tops con el mayor número de reseñas y vestidos con el segundo mayor número) es similar en la mayoría de los grupos de edad. Esto indica que la popularidad de los diferentes tipos de ropa se mantiene en gran medida constante entre los grupos de edad más jóvenes y de mediana edad.

Análisis de bigramas

El análisis de bigramas es una técnica útil para identificar patrones y tendencias en el lenguaje utilizado en las reseñas de productos. Un bigrama es un par consecutivo de palabras en un texto y puede proporcionar información valiosa sobre la frecuencia con la que ciertas palabras aparecen juntas y las combinaciones de palabras que son relevantes en las opiniones de los clientes. Al utilizar el análisis de bigramas, se espera comprender mejor las opiniones de los clientes sobre los productos de ropa en el conjunto de datos.

41.2. Análisis de tendencias de moda en textil

617

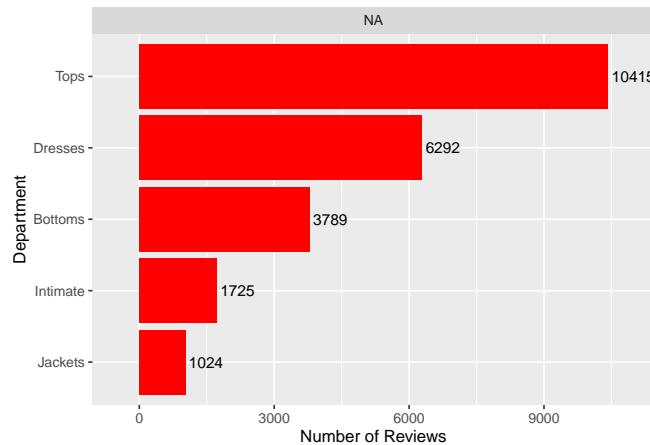


Figura 41.3: Number of Reviews by Department

```

clothesr <- clothes |> filter(!is.na(Review))
notitle <- clothesr |>
  filter(is.na>Title) |>
  select(-Title)
wtitle <- clothesr |>
  filter(!is.na>Title) |>
  unite(Review, c>Title, Review), sep = " "
main <- bind_rows(notitle, wtitle)

```

Para llevar a cabo el análisis de bigramas, se procede a procesar las palabras de las reseñas eliminando las palabras vacías (también conocidas como *stop words*), que son palabras comunes sin un significado contextual importante, y los dígitos que representan la calificación de las reseñas. Una vez procesadas las palabras, se agrupan según sus calificaciones y se representan gráficamente los 10 bigramas más comunes para cada nivel de calificación. De esta forma, se puede identificar y comprender mejor las combinaciones de palabras que son relevantes para las opiniones de los clientes y para cada nivel de calificación.

```

bigramming <- function(data) {
  cbigram <- data |> unnest_tokens(bigram, Review, token = "ngrams", n = 2)
  cbigram_sep <- cbigram |> separate(bigram, c(first, second), sep = " ")
  cbigram2 <- cbigram_sep |>
    filter(!first %in% stop_words$word, !second %in% stop_words$word,
      !str_detect(first, "\d"), !str_detect(second, "\d")) |>
    unite(bigram, c(first, second), sep = " ")
  return(cbigram2)
}

```

618 Capítulo 41. El procesamiento del lenguaje natural para tendencias de moda en textil

```
top_bigrams <- bigramming(main) |>
  mutate(Rating = factor(Rating, levels <- c(5:1))) |>
  mutate(bigram = factor(bigram, levels = rev(unique(bigram)))) |>
  group_by(Rating) |>
  count(bigram, sort = TRUE) |>
  top_n(10, n) |>
  ungroup()

top_bigrams |> ggplot(aes(bigram, n, fill = Rating)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~Rating, ncol = 3, scales = "free") +
  labs(x = NULL, y = "frequency") +
  coord_flip()
```

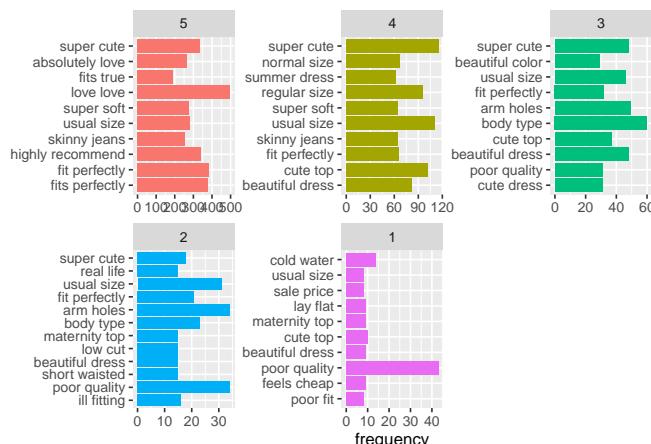


Figura 41.4: Most Common Bigrams (By Ratings)

Modelado de temas con Latent Dirichlet Allocation

El enfoque de modelado de temas de Latent Dirichlet Allocation (LDA) es una técnica ampliamente utilizada en NLP para extraer temas latentes de un corpus de texto. LDA es un algoritmo no supervisado que utiliza el aprendizaje automático para identificar patrones en grandes conjuntos de datos de texto, agrupando palabras similares en temas y asignando probabilidades a cada tema en cada documento.

En este estudio, se ha utilizado el enfoque de modelado de temas de LDA para explorar las 118 revisiones del Departamento de Tendencias. Se ajustó un modelo LDA utilizando muestreo de Gibbs y se eligió $k = 5$ para los departamentos de Bottoms, Dresses, Intimate, Jackets y Tops. A través del análisis de los resultados, se pudieron identificar las 5 palabras principales de cada tema y obtener una mejor comprensión de las características clave de las revisiones en cada departamento. De esta forma, se pudo obtener información valiosa sobre las preferencias y opiniones de los clientes en diferentes departamentos de ropa en el conjunto de datos.

41.2. Análisis de tendencias de moda en textil

619

```

library("topicmodels")
library("tm")
library("LDAvis")

trend_count <- main |>
  filter(Dept == "Trend") |>
  unnest_tokens(word, Review) |>
  anti_join(stop_words, by = "word") |>
  filter(!str_detect(word, "\\d")) |>
  count(ID, word, sort = TRUE) |>
  ungroup()

trend_dtm <- trend_count |> cast_dtm(ID, word, n)
trendy <- tidy(LDA(trend_dtm, k = 5, method = "GIBBS", control = list(seed = 4444,
                     alpha = 1)), matrix = "beta")
top_trendy <- trendy |>
  group_by(topic) |>
  top_n(5, beta) |>
  ungroup() |>
  arrange(topic, desc(beta))

top_trendy |>
  mutate(term = reorder(term, beta)) |>
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~topic, scales = "free") +
  coord_flip()

```

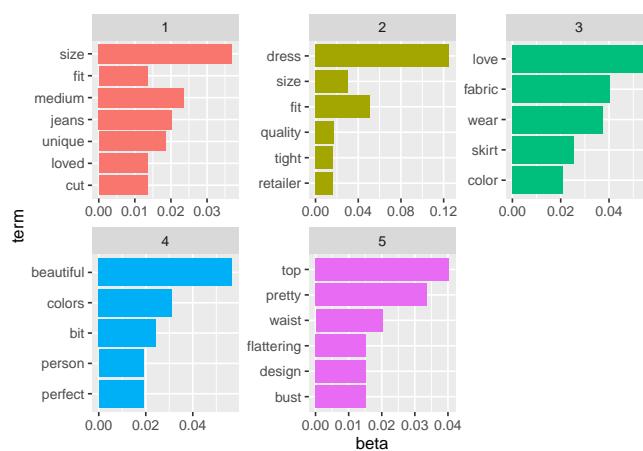


Figura 41.5: Modelo LDA (K=5)

En el modelo LDA, cada tema se representa por un conjunto de palabras que aparecen juntas con mayor frecuencia en las revisiones. Por ejemplo, al observar el tema 3, se puede identificar

620 *Capítulo 41. El procesamiento del lenguaje natural para tendencias de moda en textil*

que está caracterizado por palabras como “colors”, “wear”, “bit”, “jacket” y “price”, lo que sugiere que los clientes pueden estar comentando sobre la variedad de colores disponibles, la durabilidad de la prenda y su precio. Por otro lado, el tema 1 se caracteriza por palabras como “love”, “fit”, “fabric”, “wear” y “length”, lo que sugiere que los clientes pueden estar hablando sobre su experiencia con la prenda en términos de comodidad, ajuste y calidad de la tela. Al identificar estos temas, se pueden obtener ideas valiosas sobre las opiniones y preferencias de los clientes para mejorar la calidad de la ropa y satisfacer sus necesidades y deseos. Esto permite a las empresas tomar decisiones informadas para satisfacer las necesidades de sus clientes y mejorar la experiencia del usuario en el ámbito del comercio electrónico de ropa.

Como conclusión, destacar que el análisis de este conjunto de datos proporciona información valiosa sobre las preferencias y opiniones de los clientes en cuanto a la ropa femenina. Las reseñas de 5 estrellas son dominantes en cada departamento, y las chaquetas son las prendas que obtienen la proporción más alta de calificaciones positivas. Además, se ha observado que los clientes de entre 30 y 40 años dejan la mayoría de las reseñas y que factores como el ajuste, la comodidad/calidad del material y la estética de la prenda influyen en la calificación. La realización de análisis de datos exploratorios y de bigramas puede ayudar a las empresas a comprender mejor lo que funciona y lo que no, y seleccionar artículos con telas flexibles y cómodas puede conducir a una mayor satisfacción del cliente y mayores ventas. Por último, el modelado de temas con LDA es una herramienta útil en situaciones en las que se tienen reseñas sin marcar y puede proporcionar información valiosa sobre las características clave de las revisiones. En general, estos análisis pueden ayudar a las empresas a tomar decisiones informadas y mejorar la experiencia del usuario en el ámbito del comercio electrónico de ropa.

Capítulo 42

Detección de fraude de tarjetas de crédito

Pedro Albarracín García

42.1. Introducción

En un informe publicado en diciembre de 2021 por Nilson Report (https://nilsonreport.com/upload/content_promo/NilsonReport_Issue1209.pdf), se informó de que los emisores de tarjetas de crédito, comerciantes y consumidores sufrieron un total de 28.580 millones de dólares de pérdidas por fraude en 2020, es decir, 6,8 centavos por cada 100 dólares en volumen de compras. El fraude sólo en USA representa el 35,83 % del total mundial.

En Europa la situación no es más alentadora. Según un informe del Banco Central Europeo publicado en 2020 (<https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202008~521edb602b.en.html#toc2>), el valor total de las transacciones con tarjeta en la zona SEPA ascendieron a 4.84 billones de euros en 2018, de los cuales 1.800 millones correspondieron a operaciones fraudulentas.

Las entidades financieras trabajan a diario en el desarrollo de modelos de machine learning y deep learning que les permitan detectar, con la mayor precisión posible, aquellas operaciones de compra con tarjetas de crédito, débito o prepago que puedan ser sospechosas de fraude, o que al menos puedan ser identificadas como anómalas. En este sentido, es importante destacar que no existe una única solución posible, ya que el problema presenta, en la mayoría de los casos, múltiples variantes que hacen de éste, un problema complejo y que puede y debe ser abordado desde múltiples perspectivas y con diferentes enfoques.

En primer lugar, es posible identificar dos tipos de fraude. Por un lado, el que se comete físicamente, como, por ejemplo, la compra o la retirada de efectivo con tarjetas robadas o falsas. Por otro lado, están aquellas transacciones fraudulentas que se cometan online, en las

que no es necesaria la tarjeta física, y en las que se utilizan los datos de las tarjetas obtenidas por los delincuentes mediante técnicas como el phishing y que son utilizados posteriormente para realizar pagos online.

Otro hándicap asociado a este tipo de escenarios es el derivado de la gran diversidad de fuentes de datos que forman parte de una transacción y que pueden dar lugar a divergencias metodológicas, tanto en la recogida y transmisión de los datos, como en su posterior almacenaje, lo que ocasiona que, en muchos casos, la calidad de los datos disponibles no sea la esperada, o simplemente nos encontrremos ante datasets inconsistentes. Los datos requeridos para este caso de uso pueden categorizarse en variables relativas a:

- Cliente
- Transacción
- Geolocalización
- Comercio
- Tarjetas
- Hábitos de compra

Cada una de estas categorías, y otras que puedan aparecer aportan información que permite abordar el problema desde diferentes ángulos. Por un lado es posible enfocar el problema desde el punto de vista del cliente y sus hábitos de compra para ver si existe alguna característica anómala en una transacción, tal vez la hora de la compra, o quizás analizar los datos de geolocalización junto con los del comercio para analizar si es una compra en un comercio habitual y desde una localización conocida, etc.

42.2. Modelización del fraude en la compra con tarjetas de crédito

El objetivo de este caso de uso es la construcción de un modelo que permita detectar si una operación de compra realizada con tarjeta de crédito es fraudulenta o no. Para ello, se utilizará un dataset anonimizado de operaciones con tarjeta de crédito ya etiquetadas disponible desde la web de Kaggle en el siguiente enlace: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>. El lector debe guardar el dataset en una carpeta local llamada `data` para reproducir el análisis.

Este es un dataset con 284.807 transacciones, de las cuales 492 están etiquetadas como fraudulentas, es decir, sólo un 0.172 % del total de las transacciones. Es un dataset por lo tanto muy desequilibrado, lo que añade cierto grado de dificultad. El dataset tiene un conjunto de 31 variables, de las cuales 28 están identificadas como `V1`, ..., `V28`, una variable `Time` que registra los segundos transcurridos entre esa transacción y la primera, una variable `Amount` que registra el importe de la transacción, y la variable dependiente `class` que indica, con valor 0, que la operación es “no fraudulenta”, y con valor 1 las operaciones fraudulentas.

42.2. Modelización del fraude en la compra con tarjetas de crédito

623

Para concluir esta breve descripción del dataset, es necesario recordar que todos los valores de entrada son numéricos y que ya han sufrido algunas transformaciones. Por motivos de confidencialidad, las variables V1 a V28 no incluyen sus nombres originales ni se añade más información de contexto.

Carga de los datos y obtención de algunos descriptivos

```
# Carga del paquete readr
library("readr")

# Se crea una variable de tipo dataframe que contendrá los datos del CSV
creditcard <- read_csv("data/creditcard.csv", show_col_types = FALSE)
# psych::describe(creditcard) # descomentar para ver los descriptivos
head(creditcard)
#> # A tibble: 6 x 31
#>   Time     V1     V2     V3     V4     V5     V6     V7     V8     V9
#>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
#> 1 0    -1.36 -0.0728 2.54  1.38 -0.338  0.462  0.240  0.0987  0.364
#> 2 0    1.19   0.266  0.166  0.448  0.0600 -0.0824 -0.0788  0.0851 -0.255
#> 3 1    -1.36 -1.34   1.77  0.380 -0.503  1.80   0.791  0.248  -1.51
#> 4 1    -0.966 -0.185  1.79  -0.863 -0.0103  1.25   0.238  0.377  -1.39
#> 5 2    -1.16   0.878  1.55  0.403  -0.407  0.0959  0.593  -0.271  0.818
#> 6 2    -0.426  0.961  1.14  -0.168  0.421  -0.0297  0.476  0.260  -0.569
#> # ... with 21 more variables: V10 <dbl>, V11 <dbl>, V12 <dbl>, V13 <dbl>,
#> #   V14 <dbl>, V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>,
#> #   V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>,
#> #   V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>
```

División de los datos

A continuación es necesario dividir los datos en dos dataframes que denominados `creditcard_X` y `creditcard_y`, de esta forma se separan las variables independientes de la variable dependiente o `Class`.

```
# Se dividen los datos
creditcard_X <- creditcard[,-31]
creditcard_y <- creditcard$Class
```

Tratamiento de datos desequilibrados

Uno de los principales problemas a la hora de abordar este tipo de escenarios, es lo que se conoce como “datos desequilibrados”. Se dice que un dataset está desequilibrado cuando la variable dependiente presenta más observaciones de una clase que de otra. En el caso de transacciones fraudulentas con tarjeta de crédito, es evidente que la mayoría de las operaciones son legítimas o benignas, y que sólo un pequeño porcentaje resultan ser maliciosas. ¿Cuál es el problema?

Por lo general, los modelos entrenados con datasets desequilibrados no se comportan bien cuando tienen que generalizar, es decir, cuando tienen que realizar predicciones sobre conjuntos de datos que no han sido vistos anteriormente por el modelo. El desequilibrio de los datos es

un sesgo hacia la clase mayoritaria, por lo que, en última instancia, muestra una tendencia al sobreajuste u overfitting hacia esa clase. Existen diversas técnicas que permiten corregir esta situación:

- **Undersampling** o Submuestreo. Esta técnica consiste en reducir el número de observaciones de la clase mayoritaria, estableciendo quizás una ratio de 60/40. Esta técnica resulta efectiva si se respetan los grupos naturales que existen en los datos, así como el resto de las características presentes en la clase mayoritaria.
 - **Oversampling** o Sobremuestreo. Esta técnica consiste en aumentar el número de observaciones de la clase minoritaria mediante la creación de datos sintéticos que, al igual que la técnica anterior, respeten las características de esa clase.

Para la creación de datos sintéticos en escenarios de oversampling existen varios algoritmos que proporcionan buenos resultados. Quizás el más conocido y utilizado sea Synthetic Minority Oversampling TECnique (SMOTE). SMOTE no realiza una copia de las observaciones del dataset, sino que en su lugar genera nuevos datos de forma sintética utilizando los vecinos más cercanos de esos casos, respetando las características estadísticas de la clase. Además, los ejemplos de la clase mayoritaria también son submuestreados, lo que da lugar a un conjunto de datos más equilibrado. En R, el algoritmo SMOTE pertenece al paquete DMwR.

Para este caso particular, se utilizará una técnica simple de submuestreo basada en el paquete **unbalanced**, que actualmente no se encuentra disponible en el repositorio de CRAN por lo que, para su instalación, se debe ejecutar el siguiente código:

```
#install.packages("devtools") // Instalar si no se encuentra entre las librerías del
← sistema
library("devtools")
devtools::install_github("dalpozz/unbalanced")
library("unbalanced")
```

Una vez instalado, al igual que todas sus dependencias, se realiza el submuestreo del dataset siguiendo los siguientes pasos:

1.- Convertir la variable dependiente “Class” en factor:

```
creditcard$Class <- as.factor(creditcard$Class)
levels(creditcard$Class) <- c('0', '1')
```

2.- A continuación, ejecutar la función de submuestreo:

42.2. Modelización del fraude en la compra con tarjetas de crédito

625

```
names(undersampled_combined)[names(undersampled_combined) ==
  ↪ "undersampled_creditcard$Y"] <- "Class"
levels(undersampled_combined$Class) <- c('Legítima', 'Fraude')
```

3.- Comprobar el número de casos en el dataset sobre el que se ha ejecutado la función de submuestreo

```
creditcard$Class <- as.factor(creditcard$Class)
levels(creditcard$Class) <- c('0', '1')
```

4.- Realizar la gráfica para visualizar el número de elementos de cada clase después de realizar el submuestreo

```
library("ggplot2")
ggplot(data = undersampled_combined, aes(fill = Class))+
  geom_bar(aes(x = Class))+
  ggtitle("Número de casos de cada clase después de submuestreo",
    subtitle="Total muestrass: 984")+
  xlab("")+
  ylab("Muestras")+
  scale_y_continuous(expand = c(0,0))+
  scale_x_discrete(expand = c(0,0))+
  theme(legend.position = "ninguna",
    legend.title = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank())
```



Modelo de clasificación mediante regresión lógística

A continuación se procederá a la construcción de un modelo de regresión logística (véase Cap. ??) para una clasificación binaria en relación al fraude en transacciones con tarjeta de crédito a partir de los datos equilibrados obtenidos anteriormente. El dataframe que se utilizará, por lo tanto, será “undersampled_combined” que contiene 984 observaciones, un 50 % de las cuales son transacciones identificadas como fraude.

Lo primero, será realizar un par de pequeños cambios en el dataset, es decir, eliminar las variables `Time` y `Amount`, ya que no van a ser relevantes para el modelo, y cambiar por 0 y 1 las etiquetas “Legitima” y “Fraude”, respectivamente.

```
undersampled_combined <- subset(undersampled_combined, select = -c(Time, Amount) )
undersampled_combined$Class <- ifelse(undersampled_combined$Class == "Fraude", 1, 0)
```

Lo siguiente será dividir el conjunto de datos en los datasets de entrenamiento y test, para lo cual se aplicará la función “`split()`” con un SplitRatio de 0.80, es decir, un 80 % de los datos irán de forma aleatoria al dataset de entrenamiento, 788 observaciones, frente a las 196 observaciones que formarán el dataset de testing.

```
#install.packages("caTools") // Instalar si no se encuentra entre las librerías del
→ sistema
library("caTools")
set.seed(123)
split = sample.split(undersampled_combined$Class, SplitRatio = 0.80)
training = subset(undersampled_combined, split == TRUE)
test = subset(undersampled_combined, split == FALSE)
```

Con los datasets necesarios ya disponibles, el siguiente paso es entrenar el modelo de regresión logística que clasificará las transacciones en legítimas o fraudulentas. Para ello se utilizará el algoritmo GLM, creando un clasificador que se identificará como `undersampledModel` y al que se le pasarán los parámetros siguientes:

- **formula:** con este parámetro se indica la variable dependiente, `class`, seguida del simbolo `~` y un punto (con el punto se hace referencia al resto de variables del dataset, es decir, V1 a V28).
- **data:** el dataset con los datos de entrenamiento.
- **family:** al ser un clasificador con dos valores posibles (0, 1), se indica que será de tipo “binomial”.

```
undersampledModel = glm(Class ~ ., data = training, family = binomial())
```

Para ver,

```
summary(undersampledModel)
#>
#> Call:
#> glm(formula = Class ~ ., family = binomial(), data = training)
```

42.2. Modelización del fraude en la compra con tarjetas de crédito

627

```

#>
#> Deviance Residuals:
#>      Min       1Q   Median      3Q     Max
#> -2.2265 -0.1531  0.0000  0.0000  3.2827
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  4.6914    2.1341  2.198 0.027927 *
#> V1          -22.5178   5.8822 -3.828 0.000129 ***
#> V2           21.3999   5.7446  3.725 0.000195 ***
#> V3          -51.2534  13.5276 -3.789 0.000151 ***
#> V4           32.3808   8.3311  3.887 0.000102 ***
#> V5          -35.3374   9.3381 -3.784 0.000154 ***
#> V6          -12.1064   3.2062 -3.776 0.000159 ***
#> V7          -66.9041  17.8056 -3.757 0.000172 ***
#> V8           13.5412   3.8231  3.542 0.000397 ***
#> V9          -33.4896   8.8476 -3.785 0.000154 ***
#> V10         -78.8811  20.7460 -3.802 0.000143 ***
#> V11          55.5885  14.6046  3.806 0.000141 ***
#> V12         -99.4851  26.1453 -3.805 0.000142 ***
#> V13          0.8023   0.3577  2.243 0.024914 *
#> V14         -103.6671  27.1674 -3.816 0.000136 ***
#> V15          -1.8919   0.5720 -3.308 0.000941 ***
#> V16         -92.3210  24.4361 -3.778 0.000158 ***
#> V17         -168.2428  44.4714 -3.783 0.000155 ***
#> V18          -62.6637  16.6148 -3.772 0.000162 ***
#> V19          21.0326   5.4937  3.828 0.000129 ***
#> V20          10.8821   3.0659  3.549 0.000386 ***
#> V21          12.8444   3.4928  3.677 0.000236 ***
#> V22          1.3002   0.4491  2.895 0.003789 **
#> V23          -2.2403   0.7284 -3.076 0.002101 **
#> V24          -1.0921   0.5962 -1.832 0.067016 .
#> V25          3.5704   1.2027  2.969 0.002991 **
#> V26          0.8803   0.6295  1.399 0.161962
#> V27          17.3278   4.9277  3.516 0.000437 ***
#> V28          13.4671   3.5736  3.769 0.000164 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#> Null deviance: 1092.40 on 787 degrees of freedom
#> Residual deviance: 159.29 on 759 degrees of freedom
#> AIC: 217.29
#>
#> Number of Fisher Scoring iterations: 20

```

Con el modelo ya entrenado, se realizan las predicciones para los datos del conjunto de testing, utilizando para ello la función “predict()”. Los parámetros son simples: el primero es el modelo

o clasificador que se va a utilizar y que será “undersampledModel”; a continuación el tipo de dato que devolverá, en este caso “response”, el cual indica que el algoritmo devolverá las probabilidades de fraude listadas en un único vector, el cual estará disponible a partir de la variable “fraud_prob”; y por último, el parámetro “newdata” que hace referencia al dataset en el que se descarta la última columna por ser la que representa la variable dependiente.

```
fraud_prob = predict(undersampledModel, type = "response", newdata = test[,-29])
head(fraud_prob)
#>          4           6           7          10          13          19
#> 2.331615e-01 2.932459e-03 6.872599e-03 5.426427e-01 4.309168e-03 1.347347e-06
```

La visualización del vector con las predicciones puede parecer algo confusa, por lo que, a menudo, es preciso realizar una conversión de esas predicciones en valores 0 y 1, dependiendo del rango de valores a partir del cual se estime que una transacción es fraudulenta: por ejemplo, a partir del 60 % de probabilidad, la transacción será etiquetada como “1”; en caso contrario será etiquetada como “0”. Para ello se utiliza el siguiente código “ifelse”:

```
y_pred = ifelse(fraud_prob > 0.6, 1, 0)
```

La matriz de confusión

Como último paso del ejercicio se crea la matriz de confusión con el fin de visualizar qué tal se ha comportado el algoritmo, es decir, cuántos positivos y negativos ha logrado predecir correctamente. Para ello se creará la variable `confusionMatrix`, en la cual se almacenará el resultado de la comparación entre el vector del dataset de testing, es decir, los datos etiquetados originalmente, y el vector de sus traducciones a 0 y 1, resultado del algoritmo. El resultado, como se puede comprobar es que ha evaluado correctamente 186 de las 196 observaciones.

```
confusionMatrix = table(test[, 29], y_pred)
confusionMatrix
#>     y_pred
#>     0 1
#> 0 93 5
#> 1 7 91
```

Apéndice A

Información de la sesión

```
sessionInfo()
#> R version 4.2.1 (2022-06-23 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=Spanish_Spain.utf8  LC_CTYPE=Spanish_Spain.utf8
#> [3] LC_MONETARY=Spanish_Spain.utf8 LC_NUMERIC=C
#> [5] LC_TIME=Spanish_Spain.utf8
#>
#> attached base packages:
#> [1] stats      graphics   grDevices  utils      datasets   methods    base
#>
#> other attached packages:
#> [1] flextable_0.8.1   fontawesome_0.4.0
#>
#> loaded via a namespace (and not attached):
#> [1] zip_2.2.1        Rcpp_1.0.9       pillar_1.8.1     compiler_4.2.1
#> [5] R.methodsS3_1.8.2 R.utils_2.12.2   base64enc_0.1-3  tools_4.2.1
#> [9] digest_0.6.30    uuid_1.1-0      tibble_3.1.8    evaluate_0.16
#> [13] lifecycle_1.0.3  gtable_0.3.1   R.cache_0.16.0  pkgconfig_2.0.3
#> [17] rlang_1.0.6     DBI_1.1.3      cli_3.4.1      rstudioapi_0.14
#> [21] yaml_2.3.5      xfun_0.35     fastmap_1.1.0  stringr_1.4.1
#> [25] dplyr_1.0.10    officer_0.4.4  styler_1.8.1   xml2_1.3.3
#> [29] knitr_1.39     generics_0.1.3  gdtools_0.2.4  vctrs_0.5.1
#> [33] systemfonts_1.0.4 tidyselect_1.2.0  grid_4.2.1     glue_1.6.2
#> [37] data.table_1.14.6 R6_2.5.1      fansi_1.0.3    rmarkdown_2.14
#> [41] bookdown_0.28    purrrr_0.3.5   ggplot2_3.4.0  magrittr_2.0.3
#> [45] scales_1.2.1    htmltools_0.5.4 assertthat_0.2.1 colorspace_2.0-3
```

```
#> [49] utf8_1.2.2      stringi_1.7.8      munsell_0.5.0      R.oo_1.25.0
```

Bibliografía

- Allaire, J. (2022). *quarto: R Interface to 'Quarto' Markdown Publishing System*. R package version 1.2.
- Amat Rodrigo, J. (2017). *Clustering y heatmaps: aprendizaje no supervisado*.
- Anderberg, M. R. (1973). *Cluster analysis for applications: probability and mathematical statistics*. Academic press, New Yoor, USA.
- Ang, Q. W., Baddeley, A., and Nair, G. (2012). Geometrically corrected second order analysis of events on a linear network, with applications to ecology and criminology. *Scandinavian Journal of Statistics*, 39(4):591–617.
- Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Studies in Operational Regional Science. Springer Netherlands.
- Anselin, L. (1996). *The Moran scatterplot as an ESDA tool to assess local instability in spatial association*. Routledge. Num Pages: 16.
- Anselin, L. (2013). *Spatial econometrics: methods and models*, volume 4. Springer Science & Business Media.
- Anselin, L. (2017). A Local Indicator of Multivariate Spatial Association: Extending Geary's c. *Center for Spatial Data Science Working papers. University of Chicago*.
- Astigarraga, J. and Cruz-Alonso, V. (2022). ¡se puede entender cómo funcionan git y github! *Ecosistemas*, 31(1):2332.
- Baddeley, A., Davies, T. M., Rakshit, S., Nair, G., and McSwiggan, G. (2022). Diffusion smoothing for spatial point patterns. *Statistical Science*, 37(1):123–142.
- Baddeley, A., Nair, G., Rakshit, S., McSwiggan, G., and Davies, T. M. (2021). Analysing point patterns on networks-a review. *Spatial Statistics*, 42:100435.
- Baddeley, A., Rubak, E., and Turner, R. (2015). *Spatial Point Patterns: Methodology and Applications with R*. CRC Press.
- Baddeley, A. and Turner, R. (2005). *spatstat*: an R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12:1–42.

- Barr, C. D. and Schoenberg, F. P. (2010). On the Voronoi estimator for the intensity of an inhomogeneous planar Poisson process. *Biometrika*, 97(4):977–984.
- Baumer, B., Kaplan, D., and Horton, N. (2021). *Modern Data Science with R*. Texts in statistical science. Chapman & Hall/CRC, Boca Raton.
- Beh, E. J. and Lombardo, R. (2014). *Correspondence Analysis: Theory, Practice and New Strategies*. Wiley Series in Probability and Statistics. Wiley.
- Berlusconi, G., Calderoni, F., Parolini, N., Verani, M., and Piccardi, C. (2016). Link prediction in criminal networks: A tool for criminal intelligence analysis. *PLOS ONE*, 11(4):e0154244.
- Bernaards, C. and Jennrich, R. (2005). Gradient projection algorithms and software for arbitrary rotation criteria in factor analysis. *Educational and Psychological Measurement*, 65:676–696.
- Bivand, R. (2020). *classInt: Choose Univariate Class Intervals*. R package version 0.4-3.
- Bivand, R. (2022). *spdep: Spatial Dependence: Weighting Schemes, Statistics*. R package version 1.2-2.
- Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.
- Bock, H. H. (1974). *Automatische Classifikation*. Studia Mathematica. Vandenhoeck and Ruprecht, Göttingen, Germany.
- Boehmke, B. and Greenwell, B. M. (2019). *Hands-on machine learning with R*. CRC press.
- Boehmke, B. y Greenwell, B. (2020). *Hands-On Machine Learning with R*. Chapman and Hall/CRC. Chapman and Hall.
- Borgatti, S. P. (2022). *Analyzing Social Networks Using R: Your Essential Guide*. SAGE Publications Ltd, Estados Unidos.
- Borji, A. (2022). Generated faces in the wild: Quantitative comparison of stable diffusion, midjourney and dall-e 2. *arXiv preprint arXiv:2210.00586*.
- Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and regression trees. 1st edition.
- Brian, S. (1993). Cluster analysis 3rd ed. *Edward Arnold, London*, 169.
- Brockwell, P. J. and Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, Switzerland.

- Burkov, A. (2019). *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada.
- Carrasco-Oberto, G. I. (2020). *Cluster no jerárquicos versus cart y biplot*. PhD thesis, Universidad de Salamanca.
- Chacon, S. (2009). *Pro Git*. Apress.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., et al. (2000). Crisp-dm 1.0: Step-by-step data mining guide. *SPSS inc*, 9(13):1–73.
- Chatfield, C. and Collins, A. (1980). *Introduction to multivariate analysis*. Chapman&Hall/CRC.
- Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4.
- Chilès, J. P. and Delfiner, P. (1999). *Geostatistics: Modeling Spatial Uncertainty*. John Wiley and Sons, Ltd, Chichester, UK.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.
- Cramer, W., Guiot, J., Marini, K., Secretariat, M., and Bleu, P. (2020). Climate and environmental change in the mediterranean basin—current situation and risks for the future. *First Mediterranean Assessment Report. Union for the Mediterranean, Plan Bleu, UNEP/MAP*.
- Cressie, N. and Wikle, C. K. (2015). *Statistics for Spatio-Temporal Data*. John Wiley & Sons.
- Cressie, N. A. C. (1993). *Statistics for Spatial Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA.
- Cronie, O. and Van Lieshout, M. N. M. (2018). A non-model-based approach to bandwidth selection for kernel estimators of spatial intensity functions. *Biometrika*, 105(2):455–462.
- Cryer, J. D. and Chan, K.-S. (2010). *Time Series Analysis with Applications in R*. Springer texts in Statistics. Springer, Iowa, USA.
- Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*:1695.
- Cuadras, C. M. (2007). *Nuevos Métodos de Análisis Multivariante*. CMC Editions, Barcelona, Spain.
- Cutler, A. and Zhao, G. (1999). Fast classification using perfect random trees. *Utah State University*.
- DANE (2019). Proyecciones de población departamentales y municipales por área 2005-2020. www.dane.gov.co.
- Davies, T. M. and Baddeley, A. (2018). Fast computation of spatially adaptive kernel estimates. *Statistics and Computing*, 28(4):937–956.
- De la Fuente, S. (2011). *Análisis Factorial*. Madrid, España.

- de la Real Academia Española, D. (2023). Inteligencia artificial.
- de Leeuw, J. and Mair, P. (2009). Multidimensional scaling using majorization: SMACOF in R. *Journal of Statistical Software*, 31(3):1–30.
- Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., and Fei-Fei, L. (2012). Imagenet large scale visual recognition competition 2012 (ilsvrc2012). *See net. org/challenges/LSVRC*, 41.
- Diday, E. (1971). Une nouvelle méthode en classification automatique et reconnaissance des formes la méthode des nuées dynamiques. *Revue de statistique appliquée*, 19(2):19–33.
- Diday, E. (1973). The dynamic clusters method in nonhierarchical clustering. *International Journal of Computer & Information Sciences*, 2(1):61–88.
- Diggle, P. (1985). A kernel method for smoothing point process data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 34(2):138–147.
- Diggle, P. (2013). *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. CRC press.
- Diggle, P. and Giorgi, E. (2019). *Model-based Geostatistics for Global Public Health: Methods and Applications*. Chapman and Hall/CRC.
- Easley, David; Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, Estados Unidos.
- Edelbrock, C. (1979). Mixture model tests of hierarchical clustering algorithms: The problem of classifying everybody. *Multivariate Behavioral Research*, 14(3):367–384.
- Elhorst, J. P. (2010). Applied Spatial Econometrics: Raising the Bar. *Spatial Economic Analysis*, 5(1):9–28.
- Facebook Marketing Science (2021). *Robyn*.
- Fay, C., Rochette, S., Guyader, V., and Girard, C. (2021). *Engineering Production-Grade Shiny Apps*. Chapman and Hall/CRC.
- Firth, J. (1957). A synopsis of linguistic theory, 1930–1955. In *Selected Papers of J.R. Firth 1952–1959*, ed. Frank Palmer, 168–205. Londres: Longman.
- Fleitas, F. (2017). *La Inteligencia Artificial e Ingeniería del Conocimiento: Guía de la Inteligencia Artificial e Ingeniería del Conocimiento con ejemplos de Sistemas Expertos en el Lenguaje CLIPS*. Editorial Académica Española.
- Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, 21(3):768–769.
- Fradejas Rueda, J. M. (2022). *Cuentapalabras. Estilometría y análisis de datos con R para filólogos*. <http://www.aic.uva.es/cuentapalabras/>.
- Fruchterman, T. M. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164.

- Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.
- Gallardo San-Salvador, J. A. (2022). *Introducción al Análisis Cluster*.
- Gallardo-San Salvador, J. A. and Vera-Vera, J. F. (2004). *Técnicas aplicadas de análisis de datos multivariantes*. Universidad de Granada, Granada, Spain.
- Garcia, G. B., Suarez, O. D., Aranda, J. L. E., Tercero, J. S., and Gracia, I. S. (2015). *Learning Image Processing with OpenCV*. Packt Publishing.
- Gentile, C. and Warmuth, M. K. (1998). Linear hinge loss and average margin. *Advances in neural information processing systems*, 11.
- Getis, A. (1999). *Spatial statistics*. Longley, P., Goodchild, M., Maguire, D. y Rhind, D. (Eds.) Geographical Information Systems.
- Gilmore, R., Hutchins, S., Pastoor, D., Attali, D., Singham, L., Raja, A. M., Trimarchi, L., Khanal, K., Columbus, A., Howard, P., and Zhang, L. (2017). Awesome R Shiny.
- Giraud, T. (2022). *mapsf: Thematic Cartography*. R package version 0.4.0.
- Gohel, D. and Skintzos, P. (2022). *flextable: Functions for Tabular Reporting*. R package version 0.8.3.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Adaptive computation and machine learning. MIT Press.
- Greenacre, M. (2008). *La práctica del análisis de correspondencias*. Fundación BBVA.
- Guerry, A.-M. (1833). *Essai Sur La Statistique Morale de La France*. Crochard.
- Hamilton (1994). *Time Series Analysis*. Statistics. Princeton University Press, Princeton, NJ, USA.
- Harman, H. H. (1976). *Modern Factor Analysis (Third Edition Revised)*. Chicago, USA.
- Hartigan, J. and Wong, M. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. series c (applied statistics)*, 28(1):100–108.
- Hartigan, J. A. (1975). *Clustering algorithms*. John Wiley & Sons, Inc.
- Hastie, T. and Tibshirani, R. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Monographs on Statistics & Applied Probability. Chapman and Hall/CRC.
- Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.
- Hernangómez, D. and Fernández-Avilés, G. (2022). *Visualización y geolocalización de datos con R*. Netlify, Online.
- Hijmans, R. J. (2022). *raster: Geographic Data Analysis and Modeling*.

- Ho, D. E., Imai, K., King, G., and Stuart, E. A. (2007). Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political analysis*, 15(3):199–236.
- Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.
- Hothorn, T. and Everitt, B. (2014). *A Handbook of Statistical Analyses using R*. Routledge.
- Illian, J., Penttinen, A., Stoyan, H., and Stoyan, D. (2008). *Statistical Analysis and Modelling of Spatial Point Patterns*, volume 70. John Wiley & Sons.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer. <https://www.statlearning.com/>.
- Jenny Bryan, the STAT 545 TAs, J. H. (2021). *Happy Git and GitHub for the useR*.
- Jobson, J. D. (1992). *Applied Multivariate Data Analysis. Vol. II*. Springer test inStatistics. Springer-Verlag, NeW York, USA.
- Jockers, M. (2014). *Text analysis with R for students of literature*. Nueva York: Springer.
- Jockers, M. (2017). Introduction to the syuzhet package. <https://cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html>.
- Jones, M. C. (1993). Simple boundary correction for kernel density estimation. *Statistics and computing*, 3(3):135–146.
- Journel, A. G. and Huijbregts, C. H. J. (1978). *Mining Geostatistics*. Academic Press, New York, USA.
- Kassambara, A. (2017). *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning (Multivariate Analysis) 1st Ed.* sthada.com.
- Kaufman, L. and Rousseeuw, P. J. (1990). Divisive analysis (program diana). In Kaufman, L. and Rousseeuw, P. J., editors, *Finding Groups in Data: An Introduction to Cluster Analysis*, pages 68–125. John Wiley and Sons, Inc., Hoboken.
- Kelejian, H. H. and Prucha, I. R. (2010). Specification and estimation of spatial autoregressive models with autoregressive and heteroskedastic disturbances. *Journal of Econometrics*, 157(1):53–67.
- Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466.
- Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27(2):97–111.
- Kuhn, M. (2019). CRAN Task View: Reproducible Research. R Task View.
- Lê, S., Josse, J., and Husson, F. (2008). FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18.

- LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Lee, C.-P. and Lin, C.-J. (2013). A study on l2-loss (squared hinge-loss) multiclass svm. *Neural computation*, 25(5):1302–1323.
- Leisch, F. (2002). Sweave: Dynamic generation of statistical reports using literate data analysis. In *Compstat*, pages 575–580. Springer.
- LeSage, J. and Pace, R. K. (2009). *Introduction to Spatial Econometrics*. Chapman and Hall/CRC.
- Lewis, J. A. (1999). Statistical principles for clinical trials (ich e9): an introductory note on an international guideline. *Statistics in medicine*, 18(15):1903–1942.
- Liu, B. (2015). *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.
- Liu, X., Rivera, S. C., Moher, D., Calvert, M. J., and Denniston, A. K. (2020). Reporting guidelines for clinical trial reports for interventions involving artificial intelligence: the consort-ai extension. *BMJ*, 370.
- Lovelace, R., Nowosad, J., and Münchow, J. (2019). *Geocomputation with R*. CRC Press, Taylor and Francis Group, Boca Raton.
- Lozano Zahonero, M. (2020). Una nueva visión de la supuesta influencia de *Madame Bovary* en *La Regenta* a través de la estilometría y el análisis de sentimientos basados en lenguaje R. *Orillas: rivista d'ispanistica*, 9:573–607.
- López-González, E. and Hidalgo-Sánchez, R. (2010). Escalamiento multidimensional no métrico. un ejemplo con r empleando el algoritmo smacof. *Estudios sobre educación*, 8(1):9–35.
- MacNaughton-Smith, P., Williams, W. T., Dale, M. B., Mockett, L. G., and Dunn, C. (1964). Dissimilarity analysis: a new technique of hierarchical sub-division. *Nature*, 202:1034–1035.
- MacQueen, J. (1967). Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297.
- Mair, P., Groenen, P. J. F., and de Leeuw, J. (2022). More on multidimensional scaling in R: smacof version 2. *Journal of Statistical Software*, 102(10):1–47.
- Mardia, K., Kent, J., and Bibby, J. (1979a). *Multivariate Analysis*. London, UK.
- Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979b). *Multivariate Analysis*. Academic Press, London, UK.
- Martínez, R. G., Carrasco, R. A., García-Madariaga, J., Gallego, C. P., and Herrera-Viedma, E. (2019). A comparison between fuzzy linguistic rfm model and traditional rfm model applied to campaign management. case study of retail business. *Procedia Computer Science*, 162:281–289.

- Martínez R., Molina J. M., C. J. (2005). *Desarrollo de sistemas basados en el conocimiento*. Sanz y Torres S. L.
- Martori, J. C., Hoberg, K., and Madariaga, R. (2008). La incorporación del espacio en los métodos estadísticos: Autocorrelación espacial y segregación. *Actas del X Coloquio Internacional de Geocrítica*.
- Matheron, G. (1962). *Traité de Geostatistique Appliquée. vol I*. Éditions Technip, Paris, France.
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.
- McNicholas, P. D. (2016). Model-based clustering. *Journal of Classification*, 33(3):331–373.
- Mella, J. M. and Chasco, C. (2006). *Urban Growth and Territorial Dynamics: A Spatial-Econometric Analysis of Spain*. Edward Elgar Publishing.
- Minsky, M. and Papert, S. (1969). An introduction to computational geometry. *Cambridge tiass.*, HIT, 479:480.
- Missouri, Rokia; Sarr, I. (2015). *Social Network Analysis - Community Detection and Evolution*. Springer, Estados Unidos.
- Møller, J. and Waagepetersen, R. (2003). *Statistical Inference and Simulation for Spatial Point Processes*. CRC Press.
- Monsalve, C. (2019). Medellín necesita 2000 uniformados más para reforzar seguridad. <https://www.bluradio.com/blu360/antioquia/medellin-necesita-2-000-uniformados-mas-para-reforzar-seguridad-policia>.
- Montero, J. M. (2002). Una propuesta de corrección de continuidad asimétrica para tablas de contingencia (2x2) con totales marginales fijos. *Estadística Española*, 44(149):29–46.
- Montero, J.-M., Fernández-Avilés, G., and Mateu, J. (2015). *Spatial and spatio-temporal geostatistical modeling and kriging*. John Wiley & Sons.
- Montero, J. M. and Larraz, B. (2008). *Introducción a la Geoestadística Lineal*. Metodología y Análisis de Datos en Ciencias Sociales. Netbiblo, A Coruña, España.
- Moradi, M. (2018). *Spatial and Spatio-Temporal Point Patterns on Linear Networks*. PhD Dissertation, University Jaume I.
- Moradi, M., Cronie, O., Rubak, E., Lachieze-Rey, R., Mateu, J., and Baddeley, A. (2019). Resample-smoothing of voronoi intensity estimators. *Statistics and computing*, 29(5):995–1010.
- Morrison, D. F. (1976). *Multivariate Statistical Methods-2*. New York, NY (USA) McGraw-Hill.
- Muñoz-Reja, I. C., Carretero, A. I. G., and Cejudo, F. G. (2018). *Calidad de datos*. RA-MA Editorial.
- Mínguez Salido, R. and García Centeno, M. C. (2011). *Modelos de series temporales aplicados a rendimientos financieros*. Estadística económica y finanzas. Netbiblo, España.

- Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML 2010*, pages 807–814.
- Ng, R. T. and Han, J. (2002). Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016.
- Novikoff, A. B. (1962). On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, New York, NY, USA. Polytechnic Institute of Brooklyn.
- Okabe, A. and Sugihara, K. (2012). *Spatial Analysis Along Networks: Statistical and Computational Methods*. John Wiley & Sons.
- OpenAI (2022). Chatgpt.
- Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.
- Paul Euler, L. (1736). *Solutio problematis ad geometriam situs pertinentis*. Commentarii Academice Scientiarum Imperialis Petropolitane 8.
- Pearson, K. (1904). On the theory of contingency and its relation to association and normal correlation. In Department of Applied Mathematics, University College, U. o. L., editor, *Mathematical Contributions to the Theory of Evolution*, pages Chapter XXX, 1–34. Dulau and CO., London, UK.
- Pebesma, E. (2022). *sf: Simple Features for R*. R package version 1.0-7.
- Pebesma, E. and Bivand, R. (2022). Spatial data science: With applications in r.
- Pemberton, J. (2011). Time series analysis with applications in r, second edition. *Journal of Applied Statistics*, 38(6):1311–1332.
- Perez Sola, C. (2021). *Análisis de datos de redes sociales*. Editorial UOC, España.
- Pizarro, M., Hernangómez, D., and Fernández-Avilés, G. (2021). *climaemet: Climate AEMET Tools*.
- Pérez-Gil, J. A., Chacón, S., and Moreno, R. (2000). Validez de constructo: el uso de análisis factorial exploratorio-confirmatorio para obtener evidencias de validez. *Psicothema*, 12(Su2):442–446.
- Pérez Infante, J. I. (2006). *Las estadísticas del mercado de trabajo en España*. Ministerio de Empleo y Seguridad Social. Subdirección General de Información Administrativa y Publicaciones.
- Rakshit, S., Baddeley, A., and Nair, G. (2019a). Efficient code for second order analysis of events on a linear network. *Journal of Statistical Software*, 90:1–37.
- Rakshit, S., Davies, T., Moradi, M., McSwiggan, G., Nair, G., Mateu, J., and Baddeley, A. (2019b). Fast kernel smoothing of point patterns on a large network using two - dimensional convolution. *International Statistical Review*, 87(3):531–556.

- Rakshit, S., Nair, G., and Baddeley, A. (2017). Second-order analysis of point patterns on a network using any distance metric. *Spatial Statistics*, 22:129–154.
- Restrepo, V. (2019). Qué tan segura se siente la gente en medellín? <https://www.elcolombiano.com/antioquia/seguridad/percepcion-de-seguridad-en-medellin-encuesta-de-victimizacion-PC10033581>.
- Revelle, W. (2022). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Evanston, Illinois. R package version 2.2.5.
- Reynolds, H. T. (1984). *Analysis of Nominal Data (2nd edition)*. Quantitative Applications in the Social Sciences. Sage Publication, London, UK.
- Reynolds, R. W., Banzon, V. F., and Program, N. C. (2008). Noaa optimum interpolation 1/4 degree daily sea surface temperature (oisst) analysis, version 2. *NOAA National Centers for Environmental Information*.
- Rivera, S. C., Liu, X., Chan, A.-W., Denniston, A. K., Calvert, M. J., Ashrafi, H., Beam, A. L., Collins, G. S., Darzi, A., Deeks, J. J., et al. (2020). Guidelines for clinical trial protocols for interventions involving artificial intelligence: the spirit-ai extension. *The Lancet Digital Health*, 2(10):e549–e560.
- Rosenbaum, P. R. (2005). Observational study. *Encyclopedia of statistics in behavioral science*.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.
- Ruiz-Maya, L., Martin-Pliego, J., Montero, J. M., and Uríz, P. (1995). *Análisis Estadístico de Encuestas: Datos Cualitativos*. Alpha Centauro, Madrid, España.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Sanabria, A. M. F., Castañeda, M. P. B., Ramos, R. R. R., and Mateu, J. (2022). Identification of patterns for space-time event networks. *Applied Network Science*, 7(1):1–24.
- Schabenberger, O. and Gotway, C. A. (2005). *Statistical methods for spatial data analysis*. Texts in statistical science. Chapman & Hall/CRC, Boca Raton.
- Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and Algorithms*. The MIT Press.
- Schölkopf, B., Simard, P., Smola, A., and Vapnik, V. (1997). Prior knowledge in support vector kernels. *Advances in neural information processing systems*, 10.
- Scholkopf, B., Sung, K.-K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. (1997). Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765.
- Schubert, E. and Rousseeuw, P. J. (2021). Fast and eager k-medoids clustering: O(k) runtime improvement of the pam, clara, and clarans algorithms. *Information Systems*, 101:101804.

- Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: John Wiley & Sons.
- Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.
- Shumway, R. H. and Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics. Springer International Publishing, Cham.
- Silge, J. and Robinson, D. (2017). *Text Mining with R: A Tidy Approach*. O'Reilly Media, Inc. <https://www.tidytextmining.com>.
- Silverman, B. W. (1982). Kernel density estimation using the fast Fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(1):93–99.
- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Routledge.
- Sokal, R. R. and Rolf, F. J. (2012). *Biometry. The Principles and Practice in Statistics in Biological Research, 4th edition*. W.H. Freeman and Company, New York, USA.
- Späth, H. (1975). Cluster-analyse-algorithmen zur objektklassifizierung und datenreduktion. *Verfahren der Datenverarbeitung*.
- Tennekes, M. (2018). tmap: Thematic maps in R. *Journal of Statistical Software*, 84(6):1–39.
- Tetko, I. V., Livingstone, D. J., and Luik, A. I. (1995). Neural network studies. 1. comparison of overfitting and overtraining. *Journal of chemical information and computer sciences*, 35(5):826–833.
- Theobald, O. (2017). *Machine learning for absolute beginners: a plain English introduction*, volume 157. Scatterplot press.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58:267–288.
- Tobler, W. R. (1970a). A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46:234–240.
- Tobler, W. R. (1970b). A computer movie simulating urban growth in the Detroit region. *Economic geography*, 46(sup1):234–240.
- Toharia, L. (2012). *El mercado de trabajo en la obra de Luis Toharia*. Ministerio de Empleo y Seguridad Social.
- Tribunale di Lecco, 2a Sezione Penale (2009). Sentenza nei confronti di Alcaro Luigi + 56 (Operazione Oversize). Procedimento n. 31149/01 + 10309/03 + 42859/03 + 3885/05 RGNR.
- Tribunale di Milano, Ufficio del giudice per le indagini preliminari (2006). Ordinanza di applicazione della misura della custodia cautelare in carcere nei confronti di Alcaro Luigi + 56 (Operazione Oversize). Procedimento n. 31149/01 + 10309/03 + 42859/03 + 3885/05 RGNR.

- Uriel Jiménez, E. and Peiro Giménez, A. (2000). *Introducción al análisis de series temporales*. Estadística. Paraninfo, Madrid, España.
- Vallone, A. and Chasco, C. (2020). Spatiotemporal methods for analysis of urban system dynamics: an application to chile. *The Annals of Regional Science*, 64(2):421–454.
- Vapnik, V. N. (1997). The support vector method. In *International Conference on Artificial Neural Networks*, pages 261–271. Springer.
- Wade, C. (2020). *Hands-On Gradient Boosting with XGBoost and scikit-learn: Perform accessible machine learning and extreme gradient boosting with Python*. Packt Publishing Ltd.
- Walker, K. (2022). *crsuggest: Obtain Suggested Coordinate Reference System Information for Spatial Data*. R package version 0.4.
- Wasserman, S. (1995). *Social Network Analysis: Methods and Applications*. Cambridge University Press, Estados Unidos.
- Webb, G. I. (2011). Filtered-top-k association discovery. *WIREs Data Mining and Knowledge Discovery*, 1(3):183–192.
- Wickham, H. (2021). *Mastering shiny*. O'Reilly Media, Inc..
- Wikle, C. K., Zammit-Mangion, A., and Cressie, N. (2019). *Spatio-temporal Statistics with R*. Chapman and Hall/CRC.
- Wilks, S. S. (1935). The likelihood test of independence in contingency tables. *Ann. Math. Statist.*, 6(4):190–196.
- Winston, C., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2020). *Shiny: Web Application Framework for R*.
- Wismüller, A., Verleysen, M., Lee, J. A., and Aupetit, M. (2010). Recent advances in nonlinear dimensionality reduction, manifold and topological learning. *Conference: ESANN 2010, 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28-30, 2010, Proceedings*, pages 71–80.
- Xiao, N. (2018). Awesome Shiny Extensions.
- Xie, Y. (2017). *Dynamic Documents with R and knitr*. Chapman & Hall/CRC The R Series. CRC Press.
- Xie, Y., Allaire, J., and Grolemund, G. (2019). *R Markdown: The Definitive Guide*. Chapman and Hall/CRC the R Series. Taylor & Francis, CRC Press.
- Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.
- Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B*, 67:301–320.

Índice alfabético

(, 403
bagging, 167
'fluidPage()' :, 467

adstock, 513
agrupación, 201
ajuste automático, 122, 141, 143, 178, 190, 196
ajuste semivariográfico, 407
 automático, 407
 manual, 407

alcance, 401
algoritmo
 de árbol, 110

alpha de Cronbach, 257
ancho de silueta promedio, 221
anomalías, 581
análisis

 geoespacial, 495
 cluster, 201
 de colocaciones, 343
 de componentes principales, 235, 238, 245
 de correspondencias, 281
 de redes, 356
 de sentimientos, 344–346, 351
 de supervivencia, 564
 de texto, 612
 de textos, 341
 discriminante, 59
 discriminante cuadrático, 60
 discriminante lineal, 60
 factorial, 247, 248, 250, 252, 254, 258, 266
 factorial confirmatorio, 248
 factorial exploratorio, 248
 imagen, 257
análisis conjunto, 75
aplicaciones web interactivas, 465
aprendizaje

múltiple, 235, 246
no supervisado, 235, 236
supervisado, 235, 236
aprendizaje ensamblado, 165, 184
arista, 112, 360
asistencias, 570
asociación
 negativa, 92, 99
 perfecta e implícita de tipo 1, 99
 perfecta e implícita de tipo 2, 99
 perfecta y estricta, 99
 positiva, 92, 99
atributo, 87
Autocorrelación espacial, 413, 417
autovalor, 237–241, 246, 253–256, 264
autovector, 237–240, 245, 255

bagging, 165, 167, 170, 171, 184, 187
base de conocimiento, 597
batch, 308, 313
bigrama, 614, 615
bigramas, 612
bioestadística, 559
blog, 464
boosting, 165, 184, 187, 193
bootstrap, 167, 171, 173
botón circular, 471

c-medias, 233
caja negra, 145
camino, 363
campo aleatorio espacial, 396
características de segundo orden, 442
cargas, 242
cargas factoriales, 249, 255–261, 266
categoría, 87
centralidad, 367

- cheatsheet, 456
- chunk, 454
- clasificación, 201, 296, 299–301, 307, 309, 320, 322, 327, 335, 338
 - multiclas, 140
- clasificación binaria, 139
- cluster de variables, 203
- clusterización, 201
 - jerárquica, 201
 - no jerárquica, 201
- coeficiente
 - de correlación lineal
 - cofenético, 219
 - divisivo, 217
 - de coincidencias simple, 207
 - de congruencia, 206
 - de correlación
 - de Kendall, 206
 - de Pearson, 206
 - de Spearman, 206
 - de Czekanowski, 207
 - de Gower, 208
 - de Jacard, 207
 - de Rogers-Tanimoto, 207
 - de Russell y Rao, 207
 - de Sokal y Sneath, 207
- coeficiente de correlación parcial, 253
- colecciones, 498
- comercio, 545
- complejidad, 117, 139, 172, 173, 193
- completitud, 250
- componentes principales, 20, 203, 235, 238–240, 242, 244–246, 249, 254, 266
 - estimación, 240
 - interpretación, 242
 - número de componentes a retener, 240
 - obtención, 235
- componentes reactivos, 476
- comunalidad, 248, 250, 253–259, 264
- comunalidades, 369
- construcción, 549
- consumo eléctrico, 589
- contingencia, 87
- contraste
 - de esfericidad de Bartlett, 252
 - de razón de verosimilitudes, 264
- contraste de independencia, 90
- bilateral, 91
- contraste Chi-cuadrado, 97
- contraste razón de verosimilitudes, 97
- en tablas 2x2, 90
 - contraste aproximado, 90, 93, 95
 - contraste aproximado con corrección de continuidad, 90, 94, 95
- contraste de razón de verosimilitudes, 96
 - contraste exacto, 90, 91, 95
 - test exacto de Fisher, 91
 - total muestral fijo, 95
 - totales marginales de ambos factores fijos, 91
 - totales marginales de un factor fijos, 95
- en tablas multidimensionales, 104
- en tablas RxR, 96
 - contraste aproximado, 97
 - contraste aproximado con corrección de continuidad, 98
 - unilateral, 91
- control de versiones, 482
- control deslizante, 470
- coordenadas, 381
- coordinate reference system, 381
- coronavirus, 559
- corpus, 342
- corrección de continuidad
 - de Yates, 94
- COVID-19, 543
- COVID-19, 559
- criterio
 - de Catell, 241
 - de Kaiser, 241
 - de Kaiser, 254
 - del bastón roto, 241, 242
- criterio del gap, 221
- cross validation, 593
- CRS, 381
 - geográficos, 383
 - proyectados, 384
- curvilinear component analysis, 245
- código abierto, 482
- Daniel G. Krige, 409

Índice alfabético

645

- dashboard, 463
data augmentation, 329, 330
datos
 de patrones de puntos, 380
 espaciales, 379
 geo-referenciados, 379
 geoestadísticos, 380
 geográficos, 379
 lattice, 380
 raster, 387
 reticulares, 380
 vector, 385
datos sintéticos, 622
DBSCAN, 230, 231
deep learning, 293–297, 308, 329
DENCLUE, 230
dendrograma, 211, 220
Dependencia espacial, 413
dependencia espacial, 395, 396, 399, 402, 409, 413
 análisis estructural, 396
descenso del gradiente, 295, 307, 324
desequilibrados, 621
detección, 296
detección de emociones, 345, 351
deviance, 20
diagrama de doble escala, 530
Diagrama de Moran, 417
diagrama de Shepard, 273, 277
diagrama lineal, 527
DIANA, 217
diseño experimental, 89
 total muestral fijo, 90
 totales marginales de ambos factores fijos, 89
 totales marginales de uno de los factores fijos, 90
disimilaridad, 203
Distancia
 reproducida, 270
distancia, 151, 203, 267–271
 angular, 383
 Chebychev, 206
 coseno, 206
 de Chebychev, 206
 de Gower, 151
de Mahalanobis, 206
de Minkowski, 205
entre centroides, 211
euclídea, 151, 204
euclídea al cuadrado, 204
inter-grupos, 214, 221
intra-grupo, 229
intra-grupos, 221
Manhattan, 205, 228, 229
manhattan, 151
media, 211
Minkowski, 205
distribución, 546
dominios irregulares, 438
Dropout, 328
early stopping, 328
edad, 543
efecto pepita, 402
 puro, 402
elastic net, 35
elecciones Andaluzas, 535
elemento
 atípico, 231
 central, 231
 denso-alcanzable, 231
 denso-alcanzable directamente, 231
 denso-conectado, 231
 frontera, 231
EM, 230, 233
emociones, 345
ensayo clínico, 560
entropía, 115, 171, 174, 235
epsilon-neighborhood, 231
equilibrados, 624
ergodicidad, 39, 397
error
 de continuización, 94
error de predicción, 410
 desviación típica, 411
escala, 149
escalamiento
 multidimensional, 267, 269
 métrico, 271, 272, 274, 279
 no métrico, 271, 275, 279
escalamiento multidimensional, 235

- especificidad, 250
- estacionariedad, 39
- estadística espacial, 431
- estadístico Chi-cuadrado, 94, 97
 - ajustado, 94
 - corregido de continuidad de Yates, 94
 - corregido de continuidad de Yates, 94
- estadístico G, 97
- estandarización, 27
- estilometría, 344
- estimador de difusión, 438
- estimador de remuestreo-suavizado, 441
- estructura de dependencia espacial, 442
- estructura factorial, 251, 258, 260, 263
 - de referencia, 260
 - oblicua, 260
- estructura multidimensional, 580
- estructura simple, 256, 258, 260
 - oblicua, 260
 - ortogonal, 260
- estudio observacional, 560
- EXP (Holdout Testing o Experiments), 512
- extreme gradient boosting, 193
- facebook, 360
- factor, 87
- factores
 - comunes, 248–251, 253–259, 262–264, 266
 - de referencia, 260
 - específicos, 249, 264
 - subyacentes, 247, 265
 - únicos, 249, 250, 264
- fenómeno regionalizado, 395
- filograma, 213
- Fisher-Jenks, 390
- Forgy, 226
- frecuencia
 - esperada, 92, 97
- frequency, 552
- frontera
 - de decisión, 137
- fuentes de asociación, 98, 101, 103
- funciones núcleo, 433
- función
 - de autocorrelación (ACF), 39
 - de autocorrelación parcial (PACF), 39
- de autocovarianza, 39
- de pérdida, 140
- discriminante de Fisher, 63
- función aleatoria
 - espacial, 396
- estacionaria de segundo orden, 396
- estacionaria en sentido estricto, 396
- intrínsecamente estacionaria, 396
- no estacionaria, 396
- función de covarianza, 400
- función núcleo-calor, 438
- fútbol, 569
- geoestadística, 395
- geoprocесamiento, 495, 581
- git, 481
- github, 481
- goles, 570
- goles esperados, 572
- google earth engine, gee, 495
- gradient boosting, 172, 184, 187, 193, 194
- grado, 363
- grafo, 359, 360
 - betweenness, 370
 - eigenvector, 370
 - walktrap, 370
- gráfico de dispersión, 525
- gráfico de sedimentación, 220, 241, 254
- gráfico de silueta, 228
- hard margin, 140
- heatmap, 205
- hiperparámetro, 122, 141, 143, 150, 172, 173, 186, 187, 193
- hiperparámetros, 230, 592
- hiperplano, 140
- hoja, 112
- huella lingüistica, 344
- I de Morna, 419
- identificación de los factores, 251
- importancia, 145, 146, 170
- impureza
 - de Gini, 112, 174
- incendios forestales, 435
- indentación, 456
- independencia

Índice alfabético

647

- condicional, 104
- global, 104
- parcial, 104
- independencia condicional, 159
- índice
 - de complejidad de Hoffmann, 256
 - KMO, 252, 253
- índice de Dunn, 221
- industria, 549
- INE, 42
- información espacio-temporal, 580
- información redundante, 235
- informes, 451
 - bibliografía, 456
 - gráficos, 461
 - parámetros, 459
 - plantilla, 456
 - referencias cruzadas, 461
- instalar git, 483
- inteligencia artificial, 293
- intensidad, 433
- interfaz de usuario, 466
- interpolation, 409
- interpretación de los factores, 252, 258, 262
- Interpretación de modelos espaciales, 424
- IPC, 42
- John Snow, 379
- k-fold cross-validation, 30
- k-medianas, 229
- k-medias, 226
 - armónicas, 227
 - difuso, 227
 - recortadas, 227
 - sparse, 227
 - sparse robusto, 227
- k-medoides
 - CLARA, 228
 - CLARANS, 229
 - PAM, 227
- k-vecinos, 149, 151
- kernel, 141, 142, 147
- kriging, 409
 - ordinario, 409
 - simple, 409
- universal, 409
- Laplacian eigenmaps, 245
- latent dirichlet allocation, LDA, 612
- latex, 463
- lematización, 344
- lexicón, 345
- LibreOffice, 455
- literate programming, 452
- loadings, 242, 243, 256
- machine learning, 145, 245, 293–295, 592
- manifold learning, 235, 246
- mapa, 388
 - de coropletas, 389
 - espacio-temporal, 391
 - interactivo, 392
- mapa de calor, 545
- mapas, 581
- mapping, 410
- Mar Mediterráneo, 581
- margen, 137, 139, 140
- markdown, 451
- marketing, 511
- Marvel, 372
- matriz
 - de cargas, 243
 - de correlaciones, 239, 240, 246
 - de covarianzas, 236, 238, 240, 246
 - de puntuaciones, 244
 - cofenética, 219
 - de cargas, 251, 255, 258
 - de correlaciones, 248, 251, 252, 264, 265
 - de correlaciones anti-imagen, 252
 - de correlaciones de los residuos, 253
 - de correlaciones reducida, 255
 - de correlaciones reproducida, 253
 - de covarianzas, 257, 265
 - de disimilaridad, 268, 271, 272, 276
 - de distancia, 268
 - de distancias, 203
 - de proximidad, 267–269, 272
 - de similaridad, 268, 269
 - de transformación, 258, 260
- matriz de adyacencia, 362
- matriz de documentos, 346, 353

- maximum variance unfolding, 245
mayores de 45 años, 550
MCLUST, 230
medida de adecuación muestral, 252
medidas de asociación
 para tablas 2×2 , 207
 en tablas 2x2, 98
 cuadrado medio de la contingencia, 100
 odds ratio, 100
 Q de Yule, 99
 riesgo relativo, 100
 V de Cramer, 100
 en tablas RxC, 101
 basadas en la reducción proporcional
 del error, 102
 coeficiente de contingencia, 100, 102
 cuadrado medio de la contingencia, 102
 derivadas del estadístico Chi-cuadrado, 101
 lambda de Goodman y Kruskal, 102
 T de Tschuprow, 102
 V de Cramer, 102
meseta, 401
 parcial, 409
Messi, 569
Metodología CRISP-DM, 552
Microsoft Word, 455
minería
 de opinión, 345
 de textos, 341
missing values, 235
MMM (Marketing Mix Modeling), 511
modalidad, 87
modelado de temas, 344
modelo
 ARIMA, 40
 logarítmico lineal, 105
 sparse, 32
modelo RFM, 552
Modelos económicos espaciales, 420
monetary, 552
motor de inferencia, 597
MTA (Multitouch Attribution), 511
multicolinealidad, 235
máxima verosimilitud, 408
máxima verosimilitud compuesta, 408
máxima verosimilitud restringida, 408
método
 alpha, 257
 CHAID, 219
 de Anderson-Rubin, 265
 de Barlett, 265
 de componentes principales, 254, 255, 257
 de Fortin, 230
 de la descomposición triangular, 257
 de la distancia entre centroides, 211
 de la distancia media, 211
 de la distancia promedio, 215
 de la mediana, 212
 de Lance y Williams, 214
 de las combinaciones de Wolf, 230
 de los factores principales, 255–257, 262, 265, 266
 de máxima verosimilitud, 257, 264
 de mínimos cuadrados generalizados, 257
 de mínimos cuadrados no ponderados, 257
 de Thompson, 265
 de Ward, 212
 del análisis de la asociación, 217
 del análisis imagen, 257
 del centroide, 257
 del detector automático de interacciones, 218
 del encadenamiento intra-grupos, 213
 del vecino más cercano, 210
 del vecino más lejano, 210
 minres, 257
 modal de Wishart, 230
 TaxMap, 230
métodos basados en máxima verosimilitud, 408
mínimos cuadrados generalizados, 408
mínimos cuadrados ordinarios, 408
mínimos cuadrados ponderados, 408
n-gramas, 343, 354
Naive Bayes, 158, 159
neurona, 298, 299, 301, 307, 320, 324
nivel, 87
nodo, 112, 360
nodo raíz, 112
nube, 495
nube de palabras, 522, 524

Índice alfabético

649

- nubes de palabras, 346, 350, 353
nubes dinámicas, 226
nugget effect, 402
número
 de árboles, 173, 187
número óptimo de clusters, 220, 228

oceáanos, 579
OPTICS, 230
outliers, 241

padding, 325
palabras vacías, 342, 349, 355
pancarta, 217
pandoc, 455
parada temprana, 117
paro de muy larga duración, 548
paro registrado, 543
partial least squares, 235, 236, 246
partición, 112, 128, 174
parámetro de escala, 404
parámetro de suavizado, 434
pat, 485
patrones espaciales, 431
patrones espaciales de puntos, 431
patrón factorial, 250, 251, 258, 259, 261, 264
penalización shrinkage, 26
pequeños múltiples, 581
perceptrón, 295, 298–300, 305, 307, 320, 324
 multicapa, 305, 320, 338
perfil, 76
perfil de los clientes, 551
periódico, 535
poda, 118
pooling, 326, 327
poyección
 Robinson, 384
predicción krigada, 396
preprocesamiento, 332
procedimiento de muestreo, 89
procesamiento del lenguaje natural, NLP, 611
Procesamiento del Lenguaje Natural, PLN,
 341
proceso estocástico, 39
 espacial, 396
profundidad
 del árbol, 117, 131, 173, 187
punto
 de decisión, 112
puntuaciones factoriales, 251, 264, 265
python, 453

quarto, 451

R markdown, 451
rama, 112
random forest, 171–174, 184, 187
rango, 401
ranking de percentiles, 552
razón de riesgo, 566
reactividad, 476
recency, 552
reconocimiento, 296
red neuronal, 589, 592
 artificial, 298, 320–322
 convolucional, 296, 321, 338
red social, 360
redes lineales, 443
reducción de la dimensionalidad, 235
referencias cruzadas, 462
regionalización, 396, 400, 409
región de tolerancia, 406
región mediterránea, 579
regla, 597
regresión
 de Cox, 566
regresión lasso, 32
regresión ridge, 26
regularización, 193
repositorio local, 492
repositorio remoto, 492
reproducibilidad, 451
residuos
 de Haberman, 103
 estandarizados, 103
 ajustados, 101, 103
restauración, 545
riesgo, 564
Ronaldo, 569
rotaciones
 directas, 260
 indirectas, 260

- oblicuas, 258, 260, 261
- ortogonales, 252, 258–261
- rotación
 - BINORMALMIN, 261
 - BIQUARTIMAX, 260
 - BIQUARTIMIN, 261
 - COVARIMIN, 261
 - EQUAMAX, 260
 - OBLIMAX, 260, 261
 - OBLIMIN, 261
 - ORTOBLICUA, 261
 - ORTOMAX, 259
 - PROMAX, 261
 - QUARTIMAX, 259, 260
 - QUARTIMIN, 261
 - VARIMAX, 259–262, 265
 - VARIMAX normalizada, 259, 262
- Rstudio, 482
- S-Stress, 270
- Sammon’s mapping, 245
- script, 459
- sector, 543
- segmentación, 296
- segmentar, 551
- selección de variables, 19
- selección stepwise, 22
 - backward, 23
 - forward, 23
- semivariograma, 398–400, 402
 - comportamiento, 401
 - con meseta, 403
 - con meseta y efecto hoyo, 404
 - empírico, 406
 - J-Bessel, 404
 - logarítmico, 405
 - potencial, 405
 - sin meseta, 405
 - anisotrópico, 400
 - comportamiento, 402
 - efecto pepita puro, 404
 - esférico, 403
 - exponencial, 403
 - gausiano, 404
 - isotrópico, 400, 403
 - válido, 402, 407
- semivariograma empírico
 - direccional, 406
 - omnidireccional, 406
- serie temporal, 39
- servidor, 466
- sesgo de selección, 560
- sexo, 543
- shiny, 463
- Shrinkage, 20
- shrinkage, 26
- siete puentes de Königsberg, 359
- similaridad, 207
- similitud, 204
- sistema de referencia de coordenadas, 381
- sistema experto, 597
- sobreajuste, 117, 167, 184, 187, 193, 194, 328
- sobreentrenamiento, 595
- sobrealmuestreo, 622
- soft margin, 140
- solución factorial, 263
 - completa, 251, 266
- sparse PCA, 245
- splines, 409
- ssh, 484
- stemming, 344
- stopwords, 342, 349, 355
- Stress de Kruskal, 270, 273
- stride, 325
- sub-almuestreo, 441
- subconjunto
 - testing, 25
 - training, 25
- subalmuestreo, 622
- SVM, 139, 147
- t-distributed stochastic neighbor embedding, 245
- t-SNE, 245, 246
- tabla
 - de contingencia, 87
- tabla de contingencia, 88
- tabla de contingencia 2×2 , 218
- tablas
 - de contingencia 2x2, 89
 - de contingencia multidimensionales, 89
 - de contingencia RxC, 89

Índice alfabético

651

- formateadas, 461
 - `tabPanel()`, 469
 - `tabsetPanel()`, 469
 - tanglegrama, 212
 - tasa
 - de aprendizaje, 187
 - tasa de riqueza léxica, TTR, 343
 - temperatura superficial, 497, 579
 - tendencia, 39
 - teorema
 - de Bayes, 157–159
 - test de esfericidad, 242
 - `tibble`, 544
 - tiempo de búsqueda de empleo, 543
 - `token`, 343
 - tokenización, 343
 - traje, 597
 - truco
 - del kernel, 140
 - turismo, 545
 - twitter, 521
 - técnicas de agrupación, 201
 - jerárquicas, 208, 210
 - aglomerativas, 210
 - divisivas, 210, 215
 - no jerárquicas, 208, 210
 - basadas en la densidad de elementos, 210, 225
 - basadas en mixturas de modelos, 225
 - bi-cluster, 225, 232
 - block-cluster, 225, 232
 - clustering basado en mixturas de modelos, 233
 - clustering difuso, 233
 - co-cluster, 225, 232
 - de clusterización difusa, 225
 - de reasignación, 210, 225
 - de reducción de la dimensionalidad, 225, 232
 - directas, 225, 232
 - otras, 210
 - Q-cluster, 225, 232
 - R-cluster, 225, 232
 - two mode-cluster, 225, 232
 - técnicas híbridas, 235, 236, 244, 246
- UMA (Unified Measurement Approaches), 512
- unicidad, 250, 255, 256, 258, 266
- utilidades parciales, 75
- validación cruzada, 242
- valores regionalizados, 396
- valores semivariográficos, 406, 407
- variabilidad, 235–238, 241, 246, 247, 250
- variable regionalizada, 396
- variables estandarizadas, 239, 247
- varianza
 - intra-cluster, 213
 - compartida, 248, 266
 - común, 248, 255, 266
 - específica, 248
 - inter-grupos, 225
 - intra-grupos, 225, 226
 - no compartida, 249
 - residual, 248
 - única, 248, 250, 255, 257
- varianza a priori, 401
- virus SARS-CoV-2, 559
- Visualización, 581
- visualización, 521
- Voronoi, 440
- WAVECLUSTER, 230
- web scraping, 569
- weight decay, 329
- YAML, 453
- ángulo de tolerancia, 406
- árbol
 - de clasificación, 109, 112, 120, 165, 173, 174
 - de decisión, 109, 165, 167, 170, 171, 173, 174, 184, 187, 193
 - de regresión, 110, 127, 132, 165, 173, 174
- índice de propensión, 560